

Dobór architektury i parametrów procesu uczenia sieci neuronowych w doskonaleniu procesów produkcyjnych

Monika Kulisz



**Dobór architektury
i parametrów
procesu uczenia
sieci neuronowych
w doskonaleniu
procesów produkcyjnych**

Rada Naukowa Wydawnictwa Politechniki Lubelskiej

Przewodnicząca:

Agnieszka Rzepka

Dyrektor CINT:

Katarzyna Weinper

Przedstawiciele dyscyplin naukowych Politechniki Lubelskiej:

Marzenna Dudzińska

Małgorzata Franus

Arkadiusz Gola

Beata Kowalska

Grzegorz Komarzynieć

Jarosław Latański

Tomasz Lipecki

Zbigniew Łagodowski

Lucjan Pawłowski

Natalia Przesmycka

Halina Rarot

Magdalena Rzemieniak

Arkadiusz Syta

Wydawnictwo Politechniki Lubelskiej:

Magdalena Chołojczyk

Karolina Famulska-Ciesielska

Katarzyna Pełka-Smętek

Dobór architektury i parametrów procesu uczenia sieci neuronowych w doskonaleniu procesów produkcyjnych

Monika Kulisz



WYDAWNICTWO
POLITECHNIKI
LUBELSKIEJ

Lublin 2025

RECENZENCI:

prof. dr hab. **Anna Burduk** (Politechnika Wrocławska)

dr hab. inż. **Radosław Wajman**, prof. PŁ (Politechnika Łódzka)

AUTOR:

dr inż. **Monika Kulisz**, ORCID: 0000-0002-8111-2316

Politechnika Lubelska – ROR: <https://ror.org/024zjzd49>

Redaktor prowadzący: **Karolina Famulska-Ciesielska**

Korekta językowa: **zespół redakcyjny**

Skład i łamanie: **Łukasz Maj**

Projekt graficzny serii: **Łukasz Maj**

Zdjęcie na okładce i stronie przedtytułowej pochodzi ze strony internetowej www.freepik.com.
Jeśli nie wskazano źródła ilustracji, stanowi ona opracowanie własne autorki.

Publikacja wydana za zgodą **Rektora Politechniki Lubelskiej**

ISBN: 978-83-7947-635-0 (wersja drukowana)

ISBN: 978-83-7947-636-7 (wersja elektroniczna)

DOI: 10.35784/9788379476367

Wydawca: Wydawnictwo Politechniki Lubelskiej
www.wpl.pollub.pl
ul. Nadbystrzycka 36C, 20-618 Lublin
tel. (81) 538-46-59



WYDAWNICTWO
POLITECHNIKI
LUBELSKIEJ

Książka udostępniona jest na licencji Creative Commons Uznanie autorstwa – na tych samych warunkach 4.0 Międzynarodowe (CC BY-SA 4.0)

Nakład: 50 egz.

Spis treści

Wykaz ważniejszych skrótów i symboli	9
1. Wstęp	11
2. Cel i zakres pracy	17
3. Sztuczne sieci neuronowe	21
3.1. Sieci neuronowe – inspiracja ludzkim mózgiem	24
3.2. Perceptron wielowarstwowy	28
3.3. Rekurencyjne sieci neuronowe LSTM	31
3.4. Konwolucyjne sieci neuronowe	34
3.5. Transfer learning	39
3.6. Dobór odpowiedniej architektury sieci neuronowej	41
3.7. Kluczowe aspekty podczas uczenia sieci neuronowych	46
3.7.1. Metody optymalizacyjne w sieciach neuronowych	49
3.7.2. Techniki regularyzacji w sieciach neuronowych	53
3.8. Wskaźniki oceny jakości sieci neuronowych	54
3.8.2. Wskaźniki oceny jakości sieci neuronowych dla problemów klasyfikacyjnych	58
4. Optymalizacja parametrów obróbki skrawaniem	63
4.1. Proces frezowania stopów magnezu	65
4.2. Identyfikacja procesu na podstawie modelu doświadczalnego	67
4.3. Metodyka prowadzonych badań eksperymentalnych	69
4.4. Optymalizacja parametrów obróbki frezowaniem w celu uzyskania zadanych wartości chropowatości powierzchni	72
4.4.1. Metodyka modelowania parametrów chropowatości	73
4.4.2. Wyniki modelowania wybranych parametrów chropowatości 2D	77
4.4.3. Optymalizacja parametrów obróbki	89
4.5. Optymalizacja parametrów obróbki przy ograniczeniach siłowych	98
4.5.1. Metodyka modelowania siły skrawania	99
4.5.2. Wyniki modelowania siły skrawania	100
4.5.3. Optymalizacja parametrów obróbki	104
5. Ocena jakości wyrobów gotowych	115
5.1. Wykrywanie wad w odlewanych wyrobach gotowych – problem klasyfikacyjny	117
5.1.1. Metodyka prowadzonych badań	118
5.1.2. Wyniki działania klasyfikatorów	130
5.1.3. Ocena jakości wyrobów gotowych na podstawie wykrywania wad w odlewach	133
5.1.4. Koncepcja implementacji modelu do wykrywania wad w odlewach w kontroli jakości	136

5.2. Wykrywanie błędów kształtu na linii produkcyjnej – problem klasyfikacyjny	142
5.2.1. Metodyka prowadzonych badań oraz przygotowanie danych do modelowania	143
5.2.2. Wyniki modelowania klasyfikatora	149
5.2.3. Implementacja aplikacji do wykrywania błędów	151
5.3. Określenie twardości po odpuszczaniu wyrobów gotowych – problem regresyjny	159
5.3.1. Metodyka prowadzonych badań	160
5.3.2. Wyniki modelowania twardości po procesie odpuszczania	162
5.3.3. Ocena jakości wyrobów gotowych na podstawie twardości po procesie odpuszczania	173
6. Predykcja awarii maszyn	177
6.1. Metodyka modelowania predykcji awarii	179
6.2. Wyniki modelowania klasyfikatora dotyczącego wystąpienia awarii	182
6.3. Wyniki modelowania klasyfikatora dotyczącego określania typu awarii	192
6.4. Wyniki modelowania klasyfikatora dotyczącego określania typu awarii na ograniczonym zbiorze danych	199
6.5. Predykcja awarii maszyn z wykorzystaniem odpowiedniego interfejsu	207
7. Metodyka doboru architektury i parametrów procesu uczenia sieci neuronowych w doskonaleniu procesów produkcyjnych	217
8. Zakończenie	235
8.1. Uwagi końcowe i wnioski	236
8.2. Kierunki dalszych badań	240
Literatura	243

Dobór architektury i parametrów procesu uczenia sieci neuronowych w doskonaleniu procesów produkcyjnych

Celem pracy jest zaproponowanie sposobu doboru architektury oraz parametrów procesu uczenia sztucznych sieci neuronowych, które mogą być skutecznie zastosowane w doskonaleniu procesów produkcyjnych. Praca koncentruje się na wykorzystaniu modeli neuronowych w następujących trzech obszarach: optymalizacji parametrów obróbki skrawaniem, ocenie jakości wyrobów gotowych oraz predykcji awarii maszyn. W ramach pracy poddano analizie wybrane architektury sieci neuronowych, takie jak perceptrony wielowarstwowe, rekurencyjne sieci LSTM oraz konwolucyjne sieci neuronowe. Opracowano szczegółową metodykę doboru odpowiednich architektur oraz parametrów ich uczenia, która obejmowała kompleksowy proces przygotowania danych, selekcji odpowiednich modeli oraz ich dostosowania do specyficznych wymagań poszczególnych zastosowań przemysłowych. Kluczowym aspektem tej metodyki było uwzględnienie charakterystyki dostępnych danych oraz specyfiki zadań, co pozwoliło na optymalny dobór architektury sieci neuronowej dla każdego z analizowanych obszarów.

Słowa kluczowe: sztuczne sieci neuronowe, architektura sieci neuronowej, proces uczenia sieci neuronowej, doskonalenie procesów produkcyjnych

Selection of neural network architecture and training parameters for manufacturing process improvement

The objective of this study is to propose a methodology for selecting the architecture and training parameters of artificial neural networks that can be effectively applied to the enhancement of manufacturing processes. The research focuses on the utilization of neural network models in three key areas: optimization of machining parameters, quality assessment of finished products, and prediction of machine failures. The study examines selected neural network architectures, including multilayer perceptrons, recurrent LSTM networks, and convolutional neural networks. A detailed methodology was developed to determine suitable architectures and training parameters, encompassing a comprehensive process of data preparation, model selection, and adaptation to the specific requirements of individual industrial applications. A key part of this method was taking into account the types of data that were available and the specifics of the tasks. This let the best neural network architectures be chosen for each of the domains that were being studied.

Keywords: artificial neural networks, neural network architecture, neural network training process, manufacturing process improvement

Wykaz ważniejszych skrótów i symboli

- ANN – sztuczna sieć neuronowa (ang. *artificial neural networks*)
- AUC – krzywa AUC (ang. *area under the curve*)
- CNN – konwolucyjna sieć neuronowa (ang. *convolutional neural network*)
- BR – algorytm uczenia – regularyzacja Bayesowska
- DNN – głęboka sieć neuronowa (ang. *deep neural networks*)
- FFNN – sieć jednokierunkowa (ang. *feedforward neural networks*)
- FPR – wskaźnik fałszywie pozytywnych wyników (ang. *false positive rate*)
- GPU – jednostka przetwarzania grafiki (ang. *graphics processing unit*)
- LM – algorytm uczenia Levenberga–Marquardta
- LSTM – sieć LSTM (ang. *long short-term memory*)
- MAE – średni błąd bezwzględny (ang. *mean absolute error*)
- MAPE – średni bezwzględny błąd procentowy (ang. *mean absolute percentage error*)
- MLP – perceptron wielowarstwowy (ang. *multi layer perceptron*)
- MSE – błąd średniokwadratowy (ang. *mean squared error*)
- R – współczynnik korelacji
- RIE – względny błąd informacji (ang. *relative information error*)
- RMSE – pierwiastek błędu średniokwadratowego (ang. *root mean squared error*)
- RNN – rekurencyjne sieci neuronowe (ang. *recurrent neural network*)
- ROC – krzywa ROC (ang. *receiver operating characteristic*)
- SCG – algorytm uczenia – gradient skalowany sprzężony (ang. *scaled conjugate gradient*)
- SVM – maszyna wektorów nośnych (ang. *support vector machine*)
- TPR – wskaźnik prawdziwie pozytywnych wyników (ang. *true positive rate*)
- TPU – jednostka przetwarzania tensora (ang. *tensor processing unit*)
- a_p – głębokość skrawania
- a_e – szerokość skrawania
- f – posuw na ostrze
- F_x – składowa siły skrawania wzdłuż osi X
- F_y – składowa siły skrawania wzdłuż osi Y
- F_z – składowa siły skrawania wzdłuż osi Z
- n – prędkość obrotowa
- Ra – średnia arytmetyczna bezwzględnych rzędnych profilu chropowatości
- Rz – największa wysokość profilu chropowatości
- RSm – średnia szerokość rowków elementów profilu chropowatości
- v_c – prędkość skrawania

1. Wstęp

Doskonalenie procesów produkcyjnych stanowi jedno z najważniejszych wyzwań, przed którymi stoi współczesny przemysł. W obliczu rosnącej konkurencji na globalnym rynku oraz dynamicznie zmieniających się potrzeb konsumentów przedsiębiorstwa produkcyjne muszą nieustannie dążyć do zwiększania efektywności, redukcji kosztów oraz podnoszenia jakości oferowanych produktów. W literaturze przedmiotu podkreśla się, że doskonalenie procesów produkcyjnych polega na systematycznym usprawnianiu wszystkich aspektów produkcji: od planowania i projektowania, przez zarządzanie zasobami, aż po kontrolę jakości i logistykę [1].

Istotnym elementem doskonalenia procesów produkcyjnych jest optymalizacja zarządzania zasobami, która obejmuje nie tylko zarządzanie surowcami i materiałami, ale także efektywne wykorzystanie maszyn, narzędzi oraz pracy ludzkiej. W tym kontekście wdrażanie nowoczesnych metod zarządzania produkcją, takich jak Lean Manufacturing czy Six Sigma, umożliwia znaczną redukcję marnotrawstwa oraz zwiększenie wydajności produkcji [2,3]. Dzięki tym podejściom przedsiębiorstwa mogą osiągnąć wyższą jakość produktów przy jednoczesnym obniżeniu kosztów produkcji, co jest kluczowe w kontekście utrzymania konkurencyjności na rynku globalnym [4].

Kontrola jakości stanowi kolejny istotny obszar doskonalenia procesów produkcyjnych. Współczesne podejście do zarządzania jakością zakłada integrację narzędzi informatycznych z procesami produkcyjnymi, co umożliwia

bieżącą analizę i monitorowanie jakości produktów na różnych etapach produkcji. Zastosowanie zaawansowanych technik, takich jak statystyczne sterowanie procesem (SPC) oraz systemy wczesnego ostrzegania o odchyleniach od norm pozwala na szybkie reagowanie na wszelkie nieprawidłowości, co przekłada się na zmniejszenie liczby wadliwych produktów i zwiększenie satysfakcji klienta [5]. Równie ważnym aspektem doskonalenia procesów produkcyjnych jest elastyczność systemów produkcyjnych, która pozwala na szybkie dostosowanie się do zmieniających się warunków rynkowych. Elastyczność ta staje się szczególnie istotna w kontekście globalizacji i wzrostu złożoności łańcuchów dostaw, które wymagają szybkiego reagowania na zmiany w popycie, różnorodność produktów oraz skracanie cykli życia produktów [6].

W kontekście doskonalenia procesów produkcyjnych kluczowe jest zrozumienie, że każdy element procesu może zostać zoptymalizowany, co w konsekwencji prowadzi do poprawy całego systemu [7]. Obejmuje to zarówno techniczne aspekty związane z maszynami i narzędziami, jak i organizacyjne, dotyczące zarządzania ludźmi, czasem i zasobami [8,9]. Właściwie przeprowadzona optymalizacja procesów produkcyjnych prowadzi do zwiększenia produktywności, obniżenia kosztów operacyjnych oraz poprawy jakości produktów, co jest ważne dla utrzymania konkurencyjności na rynku [10].

W ramach doskonalenia procesów produkcyjnych wyróżnia się wiele istotnych obszarów, w tej monografii jednak uwaga zostanie skierowana na trzy z nich: optymalizację parametrów obróbki skrawaniem, ocenę jakości wyrobów oraz predykcję awarii maszyn.

Optymalizacja parametrów obróbki jest niezbędna do zapewnienia, że proces produkcyjny przebiega w sposób jak najbardziej efektywny. Dobre zarządzanie parametrami, takimi jak prędkość skrawania, posuw, głębokość obróbki czy temperatura w strefie skrawania pozwala na osiągnięcie optymalnych wyników produkcji zarówno pod względem jakości wyrobów, jak i efektywności energetycznej. Precyzyjne dostosowanie tych parametrów prowadzi do zmniejszenia zużycia narzędzi, redukcji ilości odpadów, a także minimalizacji czasu produkcji, co przekłada się na niższe koszty i wyższą rentowność produkcji [7].

Ocena jakości wyrobów to istotny element w procesie produkcyjnym, który bezpośrednio wpływa na satysfakcję klientów i reputację marki. Współczesne standardy jakości wymagają, aby każdy produkt opuszczający linię produkcyjną spełniał określone normy i specyfikacje. Skuteczna kontrola jakości nie tylko zapobiega wprowadzeniu na rynek wadliwych produktów, ale także pozwala na szybkie zidentyfikowanie i eliminowanie problemów w procesie produkcyjnym,

co zapobiega powielaniu błędów. Regularne monitorowanie i ocena jakości wyrobów są zatem niezbędne dla utrzymania wysokiego poziomu zadowolenia klientów oraz uniknięcia kosztownych reklamacji [11,12].

Zapobieganie awariom maszyn jest kluczowe dla zapewnienia ciągłości produkcji i minimalizacji kosztów związanych z przestojami. Awarie maszyn mogą prowadzić do nieplanowanych przestojów, które generują wysokie koszty oraz mogą opóźniać realizację zamówień, co negatywnie wpływa na relacje z klientami. Wprowadzenie skutecznych strategii utrzymania ruchu, które pozwalają na wczesne wykrywanie problemów i prewencyjne usuwanie usterek, jest nieodzowne dla zachowania ciągłości produkcji [13]. Przewidywanie i zapobieganie awariom pozwala na lepsze planowanie działań serwisowych, optymalne zarządzanie zasobami oraz wydłużenie żywotności maszyn, co przekłada się na zmniejszenie kosztów operacyjnych i zwiększenie efektywności produkcji [14].

Współczesne przedsiębiorstwa produkcyjne muszą nieustannie poszukiwać innowacyjnych rozwiązań, które pozwolą na doskonalenie procesów produkcyjnych. Zastosowanie technologii informatycznych, takich jak systemy ERP, sztuczna inteligencja (AI) oraz internet rzeczy (IoT), umożliwia automatyzację i integrację procesów produkcyjnych, co przyczynia się do zwiększenia efektywności oraz redukcji kosztów. W kontekście dynamicznego rozwoju technologii cyfrowych istotną rolę odgrywają sztuczne sieci neuronowe (ANN), które w ostatnich latach zyskały ogromne znaczenie. Stały się jednym z narzędzi stosowanych w przemyśle wytwórczym, oferując bowiem znaczny potencjał optymalizacji procesów, zwiększania wydajności, obniżenia kosztów i poprawy jakości produktów. Integracja ANN z produkcją jest częścią szerszego trendu w kierunku cyfryzacji i przyjęcia zaawansowanych technologii, takich jak AI, IoT i systemy cyber-fizyczne. Te elementy wspólnie napędzają transformację w kierunku inteligentnej produkcji i Przemysłu 4.0 [15,16].

Sztuczne sieci neuronowe nie tylko odgrywają kluczową rolę w optymalizacji obecnych procesów produkcyjnych, ale także mają potencjał, aby zrewolucjonizować przyszłość przemysłu. W miarę jak technologie oparte na ANN rozwijają się, mogą stać się fundamentem nowych, inteligentnych systemów zarządzania produkcją, które autonomicznie monitorują, zarządzają i optymalizują wszystkie aspekty procesu produkcyjnego. Takie systemy, dzięki zdolności do uczenia się na podstawie dużych ilości danych oraz adaptacji do zmieniających się warunków, mogą znacznie przewyższyć efektywność i precyzję tradycyjnych metod zarządzania. W konsekwencji integracja ANN z procesami wytwórczymi nie tylko umożliwi osiągnięcie lepszej jakości

produktów i większej wydajności, ale również przyczynia się do powstania w pełni zautomatyzowanych, inteligentnych fabryk przyszłości [17].

Potencjał sztucznych sieci neuronowych w doskonaleniu procesów produkcyjnych wynika z ich zdolności do przetwarzania i analizy dużych ilości danych w czasie rzeczywistym. Dzięki możliwości uczenia się na podstawie danych historycznych oraz bieżących sieci neuronowe są w stanie identyfikować wzorce, przewidywać zachowania systemów produkcyjnych oraz podejmować decyzje optymalizacyjne, które są trudne do osiągnięcia za pomocą tradycyjnych metod [18,19]. W kontekście współczesnych technologii wytwarzania takie zdolności są niezwykle cenne, gdyż umożliwiają dynamiczne dostosowywanie parametrów procesów produkcyjnych do zmieniających się warunków i wymagań.

ANN znajdują szerokie zastosowanie w przemyśle, znacząco wpływając na różne aspekty produkcji i utrzymania ruchu. Stosowane są do monitorowania wydajności przekładni turbin wiatrowych, wspierając strategię predykcyjnego utrzymania ruchu poprzez analizę danych SCADA (systemów informatycznych nadzorujących przebieg procesów produkcyjnych) [20]. W przemyśle motoryzacyjnym ANN optymalizują predykcyjne utrzymanie generatorów Diesla, zwiększając ich niezawodność i efektywność [21]. W sektorze chłodniczym przyczyniają się do oszczędności energii oraz poprawy efektywności systemów chłodniczych [22]. W przemyśle meblarskim ANN odgrywają istotną rolę w monitorowaniu defektów i poprawie kontroli jakości, co pozwala na szybkie identyfikowanie i eliminowanie wad produkcyjnych, a tym samym zwiększenie efektywności całego procesu produkcji [23]. W branży obróbki materiałowej sieci neuronowe są wykorzystywane do przewidywania chropowatości powierzchni oraz tempa usuwania materiału podczas obróbki strumieniowo-słusiennej nanokompozytów, co prowadzi do optymalizacji procesów i zwiększenia precyzji wytwarzania [24]. W kontekście optymalizacji projektowania ANN znajdują zastosowanie w poprawie wydajności wymienników ciepła oraz redukcji kosztów produkcji, co przekłada się na bardziej efektywne i ekonomiczne rozwiązania w przemyśle [25].

Ważnym aspektem stosowania sztucznych sieci neuronowych w przemyśle jest umiejętność doboru odpowiedniej architektury oraz parametrów procesu uczenia sieci, co ma kluczowe znaczenie dla uzyskania ich wysokiej jakości. Dobór architektury sieci neuronowej musi być precyzyjnie dostosowany do specyfiki danego problemu, ponieważ różne zadania – takie jak klasyfikacja czy predykcja – wymagają odmiennych podejść. Równie istotnym czynnikiem wpływającym na wybór architektury i parametrów sieci jest wielkość zestawu

danych, na których model będzie trenowany. W przypadku dużych zestawów danych bardziej złożone architektury, takie jak głębokie sieci neuronowe, mogą okazać się bardziej efektywne. Większy zasób danych pozwala na skuteczne trenowanie modeli o dużej liczbie warstw i parametrów, co z kolei może prowadzić do lepszego uchwycenia skomplikowanych zależności i wzorców w danych. Z kolei w sytuacjach, gdy dostępne są jedynie małe zestawy danych, należy zastosować inne podejście. Zdarza się, że w takich przypadkach złożone modele prowadzą do nadmiernego dopasowania, co skutkuje słabą generalizacją na danych testowych. Dlatego w przypadku małych zbiorów danych preferowane mogą być prostsze modele.

Wielkość zestawu danych wpływa także na dobór parametrów procesu uczenia, takich jak: liczba neuronów w warstwie ukrytej, liczba epok, metod optymalizacji czy technik regularyzacji. Wszystkie te czynniki podkreślają znaczenie świadomego i precyzyjnego doboru architektury oraz parametrów procesu uczenia sieci neuronowych w zależności od charakterystyki problemu oraz dostępnych danych. Umiejętne zarządzanie tymi aspektami pozwala na tworzenie modeli, które nie tylko osiągają wysoką dokładność, ale także są zdolne do efektywnej generalizacji w rzeczywistych warunkach produkcyjnych.

2. Cel i zakres pracy

Celem pracy jest zaproponowanie sposobu doboru architektury oraz parametrów procesu uczenia sztucznych sieci neuronowych, które mogą być skutecznie zastosowane w doskonaleniu procesów produkcyjnych. Autorka pracy koncentruje się na wykorzystaniu modeli neuronowych w następujących trzech obszarach: optymalizacji parametrów obróbki skrawaniem, ocenie jakości wyrobów gotowych oraz predykcji awarii maszyn. W ramach pracy poddano analizie wybrane architektury sieci neuronowych, takie jak: perceptrony wielowarstwowe (MLP), rekurencyjne sieci LSTM oraz konwolucyjne sieci neuronowe (CNN).

Problem badawczy podejmowany w niniejszej pracy dotyczy opracowania metodyki doboru odpowiednich architektur sieci neuronowych oraz parametrów ich uczenia, które najlepiej wspierają poprawę efektywności i jakości procesów produkcyjnych. Badania koncentrują się na trzech konkretnych przypadkach zastosowania: optymalizacji parametrów obróbki frezowaniem, ocenie jakości wyrobów gotowych w kontekście problemów klasyfikacyjnych oraz regresyjnych, a także predykcji awarii maszyn, w tym identyfikacji rodzaju awarii. Istotnym zagadnieniem jest ustalenie, które typy sieci neuronowych oraz jakie parametry procesu uczenia są najskuteczniejsze w odniesieniu do specyficznych wyzwań związanych z tymi obszarami produkcyjnymi.

Nowość pracy polega na oryginalnym podejściu do doboru architektur i parametrów uczenia sieci neuronowych w różnych kontekstach produkcyjnych, co pozwala na elastyczne i wszechstronne ich zastosowanie w zależności

od specyfiki problemu, uwarunkowań organizacyjnych oraz dostępnych danych. W pracy przedstawiono konkretne przykłady zastosowania sieci neuronowych, takie jak: optymalizacja parametrów obróbki frezowaniem, ocena jakości wyrobów gotowych w problemach klasyfikacyjnych oraz regresyjnych, a także predykcja awarii maszyn. Przykłady te służyły jako tło dla opracowania metodyki doboru architektury i parametrów sieci, a nie były celem samym w sobie. Praca podkreśla, że w zależności od charakterystyki zadania – czy to klasyfikacji, predykcji czy też dostępności danych – różne architektury sieci neuronowych mogą okazać się bardziej odpowiednie, co świadczy o ich zdolności do adaptacji w zróżnicowanych warunkach produkcyjnych. Przedstawione rozwiązania mają na celu nie tylko teoretyczne rozwinięcie wiedzy, ale również dostarczają praktycznych narzędzi, które mogą zostać bezpośrednio zaimplementowane w przemyśle, co stanowi istotny wkład w rozwój nowoczesnych technologii wytwarzania i obszar inżynierii produkcji.

Zakres pracy obejmuje szczegółowe omówienie zagadnień związanych z zastosowaniem sztucznych sieci neuronowych w doskonaleniu procesów produkcyjnych. Praca została podzielona na kilka rozdziałów, z których każdy koncentruje się na odrębnym aspekcie omawianej tematyki. Wstęp wprowadza w tematykę pracy. Omówiono w nim znaczenie doskonalenia procesów produkcyjnych wobec współczesnych wyzwań przemysłowych oraz roli zaawansowanych technologii, takich jak sztuczne sieci neuronowe, w optymalizacji i transformacji produkcji. Przedstawiono również cel i zakres pracy.

Rozdział trzeci poświęcony jest szczegółowemu omówieniu teorii sztucznych sieci neuronowych. Rozpoczyna się od historycznego zarysu rozwoju tej technologii, a następnie przedstawia różne typy architektur sieci neuronowych, w tym MLP, rekurencyjne sieci LSTM oraz CNN. W rozdziale omówiono również koncepcję *transfer learning*, czyli wykorzystania wcześniej wytrenowanych modeli w nowych zadaniach, co znacząco zwiększa efektywność i skraca czas trenowania modeli. Rozdział obejmuje także analizę kluczowych kwestii związanych z doбором odpowiedniej architektury sieci neuronowej, a także szczegółowe omówienie aspektów dotyczących procesu uczenia, takich jak wybór funkcji aktywacji, zastosowanie algorytmów optymalizacji oraz metod oceny jakości opracowanych modeli.

W czwartym rozdziale podjęto zagadnienie optymalizacji parametrów obróbki skrawaniem z wykorzystaniem sztucznych sieci neuronowych. Omówiono proces frezowania stopów magnezu, metodykę badań eksperymentalnych procesu frezowania oraz wyniki modelowania parametrów chropowatości powierzchni. Rozdział zawiera również szczegółową analizę optymalizacji parametrów obróbki

pod kątem uzyskania odpowiedniej chropowatości powierzchni. Dodatkowo zaprezentowano wyniki modelowania składowej F_x siły skrawania oraz optymalizację parametrów obróbki z uwzględnieniem ograniczeń siłowych wpływających na jakość procesu obróbki.

Piąty rozdział koncentruje się na ocenie jakości wyrobów gotowych z wykorzystaniem zaawansowanych metod sztucznych sieci neuronowych, obejmując szczegółową analizę zarówno problemów klasyfikacyjnych, jak i regresyjnych. Rozdział ten w szczególności dogłębnie analizuje problematykę klasyfikacji wyrobów, zwracając szczególną uwagę na wykrywanie wad w odlewach, co stanowi istotny element w zapewnieniu wysokiej jakości produktów. Zaprezentowano metodę wykrywania wad z zastosowaniem CNN. Ponadto w rozdziale przedstawiono analizę problemów związanych z klasyfikacją błędów kształtu na liniach produkcyjnych, co jest istotne dla utrzymania dokładności i powtarzalności wytwarzanych elementów. W rozdziale zawarto również omówienie zastosowania sieci neuronowych w problemach regresyjnych, z naciskiem na przewidywanie twardości wyrobów po procesie odpuszczania. Przedstawiono metodykę tworzenia modeli regresyjnych oraz ich zastosowanie w precyzyjnym przewidywaniu właściwości mechanicznych wyrobów. Dodatkowo rozdział prezentuje koncepcję implementacji modeli opartych na sieciach neuronowych do wykrywania wad w ramach systemów kontroli jakości.

Rozdział szósty pracy poświęcony jest predykcji awarii maszyn przy wykorzystaniu zaawansowanych technik sztucznych sieci neuronowych. Rozdział rozpoczyna się od omówienia metodyki modelowania predykcji awarii, następnie zaprezentowano różne architektury sieci neuronowych, w tym MLP oraz rekurencyjne sieci neuronowe LSTM. Rozdział ten zawiera również omówienie praktycznych aspektów implementacji aplikacji do predykcji awarii maszyn poprzez zaprojektowanie odpowiedniego interfejsu użytkownika.

W rozdziale siódmym skoncentrowano się na opracowaniu szczegółowej metodyki, która pozwala na skuteczny dobór zarówno architektury sieci neuronowych, jak i parametrów procesu ich uczenia. Metodyka ta jest zaprojektowana z myślą o zastosowaniu w różnych procesach produkcyjnych w celu ich doskonalenia, z uwzględnieniem specyfiki analizowanych problemów czy dostępnych danych.

W zakończeniu zawarto podsumowanie wyników przeprowadzonych badań oraz wnioski z analizy zastosowania sztucznych sieci neuronowych w procesach produkcyjnych. Przedstawiono również propozycje dalszych kierunków badań, które mogą przyczynić się do efektywniejszego wykorzystania sztucznych sieci neuronowych w przemyśle.

3. Sztuczne sieci neuronowe

Sztuczne sieci neuronowe (ang. *artificial neural networks*, ANN) stanowią fundament współczesnych technologii związanych z uczeniem maszynowym i sztuczną inteligencją. Ich inspiracja sięga biologicznych układów nerwowych, w szczególności ludzkiego mózgu, co nadaje im unikalne właściwości pozwalające na przetwarzanie informacji w sposób zbliżony do działania ludzkiego umysłu. Aby zrozumieć pełnię ich możliwości i zastosowań, warto przyjrzeć się historii ich rozwoju oraz podstawowym zasadom funkcjonowania.

Początki idei sztucznych sieci neuronowych sięgają lat 40. XX wieku, kiedy to Warren McCulloch i Walter Pitts opublikowali przełomowy artykuł na temat logicznego rachunku idei immanentnych w układach nerwowych. Ich model, znany jako neuron McCullougha–Pittsa, był pierwszym matematycznym modelem neuronu, który pozwalał na realizację funkcji logicznych [26]. To otworzyło drogę do dalszych badań nad sztucznymi systemami przypominającymi biologiczne mózgi. Również w tym okresie, w 1949 roku, Donald Hebb wprowadził ważną koncepcję w dziedzinie neuropsychologii, która miała ogromny wpływ na rozwój sztucznych sieci neuronowych. Przedstawił zasadę, która jest obecnie znana jako reguła Hebba. Stwierdza ona, że „neurony, które wspólnie działają, łączą się”. Innymi słowy, jeśli dwa neurony są aktywowane jednocześnie, połączenie między nimi jest wzmacniane. Ta zasada Hebba stała się podstawą dla stworzenia teorii uczenia się w sieciach neuronowych, która sugeruje, że synaptyczne połączenia między neuronami mogą się wzmacniać

poprzez powtarzane współdziałanie [27]. Integracja modelu McCullougha-Pittsa i reguły Hebba stanowiła fundament, na którym zbudowano wczesne modele sztucznych sieci neuronowych.

W latach 50. i 60. XX wieku nastąpił dalszy rozwój tej dziedziny, zwłaszcza dzięki pracom Franka Rosenblatt, który stworzył perceptron – podstawowy model sieci neuronowej zdolny do uczenia się na podstawie dostarczonych danych [28]. Perceptron był w stanie klasyfikować dane liniowo separowalne, co jednak stanowiło jego ograniczenie. W tym okresie także Bernard Widrow i Marcian Hoff opracowali model ADALINE (ang. *adaptive linear neuron*), który wykorzystywał algorytm delta do dostosowywania wag w sieci neuronowej [29]. Był to kolejny krok w kierunku bardziej zaawansowanych modeli sieci neuronowych.

Prawdziwy przełom nastąpił w latach 80. XX wieku wraz z wprowadzeniem algorytmu propagacji wstecznej przez Geoffreya Hinton, Davida Rumelharta i Ronalda Williamsa. Algorytm ten umożliwił trenowanie wielowarstwowych sieci neuronowych, co z kolei pozwoliło na rozwiązywanie znacznie bardziej złożonych problemów [30]. Dzięki temu sztuczne sieci neuronowe zyskały na popularności i stały się jednym z głównych narzędzi w obszarze uczenia maszynowego.

W latach 90. rozwój sieci neuronowych był częściowo ograniczony przez problemy związane z mocą obliczeniową komputerów oraz z trudnościami w trenowaniu głębszych sieci. Niemniej jednak prace nad nowymi architekturami i technikami trenowania były kontynuowane. W 1998 roku Yann LeCun i jego współpracownicy zaprezentowali CNN, które zrewolucjonizowały analizę obrazów [31].

Na początku XXI wieku nastąpił kolejny znaczący przełom, głównie dzięki rozwojowi mocy obliczeniowej oraz dostępności dużych zbiorów danych. W 2006 roku Geoffrey Hinton i jego zespół stworzyli głębokie sieci neuronowe, które stały się fundamentem nowoczesnych technologii AI [32]. Ich prace były kontynuowane przez wielu innych badaczy, co przyczyniło się do dynamicznego rozwoju tej dziedziny.

W 2012 roku Alex Krizhevsky, Ilya Sutskever i Geoffrey Hinton przedstawili sieć AlexNet, która wygrała konkurs ImageNet Large Scale Visual Recognition Challenge (ILSVRC) z dużą przewagą, pokazując skuteczność głębokich sieci neuronowych w rozpoznawaniu obrazów [33]. To wydarzenie zapoczątkowało nową erę w badaniach nad sieciami neuronowymi. Kolejnym ważnym osiągnięciem była praca nad rekurencyjnymi sieciami neuronowymi (RNN) i ich ulepszonymi wersjami, takimi jak *long short-term memory* (LSTM) oraz *gated recurrent*

units (GRU). Okazały się bardzo skuteczne w przetwarzaniu sekwencji danych, co znalazło zastosowanie w analizie tekstu i mowy [34,35].

Sztuczne sieci neuronowe znajdują szerokie zastosowanie w różnych dziedzinach. W medycynie na przykład są wykorzystywane do diagnozowania chorób na podstawie obrazów medycznych, takich jak zdjęcia rentgenowskie czy skany MRI, zwiększając obiektywność i dokładność diagnoz oraz umożliwiając spersonalizowane plany leczenia [36–38]. W finansach sieci neuronowe są stosowane do przewidywania trendów rynkowych i oceny ryzyka kredytowego [39]. Technologia ta ponadto jest używana w rozpoznawaniu mowy i przetwarzaniu języka naturalnego, co umożliwia tworzenie zaawansowanych asystentów głosowych, takich jak Siri czy Alexa [40,41]. Dodatkowo połączenie rozpoznawania głosu z innymi funkcjami AI, takimi jak wykrywanie i rozpoznawanie twarzy, może prowadzić do stworzenia bardziej spersonalizowanych i intuicyjnych asystentów AI. Systemy te są w stanie wykonywać zadania, takie jak ustawianie przypomnień, wysyłanie wiadomości e-mail i odtwarzanie muzyki, poprawiając w ten sposób interakcję człowiek-komputer i zmniejszając wysiłek człowieka w różnych sektorach [42]. Zastosowanie zaawansowanych modeli językowych, takich jak GPT-3, dodatkowo poprawia inteligencję i szybkość reakcji tych wirtualnych asystentów, umożliwiając im funkcjonowanie jako wydajne interfejsy czatu, które mogą rozumieć i wykonywać polecenia mówione [43].

W produkcji sztuczne sieci neuronowe mogą odgrywać kluczową rolę w monitorowaniu procesów produkcyjnych oraz optymalizacji linii produkcyjnych. Dzięki analizie danych z czujników oraz innych źródeł sieci neuronowe mogą identyfikować anomalie i przewidywać awarie, co pozwala na przeprowadzenie prewencyjnych działań konserwacyjnych. Na przykład sieci neuronowe mogą być wykorzystywane do monitorowania stanu maszyn i wykrywania odchyłeń od normy, co pozwala na wczesne rozpoznawanie potencjalnych problemów [44]. Potrafią one między innymi identyfikować wzorce, przewidywać przyszłe zdarzenia, segmentować dane oraz obrazy, klasyfikować elementy w zbiorach, wspierać procesy decyzyjne, a także tworzyć reprezentacje danych [45–47]. W monitorowaniu produkcji sieci neuronowe mogą analizować dane w czasie rzeczywistym, wykrywać nieprawidłowości i przewidywać przyszłe zdarzenia, co umożliwi bardziej efektywne zarządzanie procesami produkcyjnymi [48,49].

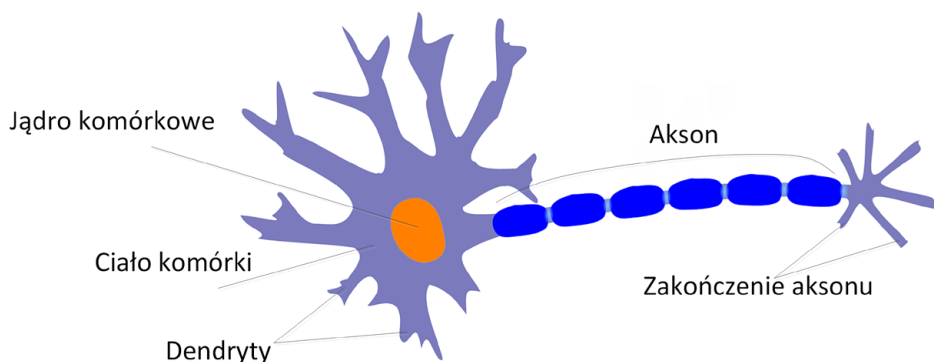
W sektorze energetycznym sieci neuronowe są wykorzystywane do prognozowania zużycia energii oraz optymalizacji jej dystrybucji. Na przykład sieci neuronowe mogą analizować dane historyczne dotyczące zużycia energii

oraz przewidywać przyszłe potrzeby, co pozwala na bardziej efektywne zarządzanie zasobami energetycznymi [50,51]. W rolnictwie ANN są stosowane do monitorowania rozwoju roślin oraz prognozowania plonów. Dzięki analizie danych z sensorów oraz zdjęć satelitarnych sieci neuronowe mogą identyfikować choroby roślin oraz przewidywać ich rozwój, co sprzyja szybkim i skutecznym działaniom prewencyjnym [52].

W logistyce sieci neuronowe mogą optymalizować trasy transportowe oraz zarządzanie łańcuchem dostaw. Analiza danych dotyczących ruchu drogowego, warunków pogodowych oraz innych czynników pozwala na optymalizację tras i minimalizację czasu dostaw, co przekłada się na redukcję kosztów oraz zwiększenie efektywności operacyjnej [53–55].

3.1. Sieci neuronowe – inspiracja ludzkim mózgiem

Sztuczne sieci neuronowe czerpią inspirację z ludzkiego mózgu, który jest niezwykle złożonym i efektywnym systemem przetwarzania informacji. Mózg ludzki składa się z miliardów neuronów, które są połączone ze sobą za pomocą synaps. Neurony komunikują się poprzez przekazywanie impulsów elektrycznych i chemicznych, co pozwala na przetwarzanie ogromnych ilości informacji w czasie rzeczywistym. Poniżej przedstawiono, jak te biologiczne inspiracje przekładają się na modele sieci neuronowych, które są wykorzystywane w uczeniu maszynowym i sztucznej inteligencji (ryc. 3.1).



Ryc. 3.1. Model ludzkiego neuronu

Opracowanie własne na podstawie pixabay.com (licencja CC0)

Neuron, podstawowa jednostka przetwarzania w mózgu, składa się z ciała komórki, dendrytów odbierających sygnały oraz aksonu przekazującego sygnały

dalej. Impulsy elektryczne, zwane potencjałami czynnościowymi, przemieszczają się wzdłuż aksonu do synaps, gdzie są przekazywane do kolejnych neuronów [56]. Ten proces komunikacji jest kluczowy dla funkcjonowania układu nerwowego, umożliwiając mózgowi wykonywanie skomplikowanych operacji poznawczych i motorycznych.

W modelach sztucznych sieci neuronowych neurony są reprezentowane przez proste jednostki obliczeniowe, które otrzymują sygnały wejściowe, przetwarzają je i generują sygnały wyjściowe. Podstawowy model matematyczny neuronu sztucznego został zaproponowany przez McCullocha i Pittsa [26]. Wykonuje on operację ważonej sumy swoich wejść, do której dodaje wartość progową, a następnie stosuje funkcję aktywacji, aby wygenerować wyjście.

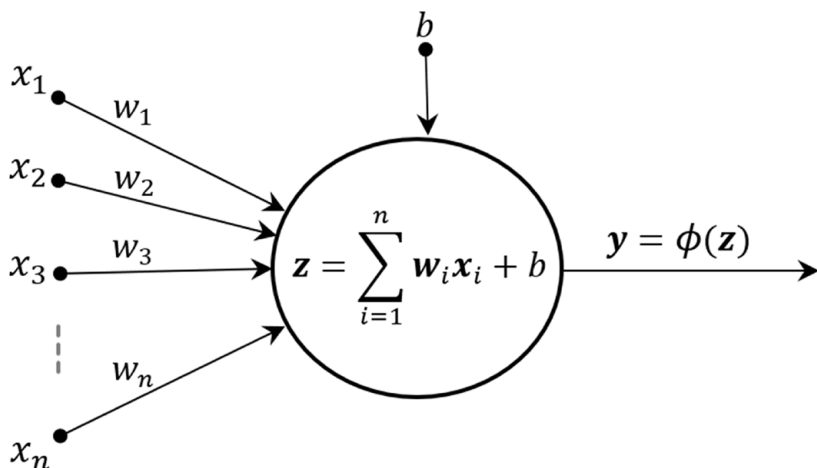
Matematyczny model neuronu można opisać następująco (ryc. 3.2):

1. Sygnały wejściowe (x): Każdy neuron otrzymuje sygnały wejściowe $x_1, x_2, \dots, x_n$ od innych neuronów lub bezpośrednio z otoczenia.
2. Wagi (w): Każdemu sygnałowi wejściowemu przypisana jest waga $w_1, w_2, \dots, w_n$, która określa siłę wpływu danego sygnału na neuron.
3. Sumowanie (Σ): Neuron sumuje swoje ważone wejścia. Sumowanie to można wyrazić wzorem:

$$z = \sum_{i=1}^n w_i * x_i + b \quad (3.1)$$

gdzie z jest sumą ważoną, a b jest wartością progową, pewną stałą (bias). Bias b nie musi być indeksowany tak jak wagi w_i , ponieważ stanowi dodatkową stałą wartość, która przesuwa całą funkcję aktywacji w przestrzeni. W klasycznym modelu neuronu, który przedstawiono na ryc. 3.2, bias jest jednolity dla danego neuronu, niezależnie od liczby jego wejść. Bias pełni funkcję przesunięcia progu aktywacji, umożliwiając neuronowi lepsze dopasowanie się do danych wejściowych poprzez przesunięcie funkcji aktywacji w poziomie, co jest niezależne od wartości sygnałów wejściowych.

4. Funkcja aktywacji (ϕ): Suma ważona z jest następnie przetwarzana przez funkcję aktywacji $\phi(z)$, która wprowadza nieliniowość do modelu. Popularne funkcje aktywacji to m.in. funkcja sigmoidalna, ReLU (ang. *rectified linear unit*) i funkcja tangens hiperboliczny (*tanh*).
5. Wyjście (y): Wynik funkcji aktywacji jest sygnałem wyjściowym neuronu: $y = \phi(z)$.



Ryc. 3.2. Model sztucznego neuronu

Funkcje aktywacji wprowadzają nieliniowość do modelu, co pozwala sieciom neuronowym na rozwiązywanie skomplikowanych problemów. Nieliniowość jest ważna, ponieważ bez niej, niezależnie od liczby warstw, model sieci neuronowej byłby równoważny jednej warstwie liniowej. Oto kilka najczęściej używanych funkcji aktywacji:

1. Funkcja *sigmoidalna*

$$\phi(z) = \frac{1}{1 + e^{-z}} \quad (3.2)$$

Funkcja *sigmoidalna* jest jedną z najstarszych i najbardziej popularnych funkcji aktywacji, szczególnie we wczesnych modelach sieci neuronowych. Przekształca dowolne wejście w wartość w przedziale $(0 \div 1)$. Jest często używana w sieciach neuronowych do klasyfikacji binarnej [57]. Jednak ma tendencję do problemu zanikania gradientu, co utrudnia trenowanie głębokich sieci [58].

2. Funkcja *ReLU*

$$\phi(z) = \max(0, z) \quad (3.3)$$

Funkcja *ReLU* (ang. *rectified linear unit*) jest jedną z najczęściej używanych funkcji aktywacji w nowoczesnych sieciach neuronowych, szczególnie w głębokich sieciach konwolucyjnych. Stosuje się ją głównie w warstwach

ukrytych nowoczesnych sieci neuronowych ze względu na prostotę i efektywność obliczeniową, co prowadzi do szybszej konwergencji podczas trenowania [59]. Jednak *ReLU* może prowadzić do problemu „umierających” neuronów, gdzie neurony mogą przestać się aktywować (dawać wyjście zero) na zawsze.

3. Funkcja tangens hiperboliczny (*tanh*)

$$\phi(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (3.4)$$

Funkcja tangens hiperboliczny jest podobna do funkcji *sigmoidalnej*, ale przekształca wartości wejściowe w przedziale $(-1, 1)$. Funkcja *tanh* jest często używana w sieciach neuronowych, gdzie wartości wyjściowe mogą być zarówno dodatnie, jak i ujemne. Jest centralnie symetryczna wokół zera, co może prowadzić do szybszej konwergencji niż funkcja *sigmoidalna* [58].

4. Funkcja *softmax*

$$\phi(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (3.5)$$

Funkcję *softmax* często używa się w warstwach wyjściowych sieci neuronowych do klasyfikacji wieloklasowej. Funkcja *softmax* przekształca wartości wejściowe w przedziale $(0, 1)$ dla każdej klasy, a suma wszystkich wyjść wynosi 1. Jest używana w klasyfikacji wieloklasowej, ponieważ zapewnia prawdopodobieństwa przypisania do poszczególnych klas [58].

5. Funkcja *Leaky ReLU*

$$\phi(z_i) = \begin{cases} z & \text{jeśli } z > 0 \\ \alpha z & \text{jeśli } z \leq 0 \end{cases} \quad (3.6)$$

gdzie α jest małą stałą (np. 0,01).

Funkcja *Leaky ReLU* jest wariantem funkcji *ReLU*, który stara się rozwiązać problem „umierających” neuronów *ReLU*. *Leaky ReLU* przekształca wartości wejściowe w przedział $(-\infty, \infty)$. Jest szeroko stosowana w nowoczesnych sieciach neuronowych, ponieważ przez małą dodatnią wartość dla $z \leq 0$, neurony mają mniejszą tendencję do „umierania” [60].

Każda z tych funkcji ma unikalne właściwości, które wpływają na efektywność uczenia i działanie sieci, umożliwiając im przetwarzanie danych w sposób analogiczny do biologicznych procesów neuronalnych. Wprowadzenie tych funkcji jest kluczowe dla głębokiego uczenia, umożliwiając modelom neuronowym adaptację i generalizację w różnych zastosowaniach sztucznej inteligencji. Generalizacja oznacza zdolność modelu do skutecznego przewidywania wyników na nowych, niewidzianych wcześniej danych testowych. Jest to kluczowa właściwość modelu, która zależy od odpowiedniego doboru danych uczących, architektury modelu i technik regularyzacji.

3.2. Perceptron wielowarstwowy

W dziedzinie sieci neuronowych istnieje wiele różnych typów i architektur, z których każda ma swoje unikalne cechy i zastosowania. Jednym z najbardziej znanych i szeroko stosowanych typów jest perceptron wielowarstwowy (z ang. *multi layer perceptron*, MLP). Koncepcja ta została zapoczątkowana przez badaczy, takich jak Rumelhart i McClelland w 1986 roku i od tego czasu stanowi podstawę wielu systemów uczenia maszynowego i sztucznej inteligencji [30]. Opis tej architektury można znaleźć w licznych podręcznikach dotyczących sieci neuronowych, takich jak prace Tadeusiewicza (1993) [61] czy Bishopa (1995) [57].

Perceptron wielowarstwowy składa się z trzech głównych typów warstw: warstwy wejściowej, jednej lub więcej warstw ukrytych oraz warstwy wyjściowej. Każda z tych warstw odgrywa kluczową rolę w przetwarzaniu informacji i generowaniu wyników.

Warstwa wejściowa składa się z neuronów, które otrzymują dane wejściowe. Liczba neuronów w tej warstwie jest równa liczbie cech w danych wejściowych. Warstwy ukryte znajdują się pomiędzy warstwą wejściową a wyjściową. Każdy neuron w warstwie ukrytej otrzymuje sygnały od neuronów z poprzedniej warstwy, przetwarza je i przesyła dalej. Neurony te obliczają ważoną sumę swoich wejść, do której dodawana jest wartość progowa (ang. *bias*). Wynik tej sumy jest następnie przetwarzany przez funkcję aktywacji, która wprowadza nieliniowość do modelu. Liczba neuronów i warstw ukrytych jest hiperparametrem, który można dostosowywać w zależności od złożoności problemu. Warstwa wyjściowa składa się z neuronów, które generują finalne wyniki sieci. Liczba neuronów w tej warstwie zależy od charakteru problemu – w przypadku klasyfikacji binarnej będzie to jeden neuron, zaś w klasyfikacji wieloklasowej liczba neuronów odpowiada liczbie klas [62].

Perceptron wielowarstwowy jest to rodzaj sieci jednokierunkowych, w których sygnały przetwarzane są tylko w jednym kierunku – od warstwy wejściowej, przez warstwy ukryte, aż do warstwy wyjściowej. W tych sieciach brak sprzężenia zwrotnego oznacza, że sygnały nie wracają do poprzednich warstw. Każdy neuron w sieci jednokierunkowej przetwarza swoje wejścia za pomocą funkcji aktywacji, co pozwala na modelowanie nieliniowych zależności. Proces uczenia najczęściej opiera się na algorytmie propagacji wstecznej, który minimalizuje błąd pomiędzy przewidywaniami sieci a rzeczywistymi wynikami.

Proces uczenia perceptronu wielowarstwowego przedstawia się następująco:

1. Inicjalizacja wag

Proces uczenia rozpoczyna się od losowej inicjalizacji wag połączeń między neuronami. Ta losowa inicjalizacja jest niezbędna, aby zapewnić różnorodność początkowych ścieżek propagacji sygnałów i uniknąć symetrii w sieci, która mogłaby utrudnić proces uczenia.

2. Propagacja w przód

Dane wejściowe są przesyłane przez sieć neuronową, warstwa po warstwie. Każdy neuron oblicza swoją ważoną sumę wejść, dodaje wartość biasu, stosuje odpowiednią funkcję aktywacji, a następnie przekazuje wynik do neuronów w następnej warstwie. Proces ten powtarza się aż do uzyskania wyjścia w ostatniej warstwie.

3. Obliczanie błędu

W warstwie wyjściowej wyniki sieci są porównywane z rzeczywistymi wartościami docelowymi za pomocą funkcji kosztu (ang. *loss function*, funkcja strat), takich jak m.in. funkcja błędu średniokwadratowego czy entropii krzyżowej. Funkcja kosztu mierzy różnicę między przewidywaniami sieci a rzeczywistymi wartościami, co stanowi podstawę do dalszych modyfikacji wag.

4. Propagacja wsteczna

Na podstawie obliczonego błędu gradienty funkcji kosztu względem wag są obliczane z wykorzystaniem reguły łańcuchowej. Gradienty te wskazują, jak bardzo każda waga wpływa na całkowity błąd. Proces ten umożliwia sieci stopniowe dostosowywanie swoich wag, aby minimalizować błąd.

5. Aktualizacja wag

Wagi są aktualizowane zgodnie z obliczonymi gradientami. Najczęściej stosowane algorytmy optymalizacji to *stochastic gradient descent* (SGD) oraz Adam. Aktualizacje te przesuwają wagi w kierunku zmniejszenia funkcji kosztu, co prowadzi do stopniowego ulepszania działania sieci. Aktualizacja wag odbywa się zgodnie z regułą:

$$waga = waga - \text{współczynnik uczenia} \times \text{gradient} \quad (3.7)$$

Perceptron wielowarstwowy jest fundamentalną architekturą w obszarze sieci neuronowych, umożliwiającą rozwiązywanie skomplikowanych problemów dzięki swojej zdolności do modelowania nieliniowych zależności. Wprowadzenie warstw ukrytych i odpowiednich funkcji aktywacji pozwala na efektywne przetwarzanie i analizowanie danych w różnych zastosowaniach.

Dzięki swojej zdolności do modelowania nieliniowych zależności między danymi sieci MLP znalazły szerokie zastosowanie w wielu różnorodnych dziedzinach, które wymagają przetwarzania danych o złożonej strukturze. Jednym z istotnych obszarów zastosowań MLP jest analiza danych tabelarycznych oraz zadania klasyfikacji i regresji. W tych kontekstach MLP pozwala na modelowanie skomplikowanych relacji między cechami wejściowymi a wynikami, co czyni je niezwykle użytecznym narzędziem w biotechnologii, gdzie są wykorzystywane do klasyfikacji wzorców oraz regresji funkcji, czerpiąc inspirację z biologicznych sieci neuronowych. Takie zastosowania podkreślają fundamentalną rolę MLP w rozwoju sztucznej inteligencji oraz ich zdolność adaptacji do różnych dziedzin, takich jak inżynieria, finanse czy medycyna [63]. Kolejnym istotnym obszarem, w którym sieci MLP są szeroko stosowane, jest przetwarzanie danych ciągłych, szczególnie w zadaniach, takich jak prognozowanie finansowe oraz analiza szeregów czasowych. Zdolność MLP do aproksymacji nieliniowych funkcji umożliwia efektywne modelowanie zależności między zmiennymi w danych czasowych, co pozwala na skuteczne przewidywanie przyszłych wartości na podstawie historycznych obserwacji. Przykładem tego zastosowania jest prognozowanie cen akcji, gdzie wykorzystanie MLP przyniosło znaczące korzyści w zakresie dokładności przewidywań w porównaniu z tradycyjnymi metodami [64].

W obszarze planowania produkcji MLP okazały się również niezwykle skuteczne. Wykorzystując dane historyczne dotyczące produkcji i sprzedaży, sieci MLP są zdolne do prognozowania przyszłego zapotrzebowania na surowce, co sprzyja bardziej efektywnemu planowaniu produkcji oraz optymalizacji zasobów magazynowych [65]. Zastosowanie MLP w prognozowaniu wskaźnika produkcji maszyn tnących w kamieniołomach pozwoliło na znaczne zwiększenie dokładności tych przewidywań, co z kolei umożliwiło optymalizację procesów cięcia kamienia. Efektem było bardziej precyzyjne planowanie i zwiększenie wydajności pracy maszyn, co bezpośrednio przełożyło się na poprawę efektywności całego procesu produkcyjnego [66]. MLP zostały ponadto zastosowane do przewidywania czasu cyklu na liniach produkcyjnych, co umożliwiło lepsze zarządzanie czasem i zasobami. Dokładne

prognozy czasu cyklu są ważne dla utrzymania płynności produkcji oraz minimalizacji przestojów, co z kolei przekłada się na wyższe zyski oraz lepszą kontrolę nad procesami produkcyjnymi [67]. W odniesieniu do nowoczesnych rozwiązań produkcyjnych, związanych z Przemysłem 4.0, sieci MLP odegrały istotną rolę w opracowaniu hybrydowego systemu wspomagania decyzji, który integruje MLP z systemami eksperckimi w kontroli procesu produkcji gipsu. Zastosowanie tego systemu przyczyniło się do precyzyjnej identyfikacji krytycznych punktów kontrolnych (CCP) w procesie produkcji, co jest niezbędne dla utrzymania wysokiej jakości produktów oraz zmniejszenia liczby defektów, szczególnie w zautomatyzowanych środowiskach produkcyjnych [68].

Perceptrony wielowarstwowe mogą znaleźć szerokie zastosowanie w inżynierii produkcji dzięki swojej zdolności do modelowania złożonych i nieliniowych zależności w danych. Mogą być wykorzystywane do optymalizacji procesów produkcyjnych, gdzie analizują sygnały z czujników i dane operacyjne, aby minimalizować przestoje i zwiększać ich efektywność. Dodatkowo MLP mogą być stosowane do prognozowania popytu na produkty na podstawie analizy danych historycznych, co jest kluczowe dla efektywnego planowania produkcji i zarządzania zapasami. W obszarze zarządzania utrzymaniem ruchu sieci neuronowe mogą przewidywać awarie maszyn, umożliwiając proaktywne działania konserwacyjne i minimalizując koszty napraw. Dzięki swojej wszechstronności i zdolności do przetwarzania złożonych danych MLP stają się nieocenionym narzędziem w nowoczesnych systemach produkcyjnych.

3.3. Rekurencyjne sieci neuronowe LSTM

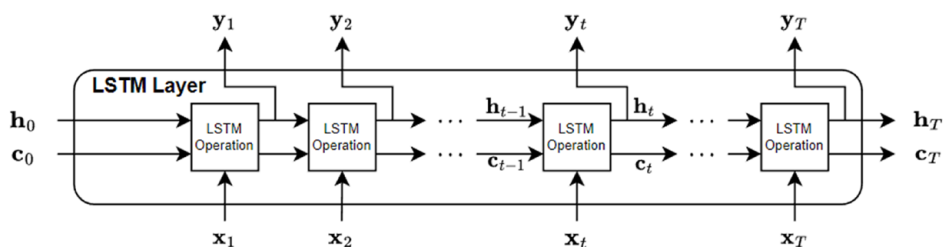
Sieci *long short-term memory* (LSTM) stanowią jeden z najbardziej zaawansowanych i wszechstronnych typów rekurencyjnych sieci neuronowych, zaprojektowanych w celu rozwiązania problemów związanych z długoterminowymi zależnościami w danych sekwencyjnych. Tradycyjne rekurencyjne sieci neuronowe (RNN) napotykają na istotne trudności podczas próby uczenia się informacji z dłuższych sekwencji danych, co jest spowodowane zjawiskiem znanym jako problem zanikającego gradientu [35,69]. W miarę propagacji błędu wstecz przez czas w procesie uczenia gradienty mogą się zmniejszać do niemal zerowych wartości lub, w innych przypadkach, eksplodować, co znacząco utrudnia proces optymalizacji modelu. To zjawisko prowadzi do sytuacji, w której sieci RNN mają trudności z utrzymaniem i wykorzystywaniem informacji, które pojawiły się w dalszej przeszłości sekwencji.

W odpowiedzi na te wyzwania Sepp Hochreiter i Jürgen Schmidhuber zaproponowali w 1997 roku architekturę LSTM, która została zaprojektowana w celu przezwyciężenia tych ograniczeń. Istotnym elementem LSTM jest jego zdolność do przechowywania informacji przez dłuższy czas bez degradacji sygnału, co czyni go idealnym narzędziem do analizy danych sekwencyjnych, takich jak teksty, serie czasowe czy sygnały. Architektura LSTM składa się z jednostek pamięci (ang. *memory cells*), które są odpowiedzialne za przechowywanie informacji, oraz trzech rodzajów bramek: zapominania (ang. *forget gate*), wejściowej (ang. *input gate*) i wyjściowej (ang. *output gate*) [35,49].

Każda z tych bramek jest siecią neuronową, która uczy się, jak manipulować przepływem informacji, aby optymalnie przechowywać i wykorzystywać dane sekwencyjne [70]. Bramki w sieciach LSTM [49,71,72]:

1. Bramka zapominania: Bramka decyduje, które informacje z poprzedniego stanu komórki powinny zostać zapomniane. Działa poprzez przyjęcie bieżącego stanu wejściowego oraz poprzedniego stanu ukrytego i przetworzenie ich przez funkcję sigmoidalną, która zwraca wartości między 0 a 1. Te wartości określają, które informacje są zachowywane (wartość bliska 1), a które zapomniane (wartość bliska 0).
2. Bramka wejściowa: Decyduje, które nowe informacje zostaną dodane do stanu komórki, steruje poziomem aktualizacji stanu komórki. Proces ten obejmuje dwa kroki. Pierwszy to zastosowanie funkcji sigmoidalnej do bieżącego wejścia i poprzedniego stanu ukrytego, co decyduje, które wartości będą aktualizowane. Drugi krok to stworzenie wektora nowych wartości kandydujących przez funkcję *tanh*, które są następnie dodawane do stanu komórki.
3. Bramka wyjściowa: Określa, które informacje z aktualnego stanu komórki zostaną użyte do obliczenia wyjścia. Stan komórki jest przekształcany przez funkcję *tanh* i mnożony przez wynik bramki wyjściowej, co daje końcowe wyjście tej komórki LSTM.

Architektura warstwy LSTM przedstawiona została na ryc. 3.3. Diagram ten ilustruje przepływ danych przez warstwę LSTM z wejściem x i wyjściem y , przy T krokach czasowych. W diagramie h_T oznacza wyjście (znane również jako stan ukryty), a c_T oznacza stan komórki w kroku czasowym t . Jeśli warstwa LSTM generuje pełną sekwencję wyjść, to wynikiem są wartości $y_1, \dots, y_T$, które odpowiadają $h_1, \dots, h_T$. Natomiast jeśli warstwa LSTM zwraca wyłącznie wynik z ostatniego kroku czasowego, to wynikiem jest y_T , które odpowiada h_T . Liczba kanałów w wyjściu odpowiada liczbie jednostek ukrytych w warstwie LSTM [73].



Ryc. 3.3. Architektura warstwy LSTM [73]

Pierwsza operacja w sieci LSTM wykorzystuje początkowy stan RNN oraz pierwszy krok czasowy sekwencji, aby obliczyć pierwsze wyjście oraz zaktualizowany stan komórki. W kroku czasowym t operacja wykorzystuje bieżący stan RNN (c_{t-1}, h_{t-1}) oraz kolejny krok czasowy sekwencji, aby obliczyć aktualne wyjście oraz zaktualizowany stan komórki c_t . Stan warstwy LSTM składa się z ukrytego stanu (znanego również jako stan wyjściowy) oraz stanu komórki. Ukryty stan w kroku czasowym t zawiera wyjście warstwy LSTM dla tego kroku czasowego. Stan komórki przechowuje informacje wyuczone z poprzednich kroków czasowych. W każdym kroku czasowym warstwa dodaje lub usuwa informacje ze stanu komórki. Te aktualizacje są kontrolowane przez odpowiednie bramki [73].

Dzięki swojej zdolności do utrzymywania długoterminowych zależności sieci LSTM znalazły szerokie zastosowanie w wielu różnorodnych dziedzinach, które wymagają analizy danych sekwencyjnych. Jednym z obszarów, w którym LSTM stosuje się na szeroką skalę, jest przetwarzanie języka naturalnego (NLP). W tej dziedzinie LSTM umożliwia modelowanie złożonych zależności między słowami i frazami w tekstach, co jest istotne dla zadań, takich jak tłumaczenie maszynowe, generowanie tekstu czy analiza sentymentu. Przykładem może być praca zespołu Sutskevera (2014), w której wykorzystano LSTM do tłumaczenia sekwencji tekstu z jednego języka na inny, co przyniosło znaczące ulepszenia w jakości tłumaczeń w porównaniu z wcześniejszymi metodami [74].

Rozpoznawanie mowy to kolejna dziedzina, w której LSTM znalazły szerokie zastosowanie. Dzięki zdolności do uchwycenia sekwencyjnej natury mowy LSTM są wykorzystywane w systemach takich jak Siri, Google Assistant czy Cortana, gdzie modele te uczą się rozpoznawać wzorce w danych mowy i przekształcać je na tekst. Graves i współpracownicy (2013) pokazali, jak LSTM może być zastosowane do rozpoznawania mowy z wysoką dokładnością, dzięki zdolności do uwzględnienia długoterminowych zależności w sygnałach mowy [75]. W dziedzinie prognozowania popytu LSTM są używane do przewidywania

zapotrzebowania na produkty czy usługi na podstawie danych historycznych, co jest istotne dla optymalizacji łańcuchów dostaw i zarządzania zasobami. Modele LSTM pozwalają na uwzględnienie sezonowych wzorców i trendów, co prowadzi do dokładniejszych prognoz.

LSTM znalazły również szerokie zastosowanie w procesach wytwarzania i produkcji, znacznie poprawiając konserwację predykcyjną, wykrywanie anomalii, monitorowanie procesów i harmonogramowanie produkcji. W konserwacji predykcyjnej sieci LSTM są wykorzystywane do przewidywania i diagnozowania awarii komponentów maszyny poprzez analizę danych czujników, zmniejszając w ten sposób czas przestojów maszyny, obniżając koszty i wydłużając cykle jej życia. Na przykład hybrydowy model CNN-LSTM osiągnął wysoką dokładność w przewidywaniu awarii jednostek produkcyjnych powietrza (APU) [76]. Integracja LSTM z CNN była również skuteczna w dynamicznym łączeniu taktycznego planowania produkcji i konserwacji predykcyjnej, zmniejszając całkowite koszty produkcji i konserwacji poprzez dokładne przewidywanie pozostałej żywotności systemów [77].

Podsumowując, sieci LSTM to zaawansowane narzędzie w dziedzinie uczenia maszynowego, które może odgrywać kluczową rolę w inżynierii produkcji. Dzięki swojej zdolności do modelowania długoterminowych zależności w danych sekwencyjnych LSTM znalazły szerokie zastosowanie w konserwacji predykcyjnej, wykrywaniu anomalii, monitorowaniu procesów oraz optymalizacji harmonogramów produkcji. Ich innowacyjna architektura, obejmująca mechanizmy bramek, pozwala na efektywne przetwarzanie informacji, co przekłada się na dokładniejsze prognozy i lepsze zarządzanie zasobami. Wprowadzenie LSTM może znacząco poprawić efektywność i niezawodność procesów produkcyjnych, czyniąc je nieocenionym narzędziem w nowoczesnej inżynierii produkcji.

3.4. Konwolucyjne sieci neuronowe

Konwolucyjne sieci neuronowe (z ang. *convolutional neural networks*, CNN) to wyspecjalizowana klasa głębokich sieci neuronowych (DNN), które zrewolucjonizowały pole widzenia komputerowego i przetwarzania obrazu, wykorzystując operację matematyczną zwaną konwolucją zamiast ogólnego mnożenia macierzy w co najmniej jednej z ich warstw [78,79]. CNN są zaprojektowane do automatycznego i adaptacyjnego uczenia się przestrzennych hierarchii elementów z obrazów wejściowych, co czyni je wysoce skutecznymi w zadaniach,

takich jak klasyfikacja obrazów, wykrywanie obiektów i segmentacja [80,81]. CNN znajdują również zastosowanie w analizie sygnałów czasowych, takich jak rozpoznawanie mowy czy analiza sekwencji wideo, dzięki swojej zdolności do wykrywania wzorców zarówno w danych przestrzennych, jak i czasowych [82].

CNN są inspirowane biologicznym systemem wzrokowym, co polega na sposobie przetwarzania obrazów przez neurony w warstwach wizualnych kory mózgowej, które reagują na specyficzne cechy wizualne, takie jak krawędzie czy tekstury. Podobnie w sieciach konwolucyjnych warstwy konwolucyjne automatycznie uczą się wykrywać te cechy bez konieczności ręcznego wyodrębniania cech z danych wejściowych [83].

Podstawowym elementem CNN jest warstwa konwolucyjna, która przekształca dane wejściowe za pomocą filtrów (jąder) przesuwających się po całym obrazie, generując mapy cech (ang. *feature maps*). Filtry te automatycznie uczą się wykrywać różne cechy, takie jak krawędzie, tekstury czy wzory w procesie trenowania sieci [84].

Architektura typowej sieci konwolucyjnej składa się z kilku warstw konwolucyjnych i poolingowych, zakończonych jedną warstwą lub kilkoma warstwami w pełni połączonymi (ang. *fully connected layer*). Klasyczny przykład takiej architektury to AlexNet. Typowa struktura CNN może wyglądać następująco [31,58,59,84]:

1. Warstwa wejściowa

Warstwa wejściowa jest pierwszą warstwą sieci, która przyjmuje surowe dane wejściowe. Dla obrazów może to być macierz pikseli o wymiarach odpowiadających wysokości, szerokości i liczbie kanałów (np. RGB). Na tym etapie dane są zwykle normalizowane, aby wartości pikseli były w odpowiednim zakresie (np. $[0, 1]$ lub $[-1, 1]$), co przyspiesza proces uczenia.

2. Pierwsza warstwa konwolucyjna

Pierwsza warstwa konwolucyjna jest kluczowym elementem CNN. Stosuje ona kilka filtrów (jąder konwolucyjnych) do danych wejściowych w celu wykrywania niskopoziomowych cech, takich jak krawędzie, tekstury, linie czy kąty. Każdy filtr przesuwany jest po obrazie i oblicza splot, co prowadzi do utworzenia mapy cech. Każda mapa cech jest wynikiem działania jednego filtra. Filtry są zwykle mniejsze niż rozmiar obrazu (np. 3×3 , 5×5), a operacja przesuwania się filtra po obrazie nazywana jest krokowaniem.

$$S\left(i, j\right) = \left(I * K\right)\left(i, j\right) = \sum_m \sum_n I\left(i + m, j + n\right) * K\left(m, n\right) \quad (3.8)$$

gdzie:

$S(i,j)$ – wartość w mapie cech w punkcie (i,j) ;

I – macierz obrazu;

K – macierz filtra;

m, n – indeksy przesunięcia filtra.

3. Warstwa poolingowa

Warstwa poolingowa (ang. *pooling layer*) redukuje wymiary map cech, co zmniejsza liczbę parametrów i obliczeń w sieci oraz pomaga w zapobieganiu przeuczeniu (ang. *overfitting*). Przeuczenie odnosi się do sytuacji, w której model zbyt dokładnie odwzorowuje dane uczące, co prowadzi do pogorszenia wyników na danych testowych. Jest to szczególnie problematyczne w przypadku małych zbiorów danych i złożonych modeli o dużej liczbie parametrów. Najczęściej stosowane techniki poolingowe to *max pooling* i *average pooling*. *Max pooling* wybiera maksymalną wartość w każdym podobszarze (np. 2×2) mapy cech. To pozwala na zachowanie najistotniejszych cech. *Average pooling* oblicza średnią wartość w każdym podobszarze, co prowadzi do wygładzenia wartości. Operacja poolingowa nie zawiera parametrów do uczenia i jest stosowana głównie w celu zmniejszenia rozmiaru reprezentacji oraz wprowadzenia pewnej translacyjnej niezmienniczości.

4. Kolejne warstwy konwolucyjne i poolingowe

Kolejne warstwy konwolucyjne i poolingowe są stosowane do wykrywania coraz bardziej złożonych i abstrakcyjnych cech na wyższych poziomach. Podczas gdy pierwsze warstwy mogą wykrywać proste krawędzie i tekstury, kolejne warstwy są w stanie rozpoznawać bardziej złożone struktury, takie jak kształty, wzory, jak również całe obiekty. Zazwyczaj architektura CNN składa się z kilku takich bloków konwolucyjno-poolingowych, które stopniowo redukują rozmiar przestrzenny danych i zwiększają liczbę kanałów cech.

5. Warstwa w pełni połączona

Warstwa w pełni połączona (ang. *fully connected layer*) jest końcową częścią sieci, która łączy wszystkie neurony z poprzedniej warstwy z każdym neuronem w tej warstwie. Warstwy w pełni połączone działają podobnie jak w tradycyjnych sieciach neuronowych i są odpowiedzialne za integrację cech wykrytych przez wcześniejsze warstwy konwolucyjne oraz dokonanie ostatecznej klasyfikacji lub przewidywania. Każdy neuron w warstwie w pełni połączonej oblicza sumę ważoną swoich wejść i przekazuje ją przez funkcję aktywacji. Typową funkcją aktywacji używaną w tych warstwach jest *ReLU* lub, w przypadku warstwy wyjściowej, *softmax* dla klasyfikacji wieloklasowej.

Jak już wspomniano wcześniej, klasyczny przykład takiej architektury to AlexNet (ryc. 3.4). Składa się z kilku warstw konwolucyjnych, przeplatanych warstwami poolingowymi oraz w pełni połączonymi warstwami [84]. Cała sieć została zaprojektowana tak, aby przetwarzać obrazy o rozmiarze 227×227 pikseli, z trzema kanałami kolorów (RGB). Pierwsza warstwa konwolucyjna zawiera 96 filtrów o rozmiarze 11×11 , z krokiem 4 pikseli, co pozwala na wyodrębnienie podstawowych cech obrazu. Po tej warstwie następuje funkcja aktywacji *ReLU*, która przyspiesza proces uczenia i zmniejsza problem zanikania gradientu. Następnie wprowadzana jest warstwa *max pooling*, która zmniejsza rozmiar map cech poprzez wybieranie maksymalnej wartości z okna o rozmiarze 3×3 , przesuwanego o 2 piksele.

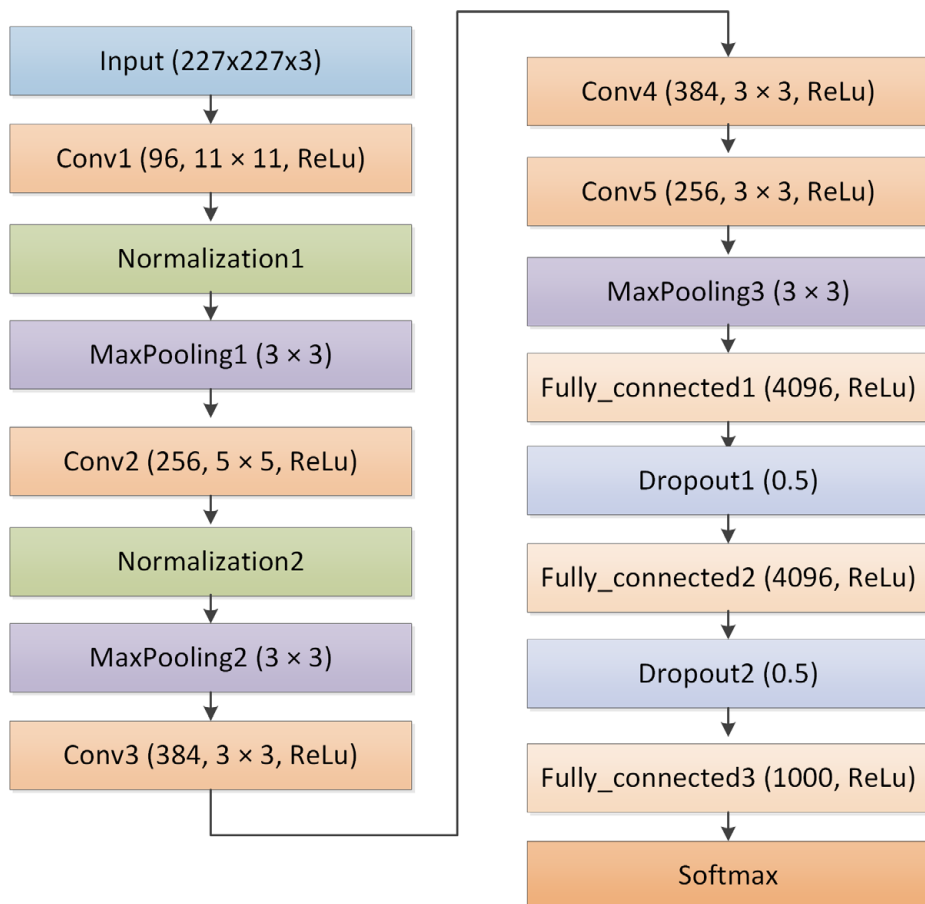
Druga warstwa konwolucyjna składa się z 256 filtrów o rozmiarze 5×5 i kroku 1, co pozwala na bardziej złożone wyodrębnienie cech z obrazów. Podobnie jak wcześniej, warstwa ta jest połączona z funkcją aktywacji *ReLU* i warstwą *max pooling*, która również zmniejsza rozmiar map cech. Trzecia, czwarta i piąta warstwa konwolucyjna mają odpowiednio 384, 384 i 256 filtrów o rozmiarze 3×3 . Każda z tych warstw również korzysta z funkcji aktywacji *ReLU*. Piąta warstwa konwolucyjna kończy się kolejną warstwą *max pooling*, która ponownie zmniejsza rozmiar cech map.

Po warstwach konwolucyjnych następują trzy w pełni połączone warstwy. Pierwsza i druga z nich mają po 4096 neuronów, a ostatnia, trzecia warstwa, zawiera 1000 neuronów, odpowiadających liczbie klas w zbiorze danych ImageNet. Warstwy te również korzystają z funkcji aktywacji *ReLU*, a na końcu trzeciej warstwy stosowana jest funkcja *softmax*, która przekształca wartości wyjściowe w prawdopodobieństwa przypisania do poszczególnych klas.

Aby zapobiec przetrenowaniu sieci, w AlexNet zastosowano technikę dropout w dwóch pierwszych w pełni połączonych warstwach, gdzie losowo wyłączane jest 50% neuronów podczas procesu uczenia. Dodatkowo, w celu zwiększenia ogólności modelu, stosowane są techniki augmentacji danych, takie jak losowe wycinanie fragmentów obrazów oraz odbicie lustrzane.

CNN mają kilka zalet, w tym zdolność do automatycznego wykrywania istotnych cech z surowych danych, co eliminuje potrzebę ręcznego tworzenia cech. Są również skalowalne, co pozwala na przetwarzanie obrazów o różnych rozdzielczościach i skali. Charakteryzują się wysoką wydajnością, osiągając bardzo dobre wyniki w zadaniach związanych z rozpoznawaniem obrazów i wizją komputerową. Jednakże CNN mają także swoje ograniczenia. Trenowanie tych sieci wymaga dużych zasobów obliczeniowych, zwłaszcza w przypadku większych zbiorów danych. Ponadto, aby osiągnąć dobrą generalizację, potrzebują

one dużej ilości danych uczących. Sieci o dużej liczbie parametrów mogą być podatne na przeuczenie, co jest szczególnie problematyczne przy niewystarczającej ilości danych uczących [85–87].



Ryc. 3.4. Architektura sieci AlexNet

CNN są stosowane w różnorodnych dziedzinach ze względu na ich zdolność do przetwarzania i analizy obrazów oraz sygnałów dźwiękowych. W rolnictwie CNN zostały z powodzeniem wykorzystane do identyfikacji i klasyfikacji chorób roślin, jak w przypadku badania dotyczącego chorób roślin bawełny, gdzie osiągnięto dokładność na poziomie 98,53% [88]. W innym badaniu CNN były używane do diagnostyki awarii mechanicznych na podstawie sygnałów dźwiękowych, co pozwoliło na osiągnięcie wysokiej dokładności w rozpoznawaniu usterek [89]. CNN znalazły również zastosowanie w ocenie jakości płoń, gdzie dzięki analizie obrazów roślin możliwe było precyzyjne określenie

jakości produktów rolnych [90]. Ponadto CNN były stosowane w produkcji opisów obrazów, łącząc techniki przetwarzania języka naturalnego i wizji komputerowej, co zwiększyło dokładność generowanych opisów w miarę wzrostu rozmiaru zbioru danych [91].

CNN odgrywają w przemyśle istotną rolę w automatyzacji i poprawie procesów produkcyjnych. Przykładem może być wykorzystanie CNN w połączeniu z wizją maszynową do wykrywania wad w kompresorach mechanicznych, co znacznie poprawiło wydajność i dokładność w rozpoznawaniu defektów [92]. CNN znalazły również zastosowanie w monitorowaniu jakości w produkcji urządzeń AGD, gdzie umożliwiły wykrywanie nietypowych dźwięków w pralkach z dokładnością sięgającą 99,3%, co znacznie usprawniło kontrolę jakości [93]. Ponadto CNN w połączeniu z głębokimi autoenkoderami były stosowane do przewidywania pozostałego czasu eksploatacji łożysk tocznych, co jest ważne dla efektywnego planowania konserwacji w systemach produkcji mechanicznej [94].

Zastosowanie CNN w przemyśle wykazuje ogromny potencjał, szczególnie w kontekście automatyzacji, poprawy efektywności procesów produkcyjnych oraz zwiększenia jakości i niezawodności produktów. Ich wszechstronność, pozwalająca na precyzyjną analizę obrazów i sygnałów, umożliwia ich zastosowanie w diagnostyce, monitorowaniu jakości, przewidywaniu właściwości materiałów oraz ocenie pozostałego czasu eksploatacji urządzeń. Perspektywy dalszego rozwoju i integracji CNN z innymi nowoczesnymi technologiami, takimi jak IoT i big data, sugerują, że CNN mogą stać się narzędziem w przyszłych innowacjach w przemyśle produkcyjnym, znacząco przyczyniając się do wzrostu konkurencyjności i efektywności w sektorze inżynierii produkcji.

3.5. Transfer learning

Transfer learning, jako zaawansowana technika w dziedzinie uczenia maszynowego, odgrywa ważną rolę w poprawie efektywności modeli głębokiego uczenia poprzez wykorzystanie wiedzy z jednego zadania lub usprawnieniu procesu uczenia w innym, powiązanym zadaniu. W tradycyjnym podejściu do uczenia maszynowego każdy model jest uczony od podstaw na nowym zbiorze danych, co wymaga dużej ilości danych i mocy obliczeniowej. Transfer learning zmienia ten paradygmat, ponieważ wykorzystuje wstępnie wytrenowane modele, aby przyspieszyć proces trenowania i poprawić uogólnienie modeli w nowych zadaniach, co skraca czas uczenia i poprawia generalizację modelu

na nowych danych. Technika ta jest szczególnie korzystna w scenariuszach, w których uzyskanych danych jest mało lub ich zdobycie jest kosztowne [95].

Transfer learning stał się szczególnie popularny w zastosowaniach związanych z przetwarzaniem obrazów, gdzie modele, takie jak ResNet, Inception czy DenseNet, wytrenowane na dużych zbiorach danych, takich jak ImageNet, mogą być łatwo dostosowane do specyficznych zadań, takich jak klasyfikacja obrazów medycznych, rozpoznawanie obiektów w obrazach przemysłowych czy analiza danych satelitarnych.

Podstawową zasadą działania transfer learningu jest możliwość adaptacji wytrenowanego modelu do nowego zadania poprzez modyfikację jego ostatnich warstw i ponowne trenowanie, co pozwala na uwzględnienie specyfiki nowych danych. W praktyce proces ten polega na wykorzystaniu modelu uprzednio wytrenowanego na dużym zbiorze danych ogólnych (np. ImageNet, który zawiera miliony obrazów z tysięcy różnych kategorii) i dostosowaniu go do specyficznych potrzeb nowego zadania. Przykładowo model ResNet, który osiągnął wysoką dokładność w zadaniach klasyfikacji obrazów, może być użyty jako baza dla klasyfikatora w dziedzinie medycznej, gdzie dostępność danych jest ograniczona [96]. Dzięki temu możliwe staje się znaczne skrócenie czasu trenowania modelu oraz zwiększenie dokładności w porównaniu z modelami trenowanymi od podstaw na niewielkich zbiorach danych.

W kontekście przemysłowym transfer learning znajduje zastosowanie w wielu obszarach, takich jak optymalizacja przepływów metabolicznych w zmodyfikowanych mikroorganizmach, gdzie pozwala na przewidywanie i optymalizację wyników w oparciu o uprzednio zebrane dane z podobnych systemów biologicznych [97]. Innym przykładem jest wykorzystanie transfer learningu w detekcji wad w obrazach przemysłowych, gdzie wiedza z obszaru analizy obrazów może być przeniesiona do nowych typów obrazów, co zwiększa efektywność systemów kontroli jakości [98]. Transfer learning wykazuje również potencjał w medycynie, gdzie ograniczona ilość danych do trenowania jest często wyzwaniem. Technika ta umożliwia wykorzystanie modeli wytrenowanych na dużych zbiorach danych medycznych do detekcji i klasyfikacji chorób na podstawie obrazów diagnostycznych, co poprawia jakość diagnostyki przy mniejszych zasobach danych [99].

Taka elastyczność transfer learningu w przetwarzaniu obrazów wynika z głębokiej reprezentacji cech, których modele uczą się w procesie trenowania na ogólnych zbiorach danych. Te cechy, takie jak krawędzie, tekstury czy kształty, są na tyle uniwersalne, że mogą być skutecznie przenoszone do nowych zadań, minimalizując potrzebę gromadzenia dużych ilości danych specyficznych dla nowego zadania. Dzięki temu transfer learning zyskał szeroką

popularność w rozwiązywaniu problemów, gdzie dostęp do dużych, specyficznych zbiorów danych jest ograniczony, ale istnieje potrzeba wysokiej dokładności i efektywności modeli analizy obrazów.

Wykorzystanie transfer learningu w różnych dziedzinach przemysłowych i badawczych podkreśla jego wszechstronność i potencjał do rozwiązywania złożonych problemów, z którymi tradycyjne metody uczenia maszynowego mogą sobie nie poradzić. Dzięki możliwości szybkiej adaptacji do nowych zadań i zwiększenia dokładności modeli transfer learning staje się nieodzownym narzędziem w nowoczesnej analityce danych.

3.6. Dobór odpowiedniej architektury sieci neuronowej

Dobór odpowiedniej architektury sieci neuronowej jest kluczowym etapem w procesie budowy skutecznych modeli uczenia maszynowego. Wybór architektury musi być uzależniony od charakterystyki danych, rodzaju problemu, a także dostępnych zasobów obliczeniowych, co warunkuje ostateczną efektywność i wydajność modelu (ryc. 3.5).



Ryc. 3.5. Czynniki wpływające na dobór odpowiedniej architektury sieci neuronowej

Dane wejściowe różnią się znacznie pod względem swojej struktury i charakterystyki, co determinuje wybór specyficznych typów sieci neuronowych. Dla danych tabelarycznych, gdzie każde wejście reprezentowane jest przez

wektor cech, najlepszym wyborem są sieci jednokierunkowe (ang. *feedforward neural networks*, FFNN), w których informacje przepływają tylko w jednym kierunku – od wejścia do wyjścia. Jednym z najpopularniejszych rodzajów FFNN jest perceptron wielowarstwowy. MLP, jak już wspomniano w poprzednich podrozdziałach, składa się z jednej warstwy wejściowej, jednej warstwy lub kilku warstw ukrytych oraz jednej warstwy wyjściowej, gdzie każda z warstw jest w pełni połączona z kolejną. MLP stosuje się w wielu zastosowaniach, takich jak klasyfikacja i regresja, ponieważ potrafi modelować złożone nieliniowe relacje między danymi wejściowymi a wyjściowymi. Te proste sieci neuronowe, złożone z warstw w pełni połączonych, pozwalają na efektywne przetwarzanie danych strukturalnych.

Alternatywnie, w przypadku danych tabelarycznych, warto również rozważyć zastosowanie algorytmów, takich jak *gradient boosting machines* (GBM), które mogą oferować lepszą wydajność niż tradycyjne sieci neuronowe. GBM to technika uczenia zespołowego, która łączy wiele słabych modeli (zazwyczaj drzew decyzyjnych) w jeden silny model. Działa on poprzez iteracyjne dodawanie nowych modeli do zespołu, gdzie każdy nowy model koryguje błędy popełnione przez poprzednie. Jest to bardzo skuteczna metoda dla danych tabelarycznych, szczególnie w przypadku problemów regresji i klasyfikacji. W literaturze GBM jest uznawany za jeden z najbardziej wydajnych algorytmów uczenia maszynowego [100].

Dla danych sekwencyjnych, takich jak serie czasowe czy tekst, bardziej odpowiednie są rekurencyjne sieci neuronowe (RNN). W przeciwieństwie do sieci jednokierunkowych RNN mają wewnętrzne stany (pamięć), które pozwalają im na modelowanie zależności czasowych. Klasyczne RNN mają jednak problemy z długotrwałymi zależnościami, co prowadzi do problemu zanikania gradientu. W literaturze RNN są szeroko stosowane w zadaniach przetwarzania języka naturalnego i przewidywania szeregów czasowych [101]. Zaawansowane warianty RNN to LSTM (ang. *long short-term memory*) i GRU (ang. *gated recurrent units*). LSTM został zaprojektowany w celu rozwiązania problemu zanikania gradientu. Wprowadza specjalne jednostki pamięci, które mogą długotrwale przechowywać informacje, co pozwala na skuteczniejsze modelowanie długotrwałych zależności w danych sekwencyjnych. LSTM znalazły szerokie zastosowanie w takich zadaniach, jak tłumaczenie maszynowe, rozpoznawanie mowy i generowanie tekstu. Nowoczesne badania pokazują, że modele LSTM w połączeniu z optymalizatorem Adam są bardziej dokładne w przewidywaniu zużycia energii w inteligentnych sieciach energetycznych, co potwierdzają niższe wartości błędów średniokwadratowych

i średnich błędów absolutnych [35,102]. GRU jest prostszy niż LSTM, ale równie skuteczny, wprowadza mechanizmy bramek, które regulują przepływ informacji wewnątrz jednostki rekurencyjnej, co umożliwia efektywne uczenie się zależności czasowych. GRU są często wybierane ze względu na ich prostszą strukturę i porównywalną wydajność do LSTM [103]. Badania pokazują, że GRU osiąga najlepsze wyniki w prognozowaniu poziomów glukozy we krwi w porównaniu z modelami RNN i LSTM [104].

W przypadku danych obrazowych najbardziej efektywnymi modelami są konwolucyjne sieci neuronowe. Architektury CNN, które szczegółowo opisano w podrozdziale 3.4, dzięki zastosowaniu warstw konwolucyjnych, są zdolne do wydobywania złożonych cech z obrazów, co jest istotne w zadaniach, takich jak klasyfikacja obrazów, detekcja obiektów czy segmentacja. Dodatkowo do generowania nowych obrazów i poprawy jakości istniejących wykorzystuje się *generative adversarial networks* (GAN). To rodzaj sieci neuronowych stosowanych do generowania nowych danych, takich jak obrazy, które są podobne do danych uczących. GAN składają się z dwóch sieci: generatora, który tworzy nowe dane, oraz dyskriminatora, który ocenia, czy dane są prawdziwe czy wygenerowane. Uczenie GAN polega na rywalizacji tych dwóch sieci, co prowadzi do generowania coraz bardziej realistycznych danych. GAN znalazły zastosowanie w zadaniach, takich jak tworzenie obrazów i poprawa jakości obrazów [105].

Rodzaj problemu, który model ma rozwiązać, ma również wpływ na wybór architektury sieci neuronowej. W problemach klasyfikacyjnych warstwa wyjściowa modelu często zawiera funkcję aktywacji *softmax*, która przekształca wyjścia sieci w prawdopodobieństwa klas. Dla prostych problemów klasyfikacyjnych można stosować FFNN, podczas gdy dla bardziej złożonych problemów z obrazami lub tekstem lepszym wyborem będą odpowiednio CNN lub RNN. W obszarze inżynierii mechanicznej sieci neuronowe są szeroko stosowane do różnych zadań klasyfikacyjnych. Przykładowo CNN były wykorzystywane do klasyfikacji sygnałów wibracji w maszynach wirujących, co pozwala na proaktywne monitorowanie stanu i diagnostykę awarii, a tym samym zwiększa precyzję oceny stanu technicznego [106]. Ponadto CNN zostały zastosowane do wykrywania elips w obrazach przemysłowych, co umożliwia precyzyjne pomiary części mechanicznych przypominających elipsy w produkcji przemysłowej [107].

W problemach regresyjnych, w których celem jest przewidywanie ciągłych wartości numerycznych, warstwa wyjściowa modelu zazwyczaj nie posiada funkcji aktywacji lub używa funkcji liniowej. W takich przypadkach proste

FFNN mogą być wystarczające, podczas gdy CNN lub RNN mogą być używane do bardziej skomplikowanych danych obrazowych lub sekwencyjnych. Na przykład sieci neuronowe były ważnym elementem w rozwijaniu modeli opartych na danych do przewidywania wydajności odwiertów na polu naftowym, co pozwoliło na dokładniejsze prognozowanie wydajności płynów w trudnych warunkach morskich [108]. Dodatkowo sieci neuronowe zostały zastosowane do prognozowania zużycia nanokompozytów stopów ZA-27 z dodatkiem nanocząstek Al_2O_3 , co pokazało ich wyższość w przewidywaniu zużycia w porównaniu z tradycyjnymi metodami [109].

Skalowalność i złożoność modelu są kolejnymi istotnymi czynnikami w wyborze architektury. Dla małych zbiorów danych należy unikać zbyt złożonych modeli, aby zapobiec przeuczeniu. W takich przypadkach proste FFNN mogą być odpowiednie, a techniki regularyzacji i augmentacji danych mogą dodatkowo poprawić wydajność modelu. Wynika to z faktu, że prostsze modele są mniej podatne na nadmierne dopasowanie i mogą lepiej uogólniać przy mniejszej liczbie próbek na klasę [110]. Natomiast dla dużych zbiorów danych głębokie sieci neuronowe, które mogą uchwycić bardziej złożone wzorce, stają się korzystniejsze. Modele CNN o wysokiej złożoności, takie jak ResNet, LeNet i AlexNet, wykazały rewolucyjny sukces w przetwarzaniu dużych zbiorów danych obrazowych, ale mogą nie działać optymalnie na mniejszych zbiorach danych ze względu na problemy z nadmiernym dopasowaniem i niestabilnością [111]. W niektórych przypadkach warto zastosować architektury hybrydowe, takie jak połączenie CNN i RNN dla danych obrazowo-sekwencyjnych. Na przykład badania wykazały, że integracja architektur RNN i CNN w wykrywaniu ataków typu Denial of Service (DoS) w architekturze mikrouslug pozwala osiągnąć wysokie wskaźniki wykrywalności, wynoszące odpowiednio 98,4% i 97,0% [112].

Dostępne zasoby obliczeniowe stanowią istotny czynnik w procesie wyboru odpowiedniej architektury modelu. W przypadku ograniczonych zasobów bardziej odpowiednie mogą okazać się lżejsze modele, takie jak mniejsze sieci CNN lub uproszczone wersje RNN, które oferują efektywność przy niższym zapotrzebowaniu na moc obliczeniową. Z kolei optymalizacja kodu oraz wykorzystanie akceleratorów sprzętowych, takich jak GPU (*graphics processing unit*, jednostka przetwarzania grafiki) czy TPU (*tensor processing unit*, jednostka przetwarzania tensora), może znacząco zwiększyć szybkość trenowania modeli, szczególnie w kontekście przetwarzania dużych zbiorów danych oraz bardziej złożonych architektur. W sytuacji, gdy dostępne zasoby obliczeniowe są znaczne, można zastosować bardziej

skomplikowane architektury, które charakteryzują się wyższą skutecznością w realizacji złożonych zadań. Niezwykle istotne jest również rozważenie strategii zarządzania pamięcią oraz efektywnego przetwarzania danych. Standardowe podejścia zakładają, że cały zbiór danych uczących zostaje załadowany do pamięci operacyjnej (RAM) przed rozpoczęciem procesu trenowania. Tego rodzaju strategia jest prosta w implementacji i umożliwia szybkie przetwarzanie danych oraz optymalne dostosowanie modelu. Jednakże w warunkach ograniczonej pamięci załadowanie pełnego zbioru danych może skutkować znacznym obciążeniem systemu, a nawet prowadzić do problemów z wydajnością. W takich przypadkach ważną rolę odgrywają techniki optymalizacji pamięci oraz zarządzania danymi. Akceleratory sprzętowe, takie jak GPU i TPU, pozwalają na równoległe przetwarzanie wielu operacji matematycznych, co jest szczególnie istotne przy trenowaniu złożonych modeli na dużych zbiorach danych. Dzięki odpowiednim technikom optymalizacji i zarządzania pamięcią możliwe jest efektywne trenowanie nawet zaawansowanych architektur, mimo ograniczeń związanych z dostępnymi zasobami obliczeniowymi.

W procesie doboru odpowiedniej architektury sieci neuronowej jedną z metod stosowanych do analizy wpływu poszczególnych komponentów na wydajność modelu jest technika ablacyjna. Polega ona na systematycznym usuwaniu lub modyfikowaniu wybranych elementów sieci, takich jak warstwy, neurony czy połączenia, a następnie ocenie, jak te zmiany wpływają na jakość wyników modelu. Dzięki temu możliwe jest zidentyfikowanie komponentów kluczowych dla skuteczności modelu oraz tych, które mogą być mniej istotne lub zbędne. Technika ablacyjna wspomaga proces optymalizacji złożonych sieci, umożliwiając projektantom identyfikację i ewentualne usunięcie niepotrzebnych elementów, co może prowadzić do uproszczenia architektury oraz poprawy efektywności obliczeniowej. Jest to metoda, która dostarcza cennych informacji na temat wpływu różnych części modelu na jego zdolność do generalizacji, co może być szczególnie przydatne przy pracy z różnorodnymi danymi i zadaniami [113].

Wybór odpowiedniej architektury sieci neuronowej zależy od specyfiki danych, rodzaju problemu oraz dostępnych zasobów. Istotnym elementem jest dobranie modelu, który najlepiej pasuje do charakterystyki zadania, co często wymaga eksperymentowania z różnymi architekturami i ich konfiguracjami.

3.7. Kluczowe aspekty podczas uczenia sieci neuronowych

Podczas uczenia sieci neuronowych ważne jest, aby zwrócić uwagę na kilka kluczowych aspektów, które mogą wpływać na skuteczność i wydajność modelu (ryc. 3.6).



Ryc. 3.6. Kluczowe aspekty podczas uczenia sieci neuronowych

Te aspekty mogą się różnić w zależności od tego, czy pracujemy z danymi regresyjnymi czy klasyfikacyjnymi. Te drugie dotyczą problemów, w których celem jest przypisanie danych wejściowych do jednej z kilku dyskretnych kategorii lub klas. Dane regresyjne odnoszą się do problemów, w których celem jest przewidywanie ciągłych wartości liczbowych.

Przygotowanie danych wejściowych jest również ważnym etapem w procesie uczenia sieci neuronowych. Dane wejściowe powinny być odpowiednio przetworzone i znormalizowane, aby uniknąć problemów z gradientami oraz przyspieszyć proces uczenia. Normalizacja danych polega na skalowaniu wartości do określonego zakresu, co pozwala na bardziej efektywne przetwarzanie danych przez sieć neuronową. Typowe metody normalizacji obejmują skalowanie do zakresu $[0, 1]$ lub $[-1, 1]$. W przypadku danych obrazowych skalowanie pikseli do zakresu $[0, 1]$ jest standardową praktyką, która stabilizuje proces uczenia się i pomaga uniknąć problemów z gradientami, takich jak ich znikanie lub eksplozja [114].

Kolejnym ważnym aspektem jest inicjalizacja wag, która jest istotna dla efektywnego uczenia sieci neuronowej. Proces ten rozpoczyna się od losowej inicjalizacji wag, co jest konieczne do przerwania symetrii między neuronami i umożliwia efektywne uczenie. Odpowiednia inicjalizacja wag jest niezwykle istotna, ponieważ zbyt małe lub zbyt duże wartości mogą prowadzić do problemów z gradientami, takich jak zanikający lub eksplodujący gradient. Wagi w sieci neuronowej pełnią rolę parametrów, które decydują o tym, jak silnie poszczególne wejścia wpływają na wyjście neuronu. Proces uczenia polega na optymalizacji tych wag, aby sieć neuronowa mogła jak najdokładniej przewidywać wyniki dla nowych danych. Początkowo wagi są zazwyczaj inicjalizowane losowo lub przy użyciu bardziej zaawansowanych metod, takich jak inicjalizacja He [115] czy Xavier [116], które pomagają w szybszym i bardziej stabilnym uczeniu sieci. Następnie wagi są modyfikowane podczas procesu trenowania, który obejmuje wiele iteracji propagacji w przód i wstecz [117].

Wybór funkcji aktywacji ma istotne znaczenie dla skuteczności sieci neuronowej, a jej rola może się różnić w zależności od tego, czy model jest stosowany do klasyfikacji czy do regresji. W modelach klasyfikacyjnych funkcje aktywacji mają na celu przekształcenie wyjść neuronów do postaci prawdopodobieństw klas. Najczęściej stosowane funkcje aktywacji w kontekście klasyfikacji to *sigmoidalna*, *tanh* oraz *softmax*. W modelach regresyjnych mają one na celu generowanie ciągłych wartości numerycznych. W przypadku regresji najczęściej stosowanymi funkcjami aktywacji są *ReLU* oraz funkcja liniowa. Funkcja liniowa, określona wzorem $\varphi(z) = z$, jest jedną z najprostszych funkcji aktywacji. Wartość wyjściowa równa się wartości wejściowej, co oznacza, że nie wprowadza nieliniowości do modelu. Często używa się jej w warstwach wyjściowych modeli regresji, gdzie model musi przewidzieć rzeczywiste wartości. Chociaż nie jest odpowiednia dla warstw ukrytych w głębokich sieciach neuronowych, ponieważ nie wprowadza nieliniowości, jest przydatna w określonych przypadkach, takich jak zadania regresyjne, gdzie liniowe zależności między wejściami a wyjściami są pożądane.

Funkcja kosztu, znana również jako funkcja strat (ang. *loss function*), jest podstawowym elementem w procesie uczenia maszynowego, który mierzy, jak dobrze model przewiduje pożądane wyniki. Stanowi miarę różnicy między wartościami przewidywanymi przez model a rzeczywistymi etykietami (danymi wyjściowymi). Głównym celem funkcji kosztu jest dostarczenie modelowi informacji zwrotnej, którą wykorzystuje się do dostosowywania wag sieci neuronowej w celu minimalizacji tej różnicy. Proces ten, nazywany optymalizacją,

pozwała na stopniowe poprawianie dokładności modelu poprzez iteracyjne zmniejszanie wartości funkcji kosztu. Powinna być dobrana odpowiednio do rodzaju problemu. Dla problemów regresyjnych typową funkcją kosztu jest błąd średniokwadratowy (MSE), podczas gdy dla problemów klasyfikacyjnych używa się zazwyczaj funkcji entropii krzyżowej [118].

Optymalizacja jest ważna w procesie uczenia sieci neuronowych, w którym algorytmy, takie jak gradient descent, Adam, AdaGrad, RMSProp, Levenberga-Marquardta, regularyzacja Bayesowska oraz gradient skalowany sprzężony są używane do aktualizacji wag na podstawie obliczonych gradientów, w celu minimalizacji funkcji kosztu. Wybór odpowiedniego algorytmu zależy od specyfiki problemu oraz dostępnych zasobów obliczeniowych. Proces ten jest iteracyjny i polega na zastosowaniu algorytmów optymalizacyjnych, które obliczają gradienty funkcji kosztu względem wag sieci, a następnie modyfikują te wagi w taki sposób, aby zmniejszyć wartość funkcji kosztu. Główne algorytmy optymalizacyjne różnią się tym, jak adaptują się do zmian gradientów, co bezpośrednio wpływa na szybkość i stabilność procesu uczenia [119].

Wybór odpowiedniej techniki regularyzacji jest również istotny dla uzyskania modelu o dobrej zdolności generalizacji. W praktyce często stosuje się kombinację różnych metod regularyzacji. Celem regularyzacji jest poprawa zdolności modelu do generalizacji, czyli jego umiejętności przewidywania na nowych, niewidzianych wcześniej danych. Regularyzacja pomaga zapobiegać przeuczeniu, które ma miejsce, gdy model uczy się zbyt szczegółowo wzorców ze zbioru uczącego, co prowadzi do słabej wydajności na danych testowych. Na przykład można zastosować L2 regularyzację, aby utrzymać małe wagi, oraz dropout, aby zapewnić, że model nie stanie się zbyt zależny od określonych neuronów. Gong (2009) wskazuje na istotność technik regularyzacji w poprawie wydajności modeli sieci neuronowych. Zastosowanie takich technik pomaga w tworzeniu modeli, które są bardziej odporne na szum i lepiej radzą sobie z nowymi danymi, co jest ważne w praktycznych zastosowaniach uczenia maszynowego [120].

Dobór odpowiedniej architektury sieci neuronowej, przygotowanie danych, inicjalizacja wag, wybór funkcji aktywacji, optymalizacja oraz regularyzacja to kluczowe aspekty, które łącznie wpływają na efektywność i wydajność modeli sieci neuronowych, a ich odpowiednie zastosowanie pozwala na osiągnięcie lepszych wyników w różnorodnych zadaniach.

3.7.1. Metody optymalizacyjne w sieciach neuronowych

W procesie uczenia sztucznych sieci neuronowych istotną rolę odgrywają metody optymalizacyjne, które odpowiadają za efektywną aktualizację wag modelu w celu minimalizacji funkcji kosztu. Dobór odpowiedniego algorytmu optymalizacyjnego ma znaczący wpływ na szybkość uczenia, stabilność procesu oraz ostateczną jakość wytrenowanego modelu. W literaturze można spotkać różne metody optymalizacyjne. Wśród najczęściej stosowanych w sieciach neuronowych znajdują się: *gradient descent* (GD), *adaptive moment estimation* (Adam), *adaptive gradient algorithm* (AdaGrad), *root mean square propagation* (RMSProp), Levenberga–Marquardta (LM), regularyzacja Bayesowska (BR) oraz gradient skalowany sprzężony (SCG). Każdy z tych algorytmów charakteryzuje się odmiennym podejściem do adaptacji wag, co czyni je bardziej lub mniej odpowiednimi do określonych typów problemów i danych.

Gradient descent (GD) jest jedną z najbardziej fundamentalnych metod optymalizacyjnych stosowanych w uczeniu maszynowym, w tym w trenowaniu sztucznych sieci neuronowych. Jego głównym celem jest minimalizacja funkcji kosztu, która mierzy, jak dobrze model dopasowuje się do danych uczących. W procesie optymalizacji GD iteracyjnie aktualizuje wagi modelu, przesuwając je w kierunku przeciwnym do gradientu funkcji kosztu względem tych wag. W standardowej wersji gradient descent gradienty są obliczane na podstawie całego zbioru danych uczących. Oznacza to, że przed każdą aktualizacją wag modelu algorytm przetwarza wszystkie próbki w zbiorze uczącym, co zapewnia dokładne obliczenie gradientu, ale może być bardzo kosztowne obliczeniowo, zwłaszcza przy dużych zbiorach danych. W takich przypadkach każda iteracja może być czasochłonna, co spowalnia proces uczenia [121].

Adam jest zaawansowanym algorytmem optymalizacyjnym, który łączy zalety dwóch innych metod: AdaGrad i RMSProp. Adam śledzi średnie pierwszego momentu (średnia gradientów) oraz drugiego momentu (kwadrat gradientów) w celu dynamicznego dostosowywania współczynnika uczenia się dla każdej wagi [122]. Automatycznie dostosowuje współczynniki uczenia się na podstawie historycznych gradientów, co pozwala na szybsze i bardziej stabilne zbieżności, oraz wykorzystuje technikę momentum, która pomaga w łagodzeniu oscylacji gradientów, co prowadzi do bardziej stabilnych i szybszych aktualizacji wag.

AdaGrad jest metodą optymalizacyjną, która dostosowuje współczynniki uczenia się dla każdej wagi na podstawie sumy kwadratów gradientów, jakie waga napotkała w czasie uczenia modelu [123]. Zmniejsza współczynniki uczenia

się dla często aktualizowanych wag i zwiększa je dla rzadziej aktualizowanych, co jest korzystne dla rzadkich danych. AdaGrad jest szczególnie efektywny dla rzadkich i gęstych danych, ale jego główną wadą jest to, że współczynniki uczenia się mogą stać się bardzo małe, co prowadzi do powolnej zbieżności.

RMSProp, podobnie jak AdaGrad, dostosowuje współczynniki uczenia się dla każdej wagi, ale robi to w sposób, który zapobiega problemowi zbyt małych współczynników uczenia się. Wykorzystuje eksponencjalnie ważoną średnią kwadratów gradientów, co pozwala na stabilniejsze i szybsze uczenie się w porównaniu z AdaGrad. RMSProp utrzymuje współczynniki uczenia się na bardziej stałym poziomie, co zapobiega problemowi powolnej zbieżności [124].

Levenberg–Marquardt (LM) to często najszybsza metoda optymalizacyjna dla małych i średnich problemów. Algorytm ten okazuje się szczególnie efektywny w przypadkach, gdy sieć neuronowa jest stosunkowo niewielka i nie wymaga ekstremalnie dużych zasobów pamięciowych [125]. Algorytm LM jest iteracyjną metodą optymalizacji, łączącą cechy metody gradientu prostego i metody Newtona. Często stosuje się go w trenowaniu sieci neuronowych ze względu na swoje szybkie tempo zbieżności. Algorytm LM aktualizuje parametry sieci neuronowej według wzoru:

$$w_{k+1} = w_k - \left[J(w_k)^T J(w_k) + \mu I \right]^{-1} J(w_k)^T e(w_k) \quad (3.9)$$

gdzie:

w_k – wektor wag w iteracji k ;

$J(w_k)$ – macierz Jacobiego funkcji błędu w iteracji k ;

μ – parametr regularyzacji;

I – macierz jednostkowa;

$e(w_k)$ – wektor błędów w iteracji k .

Macierz Jacobiego J zawiera pierwsze pochodne funkcji błędu względem wag, co pozwala na przybliżenie drugich pochodnych, jak w metodzie Newtona, ale z dodatkowym parametrem regularyzacji, który kontroluje kroki aktualizacji. Algorytm ten jest znany ze swojej efektywności w zastosowaniach, takich jak inwersja kinematyczna dla robotów [126].

Regularyzacja Bayesowska (BR) wprowadza regularyzację poprzez zastosowanie metod bayesowskich, mając na celu uniknięcie przeuczenia poprzez dodanie do funkcji celu członu regularizacyjnego penalizującego duże wartości wag. Funkcja kosztu w regularyzacji Bayesowskiej składa się z dwóch członów:

$$E(w) = E_D(w) + \lambda E_W(w) \quad (3.10)$$

gdzie:

$E_D(w)$ – funkcja błędu (np. suma kwadratów błędów);

$E_W(w)$ – człon regularyzacyjny (np. suma kwadratów wag);

λ – współczynnik regularyzacji, który kontroluje wpływ regularyzacji na funkcję kosztu.

Celem jest maksymalizacja *a posteriori* (MAP) rozkładu prawdopodobieństwa wag. MAP rozkładu prawdopodobieństwa wag to metoda wnioskowania statystycznego, która łączy informacje z danych (obserwacji) z wcześniejszą wiedzą o rozkładzie prawdopodobieństwa wag (ang. *prior*). Formuła Bayesa jest podstawą do obliczenia rozkładu *a posteriori*:

$$P(w|D) = \frac{P(D|w)P(w)}{P(D)} \quad (3.11)$$

gdzie:

$P(w|D)$ – rozkład *a posteriori*, czyli rozkład prawdopodobieństwa wag w po uwzględnieniu danych D ;

$P(D|w)$ – rozkład prawdopodobieństwa danych D przy zadanych wagach w ;

$P(w)$ – *a priori* rozkład prawdopodobieństwa wag, czyli wcześniejsza wiedza o wagach przed uwzględnieniem danych;

$P(D)$ – rozkład prawdopodobieństwa danych, który jest normalizatorem (niezależnym od wag).

Celem MAP jest znalezienie wartości wag w , które maksymalizują rozkład *a posteriori* $P(w|D)$. W praktyce oznacza to znalezienie takich wag, które są najbardziej prawdopodobne przy uwzględnieniu zarówno danych, jak i wcześniejszej wiedzy o wagach. Proces ten polega na maksymalizacji wyrażenia:

$$P(w|D) \propto P(D|w)P(w) \quad (3.12)$$

gdzie: oznaczenia jak powyżej.

Ponieważ $P(D)$ jest stałą niezależną od wag, można ją pominąć w maksymalizacji. Dlatego proces MAP sprowadza się do maksymalizacji iloczynu $P(D|w)P(w)$. MAP uwzględnia oba te elementy: zarówno dopasowanie modelu do danych, jak i wcześniejszą wiedzę o tym, jak wagi powinny się rozkładać. Maksymalizując $P(w|D)$, MAP znajduje taki zestaw wag, który jest najbardziej prawdopodobny, jeśli uwzględni się oba te źródła informacji.

Regularyzacja Bayesowska prowadzi do lepszej ogólnej wydajności oraz mniejszej podatności na przeuczenie w miarę wzrostu rozmiaru sieci, co wykazano w badaniach symulacyjnych [127].

Algorytm gradientu skalowanego sprzężonego (SCG) jest wariantem metody gradientu sprzężonego, zoptymalizowanym do zastosowań w sieciach neuronowych. SCG poprawia wydajność i stabilność poprzez wprowadzenie skalowania w kierunku poszukiwania. Algorytm SCG iteracyjnie aktualizuje wagi według wzoru:

$$w_{k+1} = w_k + \alpha_k d_k \quad (3.13)$$

gdzie:

α_k – optymalny krok w kierunku d_k ;

d_k – kierunek poszukiwania w iteracji k .

Kierunek poszukiwania d_k jest aktualizowany na podstawie gradientu funkcji celu oraz poprzednich kierunków poszukiwania:

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \quad (3.14)$$

gdzie:

g_{k+1} – gradient funkcji celu w iteracji $k+1$;

β_k – współczynnik określający, jak wiele z poprzedniego kierunku jest zachowane.

Algorytm SCG znany jest ze swojej skuteczności w zastosowaniach takich jak przetwarzanie sygnałów sonarowych [128].

Wybór odpowiedniej metody optymalizacyjnej jest niezbędny do skutecznego trenowania sztucznych sieci neuronowych, a każda z przedstawionych metod ma swoje unikalne zalety i ograniczenia. GD jest podstawową techniką, która mimo swojej prostoty, może być nieefektywna przy dużych zbiorach danych. Adam, łącząc cechy AdaGrad i RMSProp, oferuje dynamiczne dostosowywanie współczynników uczenia, co umożliwia szybszą i bardziej stabilną zbieżność. AdaGrad i RMSProp dostosowują współczynniki uczenia dla każdej wagi, ale różnią się podejściem do utrzymywania stabilności procesu uczenia. LM jest niezwykle efektywny dla małych i średnich sieci, szczególnie gdy priorytetem ma być szybkie zbieganie do minimum funkcji kosztu. BR wprowadza regularyzację, aby zapobiegać przeuczeniu, natomiast SCG optymalizuje proces uczenia przez wprowadzenie skalowania w kierunku poszukiwania. Każdy z tych algorytmów można dostosować do specyficznych warunków i potrzeb

danego projektu, co podkreśla ich wszechstronność w kontekście różnorodnych zastosowań sieci neuronowych.

3.7.2. Techniki regularyzacji w sieciach neuronowych

Regularyzacja jest ważnym elementem w procesie trenowania modeli uczenia maszynowego, który ma na celu poprawę zdolności modelu do generalizacji na nowych, nieznanych danych. W przypadku sieci neuronowych i innych modeli predykcyjnych regularyzacja pomaga zapobiegać przeuczeniu, które może prowadzić do nadmiernego dopasowania modelu do danych uczących, a w konsekwencji do niskiej wydajności na zbiorach testowych.

Wśród technik regularyzacji można wyróżnić m.in. następujące:

1. Regularyzacja L2

$$J(w) = J_0(w) + \lambda \sum_{j=1}^n w_j^2 \quad (3.15)$$

gdzie:

$J_0(w)$ – oryginalna funkcja kosztu;

w – wagi modelu;

λ – współczynnik regularyzacji.

Regularyzacja L2, znana również jako Ridge Regression, wprowadza dodatkowy człon do funkcji kosztu, proporcjonalny do sumy kwadratów wag modelu. Dodanie tego członu powoduje, że model jest karany za dużą wartość wag, co prowadzi do zmniejszenia ich wartości. Regularyzacja L2 sprawdza się szczególnie w przypadku problemów z wieloma cechami, ponieważ pomaga w utrzymaniu małych wag i zmniejsza ryzyko przeuczenia modelu. W efekcie model staje się bardziej odporny na szum w danych [129].

2. Regularyzacja L1

$$J(w) = J_0(w) + \lambda \sum_{j=1}^n |w_j| \quad (3.16)$$

Regularyzacja L1, znana również jako Lasso Regression, dodaje do funkcji kosztu karę proporcjonalną do sumy wartości bezwzględnych wag modelu. W przeciwieństwie do regularyzacji L2, regularyzacja L1 ma tendencję do prowadzenia do rzadkości modelu, co oznacza, że niektóre wagi mogą stać się dokładnie zerowe. To sprawia, że L1 staje się szczególnie użyteczna, gdy celem jest selekcja cech [130].

3. Dropout

Dropout jest techniką regularyzacji specyficzną dla sieci neuronowych. Polega na losowym „wyłączaniu” (czyli ustawianiu na zero) pewnego procentu neuronów w warstwie podczas uczenia. Przykładowo, dla danej warstwy neuronowej co epokę losowy podzbiór neuronów jest „wyłączany” i nie bierze udziału w przetwarzaniu danych. W czasie testowania wszystkie neurony są aktywne, ale ich wagi przeskalowane w celu kompensacji dropoutu podczas uczenia. Dropout zmusza sieć do bardziej niezależnej pracy neuronów i zapobiega przeuczeniu poprzez redukcję nadmiernego dopasowania się do danych uczących [131].

4. Wczesnego zatrzymania

Technika wczesnego zatrzymania (ang. *early stopping*) polega na monitorowaniu wydajności modelu na zbiorze walidacyjnym podczas uczenia i przerwaniu uczenia, gdy wydajność na zbiorze walidacyjnym przestaje się poprawiać.

Typowy algorytm wczesnego zatrzymania obejmuje:

- rozdzielenie danych na zbiór uczący i walidacyjny;
- uczenie modelu na zbiorze uczących oraz monitorowanie wydajności na zbiorze walidacyjnym;
- zatrzymanie uczenia modelu, gdy wydajność na zbiorze walidacyjnym przestaje się poprawiać przez określoną liczbę epok (np. 6).

Technika wczesnego zatrzymania jest prostą i efektywną metodą, która zapobiega nadmiernemu dostosowywaniu modelu do danych uczących, zapewniając lepszą generalizację na nowych danych [132,133].

Każda z powyżej omówionych technik wprowadza dodatkowe mechanizmy do procesu trenowania modelu, które pomagają w kontrolowaniu złożoności modelu oraz w poprawie jego zdolności do przewidywania na podstawie nowych danych. Ich zastosowanie w praktyce pozwala na budowanie bardziej niezawodnych modeli, które potrafią skutecznie przewidywać wyniki na podstawie nowych, nieznanych danych, co jest kluczowe dla sukcesu w zastosowaniach sieci neuronowych.

3.8. Wskaźniki oceny jakości sieci neuronowych

Wskaźniki oceny jakości sieci neuronowych odgrywają istotną rolę w analizie i interpretacji wyników uzyskiwanych przez modele uczenia maszynowego. Ich wybór oraz interpretacja zależą od rodzaju problemu, który jest poddawany

analizie – czy dotyczy problemu klasyfikacyjnego, czy regresyjnego. Każdy typ problemu wymaga różnych miar jakości, które pomagają w ocenie skuteczności modelu.

Dla problemów regresyjnych, w których przewidywane są wartości ciągłe, stosuje się inne wskaźniki oceny niż dla problemów klasyfikacyjnych. Jednym z podstawowych wskaźników dla problemów regresyjnych jest błąd średniokwadratowy (ang. *mean squared error*, MSE), który mierzy średnią wartość kwadratów różnic między wartościami przewidywanymi a rzeczywistymi. Kolejnym wskaźnikiem jest pierwiastek z błędu średniokwadratowego (ang. *root mean squared error*, RMSE), będący pierwiastkiem kwadratowym z MSE, który zapewnia intuicyjną interpretację błędu w tych samych jednostkach, co dane wyjściowe. Średni błąd bezwzględny (ang. *mean absolute error*, MAE) oblicza średnią wartość bezwzględnych różnic między wartościami przewidywanymi a rzeczywistymi, co jest mniej wrażliwe na ekstremalne błędy. Wskaźnik R (współczynnik korelacji) mierzy, jaka część zmienności w danych jest wyjaśniona przez model, co pozwala na ocenę ogólnego dopasowania modelu do danych.

Dodatkowo istnieją bardziej specyficzne wskaźniki, takie jak względny błąd informacji (ang. *relative information error*, RIE) oraz średni bezwzględny błąd procentowy (ang. *mean absolute percentage error*, MAPE). RIE jest miarą względną, która ocenia jakość modelu względem prostego modelu bazowego, podczas gdy MAPE mierzy średni błąd jako procent wartości rzeczywistej, co jest szczególnie przydatne w kontekście danych, gdzie wartości mają różne skale.

Problemy klasyfikacyjne dotyczą przypisania próbek do jednej z wielu klas. W takich przypadkach istotne jest określenie, jak dokładnie model przewiduje klasy dla nowych danych. Dokładność jest jednym z podstawowych wskaźników oceny, definiowanym jako procent poprawnych klasyfikacji spośród wszystkich prób. Jest on szczególnie użyteczny, gdy klasy są zbalansowane i nie ma jednej klasy dominującej. Macierz pomyłek to tabela, która przedstawia liczbę prawdziwie pozytywnych, fałszywie pozytywnych, prawdziwie negatywnych i fałszywie negatywnych klasyfikacji, co pomaga zrozumieć, które klasy są najczęściej mylone. Precyzja jest stosunkiem liczby prawdziwie pozytywnych klasyfikacji do liczby wszystkich pozytywnych klasyfikacji (prawdziwych i fałszywych), co jest szczególnie ważne w przypadku, gdy koszt fałszywych pozytywnych jest wysoki. Czułość to stosunek liczby prawdziwie pozytywnych klasyfikacji do liczby wszystkich rzeczywistych pozytywnych przypadków, co jest istotne, gdy ważne jest wykrycie wszystkich pozytywnych przypadków.

F1-score to harmoniczna średnia precyzji i czułości, która stanowi zbalansowaną miarę uwzględniającą zarówno fałszywe pozytywy, jak i fałszywe negatywy.

Wybór odpowiedniego wskaźnika oceny jakości jest istotny dla prawidłowej interpretacji wyników modelu i zależy od specyfiki analizowanego problemu. W dalszej części tego rozdziału opisano szczegółowo wymienione powyżej wskaźniki jakości oceny sieci neuronowych.

3.8.1. Wskaźniki oceny jakości sieci neuronowych dla problemów regresyjnych

W kontekście problemów regresyjnych ocena jakości modeli sieci neuronowych jest istotna, aby zapewnić ich skuteczność i dokładność. Istnieje kilka wskaźników jakości, które są powszechnie stosowane do oceny wyników predykcji modeli regresyjnych. W analizowanych w tej monografii badaniach szczegółowo omówione zostaną takie wskaźniki, jak: współczynnik korelacji (R), błąd średniokwadratowy (MSE), pierwiastek z błędu średniokwadratowego (RMSE), względny błąd informacji (RIE), średni bezwzględny błąd procentowy (MAPE) oraz średni błąd bezwzględny (MAE).

Współczynnik korelacji (R) mierzy siłę i kierunek liniowej zależności między dwiema zmiennymi. Jest obliczany jako:

$$R(y_i, \hat{y}_i) = \frac{\text{cov}(y_i, \hat{y}_i)}{\sigma_{y_i} \sigma_{\hat{y}_i}}, \quad R \in < 0, 1 > \quad (3.17)$$

gdzie:

y_i – rzeczywiste wartości;

$\hat{y}_i$ – wartości przewidywane przez model;

$\text{cov}(y_i, \hat{y}_i)$ – kowariancja rzeczywistych i przewidywanych wartości;

σ_{y_i} – odchylenie standardowe rzeczywistych wartości;

$\sigma_{\hat{y}_i}$ – odchylenie standardowe wartości przewidywanych.

Wartość R w zakresie od 0 do 1 wskazuje na liniową zależność między zmiennymi. Wartość bliska 1 oznacza silną dodatnią korelację, a wartość bliska 0 wskazuje na brak liniowej korelacji. W kontekście procesów produkcyjnych wysoki współczynnik korelacji między rzeczywistymi a przewidywanymi wartościami może wskazywać na skuteczność modelu w prognozowaniu wydajności produkcji lub jakości produktów.

Błąd średniokwadratowy (MSE) mierzy średnią kwadratów różnic między rzeczywistymi a przewidywanymi wartościami [134]. Niższe wartości MSE wskazują na lepsze dopasowanie modelu do danych. MSE jest czuły na duże błędy, ponieważ różnice są podnoszone do kwadratu. Formuła, według której jest on obliczany, to:

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (3.18)$$

gdzie: oznaczenia jak powyżej.

Pierwiastek z błędu średniokwadratowego (RMSE) to pierwiastek z MSE. RMSE bardziej penalizuje większe błędy w porównaniu z mniejszymi, co oznacza, że duże odchylenia od rzeczywistych wartości mają znacząco większy wpływ na końcowy wynik tego wskaźnika. Innymi słowy, im większy błąd predykcji, tym bardziej jest on akcentowany w wartości RMSE, co czyni ten wskaźnik szczególnie wrażliwym na duże błędy w predykcjach. Wylicza się go za pomocą następującej formuły [134–136]:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad (3.19)$$

gdzie: oznaczenia jak powyżej.

Względny błąd informacji (RIE) mierzy względny błąd predykcji, uwzględniając różnice między rzeczywistymi a przewidywanymi wartościami w kontekście całkowitej informacji w danych. Jest szczególnie użyteczny w przypadkach, gdy ważne jest zrozumienie błędów względnych, a nie bezwzględnych. Jest wyrażany matematycznie jako [137]:

$$RIE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \right| \quad (3.20)$$

gdzie: oznaczenia jak powyżej.

Średni bezwzględny błąd procentowy (MAPE) mierzy średni bezwzględny procentowy błąd między rzeczywistymi a przewidywanymi wartościami [134, 138]. MAPE wyraża błąd w procentach, co czyni go łatwym do interpretacji. Definiuje się go następująco [136]:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \right| * 100\% \quad (3.21)$$

gdzie: oznaczenia jak powyżej.

Średni błąd bezwzględny (MAE) mierzy średnią bezwzględną różnicę między rzeczywistymi a przewidywanymi wartościami [134]. MAE jest prostą i intuicyjną miarą błędu. Ponieważ nie uwzględnia kwadratów różnic, nie jest tak czuły na duże błędy, jak MSE czy RMSE. Często stosuje się go w przypadkach, gdzie równomierne traktowanie wszystkich błędów jest pożądane. Oblicza się go jako [135]:

$$MAE = \frac{1}{N} \sum_{i=1}^N (|y_i - \hat{y}_i|) \quad (3.22)$$

gdzie: oznaczenia jak powyżej.

Ocena jakości modeli regresyjnych w kontekście produkcji jest kluczowa dla zapewnienia ich skuteczności i dokładności. Wskaźniki, takie jak R, MSE, RMSE, RIE, MAPE i MAE oferują różne perspektywy na dokładność i precyzję modeli, umożliwiając wybór najbardziej odpowiednich miar w zależności od specyfiki danego zastosowania produkcyjnego. Właściwy wybór i interpretacja tych wskaźników może znacząco poprawić procesy decyzyjne w kontekście przemysłowym.

3.8.2. Wskaźniki oceny jakości sieci neuronowych dla problemów klasyfikacyjnych

Wskaźniki oceny modeli klasyfikacyjnych są niezbędne dla zrozumienia, jak dobrze model radzi sobie z przypisywaniem próbek do odpowiednich klas. Istnieje wiele miar, które można wykorzystać do oceny wydajności modelu, z których każda dostarcza unikalnych informacji na temat jego skuteczności. Klasyczne wskaźniki, takie jak dokładność, precyzja, czułość i F1-score, pozwalają na ocenę różnych aspektów wyników klasyfikacji. Analiza tych wskaźników wspierana przez narzędzia, takie jak macierz pomyłek czy krzywa ROC, pozwala na głębokie zrozumienie wydajności modelu i jest nieodzowna w procesie doskonalenia algorytmów klasyfikacyjnych. Wprowadzenie tych miar jest niezbędne do wyboru najlepszego modelu do konkretnego zadania oraz identyfikacji obszarów wymagających poprawy.

Macierz pomyłek jest narzędziem wykorzystywanym do oceny wydajności algorytmów klasyfikacyjnych. Jest to tabela używana do wizualizacji wyników działania algorytmu klasyfikacji, która porównuje rzeczywiste klasy próbek z klasami przewidywanymi przez model. Pozwala ona na szczegółową analizę błędów klasyfikacji. Strukturę macierzy pomyłek przedstawia tabela 3.1.

Tabela 3.1. Struktura macierzy pomyłek

	Predykcja negatywna	Predykcja pozytywna
Stan rzeczywisty przypadki negatywne	FN	FP
Stan rzeczywisty przypadki pozytywne	TN	TP

W przypadku klasyfikacji binarnej macierz pomyłek składa się z czterech podstawowych elementów:

- TP (True Positives) – prawdziwie pozytywne klasyfikacje, przypadki poprawnie zidentyfikowane jako pozytywne;
- TN (True Negatives) – prawdziwie negatywne klasyfikacje, przypadki poprawnie zidentyfikowane jako negatywne;
- FP (False Positives) – fałszywie pozytywne klasyfikacje, przypadki błędnie zidentyfikowane jako pozytywne;
- FN (False Negatives) – fałszywie negatywne klasyfikacje, przypadki błędnie zidentyfikowane jako negatywne.

Macierz pomyłek dostarcza pełnego obrazu działania modelu klasyfikacji, pokazując, ile przypadków każdej klasy zostało poprawnie i błędnie sklasyfikowanych. Pozwala to na identyfikację specyficznych problemów modelu, takich jak tendencja do fałszywie pozytywnych lub fałszywie negatywnych klasyfikacji. Analizując macierz pomyłek, można wyciągnąć wnioski na temat koniecznych usprawnień modelu [139,140]. W obszarze produkcji macierz pomyłek może pomóc zidentyfikować, które rodzaje wad są najczęściej błędnie klasyfikowane. Na przykład jeżeli system kontroli jakości w fabryce ma tendencję do błędnej klasyfikacji pewnych typów wad, macierz pomyłek pomoże zidentyfikować te problemy i skierować wysiłki na ich poprawę.

Macierz pomyłek stanowi podstawę do obliczenia innych kluczowych wskaźników oceny, takich jak dokładność, precyzja, czułość i F1-score. Dokładność to miara, która określa procent poprawnych klasyfikacji spośród wszystkich prób, niezależnie od klasy. Dokładność to intuicyjny i łatwy do zrozumienia wskaźnik, szczególnie użyteczny w sytuacjach, gdy klasy są zbalansowane. W przeciwnych przypadkach (np. gdy jedna klasa znacznie przewyższa liczebnie inną), dokładność może być myląca. Na przykład model, który zawsze przewiduje dominującą klasę, może mieć wysoką dokładność, mimo że nie rozpoznaje mniejszościowych klas poprawnie [139,140]. W produkcyjnych przypadkach, gdzie jedna klasa może być znacznie liczniejsza, dokładność może być myląca. Na przykład w systemie monitorującym jakość produktów

na linii produkcyjnej, jeżeli większość produktów jest bez wad, model przewidujący, że wszystkie produkty są dobre, może osiągnąć wysoką dokładność, pomimo że nie wykrywa wadliwych produktów.

$$Dokładność = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.23)$$

gdzie: oznaczenia jak powyżej.

Precyzja to miara dokładności przewidywań modelu, która określa procent prawdziwie pozytywnych klasyfikacji wśród wszystkich przypadków sklasyfikowanych jako pozytywne. Precyzja ma kluczowe znaczenie w sytuacjach, gdy koszt fałszywie pozytywnych klasyfikacji jest wysoki. Na przykład w systemach medycznych fałszywie pozytywne diagnozy mogą prowadzić do niepotrzebnych i kosztownych badań oraz leczenia. W obszarze produkcji fałszywie pozytywne klasyfikacje mogą oznaczać, że dobry produkt został błędnie oznaczony jako wadliwy, co prowadzi do niepotrzebnych kosztów związanych z jego ponowną kontrolą lub odrzuceniem. Wysoka precyzja oznacza, że model rzadko przypisuje wadę dobremu produktowi [139–141]:

$$Precyzja = \frac{TP}{TP + FP} \quad (3.24)$$

gdzie: oznaczenia jak powyżej.

Czułość to miara skuteczności modelu w wykrywaniu wszystkich pozytywnych przypadków. Wskaźnik ten odgrywa istotną rolę w sytuacjach, gdy priorytetem jest wykrycie wszystkich pozytywnych przypadków. W obszarze produkcji wysoka czułość oznacza, że model rzadko przeocza wadliwy produkt, co jest ważne dla utrzymania wysokiej jakości produkcji i minimalizacji ryzyka dostarczenia wadliwych produktów do klienta [139,140]:

$$Czułość = \frac{TP}{TP + FN} \quad (3.25)$$

gdzie: oznaczenia jak powyżej.

F1-score to harmoniczna średnia precyzji i czułości, zapewniająca zbalansowaną miarę wydajności modelu. F1-score uwzględnia zarówno fałszywe pozytywne, jak i fałszywe negatywne, będąc wskaźnikiem przydatnym, gdy istnieje potrzeba równowagi między precyzją a czułością. W produkcji F1-score jest szczególnie ważny, gdy zarówno fałszywie pozytywne, jak i fałszywie negatywne wyniki są kosztowne. Na przykład równowaga między wykrywaniem wszystkich wadliwych produktów (wysoka czułość) a minimalizowaniem

liczby dobrych produktów błędnie oznaczonych jako wadliwe (wysoka precyzja) jest istotne dla efektywnego systemu kontroli jakości [139,140]:

$$F1Score = 2 \frac{Precision * Sensitivity}{Precision + Sensitivity} \quad (3.26)$$

gdzie: oznaczenia jak powyżej.

Krzywa ROC (ang. *receiver operating characteristic*) to narzędzie używane do oceny jakości modeli klasyfikacyjnych, szczególnie w kontekście problemów z niezbilansowanymi danymi. Krzywa ROC przedstawia zależność między czułością (wskaźnikiem prawdziwie pozytywnych wyników – TPR) a specyficznością (1 – wskaźnikiem fałszywie pozytywnych wyników – FPR) dla różnych progów decyzyjnych [142]. Wykres ten pozwala na ocenę, jak dobrze model radzi sobie z rozróżnianiem między klasami. Aby stworzyć krzywą ROC, dla każdego możliwego progu decyzyjnego wylicza się wskaźniki czułości i specyficzności, a następnie przedstawia się je na wykresie, gdzie:

- oś x (pozioma) reprezentuje 1 – specyficzność (wskaźnik fałszywie pozytywnych wyników);
- oś y (pionowa) reprezentuje czułość (wskaźnik prawdziwie pozytywnych wyników).

Krzywą ROC można wykorzystać również jako miernik jakości klasyfikatora, wyznaczając pole powierzchni pod krzywą (AUC – ang. *area under the curve*). Wartość AUC mieści się w przedziale od 0 do 1, gdzie wartość 1 oznacza model doskonały, a wartość 0,5 model losowy. Im wyższa wartość AUC, tym lepsza jest zdolność modelu do rozróżniania między klasami. Krzywa ROC może być szczególnie przydatna do oceny, jak zmieniają się wyniki klasyfikacji wraz ze zmianą progu decyzyjnego [143]. Na przykład w systemach kontroli jakości można być bardziej zainteresowanym minimalizowaniem fałszywych negatywów (przegapionych wadliwych produktów) kosztem większej liczby fałszywie pozytywnych klasyfikacji (błędne oznaczenie dobrego produktu jako wadliwy). Analiza krzywej ROC pozwala na znalezienie optymalnego punktu pracy, który najlepiej odpowiada specyficznym wymaganiom danego procesu produkcyjnego. Krzywa ROC oraz AUC stanowią narzędzie uzupełniające inne miary jakości klasyfikacji, takie jak dokładność, precyzja, czułość i F1-score [144,145].

Ocenianie jakości modeli klasyfikacyjnych jest kluczowym elementem procesu budowy i wdrażania sieci neuronowych. W kontekście problemów klasyfikacyjnych odpowiednie wskaźniki pozwalają na zrozumienie, jak dobrze model radzi sobie z przypisywaniem obserwacji do odpowiednich kategorii.

Dobór właściwych miar oceny jest istotny zarówno dla samego procesu trenowania modelu, jak i dla późniejszej oceny jego wydajności w środowisku produkcyjnym. Dzięki temu można nie tylko określić, który model najlepiej spełnia określone wymagania, ale także zidentyfikować potencjalne obszary do poprawy.

4. Optymalizacja parametrów obróbki skrawaniem

Optymalizacja parametrów obróbki skrawaniem z wykorzystaniem sztucznych sieci neuronowych stanowi jedno z najnowocześniejszych podejść w inżynierii produkcji, które integruje zaawansowane techniki modelowania i analizy z praktyką przemysłową. Współczesne procesy produkcyjne charakteryzujące się rosnącymi wymaganiami dotyczącymi precyzji, efektywności oraz jakości produktów, stawiają przed inżynierami wyzwania, którym tradycyjne metody optymalizacji nie zawsze są w stanie sprostać. W tym kontekście zastosowanie ANN pozwala na znaczący postęp w dziedzinie optymalizacji procesów obróbkowych, umożliwiając nie tylko precyzyjne dostosowanie parametrów obróbki, ale również dynamiczne reagowanie na zmieniające się warunki produkcyjne.

Wprowadzenie sztucznych sieci neuronowych do optymalizacji procesów obróbki wynika z ich zdolności do modelowania nieliniowych, złożonych relacji między wieloma zmiennymi. Tradycyjne metody, oparte na analizach statystycznych lub metodach prób i błędów, często okazują się niewystarczające, zwłaszcza w przypadku obróbki materiałów o trudnych właściwościach fizykochemicznych, takich jak stopy magnezu. Stopy te, ze względu na swoją niską gęstość, wysoką wytrzymałość w stosunku do masy oraz doskonałe właściwości tłumienia drgań, znajdują szerokie zastosowanie w zaawansowanych branżach przemysłowych, takich jak lotnictwo, motoryzacja czy elektronika. Jednak ich obróbka stawia specyficzne wyzwania technologiczne, związane z wysoką reaktywnością chemiczną i niską temperaturą zapłonu, co zwiększa

ryzyko wystąpienia niebezpiecznych sytuacji, takich jak samozapłon podczas procesu skrawania.

Zastosowanie ANN w optymalizacji parametrów obróbki pozwala na precyzyjne modelowanie tych ryzyk oraz na identyfikację optymalnych warunków pracy, które minimalizują możliwość wystąpienia niepożądanych zdarzeń. Dzięki zdolności do uczenia się na podstawie danych doświadczalnych sieci neuronowe są w stanie przewidywać wpływ zmian parametrów obróbki, takich jak prędkość skrawania, posuw czy głębokość skrawania, na końcowe właściwości powierzchni, zużycie narzędzi oraz bezpieczeństwo procesu. Co więcej, ANN mogą być wykorzystane do dynamicznej optymalizacji procesu w czasie rzeczywistym, co ma szczególnie znaczenie w produkcji seryjnej, gdzie minimalizacja odchyłek od założonych parametrów jest istotna dla utrzymania wysokiej jakości produktów.

W niniejszym rozdziale omówione zostaną: metodyka identyfikacji procesu na podstawie modelu doświadczalnego oraz zaawansowanych technik modelowania parametrów chropowatości powierzchni, które stanowią miarę jakości wyrobu końcowego. Współczesne wymagania przemysłowe dotyczące jakości powierzchni są coraz bardziej rygorystyczne, co wymusza konieczność precyzyjnej kontroli parametrów obróbki. ANN, dzięki swojej zdolności do dokładnego modelowania relacji między parametrami procesu a właściwościami powierzchni, pozwalają na optymalizację tych parametrów w celu uzyskania pożądanej chropowatości, co bezpośrednio przekłada się na funkcjonalność i trwałość produktów.

Rozdział ten kończy szczegółowa analiza optymalizacji parametrów obróbki z uwzględnieniem ograniczeń siłowych, co stanowi kolejny istotny element efektywnego zarządzania procesami produkcyjnymi. Siły skrawania mają bezpośredni wpływ na jakość powierzchni oraz na trwałość narzędzi, a ich optymalizacja jest niezbędna do zapewnienia stabilności procesu oraz do minimalizacji kosztów produkcji. ANN, poprzez modelowanie sił skrawania, umożliwiają identyfikację takich kombinacji parametrów, które minimalizują negatywne efekty siły skrawania, jednocześnie maksymalizując wydajność procesu. Dzięki temu możliwe jest nie tylko zapewnienie wysokiej jakości produktów, ale również zwiększenie efektywności energetycznej procesów produkcyjnych, co ma istotne znaczenie w obliczu globalnych trendów zmierzających do zrównoważonego rozwoju i redukcji emisji gazów cieplarnianych.

Optymalizacja parametrów obróbki skrawaniem z wykorzystaniem ANN to zaawansowane narzędzie, które może umożliwić inżynierom nie tylko zwiększenie efektywności procesów produkcyjnych, ale również zapewnienie

najwyższych standardów jakości i bezpieczeństwa. Rozdział ten dostarczy kompleksowego wglądu w te zagadnienia, przedstawione zostaną zarówno teoretyczne podstawy, jak i praktyczne zastosowania ANN w optymalizacji procesów obróbki materiałów, z uwzględnieniem specyficznych wyzwań związanych z obróbką trudnych materiałów, takich jak stopy magnezu.

4.1. Proces frezowania stopów magnezu

Frezowanie stanowi proces obróbki skrawaniem, który znajduje szerokie zastosowanie w produkcji komponentów z różnych materiałów, w tym szczególnie stopów magnezu. Stopy magnezu ze względu na swoje unikalne właściwości fizykochemiczne, takie jak niska gęstość, wysoka wytrzymałość w stosunku do masy oraz doskonała zdolność tłumienia drgań, zyskują coraz większą popularność w zaawansowanych zastosowaniach inżynieryjnych [146]. Niska gęstość stopów magnezu sprawia, że są one wyjątkowo atrakcyjne w przemyśle lotniczym, motoryzacyjnym oraz elektronicznym [147].

Dzięki wysokiej wytrzymałości mechanicznej w stosunku do masy stopy magnezu pozwalają na projektowanie lekkich, a zarazem wytrzymałych konstrukcji, co jest szczególnie istotne w aplikacjach lotniczych, gdzie każda redukcja masy przekłada się na znaczne oszczędności paliwa i zwiększenie zasięgu lotu [148]. W motoryzacji zastosowanie stopów magnezu prowadzi do zmniejszenia masy pojazdów, co w konsekwencji redukuje zużycie paliwa oraz emisję spalin, a tym samym przyczynia się do ochrony środowiska [149]. Stopy magnezu ponadto charakteryzują się doskonałymi właściwościami tłumienia drgań, co jest szczególnie korzystne w przemyśle elektronicznym, gdzie minimalizacja wibracji jest kluczowa dla poprawy wydajności i trwałości urządzeń elektronicznych [150].

Wykorzystanie stopów magnezu w zaawansowanych aplikacjach inżynieryjnych pozwala na znaczną redukcję masy komponentów, co jest niezwykle istotne w obliczu globalnych trendów zmierzających do poprawy efektywności energetycznej oraz redukcji emisji gazów cieplarnianych [151].

Jednakże frezowanie stopów magnezu wiąże się z pewnymi istotnymi wyzwaniami technologicznymi, wynikającymi z unikalnych właściwości fizykochemicznych tych materiałów. Magnez i jego stopy charakteryzują się wysoką reaktywnością chemiczną oraz niską temperaturą zapłonu, co czyni je szczególnie podatnymi na samozapłon podczas procesów obróbki skrawaniem. W wyniku oddziaływania sił skrawających oraz tarcia generowane są wysokie temperatury, które mogą przekraczać próg zapłonu magnezu, prowadząc do

egzotermicznych reakcji spalania, trudnych do opanowania i stwarzających poważne zagrożenie dla bezpieczeństwa pracy [151].

Drobne wióry, pył czy cienkie elementy magnezu powstające podczas frezowania są szczególnie niebezpieczne, gdyż ich stosunek powierzchni do objętości jest wysoki, co sprzyja szybkiemu nagrzewaniu i zapłonowi. W celu minimalizacji ryzyka samozapłonu istotne jest przestrzeganie ściśle określonych parametrów obróbki. Optymalna prędkość skrawania powinna być dostosowana do właściwości obrabianego stopu magnezu, aby zredukować generowanie nadmiernego ciepła. Dodatkowo skuteczne chłodzenie narzędzi skrawających oraz powierzchni obrabianych jest niezbędne, aby efektywnie odprowadzać ciepło i zapobiec osiągnięciu temperatury zapłonu [151].

Właściwe zarządzanie procesem frezowania stopów magnezu ma duże znaczenie nie tylko z powodów ekonomicznych, ale przede wszystkim z punktu widzenia bezpieczeństwa i efektywności produkcji. Zarządzanie tym procesem wymaga ścisłej kontroli parametrów obróbki oraz zastosowania zaawansowanych technologii, które pozwalają na optymalizację procesu pod kątem jakości otrzymanych wyrobów, kosztów oraz bezpieczeństwa. Zaawansowane strategie obróbki, takie jak frezowanie trochoidalne, mogą znacznie zwiększyć stabilność procesu poprzez zmniejszenie sił skrawania i drgań, które mają znaczenie dla zachowania dokładności i jakości wytwarzanych elementów [152,153].

Modelowanie procesu frezowania stopów magnezu stanowi istotny krok w kierunku zwiększenia efektywności i bezpieczeństwa obróbki tych materiałów. Modelowanie komputerowe pozwala na symulację różnych scenariuszy obróbki, co umożliwia identyfikację optymalnych parametrów procesu bez konieczności przeprowadzania kosztownych i czasochłonnych eksperymentów. Dzięki modelowaniu można przewidzieć zachowanie materiału w różnych warunkach obróbki, co pozwala na lepsze zrozumienie mechanizmów skrawania oraz minimalizację ryzyka zapłonu.

Komputerowe modelowanie procesu frezowania stopów magnezu umożliwia symulację różnych kombinacji prędkości skrawania, posuwu oraz głębokości skrawania, co pozwala na określenie najbardziej efektywnych ustawień zapewniających wysoką jakość powierzchni oraz minimalne zużycie narzędzi. Na przykład badania przeprowadzone przez Ritti i Bhat (2021) wykazały, że odpowiednie modelowanie narzędzi do procesów, takich jak *friction stir processing* (obróbka metodą zgrzewania tarcowego), pozwala na projektowanie narzędzi o minimalnej deformacji, niskim naprężeniu oraz wysokiej trwałości zmęczeniowej, co jest istotne dla skutecznego przetwarzania stopów magnezu [154].

Modelowanie procesów pozwala na redukcję liczby niezbędnych prób eksperymentalnych, co przekłada się na znaczne oszczędności finansowe, gdyż pozwala na identyfikację problemów przed rozpoczęciem rzeczywistej obróbki, minimalizując straty materiałowe i czasowe. Ponadto dzięki możliwości symulacji różnych scenariuszy można lepiej przewidzieć potencjalne zagrożenia, takie jak zapłon materiału, i wdrożyć odpowiednie środki zapobiegawcze, co zwiększa bezpieczeństwo procesu. Modelowanie umożliwia również szybkie testowanie innowacyjnych rozwiązań i technologii obróbkowych, co przyspiesza wdrażanie nowych metod produkcji.

Modelowanie procesu frezowania stopów magnezu jest niezwykle złożonym zadaniem, wymagającym precyzyjnej identyfikacji wszystkich kluczowych parametrów i zależności. Właściwe zrozumienie i opisanie mechanizmów zachodzących podczas frezowania jest podstawą do stworzenia wiarygodnego modelu, który może służyć do optymalizacji procesów produkcyjnych. Aby osiągnąć taki poziom precyzji i dokładności w modelowaniu, niezbędne jest przeprowadzenie dokładnej identyfikacji procesu obróbki.

4.2. Identyfikacja procesu na podstawie modelu doświadczalnego

Identyfikacja procesów ma charakter iteracyjny i polega na porównywaniu modeli o różnych strukturach danych. Model jest uproszczonym przedstawieniem rzeczywistości, skonstruowanym w konkretnym celu, eliminującym szczegóły nieistotne dla jego osiągnięcia. Modele umożliwiają lepsze zrozumienie badanych zjawisk, co jest niezbędne w badaniach naukowych. Model doświadczalny odwzorowuje przebieg procesu oraz jego cechy, co jest kluczowe w rozwiązywaniu problemów praktycznych, zwłaszcza gdy matematyczne opisanie procesu bywa skomplikowane [155–157].

Proces identyfikacji rozpoczyna się od oszacowania parametrów prostych modeli strukturalnych. Jeżeli model nie spełnia wymagań dotyczących dokładności, stopniowo zwiększa się jego złożoność. Ostatecznie wybiera się najprostszы model, który najlepiej opisuje dynamikę systemu. Ważne jest, aby rozpocząć od modeli prostych, ponieważ bardziej złożone modele nie zawsze są dokładniejsze. Ponadto, przy zwiększeniu złożoności modelu wzrasta niepewność oszacowań parametrów i wymagana jest większa ilość danych [158,159].

Identyfikacja procesu obróbki polega na szczegółowym badaniu i analizie wszystkich czynników wpływających na przebieg frezowania, takich jak

prędkość skrawania, posuw, głębokość skrawania, rodzaj narzędzia, warunki chłodzenia czy właściwości materiału. Każdy z tych parametrów ma bezpośredni wpływ na efektywność obróbki, jakość powierzchni, zużycie narzędzi oraz bezpieczeństwo procesu. Na przykład prędkość skrawania i posuw determinują ilość ciepła generowanego podczas obróbki, co jest istotne dla minimalizowania ryzyka zapłonu łatwopalnych stopów magnezu.

Rzetelna identyfikacja procesu umożliwia zrozumienie wzajemnych zależności między parametrami obróbki a wynikowymi właściwościami produktu. Pozwala to na przewidywanie, w jaki sposób zmiany w jednym parametrze wpłyną na inne aspekty procesu, co jest niezbędne do optymalizacji parametrów obróbki w celu osiągnięcia najlepszych wyników. Bez tej wiedzy próby optymalizacji mogłyby prowadzić do nieprzewidywalnych rezultatów, zwiększając ryzyko nieefektywnej produkcji lub niebezpiecznych sytuacji.

Znajomość dynamicznych właściwości procesu frezowania umożliwiłaby przewidywanie istotnych parametrów wyjściowych, takich jak chropowatość powierzchni czy siła skrawania. Modelowanie zjawisk zachodzących w strefie skrawania jest jednak złożone i nieliniowe. Wynika to z dynamicznych interakcji siłowych między narzędziem a przedmiotem obrabianym oraz z niestacjonarności procesu, jak w przypadku frezowania. Frez kontaktuje się z materiałem w zmieniających się warunkach, co powoduje zmiany w sztywności i warunkach skrawania. Aby przewidzieć te zmiany, należy traktować proces frezowania jako system sterowania i przeprowadzić jego identyfikację, określając wejścia i wyjścia modelu. Model „czarnej skrzynki” jest tutaj szczególnie użyteczny, ponieważ dopasowuje dane wejściowe i wyjściowe uzyskane z systemu.

Jako sterowalne i dostępne dane wejściowe do modelu należy uznać parametry obróbki (np. prędkość skrawania (v_c), posuw na ostrze (f) oraz głębokość skrawania (a_p)), mając jednak świadomość, że powstałe siły skrawania oraz drgania mogą zmieniać się ze względu na lokalne warunki procesu w różnych punktach toru narzędzia skrawającego. Ważnym krokiem w procesie identyfikacji jest również zbieranie danych eksperymentalnych. Pomiar sił skrawania czy analiza jakości powierzchni obrabianego materiału pozwalają na stworzenie dokładnych baz danych, które są niezbędne do kalibracji i weryfikacji opracowywanego modelu. Dane te mogą pochodzić z rzeczywistych prób frezowania lub z zaawansowanych eksperymentów laboratoryjnych.

Po zebraniu danych i ich analizie możliwe jest stworzenie modelu, który odzwierciedla rzeczywiste zależności zachodzące podczas procesu frezowania. Model taki może być następnie wykorzystany do symulacji różnych scenariuszy

obróbki. Właściwa identyfikacja procesu frezowania stopów magnezu jest istotnym elementem skutecznego modelowania. Pozwala na dokładne zrozumienie mechanizmów skrawania oraz zależności między parametrami procesu a wynikami obróbki. Dzięki temu możliwe jest stworzenie wiarygodnego modelu doświadczalnego, który przyczynia się do optymalizacji procesu, zwiększenia jego efektywności oraz zapewnienia bezpieczeństwa produkcji.

4.3. Metodyka prowadzonych badań eksperymentalnych

Badania eksperymentalne dotyczące obróbki frezowaniem przeprowadzono w Katedrze Podstaw Inżynierii Produkcji Wydziału Mechanicznego Politechniki Lubelskiej. Badania wykonano dla stopu AZ91D. Jego skład chemiczny przedstawia tabela 4.1. Eksperyment przeprowadzono na pionowym centrum Avia VMC 800HS z systemem sterowania Heidenhain iTNC 530 z maksymalną prędkością obrotową wrzeciona 24 000 obr/min i maksymalną prędkością posuwu 40 m/min (ryc. 4.1).

Tabela 4.1. Skład chemiczny stopu magnezu AZ91D [160]

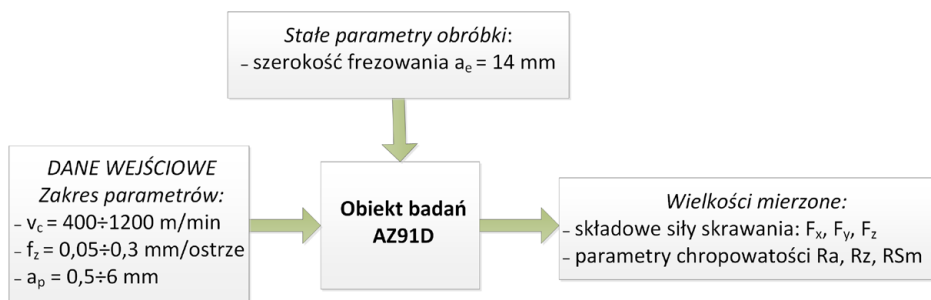
Skład chemiczny	Al [%]	Zn [%]	Mn [%]	Si [%]	Cu [%]	Fe [%]	Ni [%]	Be [%]	Mg [%]
AZ91D	8,91	0,66	0,22	0,016	0,002	0,002	0,001	0,001	pozostałe



Ryc. 4.1. Centrum Avia VMC 800HS

W badaniach dotyczących chropowatości powierzchni wykorzystano frez o średnicy $\varnothing 16$ mm z wymiennymi ostrzami z polikrystalicznego diamentu (PCD) firmy Guhring z kątem pochylenia $\lambda_s = 0^\circ$. Dla poprawy stabilności obróbki frez zamocowano w uchwycie narzędziowym Shrinkfit SFD z adapterem HSK-A63 firmy Seco. Układ narzędzie-uchwyt został dynamicznie wyważony zgodnie z normą ISO 21940-11: 2016 [160]. W przypadku badań składowych sił skrawania wykorzystano również frez o średnicy $\varnothing 16$ mm, jednak z innym kątem pochylenia – $\lambda_s = 20^\circ$ [161].

Proces frezowania przeprowadzono przy następujących parametrach obróbki: szerokość skrawania $a_e = 15$ [mm], prędkość skrawania $v_c = 400 \div 1200$ [m/min], posuw na ostrze $f_z = 0,05 \div 0,3$ [mm/ostrze] oraz głębokość skrawania $a_p = 0,5 \div 6$ [mm]. Schemat obiektu badań przedstawia ryc. 4.2.



Ryc. 4.2. Schemat obiektu badań

Parametry chropowatości 2D zostały zmierzone za pomocą profilometru Hommel Roughness Tester T1000 od Hommel Etamic (ryc. 4.3), z wykorzystaniem cyfrowego filtra Gaussa (M1), zgodnie z międzynarodową normą ISO 11562. Szczegółowe parametry pomiaru obejmowały: długość przejazdu $lt = 4,8$ [mm] oraz długość odcinka pomiarowego $ln = 0,8$ [mm], prędkość skanowania $vt = 0,5$ [mm/s], a także zakresy pomiarowe i rozdzielczość $M = \pm 320$ [μm]/ $0,04$ [μm]. Każdy pomiar był wykonywany pięciokrotnie na powierzchni, aby zapewnić wystarczającą ilość danych do obliczenia wartości średnich i odchylenia standardowego wyników.

Do pomiaru składowych sił skrawania zastosowano siłomierz piezoelektryczny Kistler 9257B (ryc. 4.4) wykonujący pomiar w zakresie -5 [kN] $\div$ 5 [kN] wraz ze wzmacniaczem 5017B oraz analizator LMS SCADAS Mobile SCM05.

Wyniki badań eksperymentalnych: wybranych parametrów chropowatości (Ra , Rz , RSm) [160] oraz składowych sił skrawania (F_x , F_y , F_z) [161] zostały wykorzystane do przygotowania modeli wykorzystywanych do optymalizacji parametrów obróbki.



Ryc. 4.3. Profilometr Hommel Roughness Tester T1000



Ryc. 4.4. Siłomierz piezoelektryczny Kistler 9257B

4.4. Optymalizacja parametrów obróbki frezowaniem w celu uzyskania zadanych wartości chropowatości powierzchni

Optymalizacja parametrów obróbki frezowaniem w celu uzyskania zadanych wartości chropowatości powierzchni jest kluczowym aspektem w procesie produkcji elementów wykonanych z różnych materiałów, w tym ze stopów magnezu. Chropowatość powierzchni, będąca miarą mikrogeometrii powierzchni, odgrywa istotną rolę w określeniu jakości produktu końcowego. Właściwie zoptymalizowane parametry obróbki wpływają nie tylko na wygląd i funkcjonalność komponentów, ale również na ich trwałość i wydajność w eksploatacji.

Z punktu widzenia jakości produkcji kontrola chropowatości powierzchni jest niezbędna do spełnienia specyfikacji technicznych oraz wymagań klientów. Elementy o zbyt wysokiej chropowatości mogą prowadzić do zwiększonego tarcia, szybszego zużycia i obniżenia wydajności operacyjnej. Z kolei powierzchnie o zbyt niskiej chropowatości mogą być wynikiem nadmiernej obróbki, co niepotrzebnie zwiększa koszty produkcji i zużycie narzędzi. Dlatego precyzyjne ustalenie i utrzymanie optymalnej chropowatości jest ważne dla zapewnienia wysokiej jakości produktów przy minimalnych kosztach produkcji.

Zarządzanie produkcją wymaga zatem ciągłej optymalizacji parametrów obróbki, takich jak: prędkość skrawania, posuw, głębokość skrawania oraz warunki chłodzenia. Poprzez dokładne dostosowanie tych parametrów można uzyskać pożądane wartości chropowatości, co bezpośrednio przekłada się na poprawę wydajności procesów produkcyjnych. Optymalizacja parametrów obróbki prowadzi do redukcji czasu cyklu produkcyjnego, zmniejszenia zużycia narzędzi oraz ograniczenia liczby wadliwych produktów, co z kolei zwiększa ogólną efektywność produkcji [162–164].

W kontekście stopów magnezu optymalizacja parametrów obróbki jest szczególnie istotna ze względu na specyficzne właściwości tego materiału, takie jak niska gęstość i wysoka wytrzymałość. Właściwa obróbka magnezu wymaga precyzyjnego zarządzania procesem, aby uniknąć problemów związanych z jego łatwopalnością oraz zapewnić wymagane parametry powierzchniowe. Badania pokazują, że zastosowanie odpowiednich parametrów skrawania, takich jak odpowiednia prędkość, posuw i głębokość skrawania, ma znaczenie dla osiągnięcia optymalnej chropowatości powierzchni i minimalizacji zużycia narzędzi [165].

4.4.1. Metodyka modelowania parametrów chropowości

Współczesna technologia produkcji charakteryzuje się coraz większymi wymaganiami dotyczącymi jakości powierzchni. Kontrola chropowości powierzchni jest elementem, który wpływa na funkcjonalność i trwałość wytworzonych elementów. Parametry chropowości, takie jak Ra , Rz , i RSm , są powszechnie stosowane w celu oceny jakości powierzchni w różnych gałęziach przemysłu, od automatyki po zaawansowane technologie biomedyczne. Ra (średnia arytmetyczna bezwzględnych rzędnych profilu chropowości) [166] jest jednym z najczęściej używanych parametrów chropowości, służącym do określania ogólnej tekstury powierzchni poprzez średnią wartość bezwzględnych odchylek profilu od linii środkowej. Ra dostarcza podstawowych informacji o stanie powierzchni i jest uznawany za wskaźnik ogólnego stopnia wygładzenia. Jest szeroko stosowany, mimo że nie umożliwia wnioskowania o wielu istotnych cechach profilu chropowości, dlatego zaleca się jego stosowanie do monitorowania ustabilizowanych procesów [167,168].

$$Ra = \frac{1}{l} \int_0^l |z(x)| dx \quad (4.1)$$

gdzie:

l – długość odcinka elementarnego;

$z(x)$ – wartości rzędnych wewnątrz odcinka elementarnego.

Rz (największa wysokość profilu chropowości) [166] jest parametrem, który daje bardziej szczegółowy obraz topografii powierzchni. Rz jest definiowany jako suma wysokości najwyższego wzniesienia profilu oraz głębokości najniższego wgłębienia profilu wewnątrz odcinka elementarnego. Rz jest użyteczny w aplikacjach, gdzie szczyty i doliny na powierzchni mają bezpośredni wpływ na działanie i żywotność produktu, np. w elementach stykowych maszyn, gdzie mogą wpływać na tarcie i zużycie.

RSm (średnia szerokość rowków elementów profilu chropowości) to parametr określający średnią szerokość elementów profilu wewnątrz odcinka elementarnego [166,167]. RSm jest przydatny w ocenie regularności i rozkładu cech tekstury powierzchni, co ma istotne znaczenie w przypadku materiałów, które muszą spełniać określone kryteria funkcjonalne dotyczące przyczepności, smarowania czy przepływu płynów po powierzchni [169].

$$RSm = \frac{1}{m} \sum_{i=1}^m Xs_i \quad (4.2)$$

gdzie:

Xs_i – szerokość i -tego elementu profilu;

m – liczba elementów profilu.

Wybór tych trzech parametrów dla modelowania chropowatości 2D jest uzasadniony ich zdolnością do zapewnienia kompleksowej charakterystyki powierzchni, co ma znaczenie w wielu zastosowaniach technicznych i przemysłowych. Ra , Rz i RSm wspólnie dostarczają pełnego obrazu różnych aspektów chropowatości, umożliwiając lepsze zrozumienie wpływu obróbki powierzchniowej na właściwości użytkowe i funkcjonalne produktów. Ponadto analiza tych parametrów pozwala na optymalizację procesów produkcyjnych i poprawę jakości końcowych produktów, co jest decydujące w utrzymaniu konkurencyjności na rynku.

W warstwie wejściowej zdefiniowano trzy neurony wejściowe, tj. zmienne parametry obróbki v_c , f_z i a_p . W warstwie wyjściowej zdefiniowano jeden neuron – modelowany parametr. Zestaw danych eksperymentalnych składający się z 46 obserwacji ustalono w proporcji 75% : 25%, gdzie pierwsza wartość oznacza udział danych uczących, a druga – danych walidacyjnych. Podział ten był ustalony przed rozpoczęciem procesu trenowania i nie zmieniał się w kolejnych epokach. Ze względu na małą ilość danych pominięto dane testowe. Taki podział pozwala na efektywniejsze wykorzystanie ograniczonej ilości danych do trenowania modelu, umożliwiając jednocześnie walidację i dostosowanie hiperparametrów. Walidacja jest niezbędna do monitorowania nadmiernego dopasowania i regulacji modelu w trakcie uczenia [58].

Modelowanie przeprowadzono za pomocą sieci neuronowych MLP w oprogramowaniu Matlab 2023a. Wybór architektury sieci neuronowej MLP w niniejszym badaniu został podyktowany specyficznymi wymaganiami wynikającymi z bardzo małego zestawu danych. Mała liczba próbek (46 obserwacji) wymagała zastosowania modelu, który z jednej strony jest wystarczająco złożony, aby uchwycić istotne zależności w danych, a z drugiej na tyle prosty, aby uniknąć problemów związanych z nadmiernym dopasowaniem. Złożone architektury, takie jak głębokie sieci neuronowe, mogłyby prowadzić do przeuczenia modelu, co znacznie obniżyłoby jego zdolność do generalizacji na nowych, nieznanych danych. W związku z tym zdecydowano się na użycie MLP z jedną warstwą ukrytą, której liczba neuronów od 2 do 12 była iteracyjnie dostosowywana w celu znalezienia optymalnej konfiguracji. Dodatkowo, przy tak małej ilości danych

zalecane jest, aby liczba neuronów w warstwie ukrytej nie przekraczała liczby obserwacji w zbiorze danych. Zalecenie to wynika z doświadczeń, które pokazują, że przekroczenie tej liczby może prowadzić do sytuacji, w której model zbyt dokładnie zapamiętuje dane uczące, zamiast uczyć się ogólnych wzorców – co z kolei skutkuje nadmiernym dopasowaniem. W kontekście małych zbiorów danych mniejsza liczba neuronów pomaga w zachowaniu zdolności modelu do generalizacji, co umożliwia lepsze radzenie sobie z danymi, które nie były widziane podczas treningu. Taka konfiguracja wspomaga również stabilność modelu i redukuje ryzyko wystąpienia problemów związanych z przeuczeniem, co jest kluczowe przy pracy z ograniczonymi zestawami danych.

Podczas doboru parametrów uczenia sieci zastosowano domyślną metodę inicjalizacji wag, która zapewnia stabilne warunki początkowe dla procesu uczenia. Odpowiednia inicjalizacja wag pozwala modelowi na skuteczne rozpoczęcie procesu optymalizacji, co jest szczególnie ważne przy ograniczonej ilości danych.

W warstwie ukrytej sieci zastosowano funkcję aktywacji typu sigmoidalnego, która jest szeroko stosowana w modelowaniu nieliniowych zależności między zmiennymi wejściowymi a wyjściowymi. Funkcja ta umożliwia modelowi uchwycenie skomplikowanych wzorców w danych wejściowych, co jest niezbędne dla osiągnięcia wysokiej dokładności predykcji. Do oceny jakości modelu zastosowano funkcję kosztu MSE, która jest powszechnie używana w problemach regresji.

Proces trenowania sieci neuronowej został przeprowadzony z wykorzystaniem kilku metod optymalizacyjnych: algorytmu Levenberga–Marquardta (LM), regularyzacji Bayesowskiej (BR) oraz gradientu skalowanego sprzężonego (SCG). W procesie projektowania sieci neuronowych istotne jest określenie optymalnej liczby neuronów, która pozwoli uzyskać najlepszą jakość modelu. Początkowym etapem było zatem zastosowanie algorytmu LM, szeroko stosowanego ze względu na swoje efektywne właściwości szybkiej zbieżności, szczególnie w problemach, gdzie stosunkowo mniejsza ilość danych może stanowić wyzwanie. Dzięki temu podejściu udało się wyłonić konfigurację sieci, która zapewniła najwyższą jakość przetwarzania danych. Następnie, wykorzystując ustaloną liczbę neuronów, przystąpiono do trenowania pozostałych sieci z wykorzystaniem innych algorytmów: SCG oraz BR. To podejście pozwoliło na bezpośrednie porównanie efektywności poszczególnych algorytmów optymalizacyjnych przy identycznej architekturze sieci, co umożliwiło rzetelną ocenę ich wydajności i efektywności w różnych scenariuszach.

W celu uniknięcia przeuczenia sieci został zastosowany mechanizm wczesnego zatrzymania, który ustawiono na poziomie 6, co oznacza, że proces

trenowania modelu zostanie zatrzymany, gdy przez 6 kolejnych epok nie nastąpi poprawa wyników na zbiorze walidacyjnym.

Ocena wydajności i jakości sieci została przeprowadzona przy użyciu wskaźników jakościowych, takich jak: R, MSE, RMSE, RIE, MAPE i MAE, które zostały szczegółowo opisane we wcześniejszych podrozdziałach.

Dodatkowo, aby porównać skuteczność sieci neuronowych, wykonano modelowanie z wykorzystaniem *gradient boosting machines* (GBM). GBM opiera się na idei połączenia wielu słabych modeli, zazwyczaj drzew decyzyjnych, w celu stworzenia jednego, silnego predyktora. Iteracyjnie buduje on model, dodając w każdej iteracji nowy słaby model, który redukuje błędy poprzednich modeli. Każdy nowy model stara się poprawić wynik poprzez minimalizację funkcji kosztu (np. błędu kwadratowego w przypadku regresji). Na początku ustawia się początkową wartość predykcji F_0 , która jest zazwyczaj średnią wartością zmiennej celu w przypadku regresji.

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma) \quad (4.3)$$

gdzie:

L – funkcja kosztu (np. średni błąd kwadratowy);

y_i – rzeczywista wartość zmiennej celu;

γ – wartość początkowej predykcji.

W każdej iteracji m GBM dodaje nowy słaby model $h_m(x)$, który jest trenowany na resztach (błędach) dotychczasowego zespołu modeli.

$$F_m(x) = F_{m-1}(x) + \nu h_m(x) \quad (4.4)$$

gdzie:

$F_m(x)$ – predykcja zespołu modeli po m -tej iteracji;

ν – współczynnik uczenia;

$h_m(x)$ – słaby model trenowany w m -tej iteracji.

Dane podczas modelowania GBM zostały podzielone na zestaw uczący oraz testowy przy użyciu walidacji krzyżowej z 25% próbą testową. Modele GBM trenowano przy użyciu różnej liczby drzew, od 50 do 1000 z krokiem co 50, tak aby znaleźć najlepszy model na podstawie minimalnego błędu MSE. Najlepszy model był wybierany i oceniany na podstawie zestawu testowego, a wyniki zostały wykorzystane do oceny jakości modelu w porównaniu z ANN.

4.4.2. Wyniki modelowania wybranych parametrów chropowatości 2D

Pierwszym modelowanym parametrem chropowatości był parametr R_a . Wykorzystując algorytm LM do uczenia, najlepsze parametry jakościowe otrzymano dla sieci MLP z warstwą ukrytą zawierającą sześć neuronów, co udało się osiągnąć w ciągu 7 epok. Następnie dla sześciu neuronów w warstwie ukrytej trenowano sieci, wykorzystując dwa pozostałe algorytmy: SCG oraz BR. Tabela 4.2 przedstawia wskaźniki wydajności trzech modeli sieci neuronowych wytrenowanych przy użyciu różnych algorytmów: LM, SCG i BR. Wydajność każdego modelu jest oceniana na podstawie wskaźników jakości, w tym R, MSE, RMSE, RIE, MAPE i MAE.

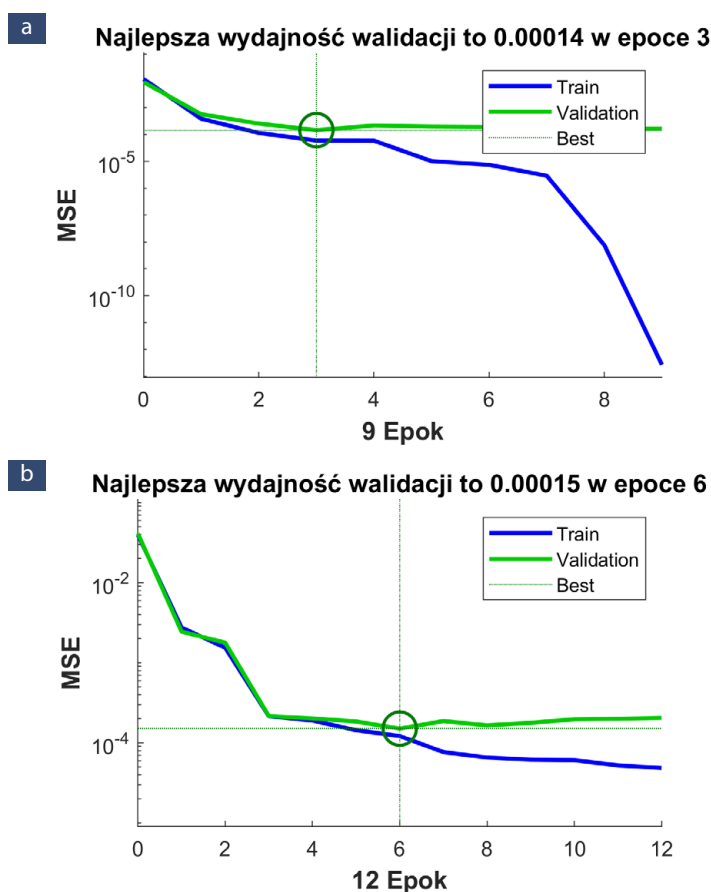
Tabela 4.2. Wskaźniki jakości dla różnych algorytmów uczenia sieci MLP dla parametru R_a

Algorytm uczenia sieci MLP	LM	SCG	BR
Epoka	9	12	89
R (cały zbiór danych)	0,9869	0,9749	0,9723
MSE	0,00008	0,00012	0,00011
RMSE	0,009	0,0114	0,0106
RIE	0,0232	0,0292	0,0274
MAPE	0,0166	0,0261	0,0199
MAE	0,0062	0,0101	0,0085

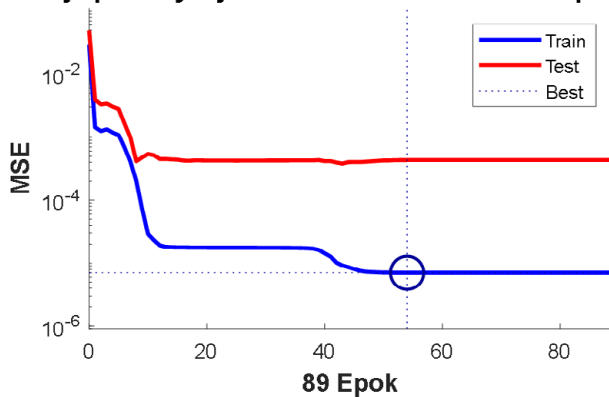
W wyniku przeprowadzenia analizy porównawczej można stwierdzić, że w odniesieniu do wskaźników jakości sieci, R, który odnosi się do korelacji między wartościami przewidywanymi a rzeczywistymi na całym zbiorze danych, wszystkie trzy algorytmy osiągnęły podobne wyniki, uzyskując nieco wyższą wartość z LM (0,9869) w porównaniu z SCG (0,9749) i BR (0,9723). To wskazuje, że LM może być nieco lepszy w dopasowaniu danych.

Analiza MSE oraz RMSE również pokazuje różnice między algorytmami. Najniższy MSE (0,00008) oraz RMSE (0,009) osiągnął LM, co sugeruje, że ten algorytm był najdokładniejszy. SCG i BR miały podobne, ale wyższe wartości MSE i RMSE, co może wskazywać na nieco gorszą dokładność tych metod w porównaniu z LM. Dalsze wskaźniki, takie jak RIE, MAPE, i MAE, które odnoszą się do różnych aspektów błędów przewidywania, także ukazują lepsze wyniki dla LM. SCG i BR osiągnęły podobne, ale ogólnie gorsze wartości dla tych wskaźników.

Ryc. 4.5 ilustruje najlepszą wydajność modeli z różnymi algorytmami optymalizacyjnymi: a) LM, b) SCG, c) BR. Wykresy te dostarczają informacji o dynamice uczenia się modeli. Każdy z algorytmów prezentuje różne charakterystyki zbieżności i wydajności, co może być użyteczne przy wyborze odpowiedniego modelu do konkretnego rodzaju danych i problemu, który ma zostać rozwiązany. Algorytm LM osiąga najlepszą wydajność walidacyjną już w 3. epoce. Wydajność na danych uczących i walidacyjnych dla algorytmu SCG wykazuje dużą stabilność po osiągnięciu minimum, co sugeruje dobre dopasowanie modelu do danych. Wykres c) ilustruje działanie algorytmu BR, który stopniowo poprawia swoją wydajność, osiągając najlepszy wynik w 54. epoce. Mimo większej liczby epok potrzebnych do osiągnięcia tego wyniku, wydaje się, że ten algorytm zapewnia bardzo stabilną poprawę wydajności modelu w czasie.

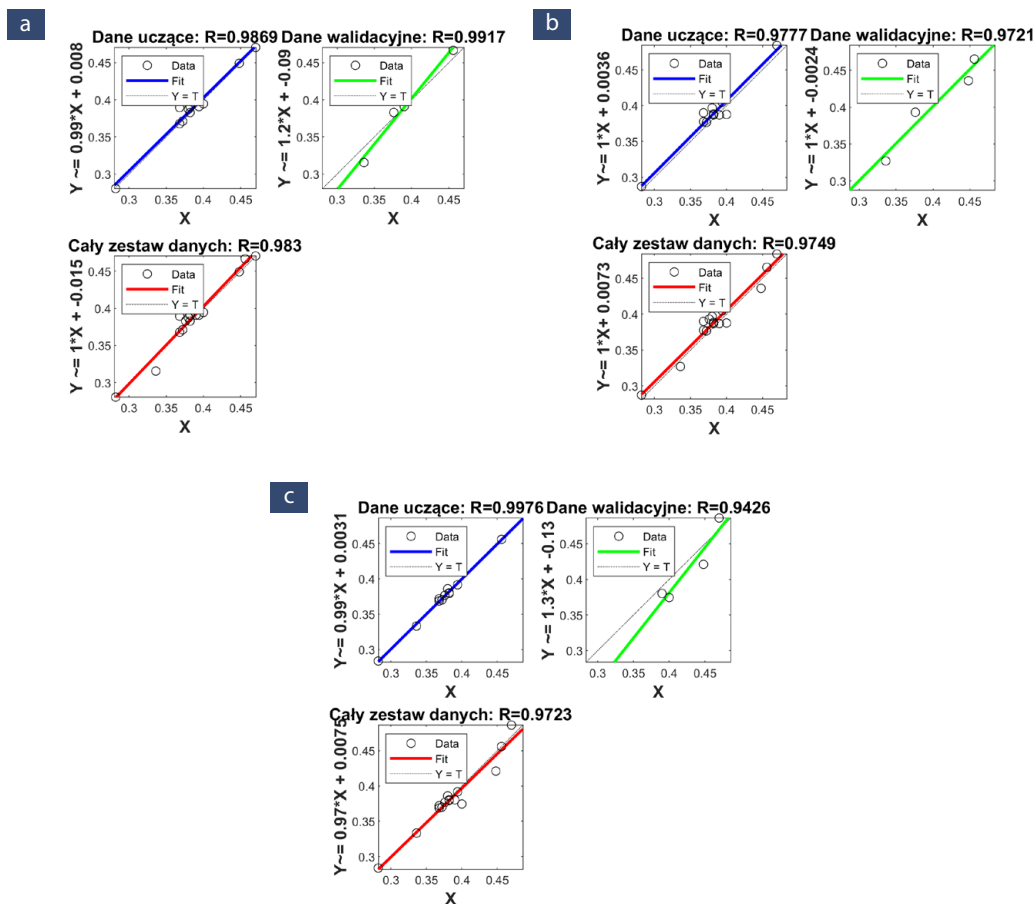


c Najlepsza wydajność modelu to 7.037e-06 w epoce 54



Ryc. 4.5. Najlepsza wydajność modelu Ra dla sieci MLP z algorytmem uczenia:

a) LM, b) SCG, c) BR



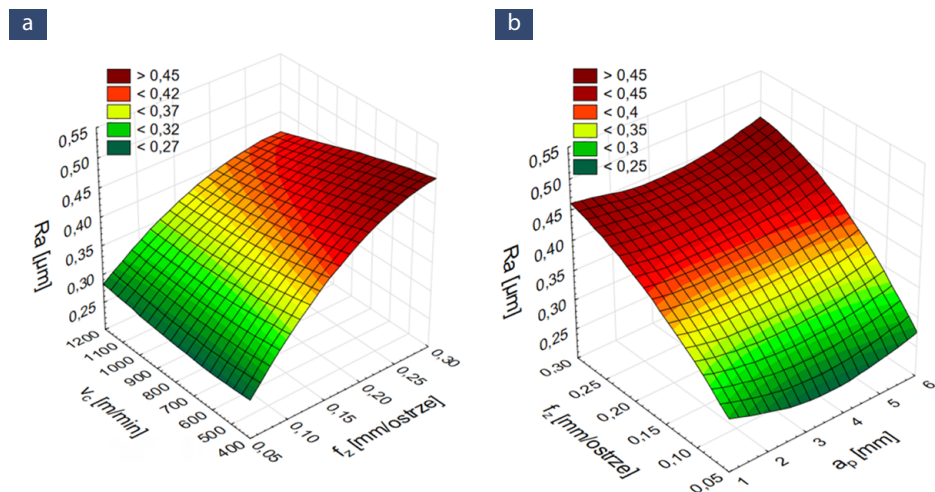
Ryc. 4.6. Wykresy korelacji dla modelu Ra dla sieci MLP z algorytmem uczenia:

a) LM, b) SCG, c) BR

Na ryc. 4.6 przedstawiono wykresy korelacji dla zestawów danych uczących, walidacyjnych i całych zestawów danych dla modeli z różnymi algorytmami uczenia: a) LM, b) SCG, c) BR.

Podsumowując, analiza porównawcza wyników dla modelowania parametru chropowatości Ra wskazuje, że algorytm Levenberga–Marquardta wydaje się najbardziej efektywny pod względem szybkości uczenia, dokładności oraz ogólnej jakości wyników w porównaniu z SCG oraz BR.

Na ryc. 4.7 przedstawiono dane dotyczące chropowatości powierzchni uzyskane z modelu MLP z wykorzystaniem algorytmu LM. Ze względu na fakt, że w badaniu uwzględniono trzy parametry wejściowe, zmiany wartości parametru chropowatości Ra przedstawiono dwukrotnie pod wpływem pary parametrów skrawania: v_c/f_z i f_z/a_p . Symulacje zostały uruchomione po wprowadzeniu danych wejściowych (tj. parametrów frezowania v_c , a_p , f_z) do oprogramowania Matlab i otrzymano wyjście z modelu w postaci parametru Ra .



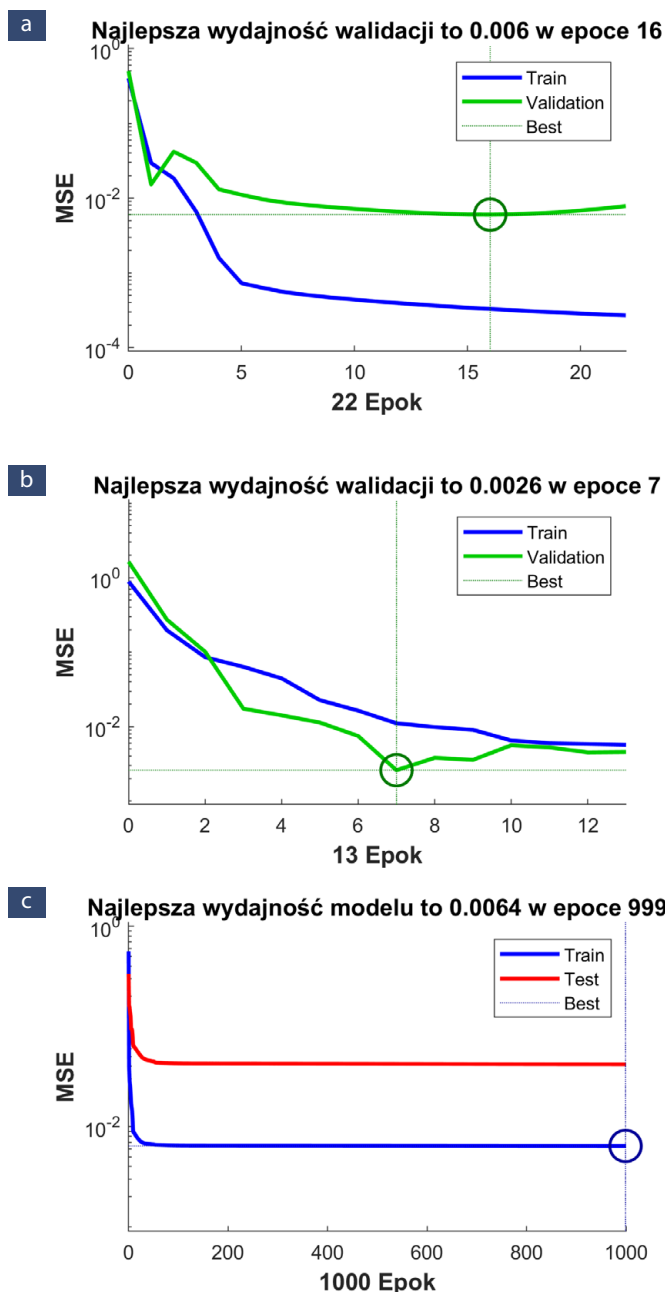
Ryc. 4.7. Parametr chropowatości Ra uzyskany z modelu dla sieci MLP w zależności od v_c/f_z i f_z/a_p

W przypadku kolejnego modelowanego parametru chropowatości Rz najlepszą wydajność i jakość sieci MLP z wykorzystaniem algorytmu LM otrzymano dla dziewięciu neuronów w warstwie ukrytej. Model ten otrzymano w 22 epokach. Następnie dla takiej samej liczby neuronów w warstwie ukrytej trenowano sieci, wykorzystując dwa pozostałe algorytmy: SCG oraz BR. Wskaźniki jakościowe najlepszych sieci otrzymanych podczas trenowania z różnymi algorytmami uczenia przedstawia tabela 4.3.

Tabela 4.3. Wskaźniki jakości dla różnych algorytmów uczenia sieci MLP dla parametru Rz

Algorytm uczenia sieci MLP	LM	SCG	BR
Epoka	22	13	1000
R (cały zbiór danych)	0,9894	0,9353	0,9119
MSE	0,0017	0,0089	0,0153
RMSE	0,0418	0,0946	0,1237
RIE	0,0165	0,0373	0,0487
MAPE	0,0090	0,0324	0,0404
MAE	0,0235	0,0792	0,0950

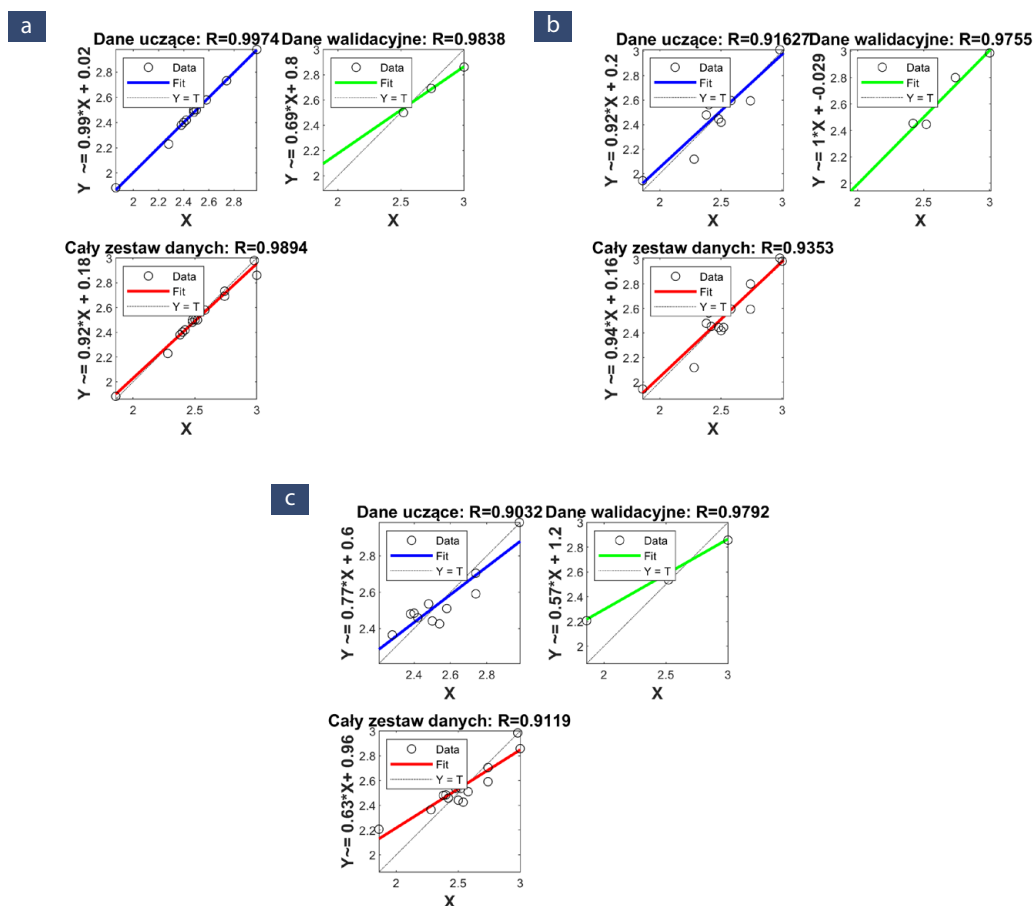
W wyniku przeprowadzenia analizy porównawczej można stwierdzić, że algorytm LM potrzebował zaledwie 22 epok, aby osiągnąć wysoką wartość współczynnika korelacji na poziomie 0,9894, co sugeruje bardzo dobrą zdolność modelu do wyjaśniania zmienności danych. Dodatkowo najniższe wartości błędów MSE, RMSE, MAPE oraz MAE świadczą o wysokiej dokładności predykcji tego modelu. Wyniki te wskazują na efektywność algorytmu LM w szybkim osiągnięciu wiarygodnych wyników z minimalną liczbą obliczeń. Algorytm SCG, choć potrzebował mniej epok (13), wykazuje nieco gorsze wskaźniki jakości w porównaniu z LM. Wartość R wynosi 0,9353, co jest nadal dobrym wynikiem, ale wskazuje na mniejszą zdolność do adekwatnego modelowania danych w porównaniu z LM. Wartości błędów są wyższe niż w przypadku LM, co potwierdza mniejszą precyzję tego algorytmu w modelowaniu danych. Ostatni algorytm BR wymagał znacznie więcej epok (1000), co może wskazywać na trudności w osiągnięciu optymalnej zbieżności. Jego wartość R to 0,9119, co jest najniższym wynikiem wśród omawianych modeli, a wysokie wartości błędów wskazują na znacznie gorszą dokładność predykcji w porównaniu z pozostałymi algorytmami.



Ryc. 4.8. Najlepsza wydajność modelu Rz dla sieci MLP z algorytmem uczenia: a) LM, b) SCG, c) BR

Na ryc. 4.8 można zauważyć trzy wykresy, które przedstawiają proces uczenia sieci neuronowych za pomocą analizowanych algorytmów: LM, SCG i BR. Każdy wykres ilustruje zmiany w wartości błędu średniokwadratowego (MSE) na przestrzeni kolejnych epok dla zestawów danych uczących i walidacyjnych, a także wskazuje najlepsze wyniki uzyskane w procesie

walidacji. Dla algorytmu LM szybkie zmniejszenie wartości MSE w początkowej fazie uczenia sugeruje efektywną zbieżność algorytmu. Najlepsza wydajność modelu na danych walidacyjnych, z wartością MSE równą 0,006, została osiągnięta w 16. epoce, co pokazuje, że model szybko osiągnął optymalne dopasowanie. Algorytm SCG wykazuje bardzo dobrą wydajność, z niskim błędem MSE na poziomie 0,0026, który osiągnięty został już w 7. epoce. Stabilność wartości MSE na niskim poziomie przez resztę procesu uczenia wskazuje na skuteczność tego algorytmu w kontekście dopasowania i generalizacji modelu. Z kolei proces uczenia przy użyciu algorytmu BR jest znacznie dłuższy, rozciąga się aż do 1000 epok, z najlepszym wynikiem MSE osiągniętym na poziomie 0,0064. Długi czas potrzebny do osiągnięcia optymalnej wartości MSE wskazuje na wolniejszą zbieżność tego algorytmu w porównaniu z pozostałymi.

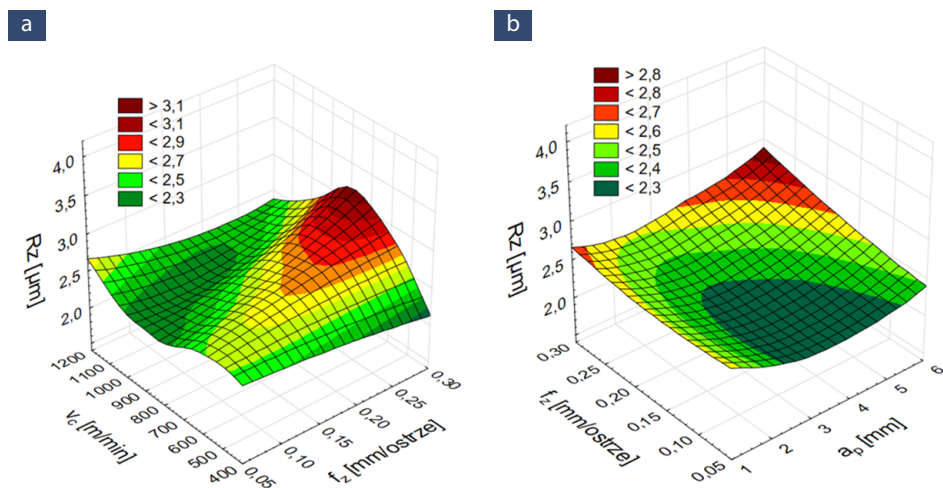


Ryc. 4.9. Wykresy korelacji dla modelu Rz dla sieci MLP z algorytmem uczenia: a) LM, b) SCG, c) BR

Ryc. 4.9 ilustruje wykresy korelacyjne dla różnych zestawów danych – uczących, walidacyjnych oraz całego zestawu danych – stosowanych w modelach z użyciem różnych algorytmów uczenia: a) LM, b) SCG, c) BR.

Podsumowując, algorytm LM okazał się najbardziej efektywny zarówno pod względem szybkości osiągania zbieżności, jak i dokładności modelowania. Algorytm SCG również prezentuje dobre wyniki, lecz nie dorównuje algorytmowi LM. BR, mimo że jest w stanie osiągnąć zbieżność, wymaga znacznie większego nakładu obliczeniowego i czasowego, a jego wyniki są mniej precyzyjne.

Na ryc. 4.10 przedstawiono wyniki dotyczące parametru chropowatości powierzchni R_z uzyskane z modelu MLP korzystającego z algorytmu LM. W analizie uwzględniono trzy zmienne wejściowe, a zmiany w wartościach parametru chropowatości R_z zaprezentowano w kontekście dwóch kombinacji parametrów skrawania: v_c/f_z i f_z/a_p . Symulacje przeprowadzono po wprowadzeniu danych wejściowych – parametrów frezowania v_c , a_p , f_z – do oprogramowania Matlab, skąd uzyskano wyniki w postaci wartości parametru R_z .



Ryc. 4.10. Parametr chropowatości R_z uzyskany z modelu dla sieci MLP w zależności od v_c/f_z i f_z/a_p

W przypadku modelowania ostatniego z wybranych parametrów chropowatości RSm najwyższą skuteczność oraz jakość sieci MLP uzyskano przy użyciu algorytmu LM, stosując dziesięć neuronów w warstwie ukrytej. Optymalny model wygenerowano po zaledwie 10 epokach. Ustalając tę samą liczbę neuronów, kontynuowano następnie uczenie sieci za pomocą dwóch pozostałych algorytmów: SCG oraz BR.

Wyniki jakości najlepszych modeli, które uzyskano w wyniku tych procesów uczenia, prezentuje tabela 4.4. Porównując otrzymane wyniki, można zauważyć istotne różnice w wydajności każdego z algorytmów. Algorytm LM, który osiągnął optymalizację modelu już po 10 epokach, wykazuje bardzo dobre wyniki, z wartością współczynnika korelacji wynoszącą 0,911, co wskazuje na wysoką zdolność modelu do wyjaśniania zmienności danych. Dodatkowo wartość MSE oraz pozostałych błędów jest najniższa spośród porównywanych algorytmów, co świadczy o bardzo dokładnym dopasowaniu modelu do danych. Algorytm SCG, który potrzebował 12 epok do osiągnięcia optymalizacji, wykazuje niższe wartości R na poziomie 0,752. Jest to znacznie słabsza zdolność do modelowania danych w porównaniu z LM, co sugeruje, że może nie być on tak efektywny w przypadku tego konkretnego zestawu danych. W przypadku algorytmu BR, który wymagał największej liczby epok (55), do osiągnięcia najlepszego dopasowania, wartości R są niższe niż dla LM, wynoszące 0,7743. Wskazuje to na umiarkowaną zdolność modelu do przewidywania zmienności danych.

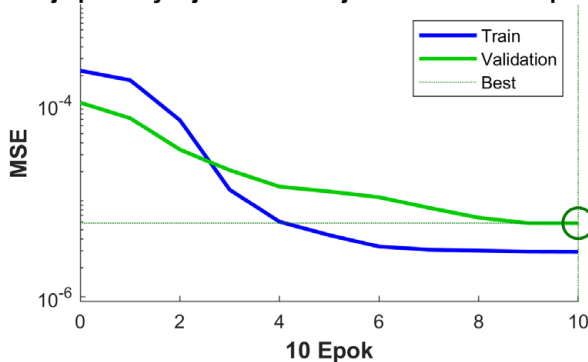
Tabela 4.4. Wskaźniki jakości dla różnych algorytmów uczenia sieci MLP dla parametru R_{Sm}

Algorytm uczenia sieci MLP	LM	SCG	BR
Epoka	10	12	55
R (cały zbiór danych)	0,911	0,752	0,7743
MSE	0,000004	0,000009	0,000014
RMSE	0,0019	0,0030	0,0038
RIE	0,0493	0,0780	0,0971
MAPE	0,0256	0,0673	0,0492
MAE	0,0010	0,0025	0,0017

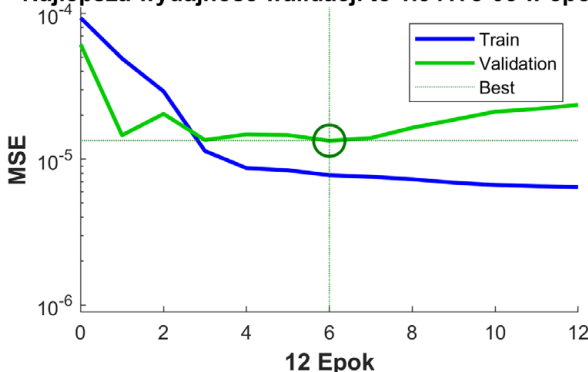
Na ryc. 4.11 przedstawiono trzy wykresy zmian wartości MSE podczas procesu uczenia sieci neuronowych z wykorzystaniem trzech algorytmów: a) LM, b) SCG, c) BR, w kontekście modelowania parametru chropowatości R_{Sm} . Na wykresie a) dla algorytmu LM można zaobserwować szybki spadek wartości MSE, który osiąga swoje minimum już w 10. epoce. MSE na danych uczących kontynuuje delikatne obniżanie się, co wskazuje na efektywność i stabilność tego algorytmu w szybkim osiąganiu optymalnych wyników. Wykres b) ilustrujący algorytm SCG pokazuje, że najlepszy wynik walidacji zostaje osiągnięty w 6. epoce. Wartości MSE stabilizują się na stosunkowo niskim poziomie zarówno dla danych uczących, jak i walidacyjnych, co świadczy o dobrej generalizacji modelu, chociaż nie osiąga on tak niskich wartości MSE,

jak algorytm LM. Na wykresie c), gdzie zastosowano algorytm BR, zaobserwować można dłuższy proces uczenia, który trwa do 55 epok. Najlepszy wynik uczenia sieci osiągnięty zostaje pod koniec tego okresu.

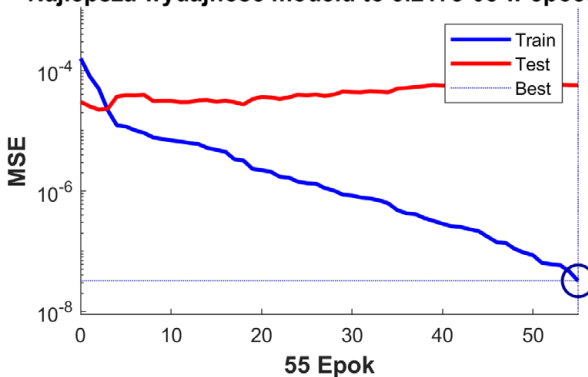
a Najlepsza wydajność walidacji to $5.8433\text{e-}06$ w epoce 10



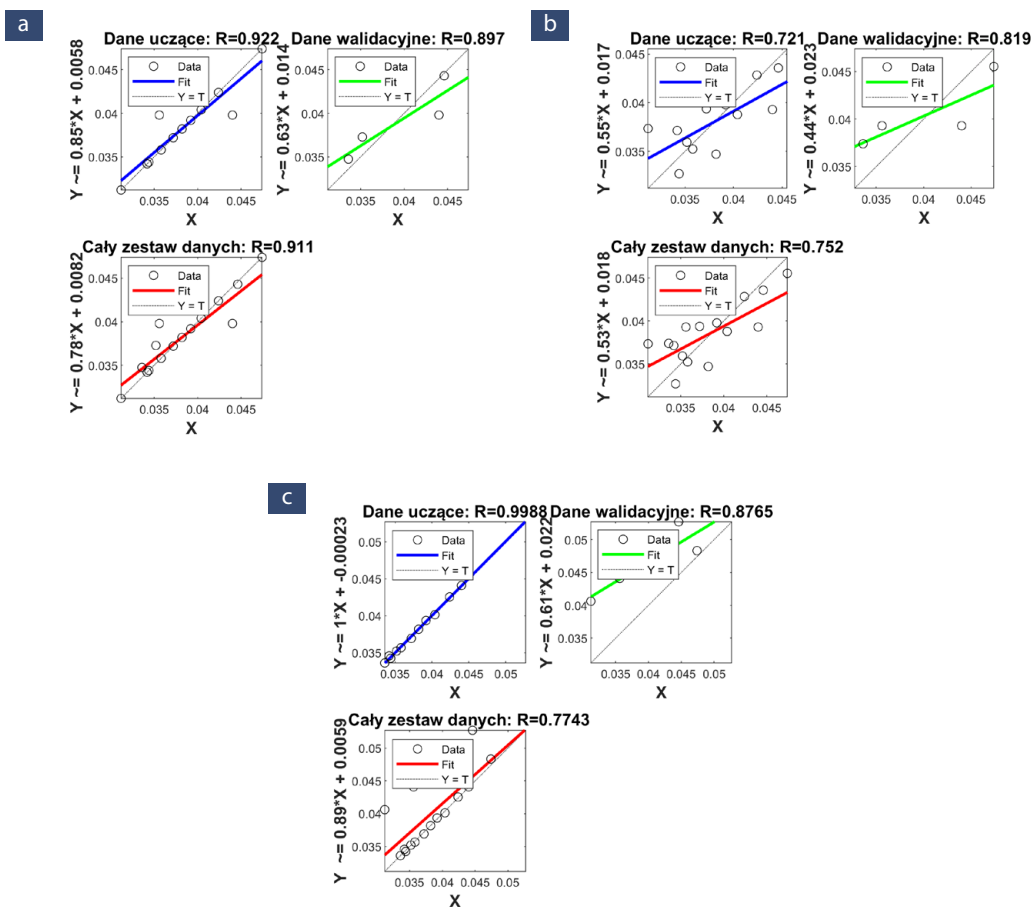
b Najlepsza wydajność walidacji to $1.3417\text{e-}05$ w epoce 6



c Najlepsza wydajność modelu to $3.217\text{e-}08$ w epoce 55



Ryc. 4.11. Najlepsza wydajność modelu RSm dla sieci MLP z algorytmem uczenia:
a) LM, b) SCG, c) BR

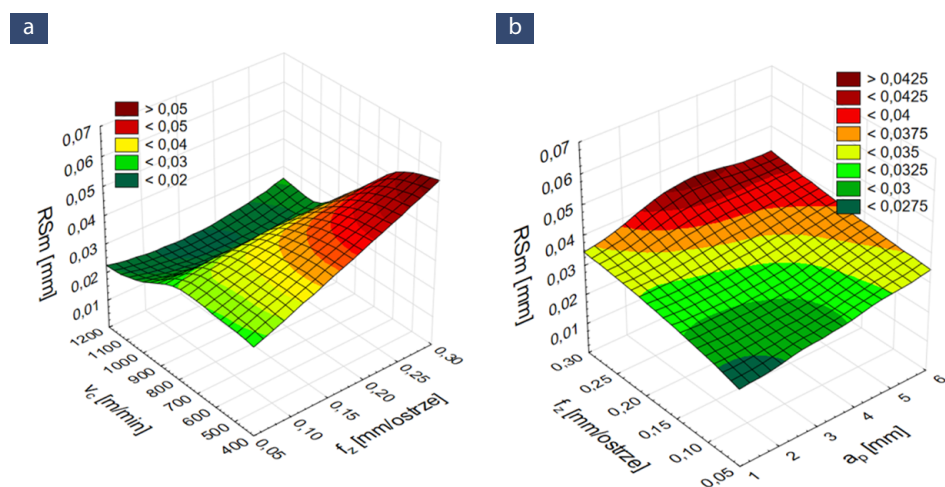


Ryc. 4.12. Wykresy korelacji dla modelu RSm dla sieci MLP z algorytmem uczenia:
a) LM, b) SCG, c) BR

Na ryc. 4.12 przedstawiono wykresy korelacyjne dla modelu RSm wykorzystujące trzy różne algorytmy uczenia: a) LM, b) SCG, c) BR. Wykresy te ilustrują zależność między rzeczywistymi wartościami danych (oznaczone jako „Data”) a wartościami przewidzianymi przez model (oznaczone jako „Fit”), a także pokazują współczynniki korelacji R dla różnych zbiorów danych: uczących, walidacyjnych oraz całego zestawu danych. Dla algorytmu LM wartości przewidziane są bardzo zbliżone do rzeczywistych. Wysokie współczynniki korelacji dla danych uczących (0,922) i walidacyjnych (0,897), a także dla całego zbioru danych (0,911) wskazują na dobrą zdolność modelu do przewidywania i generalizacji. Algorytm SCG prezentuje się nieco inaczej. Mimo że współczynniki dla danych uczących (0,721) i walidacyjnych (0,819) są niższe niż w LM, model utrzymuje umiarkowaną zdolność do generalizacji, jak to widać w wyniku $R=0,752$ dla całego zbioru danych. Algorytm BR prezentuje słabsze

wyniki od pozostałych modeli. Mimo że współczynnik korelacji R dla danych uczących osiąga wysoką wartość 0,9988, co wskazuje na doskonałe dopasowanie modelu do danych uczących, to jednak wartość dla danych walidacyjnych jest dużo niższa (0,8765). Może to sugerować, że model jest nieco przeuczony w stosunku do danych uczących, co wpływa na jego generalizację na danych walidacyjnych z wynikiem 0,7743 dla całego zbioru danych. Podsumowując, algorytm LM okazał się najbardziej efektywny zarówno pod względem szybkości osiągnięcia zbieżności, jak i dokładności modelowania.

Na ryc. 4.13 zaprezentowano wyniki dotyczące parametru chropowatości powierzchni RSm , które zostały uzyskane za pomocą modelu wykorzystującego algorytm LM. Analiza opierała się na trzech zmiennych wejściowych, a wynikowe zmiany wartości parametru chropowatości RSm zostały przedstawione w odniesieniu do dwóch kombinacji parametrów skrawania: v_c/f_z i f_z/a_p .



Ryc. 4.13. Parametr chropowatości RSm uzyskany z modelu dla sieci MLP w zależności od v_c/f_z i f_z/a_p .

Podsumowując, analiza porównawcza różnych algorytmów uczenia sieci neuronowych wykazała, że algorytm LM osiąga najlepsze wyniki pod względem szybkości uczenia, dokładności przewidywań oraz ogólnej jakości modeli dla wszystkich analizowanych parametrów chropowatości Ra , Rz i RSm . Algorytmy SCG i BR, mimo pewnych zalet, nie dorównują efektywności algorytmowi LM, co czyni go najskuteczniejszym narzędziem do modelowania chropowatości powierzchni w przedstawionych badaniach.

Następnie przeprowadzono modelowanie parametrów chropowatości Ra , Rz i RSm z wykorzystaniem GBM. Dla każdego z parametrów przetestowano liczbę drzew od 50 do 1000. Otrzymane wyniki przedstawia tabela 4.5.

Tabela 4.5. Wskaźniki jakości dla modelowania parametrów chropowatości R_a , R_z i RSm z wykorzystaniem GBM

Parametr chropowatości	R_a	R_z	RSm
Liczba drzew	500	50	500
MSE	0,0019	0,0266	0,000031
RMSE	0,0445	0,1633	0,0055
RIE	0,1098	0,0648	0,1352
MAPE	0,0497	0,0502	0,1323
MAE	0,0231	0,125	0,0054

W przypadku modelowania parametrów chropowatości z wykorzystaniem GBM najlepsze wyniki uzyskano przy liczbie drzew odpowiednio 500 dla R_a , 50 dla R_z oraz 500 dla RSm . Modele GBM również wykazały wysoką dokładność predykcji, jednakże ich jakość jest nieco gorsza w porównaniu z najlepszymi wynikami uzyskanymi przez MLP z algorytmem LM. Wartości MSE, RMSE, MAPE i MAE są wyższe dla GBM niż dla LM.

Podsumowując, algorytm LM w sieciach neuronowych MLP okazał się najskuteczniejszym narzędziem do modelowania parametrów chropowatości powierzchni 2D, oferując najlepszą kombinację szybkości uczenia, dokładności oraz ogólnej jakości modeli. GBM, choć również efektywny, nie osiągnął tak wysokiej dokładności, jak LM, co sugeruje, że MLP z algorytmem LM jest bardziej odpowiednie do modelowania tego typu danych. W związku z powyższym modele otrzymane z wykorzystaniem algorytmu LM zostaną wykorzystane w dalszej analizie optymalizacyjnej.

4.4.3. Optymalizacja parametrów obróbki

Kolejnym etapem prowadzonych badań był proces optymalizacji parametrów obróbki v_c , f_z i a_p , mający na celu uzyskanie określonych wartości parametrów chropowatości powierzchni: R_a , R_z oraz RSm . Optymalizacja została przeprowadzona przy użyciu najlepszych modeli sieci neuronowych opracowanych na wcześniejszym etapie badań dla każdego z tych parametrów, które zostały zaprezentowane w podrozdziale 4.4.2.

Aby zoptymalizować parametry obróbki, najpierw określono zakresy oraz kroki zmian dla trzech głównych parametrów wejściowych: prędkości skrawania (v_c), posuwu na ostrze (f_z) oraz głębokości skrawania (a_p). Zakresy te zostały dobrane na podstawie praktycznych wartości stosowanych w procesie obróbki,

czyli w zakresach takich jak podczas prowadzenia badań doświadczalnych ($v_c = 400 \div 1200$ [m/min], $f_z = 0,05 \div 0,3$ [mm/ostre] oraz $a_p = 0,5 \div 6$ [mm]). Założono, że prędkość skrawania może się zmieniać co 50 [m/min], posuw na ostre co 0,05 [mm/ostre], zaś głębokość skrawania co 0,5 [mm].

Następnie zdefiniowano zakresy docelowe dla parametrów wyjściowych, które założono do uzyskania w wyniku optymalizacji. Zakresy te odpowiadają minimalnym i maksymalnym wartościom Ra , Rz oraz RSm , które są akceptowalne pod względem jakości powierzchni po obróbce.

W procesie optymalizacji przestrzeń parametrów została przeszukana w celu znalezienia takich wartości v_c , f_z i a_p , które minimalizują błąd w stosunku do zadanych zakresów Ra , Rz i RSm . Dla każdego parametru chropowatości zdefiniowano funkcję błędu w stosunku do zakresu docelowego:

$$E_{Ra}(v_c, f_z, a_p) = \begin{cases} RaTarget(1) - netRa(v_c, f_z, a_p) & \text{jeżeli } netRa(v_c, f_z, a_p) < RaTarget(1) \\ netRa(v_c, f_z, a_p) - RaTarget(2) & \text{jeżeli } netRa(v_c, f_z, a_p) > RaTarget(2) \\ 0 & \text{w innym przypadku} \end{cases}$$

$$E_{Rz}(v_c, f_z, a_p) = \begin{cases} RzTarget(1) - netRz(v_c, f_z, a_p), & \text{jeżeli } netRz(v_c, f_z, a_p) < RzTarget(1) \\ netRz(v_c, f_z, a_p) - RzTarget(2), & \text{jeżeli } netRz(v_c, f_z, a_p) > RzTarget(2) \\ 0, & \text{w innym przypadku} \end{cases} \quad (4.5)$$

$$E_{RSm}(v_c, f_z, a_p) = \begin{cases} RSmTarget(1) - netRSm(v_c, f_z, a_p), & \text{jeżeli } netRSm(v_c, f_z, a_p) < RSmTarget(1) \\ netRSm(v_c, f_z, a_p) - RSmTarget(2), & \text{jeżeli } netRSm(v_c, f_z, a_p) > RSmTarget(2) \\ 0, & \text{w innym przypadku} \end{cases}$$

gdzie:

$E_{Ra}(v_c, f_z, a_p)$, $E_{Rz}(v_c, f_z, a_p)$, $E_{RSm}(v_c, f_z, a_p)$ – funkcje błędu dla poszczególnych parametrów chropowatości Ra , Rz i RSm ;

$RaTarget(1)$, $RzTarget(1)$, $RSmTarget(1)$ – dolne granice zakresów docelowych parametrów chropowatości Ra , Rz i RSm ;

$RaTarget(2)$, $RzTarget(2)$, $RSmTarget(2)$ – górne granice zakresów docelowych parametrów chropowatości Ra , Rz i RSm ;

$netRa(v_c, f_z, a_p)$, $netRz(v_c, f_z, a_p)$, $netRSm(v_c, f_z, a_p)$ – wyliczona wartość parametrów chropowatości przez sieć neuronową dla zadanych wartości v_c , f_z , a_p odpowiednio dla parametru Ra , Rz i RSm .

Łączny błąd $E_{suma}(v_c, f_z, a_p)$ jest sumą poszczególnych błędów:

$$E_{suma}(v_c, f_z, a_p) = E_{Ra}(v_c, f_z, a_p) + E_{Rz}(v_c, f_z, a_p) + E_{RSm}(v_c, f_z, a_p). \quad (4.6)$$

Funkcja optymalizacyjna może być zapisana jako:

$$\left(v_c^*, f_z^*, a_p^*\right) = \arg \min_{v_c, f_z, a_p} E_{\text{suma}}\left(v_c, f_z, a_p\right) \quad (4.7)$$

Opracowany algorytm 4.1 przeszukiwał wszystkie kombinacje zadanych wartości wejściowych, a następnie obliczał wartość wyjściową przy użyciu wcześniej wytrenowanych sieci neuronowych.

Algorytm 4.1. Optymalizacja parametrów obróbki

```
% Inicjalizacja zmiennych do przechowywania najlepszych wyników
bestParams = [];
bestError = inf;
% Przeszukiwanie przestrzeni parametrów
for vc = vcValues
    for fz = fzValues
        for ap = apValues
            % Aktualne wartości parametrów wejściowych
            params = [vc, fz, ap];
            % Wyznaczanie wartości wyjściowych z sieci neuronowych
            Ra = netRa(params');
            Rz = netRz(params');
            RSm = netRSm(params');
            % Obliczanie błędu
            RaError = calculateError(Ra, RaTarget);
            RzError = calculateError(Rz, RzTarget);
            RSmError = calculateError(RSm, RSmTarget);
            totalError = RaError + RzError + RSmError;
            % Aktualizacja najlepszych wyników
            if totalError < bestError
                bestError = totalError;
                bestParams = params;
                bestRa = Ra;
                bestRz = Rz;
                bestRSm = RSm;
            end
        end
    end
end
```

Algorytm 4.1. Optymalizacja parametrów obróbki, cd.

```
% Tworzenie komunikatu wyników
resultMessage = sprintf('Optymalne parametry:\nvc = %.2f\nfz = %.2f\n\nap = %.2f\nOdchylenie od założonego zakresu = %.4f\n\nOtrzymane wartości:\nRa = %.4f\nRz = %.4f\nRSm = %.4f\n', ...
    bestParams(1), bestParams(2), bestParams(3), bestError, bestRa,
    bestRz, bestRSm);
% Wyświetlanie wyników w osobnym oknie dialogowym
msgbox(resultMessage, 'Wyniki optymalizacji');
% Definicja funkcji do obliczania błędu
function error = calculateError(value, targetRange)
    if value < targetRange(1)
        error = targetRange(1) - value;
    elseif value > targetRange(2)
        error = value - targetRange(2);
    else
        error = 0; % Wartość jest w zakresie docelowym
    end
end
```

Aby korzystanie z algorytmu optymalizacyjnego było bardziej przyjazne i intuicyjne dla użytkownika, stworzono interfejs graficzny, który pozwala na łatwe wprowadzanie danych wejściowych oraz szybkie uzyskiwanie wyników optymalizacji. Dzięki temu użytkownicy mogą skupić się na analizie wyników bez konieczności zagłębiania się w szczegóły kodu źródłowego.

Interfejs graficzny został zaprojektowany tak, aby był prosty i intuicyjny w obsłudze (ryc. 4.20). W górnej części interfejsu znajduje się nagłówek „Optymalizacja parametrów chropowatości”, który jasno informuje użytkownika o celu aplikacji. Centralną część stanowią pola tekstowe, w których użytkownik może wprowadzić zakresy docelowe dla trzech parametrów chropowatości: Ra [μm], Rz [μm] oraz RSm [mm]. Każdy z tych parametrów ma osobne pole tekstowe umożliwiające wprowadzenie wartości minimalnych i maksymalnych.

Pod polami wprowadzania znajduje się przycisk „Optymalizuj”. Po jego kliknięciu uruchamiany jest algorytm optymalizacyjny, który przeszukuje przestrzeń parametrów, aby znaleźć optymalne wartości. W dolnej części interfejsu wyświetlane będą wyniki przeprowadzonej optymalizacji. W tej sekcji użytkownik może zobaczyć optymalne parametry, czyli najlepsze wartości prędkości skrawania (v_c), posuwu na ostrze (f_z) oraz głębokości skrawania (a_p) uzyskane

w wyniku optymalizacji. Dodatkowo wyświetlana będzie wartość błędu, informująca o tym, jak dobrze uzyskane wyniki mieszczą się w zadanych zakresach docelowych. Ostatecznie prezentowane będą również przewidywane wartości parametrów R_a , R_z i R_{Sm} , które zostały obliczone na podstawie optymalnych parametrów wejściowych.

Ryc. 4.14. Interfejs użytkownika aplikacji do optymalizacji parametrów chropowości

Przedstawiony poniżej kod ilustruje sposób implementacji opisanego interfejsu graficznego (algorytm 4.2).

Algorytm 4.2. Implementacja interfejsu graficznego do optymalizacji parametrów chropowości

```
function Optymalizacja_Chropowatosci()
    % Ładowanie wytrenowanych sieci
    load('netRa.mat', 'netRa'); % Sieć Ra
    load('netRz.mat', 'netRz'); % Sieć Rz
    load('netRSm.mat', 'netRSm'); % Sieć RSm
    % Tworzenie interfejsu graficznego
    fig = figure('Position', [100, 100, 600, 600], 'Name',
'Optymalizacja Parametrów Chropowości', 'Color', [1, 0.85, 0.8]);
    % Nagłówek
    uicontrol('Style', 'text', 'Position', [100, 550, 450, 40],
'String', 'Optymalizacja parametrów chropowości', 'FontSize',
16, 'FontWeight', 'bold', 'BackgroundColor', [1, 0.85, 0.8],
'HorizontalAlignment', 'center');
```

Algorytm 4.2. Implementacja interfejsu graficznego do optymalizacji parametrów chropowatości, cd.

```
% Ramka wokół pól wprowadzania danych
uicontrol('Style', 'frame', 'Position', [25, 320, 550, 220],
'BackgroundColor', [0.9, 0.9, 0.9]);
% Pola wprowadzania zakresów docelowych
uicontrol('Style', 'text', 'Position', [50, 500, 200, 20],
'String', 'Ra [μm] (min-max)', 'BackgroundColor', [0.9, 0.9, 0.9],
'FontSize', 13, 'HorizontalAlignment', 'left');
raMinInput = uicontrol('Style', 'edit', 'Position', [300, 500, 80,
20], 'FontSize', 12);
raMaxInput = uicontrol('Style', 'edit', 'Position', [400, 500, 80,
20], 'FontSize', 12);
uicontrol('Style', 'text', 'Position', [50, 450, 200, 20],
'String', 'Rz [μm] (min-max)', 'BackgroundColor', [0.9, 0.9, 0.9],
'FontSize', 13, 'HorizontalAlignment', 'left');
rzMinInput = uicontrol('Style', 'edit', 'Position', [300, 450, 80,
20], 'FontSize', 12);
rzMaxInput = uicontrol('Style', 'edit', 'Position', [400, 450, 80,
20], 'FontSize', 12);
uicontrol('Style', 'text', 'Position', [50, 400, 200, 20],
'String', 'RSm [mm] (min-max)', 'BackgroundColor', [0.9, 0.9, 0.9],
'FontSize', 13, 'HorizontalAlignment', 'left');
rsmMinInput = uicontrol('Style', 'edit', 'Position', [300, 400,
80, 20], 'FontSize', 12);
rsmMaxInput = uicontrol('Style', 'edit', 'Position', [400, 400,
80, 20], 'FontSize', 12);
% Przycisk do optymalizacji
optimizeButton = uicontrol('Style', 'pushbutton', 'Position',
[250, 330, 100, 40], 'String', 'Optymalizuj', 'BackgroundColor',
[0.2, 0.6, 0.8], 'ForegroundColor', 'white', 'FontWeight', 'bold',
'FontSize', 12, 'Callback', @optimizeCallback);
% Pole do wyświetlania wyników
resultText = uicontrol('Style', 'text', 'Position', [50, 50, 500,
250], 'String', '', 'FontSize', 14, 'BackgroundColor', [1, 0.85,
0.8], 'HorizontalAlignment', 'left');
% Callback do przycisku optymalizacji
function optimizeCallback(~, ~)
% Pobieranie danych wejściowych
RaTarget = [str2double(get(raMinInput, 'String')),
str2double(get(raMaxInput, 'String'))];
RzTarget = [str2double(get(rzMinInput, 'String')),
str2double(get(rzMaxInput, 'String'))];
```

Algorytm 4.2. Implementacja interfejsu graficznego do optymalizacji parametrów chropowatości, cd.

```
RSmTarget = [str2double(get(rsmMinInput, 'String')),
str2double(get(rsmMaxInput, 'String'))];
% Sprawdzenie poprawności danych wejściowych
if any(isnan([RaTarget, RzTarget, RSmTarget]))
    set(resultText, 'String', 'Błąd: Proszę wprowadzić poprawne
dane wejściowe.');
```

```
    return;
end
% Definicja zakresów przeszukiwania
vcValues = 400:50:1200;
fzValues = 0.05:0.05:0.3;
apValues = 0.5:0.5:6;
% Inicjalizacja zmiennych do przechowywania najlepszych wyników
bestParams = [];
bestError = inf;
bestRa = NaN;
bestRz = NaN;
bestRSm = NaN;
% Przeszukiwanie przestrzeni parametrów
for vc = vcValues
    for fz = fzValues
        for ap = apValues
            % Aktualne wartości parametrów wejściowych
            params = [vc, fz, ap];
% Wyznaczanie wartości wyjściowych z sieci neuronowych
            Ra = netRa(params');
            Rz = netRz(params');
            RSm = netRSm(params');

            % Obliczanie błędu
            RaError = calculateError(Ra, RaTarget);
            RzError = calculateError(Rz, RzTarget);
            RSmError = calculateError(RSm, RSmTarget);
            totalError = RaError + RzError + RSmError;
            % Aktualizacja najlepszych wyników
            if totalError < bestError
                bestError = totalError;
                bestParams = params;
                bestRa = Ra;
                bestRz = Rz;
                bestRSm = RSm;
```

Algorytm 4.2. Implementacja interfejsu graficznego do optymalizacji parametrów chropowatości, cd.

```
        end
    end
end

end
% Tworzenie komunikatu wyników
resultMessage = sprintf('Optymalne parametry:\nvc = %.2f m/
min\nfz = %.2f mm\ostrze\nap = %.2f mm\nOdchylenie od założonego
zakresu = %.4f\n\nOtrzymane wartości:\nRa = %.4f μm\nRz = %.4f μm\
nRSm = %.4f mm\n', ...
bestParams(1), bestParams(2), bestParams(3), bestError, bestRa,
bestRz, bestRSm);

% Wyświetlanie wyników w interfejsie
set(resultText, 'String', resultMessage);
end
% Definicja funkcji do obliczania błędu
function error = calculateError(value, targetRange)
    if value < targetRange(1)
        error = targetRange(1) - value;
    elseif value > targetRange(2)
        error = value - targetRange(2);
    else
        error = 0; % Wartość jest w zakresie docelowym
    end
end
end
end
```

Interfejs został zaprojektowany z myślą o zapewnieniu przejrzystości oraz łatwości obsługi, co umożliwi szybkie uzyskanie rezultatów i analizę ich zgodności z założonymi wymaganiami. Dzięki graficznemu przedstawieniu procesu optymalizacji użytkownik może w łatwy sposób monitorować wyniki i dokonywać potrzebnych zmian w parametrach wejściowych.

Przykład 1. działania algorytmu dla założonych parametrów obróbki: Ra w zakresie $0,35 \div 0,39$ [μm], Rz w zakresie $2,3 \div 2,5$ [μm] oraz RSm w zakresie $0,04 \div 0,05$ [mm] przedstawia ryc. 4.15. Po weryfikacji otrzymanych wartości parametrów chropowatości można stwierdzić, że zawierają się w założonych granicach.

Optymalizacja parametrów chropowatości

Ra [μm] (min-max)	0.35	0.39
Rz [μm] (min-max)	2.3	2.5
RSm [mm] (min-max)	0.04	0.05

Optymalne parametry:
 $v_c = 850.00 \text{ m/min}$
 $f_z = 0.15 \text{ mm/ostre}$
 $a_p = 6.00 \text{ mm}$
 Odchylenie od założonego zakresu = 0.0000

Otrzymane wartości:
 $Ra = 0.3894 \mu\text{m}$
 $Rz = 2.3551 \mu\text{m}$
 $RSm = 0.0401 \text{ mm}$

Ryc. 4.15. Przykład 1. działania algorytmu optymalizującego

Przykład 2. działania algorytmu dla założonych parametrów obróbki w następujących zakresach: Ra $0,39 \div 0,42 [\mu\text{m}]$, Rz $2,7 \div 2,8 [\mu\text{m}]$ oraz RSm $0,035 \div 0,04 [\text{mm}]$ przedstawia ryc. 4.16. Po weryfikacji otrzymanych wartości parametrów chropowatości można stwierdzić, że mieszczą się w założonych granicach, podobnie jak miało to miejsce w poprzednim przypadku.

Na podstawie przeprowadzonych badań oraz optymalizacji parametrów obróbki można stwierdzić, że zaproponowany algorytm skutecznie znajduje optymalne wartości parametrów wejściowych (v_c, f_z, a_p), które prowadzą do uzyskania założonych wartości parametrów chropowatości powierzchni (Ra, Rz, RSm). Wyniki pokazują, że zarówno w pierwszym, jak i drugim przykładzie otrzymane wartości parametrów chropowatości mieszczą się w zakładanych granicach, potwierdzając efektywność zastosowanego podejścia. Optymalizacja parametrów obróbki przy użyciu sieci neuronowych pozwala na precyzyjne dostosowanie procesu do wymagań jakościowych powierzchni.

Optymalizacja parametrów chropowatości

Ra [μm] (min-max)	0.39	0.42
Rz [μm] (min-max)	2.7	2.8
RSm [mm] (min-max)	0.035	0.04
Optymalizuj		

Optymalne parametry:
 vc = 550.00 m/min
 fz = 0.15 mm/ostre
 ap = 6.00 mm
 Odchylenie od założonego zakresu = 0.0000

Otrzymane wartości:
 Ra = 0.3942 μm
 Rz = 2.7050 μm
 RSm = 0.0374 mm

Ryc. 4.16. Przykład 2. działania algorytmu optymalizującego

4.5. Optymalizacja parametrów obróbki przy ograniczeniach siłowych

Modelowanie sił skrawania odgrywa ważną rolę w procesie wytwarzania elementów maszyn i urządzeń, zwłaszcza przy użyciu nowoczesnych materiałów, takich jak stopy magnezu. W przemyśle lotniczym, gdzie dąży się do zmniejszenia masy elementów konstrukcyjnych, coraz częściej obrabia się części cienkościenne technologią frezowania, w tym frezowania HSM (High-Speed Machining). Siły skrawania, będące jednym z głównych wskaźników skrawalności, mają istotny wpływ na odkształcenia elementów obrabianych, a ich nadmierna wartość może prowadzić do pogorszenia jakości powierzchni oraz zwiększenia drgań w systemie OUPN (Obrabiarka-Uchwyt-Przedmiot-Narzędzie) [170,171].

Modelowanie tych sił jest więc istotne dla optymalizacji procesu obróbki, pozwala na określenie parametrów skrawania, które minimalizują negatywne skutki, takie jak nadmierne odkształcenia czy pogorszenie jakości powierzchni. Tradycyjne metody analityczne często zawodzą z powodu

złożoności i nieliniowości procesów obróbkowych. Stąd zastosowanie sztucznych sieci neuronowych skutkuje stworzeniem dokładniejszych modeli przy czynowo-skutkowych między parametrami skrawania a siłami skrawania.

Jednak samo zamodelowanie składowej czy składowych siły skrawania nie przekłada się na wskazanie dla technologa konkretnych parametrów obróbki, otrzymanego modelu nie można bezpośrednio wykorzystać do doboru optymalnych parametrów obróbki. Zakładając, że proces produkcji będzie przebiegał w warunkach odpowiadających tym, w których stworzono model, konieczne jest wyselekcjonowanie z wielowymiarowej przestrzeni parametrów kontrolnych takiego zestawu, który zapewni osiągnięcie wymagań zapisanych w dokumentacji technicznej produktu. Ponadto nie może to być dowolny zestaw spełniający te wymagania (nieprzekroczenie wartości siły), ale taki, który gwarantuje maksymalną wydajność procesu. W takich przypadkach wykorzystuje się tzw. inżynierię odwrotną, której celem jest znalezienie optymalnej przestrzeni parametrów wejściowych prowadzących do osiągnięcia zamierzonego parametru wyjściowego. Proces ten polega na selekcji zestawu parametrów, który spełnia określone kryteria, takie jak np. wydajność procesu.

Taki model siły skrawania może jednak zostać wykorzystany do stworzenia algorytmu optymalizującego parametry obróbki, którego celem będzie wyznaczenie jednego rozwiązania spełniającego maksymalną wydajność obróbki.

4.5.1. Metodyka modelowania siły skrawania

W ramach badań nad modelowaniem sił skrawania, ze względu na dominujący wpływ składowej siły F_x na odkształcenia przedmiotu obrabianego, przyjęto ten parametr jako wielkość wyjściową w budowanym modelu i tylko tą składową modelowano. Proces modelowania przeprowadzono za pomocą sieci neuronowych MLP w oprogramowaniu Matlab 2023a. W ramach badań nad modelowaniem sił skrawania zdecydowano się na zastosowanie sieci neuronowej MLP ze względu na specyficzne wymagania wynikające z bardzo małego zestawu danych. Liczba próbek (46 obserwacji) wymagała wyboru modelu, który jest wystarczająco złożony, aby uchwycić istotne zależności w danych, ale jednocześnie na tyle prosty, aby uniknąć problemów związanych z nadmiernym dopasowaniem. Z tego powodu zastosowano MLP z jedną warstwą ukrytą, której liczba neuronów była iteracyjnie dostosowywana w celu znalezienia optymalnej konfiguracji, w zakresie od 2 do 12 neuronów.

Warstwa wejściowa modelu składała się z trzech neuronów wejściowych, reprezentujących zmienne parametry obróbki: prędkość skrawania (v_c), posuw na ostrze (f_z) i głębokość skrawania (a_p). Warstwa wyjściowa składała się z jednego neuronu, który modelował składową F_x siły skrawania.

W badaniu zastosowano domyślną metodę inicjalizacji wag, która zapewnia stabilne warunki początkowe dla procesu uczenia. W warstwie ukrytej sieci zastosowano funkcję aktywacji typu sigmoidalnego. W procesie oceny jakości modelu zastosowano funkcję kosztu typu MSE. W badaniu analizowano trzy algorytmy uczenia: LM, BR oraz SCG. Celem było określenie optymalnej liczby neuronów w warstwie ukrytej, która zapewni najlepszą jakość modelu. Proces rozpoczęto od zastosowania algorytmu LM ze względu na jego efektywne właściwości szybkiej zbieżności, szczególnie w problemach z ograniczoną ilością danych. Iteracyjnie testowano różne konfiguracje liczby neuronów w warstwie ukrytej (od 2 do 12), używając algorytmu LM. W wyniku tego podejścia udało się wyłonić konfigurację sieci, która zapewniła najwyższą jakość przetwarzania danych. Następnie, wykorzystując ustaloną liczbę neuronów, przystąpiono do trenowania sieci za pomocą pozostałych algorytmów: SCG oraz BR. Procedura została przeprowadzona analogicznie, jak w przypadku modelowania parametrów chropowości. Dzięki identycznej architekturze sieci możliwe było bezpośrednie porównanie efektywności poszczególnych algorytmów.

Zestaw danych eksperymentalnych składał się z 46 obserwacji, podzielonych w proporcji 75% : 25%, gdzie pierwsza wartość oznaczała udział danych uczących, a druga – danych walidacyjnych. Ze względu na małą ilość danych pominięto dane testowe. Aby uniknąć przeuczenia sieci, zastosowano mechanizm wczesnego zatrzymania, ustawiony na poziomie 6. Wydajność i jakość sieci oceniano przy użyciu wskaźników jakościowych, takich jak: R, MSE, RMSE, RIE, MAPE i MAE.

4.5.2. Wyniki modelowania siły skrawania

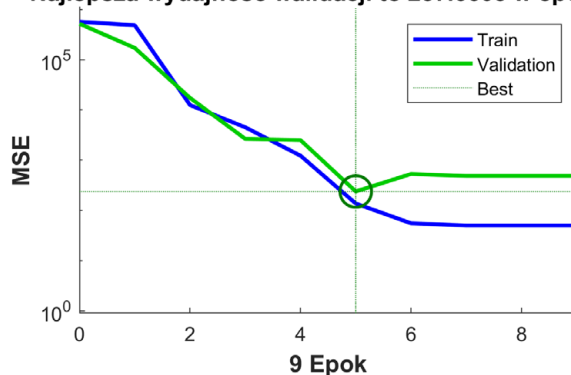
Składową siły skrawania, która została zamodelowana z wykorzystaniem sieci neuronowych MLP, była składowa F_x . W procesie uczenia, wykorzystując algorytm LM, uzyskano najlepsze parametry jakościowe dla sieci neuronowej z siedmioma neuronami w warstwie ukrytej. Osiągnięto to po 9 epokach uczenia. Następnie dla warstwy ukrytej zawierającej siedem neuronów trenowano sieci neuronowe MLP z wykorzystaniem dwóch pozostałych algorytmów: SCG oraz BR. Tabela 4.6 przedstawia wskaźniki jakości trzech modeli sieci neuronowych. Algorytm LM wymagał 9 epok do osiągnięcia najlepszych wyników, podczas gdy SCG i BR wymagały odpowiednio 19 i 83 epok. Współczynnik korelacji R dla

LM wyniósł 0,999, co wskazuje na najwyższą dokładność w porównaniu z SCG i BR. Pod względem MSE algorytm LM osiągnął wartość 160,59, co jest znacznie niższe niż dla SCG i BR. Podobnie przedstawiają się wartości dla wszystkich pozostałych analizowanych błędów. Na podstawie tych wskaźników można stwierdzić, że algorytm LM zapewnia najlepszą jakość modelowania składowej F_x w porównaniu z algorytmami SCG i BR.

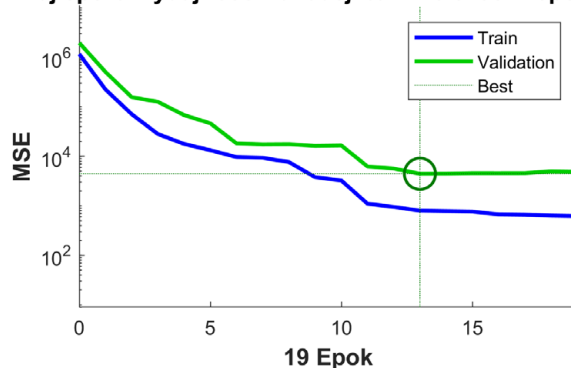
Tabela 4.6. Wskaźniki jakości dla różnych algorytmów uczenia sieci MLP dla składowej F_x

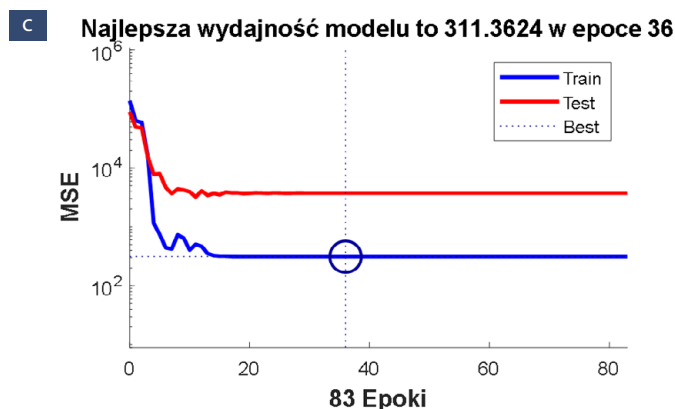
Algorytm uczenia sieci MLP	LM	SCG	BR
Epoka	9	19	83
R (cały zbiór danych)	0,999	0,989	0,993
MSE	160,59	1657,24	1120,38
RMSE	12,67	40,71	33,47
RIE	0,0168	0,0542	0,0446
MAPE	0,0193	0,0439	0,0317
MAE	10,56	29,67	20,75

a Najlepsza wydajność walidacji to 237.6603 w epoce 5



b Najlepsza wydajność walidacji to 4443.5793 w epoce 13

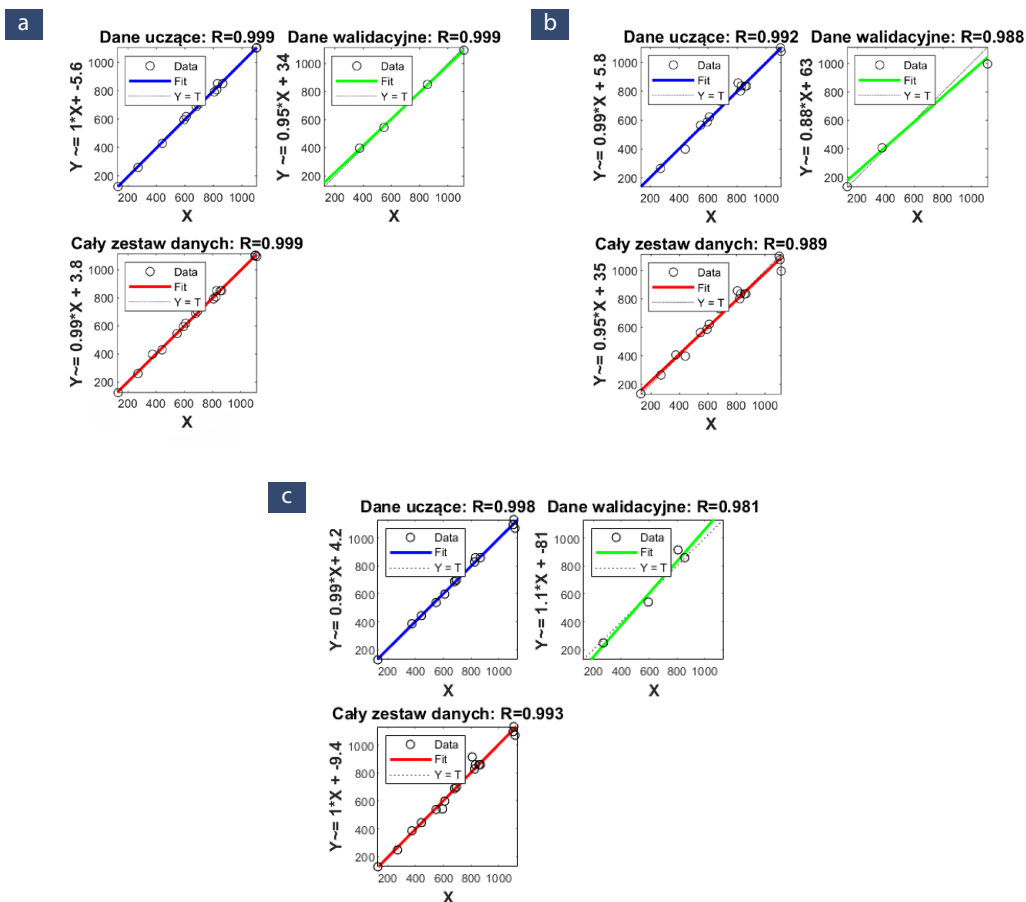




Ryc. 4.17. Najlepsza wydajność modelu F_x dla sieci MLP z algorytmem uczenia: a) LM, b) SCG, c) BR

Ryc. 4.17 ilustruje wykresy z najlepszą wydajnością modelu F_x dla sieci z trzema różnymi algorytmami uczenia: LM, SCG i BR. Analiza wykresów wskazuje na różnice w efektywności i zbieżności trzech algorytmów uczenia sieci neuronowych dla modelu F_x . Algorytm LM wykazał się najniższym MSE podczas walidacji, co sugeruje jego najwyższą precyzję w modelowaniu składowej F_x . Zbieżność algorytmu LM jest również szybsza, osiąga najlepsze wyniki już po 16 epokach. Algorytm SCG, choć wymagał mniej epok do osiągnięcia najlepszej wydajności walidacji w porównaniu z BR, charakteryzował się znacznie wyższym MSE, co wskazuje na jego niższą dokładność. Algorytm BR, mimo większej liczby epok, osiągnął lepsze wyniki niż SCG, ale nie dorównał precyzji LM. Wnioski te sugerują, że dla modelowania składowej F_x w analizowanym kontekście algorytm LM jest najbardziej efektywny i precyzyjny.

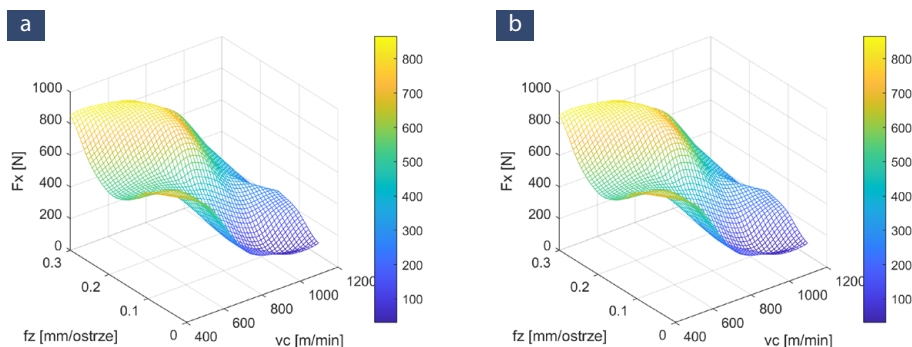
Na ryc. 4.18 przedstawiono wykresy korelacji dla zestawów danych uczących, walidacyjnych i całych zestawów danych dla modeli MLP z różnymi algorytmami uczenia: a) LM, b) SCG, c) BR.



Ryc. 4.18. Wykresy korelacji dla modelu F_x z algorytmem uczenia: a) LM, b) SCG, c) BR

Podsumowując, analiza porównawcza wyników dla modelowania składowej siły skrawania F_x wskazuje, że algorytm LM wydaje się najbardziej efektywny pod względem szybkości uczenia, dokładności oraz ogólnej jakości wyników w porównaniu z dwoma pozostałymi.

Na ryc. 4.19 zilustrowano wielkość składowej F_x siły skrawania uzyskaną z modelu MLP z wykorzystaniem algorytmu LM. W badaniu uwzględniono trzy parametry wejściowe, dlatego zmiany wartości składowej F_x zostały zilustrowane w kontekście par parametrów skrawania: v_c/f_z i f_z/a_p . Symulacje przeprowadzono, wprowadzając dane wejściowe (tj. parametry frezowania v_c, a_p, f_z) do oprogramowania Matlab, a następnie uzyskano wynik z modelu w postaci składowej F_x siły skrawania.



Ryc. 4.19. Składowa F_x siły skrawania uzyskana z modelu MLP z algorytmem uczenia LM w zależności od v_c/f_z i f_z/a_p

4.5.3. Optymalizacja parametrów obróbki

W procesie optymalizacji parametrów obróbki celem jest znalezienie optymalnych wartości prędkości skrawania (v_c), posuwu na ostrze (f_z) oraz głębokości skrawania (a_p), które minimalizują siłę skrawania F_x przy jednoczesnym zachowaniu wysokiej wydajności procesu (maksymalizowaniu wydajności objętościowej frezowania). Optymalizacja została przeprowadzona przy użyciu modelu sieci neuronowej MLP, który został wcześniej wytrenowany na danych eksperymentalnych. Model ten przewiduje wartość składowej F_x siły skrawania na podstawie wartości parametrów obróbki v_c , f_z i a_p .

Zakresy oraz kroki zmian dla trzech głównych parametrów wejściowych zostały określone na podstawie praktycznych wartości stosowanych w procesie obróbki frezowaniem:

- prędkość skrawania (v_c): 400÷1200 [m/min] (krok: 10 [m/min]);
- posuw na ostrze (f_z): 0,05÷0,3 [mm/ostre] (krok: 0,01 [mm/ostre]);
- głębokość skrawania (a_p): 0,5÷6 [mm] (krok: 0,1 [mm]).

Zakresy docelowe dla parametru F_x zostały zdefiniowane na podstawie dopuszczalnych wartości sił skrawania, które nie powodują nadmiernych odkształceń obrabianego przedmiotu.

Funkcja celu w optymalizacji parametrów obróbki jest zdefiniowana jako negatywna wydajność objętościowa frezowania. Negatywną wartość stosuje się, aby optymalizacja polegała na maksymalizowaniu wydajności poprzez minimalizację negatywnej wartości.

Wydajność objętościowa frezowania jest obliczana na podstawie prędkości obrotowej wrzeciona, prędkości posuwowej oraz szerokości i głębokości frezowania.

- Prędkość obrotowa wrzeciona [172]:

$$n = \frac{v_c * 1000}{D * \pi} \quad (4.8)$$

gdzie:

n – prędkość obrotowa wrzeciona [min^{-1}];

v_c – prędkość skrawania [m/min];

D – średnica narzędzia [mm].

- Prędkość posuwowa [172]:

$$v_f = f_z * z * n \quad (4.9)$$

gdzie:

v_f – prędkość posuwowa [mm/min];

f_z – posuw na ostrze [mm/ostrze];

z – liczba ostrzy frezu.

- Wydażność objętościowa frezowania[173]:

$$Q = \frac{a_e * a_p * v_f}{1000} \quad (4.10)$$

gdzie:

Q – wydażność objętościowa frezowania [cm^3/min];

a_e – szerokość frezowania [mm];

a_p – głębokość skrawania [mm].

Matematycznie funkcja celu jest zdefiniowana jako:

$$\text{Wydażność} = -Q \quad (4.10)$$

W kodzie Matlab funkcja celu jest zdefiniowana jako algorytm 4.3.

Algorytm 4.3. Definicja funkcji celu

```
% Definicja funkcji celu (wydażność objętościowa frezowania)
function Q = calculateVolumeEfficiency(vc, fz, ap)
n = (vc * 1000) / (D * pi); % Prędkość obrotowa wrzeciona [min^-1]
vf = fz * z * n; % Prędkość posuwowa [mm/min]
Q = (ae * ap * vf) / 1000; % Wydażność objętościowa [cm^3/min]
end
% Definicja funkcji celu (negatywna wydażność objętościowa)
objectiveFunction = @(x) -calculateVolumeEfficiency(x(1),
x(2), x(3));
```

Dla składowej siły F_x zdefiniowano funkcję błędu w stosunku do zakresu docelowego, aby określić, czy wyliczone wartości siły mieszczą się w zadanych granicach:

$$E_{F_x}(v_c, f_z, a_p) = \begin{cases} F_{x_{\max}} - \text{netF } x(v_c, f_z, a_p), & \text{jeżeli } \text{netF } x(v_c, f_z, a_p) < F_{x_{\max}} \\ \text{netF } x(v_c, f_z, a_p) - F_{x_{\max}}, & \text{jeżeli } \text{netF } x(v_c, f_z, a_p) > F_{x_{\max}} \\ 0, & \text{w innym przypadku} \end{cases} \quad (4.12)$$

gdzie:

$E_{F_x}(v_c, f_z, a_p)$ – funkcja błędu dla składowej siły F_x ;

$F_{x_{\max}}$ – maksymalna dopuszczalna wartość siły $F_{x_{\max}}$;

$\text{netF } x(v_c, f_z, a_p)$ – wyliczona wartość składowej siły F_x przez sieć neuronową dla zadanych wartości v_c, f_z, a_p .

Procedura optymalizacji polega na przeszukiwaniu przestrzeni parametrów v_c, f_z i a_p w zadanych zakresach i krokach zmian, aby znaleźć kombinację parametrów, która minimalizuje funkcję celu i jednocześnie spełnia ograniczenia. Przeszukiwanie przestrzeni parametrów realizowane jest w wyniku działania kodu przedstawionego jako algorytm 4.4.

Algorytm 4.4. Przeszukiwanie przestrzeni parametrów

```
% Przeszukiwanie przestrzeni parametrów
for vc = vc_values
    for fz = fz_values
        for ap = ap_values
            % Aktualne wartości parametrów wejściowych
            params = [vc, fz, ap];

            % Wyznaczanie wartości wyjściowych z sieci neuronowej
            Fx = net(params');

            % Obliczanie błędu
            if Fx < Fx_max
                error = Fx_max - Fx;
            else
                error = Fx - Fx_max;
            end

            % Aktualizacja najlepszych wyników
            if error < bestError
                bestError = error;
                bestParams = params;
            end
        end
    end
end
```


Algorytm 4.4. Przeszukiwanie przestrzeni parametrów, cd.

```
bestFx = Fx;  
    end  
end  
end  
end
```

Kolejny etap to zaokrąglenie wyników otrzymanych po optymalizacji parametrów wejściowych: v_c do najbliższej wielokrotności 10 [m/min], f_z do dwóch miejsc po przecinku (0,01 [mm]), a a_p do jednego miejsca po przecinku (0,1 [mm]). Dalej następuje wyliczenie wartości składowej F_x dla zaokrąglonych parametrów wejściowych oraz wyświetlenie wyników. W celu łatwiejszego korzystania z przygotowanego algorytmu zaproponowano prosty interfejs użytkownika. Został przedstawiony na ryc. 4.20.

Wartość Graniczna	
Graniczna wartość F_x :	500

Parametry Wejściowe	
Minimalna prędkość v_c :	400
Minimalny posuw f_z :	0.05
Minimalna głębokość a_p :	0.5
Maksymalna prędkość v_c :	1200
Maksymalny posuw f_z :	0.3
Maksymalna głębokość a_p :	6

Optymalizuj

Wyniki Optymalizacji	
Optymalne v_c :	
Optymalny f_z :	
Optymalny a_p :	
Maksymalna wydajność:	
Wyliczone F_x :	

Ryc. 4.20. Interfejs użytkownika aplikacji doboru optymalnych parametrów obróbki

Interfejs graficzny składa się z trzech głównych paneli: „Wartość Graniczna”, „Parametry Wejściowe” oraz „Wyniki Optymalizacji”. Panel „Wartość Graniczna” zawiera pole edycyjne służące do wprowadzania wartości granicznej siły F_x . Użytkownik wprowadza tutaj maksymalną dopuszczalną wartość siły skrawania. Pole edycyjne do wprowadzenia wartości granicznej siły F_x ma domyślną wartość ustawioną na 500. Panel „Parametry Wejściowe” zawiera pola edycyjne służące do

wprowadzania zakresu przestrzeni parametrów wejściowych, w których poszukuje się optymalnych wartości. Użytkownik może tutaj określić minimalne i maksymalne wartości prędkości skrawania (v_c), posuwu na ostrze (f_z) oraz głębokości skrawania (a_p). Panel „Wyniki Optymalizacji” wyświetla wyniki procesu optymalizacji, w tym optymalne wartości parametrów obróbki oraz obliczoną maksymalną wydajność i przewidywaną wartość F_x . Pola tekstowe w panelu obejmują:

- Optymalne v_c : Wyświetla optymalną wartość prędkości skrawania;
- Optymalny f_z : Wyświetla optymalną wartość posuwu na ostrze;
- Optymalny a_p : Wyświetla optymalną wartość głębokości skrawania;
- Maksymalna wydajność: Wyświetla maksymalną wydajność objętościową frezowania;
- Wyliczone F_x : Wyświetla przewidywaną wartość siły skrawania F_x na podstawie optymalnych parametrów.

Przycisk „Optymalizuj” uruchamia proces optymalizacji na podstawie wprowadzonych danych. Po jego naciśnięciu funkcja *callback* (opisana w algorytmie 4.5) przeprowadza optymalizację parametrów skrawania (uruchamia odpowiedni fragment kodu), wyznaczając optymalne wartości v_c , f_z i a_p , które minimalizują siłę skrawania F_x przy jednoczesnym maksymalizowaniu wydajności objętościowej frezowania. Interfejs graficzny został zaprojektowany w sposób intuicyjny, umożliwiając użytkownikowi łatwe wprowadzenie niezbędnych danych wejściowych, uruchomienie procesu optymalizacji oraz przeglądanie wyników. Budowa pełnego algorytmu optymalizacyjnego łącznie z interfejsem została przedstawiona jako algorytm 4.5.

Algorytm 4.5. Algorytm optymalizacyjny parametrów obróbki przy ograniczeniach siłowych

```
function skrawanie_optimizer
    % Wczytaj lub zdefiniuj tutaj swoją sieć neuronową 'net'
    load('net.mat', 'net'); % Przykład wczytania sieci z pliku%
    Tworzenie głównego okna GUI
    f = figure('Position', [300, 300, 500, 650], 'Name',
        'Optymalizacja Parametrów Skrawania', 'NumberTitle', 'off');
    % Panel do wprowadzenia wartości granicznej Fx
    p1 = uipanel('Title', 'Wartość Graniczna', 'FontSize', 12,
        'Position', [0.05 0.85 0.9 0.1]);
    uicontrol('Parent', p1, 'Style', 'text', 'Position', [20, 20,
        140, 20], 'String', 'Graniczna wartość Fx:', 'HorizontalAlignment',
        'left');
    fxEdit = uicontrol('Parent', p1, 'Style', 'edit', 'Position',
        [180, 20, 100, 20], 'String', '500');
    % Panel do wprowadzenia parametrów wejściowych
```

Algorytm 4.5. Algorytm optymalizacyjny parametrów obróbki przy ograniczeniach siłowych, cd.

```
p2 = uipanel('Title', 'Parametry Wejściowe', 'FontSize', 12,
'Position', [0.05 0.45 0.9 0.4]);
uicontrol('Parent', p2, 'Style', 'text', 'Position', [20, 180,
140, 20], 'String', 'Minimalna prędkość vc:', 'HorizontalAlignment',
'left');
vcMinEdit = uicontrol('Parent', p2, 'Style', 'edit', 'Position',
[180, 180, 100, 20], 'String', '400');
uicontrol('Parent', p2, 'Style', 'text', 'Position', [20, 150, 140,
20], 'String', 'Minimalny posuw fz:', 'HorizontalAlignment', 'left');
fzMinEdit = uicontrol('Parent', p2, 'Style', 'edit', 'Position',
[180, 150, 100, 20], 'String', '0.05');
uicontrol('Parent', p2, 'Style', 'text', 'Position', [20, 120,
140, 20], 'String', 'Minimalna głębokość ap:', 'HorizontalAlignment',
'left');
apMinEdit = uicontrol('Parent', p2, 'Style', 'edit', 'Position',
[180, 120, 100, 20], 'String', '0.5');
uicontrol('Parent', p2, 'Style', 'text', 'Position', [20, 90, 140,
20], 'String', 'Maksymalna prędkość vc:', 'HorizontalAlignment',
'left');
vcMaxEdit = uicontrol('Parent', p2, 'Style', 'edit', 'Position',
[180, 90, 100, 20], 'String', '1200');
uicontrol('Parent', p2, 'Style', 'text', 'Position', [20, 60, 140,
20], 'String', 'Maksymalny posuw fz:', 'HorizontalAlignment', 'left');
fzMaxEdit = uicontrol('Parent', p2, 'Style', 'edit', 'Position',
[180, 60, 100, 20], 'String', '0.3');
uicontrol('Parent', p2, 'Style', 'text', 'Position', [20, 30, 140,
20], 'String', 'Maksymalna głębokość ap:', 'HorizontalAlignment',
'left');
apMaxEdit = uicontrol('Parent', p2, 'Style', 'edit', 'Position',
[180, 30, 100, 20], 'String', '6');
% Przycisk do wywołania optymalizacji
uicontrol('Style', 'pushbutton', 'Position', [190, 260, 120, 30],
'String', 'Optymalizuj', 'Callback', @(~,~) optimizeCallback(net));
% Panel do wyświetlania wyników
p3 = uipanel('Title', 'Wyniki Optymalizacji', 'FontSize', 12,
'Position', [0.05 0.05 0.9 0.35]);
uicontrol('Parent', p3, 'Style', 'text', 'Position', [20, 130, 140,
20], 'String', 'Optymalne vc:', 'HorizontalAlignment', 'left');
vcOptText = uicontrol('Parent', p3, 'Style', 'text', 'Position',
[180, 130, 100, 20], 'String', '', 'BackgroundColor', 'white');
uicontrol('Parent', p3, 'Style', 'text', 'Position', [20, 100,
140, 20], 'String', 'Optymalny fz:', 'HorizontalAlignment', 'left');
```

Algorytm 4.5. Algorytm optymalizacyjny parametrów obróbki przy ograniczeniach siłowych, cd.

```
fzOptText = uicontrol('Parent', p3, 'Style', 'text', 'Position',  
[180, 100, 100, 20], 'String', '', 'BackgroundColor', 'white');  
uicontrol('Parent', p3, 'Style', 'text', 'Position', [20, 70, 140,  
20], 'String', 'Optymalny ap:', 'HorizontalAlignment', 'left');  
apOptText = uicontrol('Parent', p3, 'Style', 'text', 'Position',  
[180, 70, 100, 20], 'String', '', 'BackgroundColor', 'white');  
uicontrol('Parent', p3, 'Style', 'text', 'Position', [20, 40, 140,  
20], 'String', 'Maksymalna wydajność:', 'HorizontalAlignment', 'left');  
maxEffText = uicontrol('Parent', p3, 'Style', 'text', 'Position',  
[180, 40, 100, 20], 'String', '', 'BackgroundColor', 'white');  
uicontrol('Parent', p3, 'Style', 'text', 'Position', [20, 10, 140, 20],  
'String', 'Wyliczone Fx:', 'HorizontalAlignment', 'left');  
fxCalcText = uicontrol('Parent', p3, 'Style', 'text', 'Position',  
[180, 10, 100, 20], 'String', '', 'BackgroundColor', 'white');  
% Funkcja callback dla przycisku optymalizacji  
function optimizeCallback(net)  
    % Pobieranie wartości z pól edycyjnych  
    Fx_max = str2double(get(fxEdit, 'String'));  
    vc_min = str2double(get(vcMinEdit, 'String'));  
    fz_min = str2double(get(fzMinEdit, 'String'));  
    ap_min = str2double(get(apMinEdit, 'String'));  
    vc_max = str2double(get(vcMaxEdit, 'String'));  
    fz_max = str2double(get(fzMaxEdit, 'String'));  
    ap_max = str2double(get(apMaxEdit, 'String'));  
    D = 16; % Średnica narzędzia [mm]  
    z = 2; % Liczba ostrzy frezu  
    ae = 15; % Szerokość frezowania [mm]  
    % Zakresy i kroki zmian parametrów wejściowych  
    vc_values = vc_min:10:vc_max;  
    fz_values = fz_min:0.01:fz_max;  
    ap_values = ap_min:0.1:ap_max;  
    % Definicja funkcji celu (wydajność objętościowa frezowania)  
    function Q = calculateVolumeEfficiency(vc, fz, ap)  
        n = (vc * 1000) / (D * pi); % Prędkość obrotowa  
        wrzeczona [min^-1]  
        vf = fz * z * n; % Prędkość posuwowa [mm/min]  
        Q = (ae * ap * vf) / 1000; % Wydajność  
        objętościowa [cm^3/min]  
    end  
    % Definicja funkcji celu (negatywna wydajność objętościowa)  
    objectiveFunction = @(x) -calculateVolumeEfficiency(x(1), x(2), x(3));
```

Algorytm 4.5. Algorytm optymalizacyjny parametrów obróbki przy ograniczeniach siłowych, cd.

```
% Definicja funkcji ograniczeń
function [c, ceq] = constraints(x, net, Fx_max)
    % Wejścia do sieci neuronowej
    inputs = [x(1); x(2); x(3)];
    % Przewidywana wartość Fx z sieci neuronowej
    Fx_pred = net(inputs);
    % Ograniczenia nierównościowe (Fx <= Fx_max)
    c = Fx_pred - Fx_max;
    % Brak ograniczeń równościowych
    ceq = [];
end

% Inicjalizacja zmiennych do przechowywania najlepszych wyników
bestParams = [];
bestError = inf;
% Przeszukiwanie przestrzeni parametrów
for vc = vc_values
    for fz = fz_values
        for ap = ap_values
            % Aktualne wartości parametrów wejściowych
            params = [vc, fz, ap];
            % Wyznaczanie wartości wyjściowych z sieci neuronowej
            Fx = net(params');
            % Obliczanie błędu
            if Fx < Fx_max
                error = Fx_max - Fx;
            else
                error = Fx - Fx_max;
            end
            % Aktualizacja najlepszych wyników
            if error < bestError
                bestError = error;
                bestParams = params;
                bestFx = Fx;
            end
        end
    end
end

% Zaokrąglenie wyników
vc_opt_rounded = round(bestParams(1) / 10) * 10;
fz_opt_rounded = round(bestParams(2) * 100) / 100;
ap_opt_rounded = round(bestParams(3) * 10) / 10;
```

Algorytm 4.5. Algorytm optymalizacyjny parametrów obróbki przy ograniczeniach siłowych, cd.

```
% Wyznaczanie wartości wyjściowych z sieci neuronowej dla
zaokrąglonych wyników
inputs_opt_rounded = [vc_opt_rounded; fz_opt_rounded;
ap_opt_rounded];
Fx_pred_opt_rounded = net(inputs_opt_rounded);
maxEfficiency = calculateVolumeEfficiency(vc_opt_rounded, fz_
opt_rounded, ap_opt_rounded);
% Wyświetlenie wyników w GUI
set(vcOptText, 'String', num2str(vc_opt_rounded, '%.2f'));
set(fzOptText, 'String', num2str(fz_opt_rounded, '%.2f'));
set(apOptText, 'String', num2str(ap_opt_rounded, '%.2f'));
set(maxEffText, 'String', num2str(maxEfficiency, '%.2f'));
set(fxCalcText, 'String', num2str(Fx_pred_opt_rounded,
'%.2f'));
end
end
```

Przykład 1. działania algorytmu dla założonej wartości granicznej $F_x = 500$ [N], przedstawia ryc. 4.21, zaś przykład 2. dla założonej wartości granicznej $F_x = 600$ [N] – ryc. 4.22. Po weryfikacji otrzymanych wartości parametrów można stwierdzić, że zawierają się one w założonych granicach.

W ramach przeprowadzonych analiz opracowano algorytm optymalizacyjny parametrów obróbki, mający na celu znalezienie optymalnych wartości prędkości skrawania (v_c), posuwu na ostrze (f_z) oraz głębokości skrawania (a_p), które minimalizują składową siły skrawania F_x , przy jednoczesnym maksymalizowaniu wydajności objętościowej frezowania. Optymalizacja została przeprowadzona przy użyciu modelu sieci neuronowej wytrenowanego na danych eksperymentalnych, przewidującego wartość składowej F_x siły skrawania na podstawie parametrów obróbki. Przykłady działania algorytmu dla różnych wartości granicznych F_x potwierdziły, że otrzymane parametry mieszczą się w założonych granicach, co świadczy o skuteczności zaproponowanego podejścia.

Wartość Graniczna	
Graniczna wartość F_x :	500

Parametry Wejściowe	
Minimalna prędkość v_c :	400
Minimalny posuw f_z :	0.05
Minimalna głębokość a_p :	0.5
Maksymalna prędkość v_c :	1200
Maksymalny posuw f_z :	0.3
Maksymalna głębokość a_p :	6

Optymalizuj

Wyniki Optymalizacji	
Optymalne v_c :	560.00
Optymalny f_z :	0.20
Optymalny a_p :	2.80
Maksymalna wydajność:	187.17
Wyliczone F_x :	500.00

Ryc. 4.21. Przykład 1. działania algorytmu optymalizacyjnego dla założonej wartości granicznej $F_x = 500$ [N]

Wartość Graniczna	
Graniczna wartość F_x :	600

Parametry Wejściowe	
Minimalna prędkość v_c :	400
Minimalny posuw f_z :	0.05
Minimalna głębokość a_p :	0.5
Maksymalna prędkość v_c :	1200
Maksymalny posuw f_z :	0.3
Maksymalna głębokość a_p :	6

Optymalizuj

Wyniki Optymalizacji	
Optymalne v_c :	1160.00
Optymalny f_z :	0.22
Optymalny a_p :	5.40
Maksymalna wydajność:	822.48
Wyliczone F_x :	600.00

Ryc. 4.22. Przykład 2. działania algorytmu optymalizacyjnego dla założonej wartości granicznej $F_x = 600$ [N]

5. Ocena jakości wyrobów gotowych

Ocena jakości wyrobów gotowych z wykorzystaniem sztucznych sieci neuronowych w procesach produkcyjnych jest zagadnieniem, które w dużym stopniu determinuje efektywność i konkurencyjność współczesnych zakładów produkcyjnych. W zależności od podejścia do problemu jakości analiza ta może przyjmować charakter zarówno klasyfikacyjny, jak i regresyjny, co ma znaczenie dla doboru odpowiednich metod i narzędzi analizy. ANN, dzięki swojej elastyczności i zdolności do modelowania złożonych zależności, oferują szerokie spektrum możliwości w tym zakresie, jednakże wybór odpowiedniego modelu zależy od specyfiki zadania i rodzaju analizowanych danych.

Klasyfikacja i regresja jako dwa podstawowe podejścia w ocenie jakości różnią się zasadniczo pod względem celu analizy. Problemy klasyfikacyjne koncentrują się na przypisaniu badanego obiektu do jednej z określonych klas, co ma szczególne znaczenie w przypadkach, gdzie konieczne jest jednoznaczne określenie, czy dany wyrób spełnia określone standardy jakości, czy też nie. Przykładem może być wykrywanie wad odlewniczych, gdzie klasyfikacja obrazu jako „bez wad” lub „z wadami” decyduje o dalszym losie produktu w procesie produkcyjnym. Z kolei regresję stosuje się w przypadkach, gdzie istotne jest przewidywanie ciągłych wartości związanych z jakością, takich jak grubość powłoki, chropowatość powierzchni czy inne parametry fizyczne, które wymagają dokładnej kwantyfikacji, a nie tylko binarnego podziału na klasy.

W niniejszym rozdziale, w kolejnych podrozdziałach omówione zostaną oba podejścia z uwzględnieniem specyficznych przypadków zastosowań ANN w procesach produkcyjnych. W pierwszej kolejności przedstawiony zostanie problem klasyfikacyjny związany z wykrywaniem wad w odlewach. W ramach tych badań zastosowano technikę transfer learning, która umożliwia adaptację wcześniej wytrenowanych modeli do specyficznych zadań przemysłowych. Takie podejście pozwala na skrócenie czasu potrzebnego na uczenie sieci oraz na znaczną poprawę efektywności kontroli jakości. Drugi przypadek klasyfikacyjny dotyczy problemu wykrywania błędów kształtu na dynamicznych liniach produkcyjnych, gdzie obrazy obiektów są rozmyte z powodu ich ruchu na taśmociągu. Tutaj również zastosowanie znajdują techniki klasyfikacyjne, które pozwalają na precyzyjną identyfikację i lokalizację obiektów o niepożądanym kształtach, co jest istotne dla np. automatyzacji kontroli jakości w czasie rzeczywistym.

W rozdziale zostanie również omówiony problem regresyjny, gdzie ANN wykorzystywane są do przewidywania twardości wyrobów gotowych po procesie odpuszczania, co jest ważnym aspektem w ocenie jakości materiałów metalowych. Odpuszczanie jako proces cieplny stosowany w przemyśle metalurgicznym ma na celu poprawę właściwości mechanicznych materiału, takich jak twardość, wytrzymałość i plastyczność. Precyzyjne określenie twardości po odpuszczaniu jest niezbędne, aby zapewnić, że wyrób końcowy spełnia wymagania techniczne oraz normy jakościowe.

Kolejnym istotnym aspektem, który zostanie omówiony, jest różnica w analizie danych w zależności od tego, czy rejestrujemy pojedynczy obiekt, czy obraz z linii produkcyjnej, na którym znajduje się wiele obiektów jednocześnie. Analiza pojedynczych obiektów, jak na przykład detekcja defektów w indywidualnych odlewach, jest relatywnie prostsza, ponieważ model koncentruje się na jednym, dobrze zdefiniowanym elemencie. W takich przypadkach istotny jest wybór sieci neuronowej o odpowiedniej głębokości i zdolności do rozpoznawania wzorców na poziomie detalu, co pozwala na dokładną klasyfikację lub regresję.

Z kolei w przypadku analizy obrazów z linii produkcyjnej, gdzie na jednym obrazie znajduje się wiele obiektów, pojawia się dodatkowe wyzwanie związane z koniecznością lokalizacji, segmentacji oraz klasyfikacji wielu elementów jednocześnie. W takich przypadkach wybór odpowiedniej architektury sieci neuronowej, jak również parametrów modelu, zależy od specyfiki zadania. CNN w takich przypadkach mogą być bardziej efektywne ze względu na ich zdolność do analizowania skomplikowanych układów obiektów oraz wykrywania subtelnych różnic między nimi. Dodatkowo w takich scenariuszach często

stosuje się techniki augmentacji danych oraz transfer learning, aby poprawić zdolność modelu do generalizacji na nowych danych produkcyjnych.

Wybór odpowiedniego podejścia do oceny jakości wyrobów gotowych za pomocą ANN zależy od wielu czynników, w tym od charakterystyki analizowanych danych oraz specyficznych wymagań procesu produkcyjnego. W dalszych częściach rozdziału omówione zostaną szczegółowo przykłady zastosowań zarówno klasyfikacyjnych, jak i regresyjnych modeli ANN, z uwzględnieniem różnic w analizie obrazów pojedynczych obiektów oraz tych pochodzących z linii produkcyjnych. Rozdział ten dostarczy wszechstronnego wglądu w możliwości, jakie oferują sztuczne sieci neuronowe w kontekście oceny jakości wyrobów gotowych, oraz przedstawi perspektywy ich zastosowania w różnych procesach produkcyjnych.

5.1. Wykrywanie wad w odlewanych wyrobach gotowych – problem klasyfikacyjny

Wykrywanie wad w odlewanych wyrobach gotowych stanowi jedno z wyzwań współczesnej produkcji przemysłowej, szczególnie w branży metalurgicznej. Odlewanie jest zaawansowanym procesem produkcyjnym, polegającym na wlewaniu ciekłego materiału, najczęściej metalu, do formy zawierającej wnękę o precyzyjnie zaprojektowanym kształcie, który odpowiada wymiarom gotowego wyrobu. Po wypełnieniu formy materiał jest pozostawiany do stwardnienia, co pozwala na uzyskanie pożądanego kształtu i struktury odlewu. Proces ten, choć powszechnie stosowany, wiąże się z ryzykiem powstawania różnorodnych wad, które mogą znacząco obniżyć jakość wyrobów finalnych. Do najczęściej występujących wad odlewniczych należą pęcherze gazowe, mikropory, zadziory, wady skurczowe, wady materiału formy, wady wlewanego metalu oraz wady metalurgiczne. Każda z tych nieregularności może prowadzić do istotnych problemów funkcjonalnych lub estetycznych wyrobu, co w konsekwencji wpływa na jego wartość rynkową i trwałość.

Wady odlewnicze są szczególnie niepożądane w przemyśle, ponieważ mogą prowadzić do znacznych strat finansowych. W przypadku ich wykrycia cała partia produkcyjna może zostać odrzucona, co nie tylko generuje bezpośrednie koszty związane z utratą surowców i energii, ale także wpływa na reputację producenta oraz opóźnia realizację zamówień. Dlatego często firma zaangażowana w procesy odlewnicze posiada dedykowany dział kontroli jakości, którego zadaniem jest identyfikacja wadliwych produktów przed ich wprowadzeniem na rynek.

Głównym problemem tradycyjnych metod kontroli jakości jest ich manualny charakter. Mimo że proces prowadzony jest przez wykwalifikowanych specjalistów, wymaga dużo czasu i podatny na błędy ludzkie. Ręczne inspekcje nie są w pełni dokładne, a ich efektywność zależy od wielu czynników, takich jak zmęczenie inspektorów, złożoność kontrolowanych wyrobów oraz warunki oświetleniowe. W rezultacie nawet niewielkie defekty mogą zostać przeoczone, co prowadzi do dalszych komplikacji w procesie produkcyjnym.

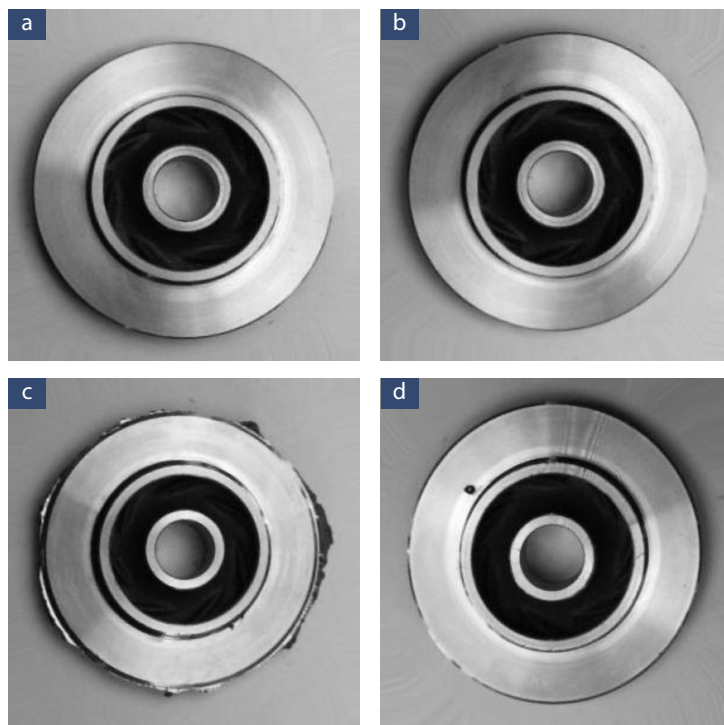
W obliczu tych wyzwań automatyzacja procesu wykrywania wad staje się nie tylko pożądana, ale wręcz niezbędna. Rozwój technologii głębokiego uczenia, w szczególności wstępnie wytrenowanych CNN, otwiera nowe możliwości w dziedzinie automatycznej kontroli jakości. Wykorzystanie technik transfer learning umożliwia adaptację sieci neuronowych, które zostały wcześniej wytrenowane na dużych zbiorach danych obrazowych, do specyficznych zadań związanych z wykrywaniem defektów w odlewach [98,174,175]. Dzięki modyfikacji warstw końcowych takich sieci, jak ResNet50, ResNet101 czy Darknet53, oraz ich ponownemu trenowaniu na mniejszych, specyficznych zbiorach danych, możliwe jest osiągnięcie wysokiej skuteczności w identyfikacji wadliwych wyrobów.

Zastosowanie wstępnie wytrenowanych sieci CNN w przemyśle odlewniczym ma zatem ogromny potencjał. Nie tylko redukuje czas i zasoby potrzebne do trenowania nowych modeli, ale także podnosi jakość i dokładność procesu kontroli, minimalizując ryzyko związane z błędami ludzkimi. W efekcie automatyzacja procesu wykrywania wad z wykorzystaniem zaawansowanych technik głębokiego uczenia może przyczynić się do znacznego zwiększenia efektywności produkcji, redukcji strat oraz poprawy konkurencyjności przedsiębiorstw na rynku. W kontekście dynamicznie rozwijającej się technologii i rosnących wymagań jakościowych wprowadzenie takich innowacyjnych rozwiązań staje się elementem strategii rozwoju nowoczesnych zakładów produkcyjnych.

5.1.1. Metodyka prowadzonych badań

Celem prowadzonych w tej części badań jest ocena skuteczności różnych sieci CNN w klasyfikacji obrazów na dwie kategorie: odlew bez wad (OK) oraz odlew z wadami (defektami). Algorytmy klasyfikacyjne były testowane przy użyciu publicznie dostępnego zestawu obrazów pod nazwą „casting product image data for quality inspection”, który został udostępniony w licencji CC BY-NC-ND 4.0 [176].

Baza danych dotyczy produktów wytwarzanych w procesie odlewania, zawiera zdjęcia wykonane z góry produktu, jakim jest wirnik pompy głębinowej. Łącznie w bazie znajduje się 7348 zdjęć w rozdzielczości 300×300 pikseli w skali szarości, na których zastosowano augmentację. W bazie występuje 3137 zdjęć bez wad oraz 4167 zdjęć z wadami odlewów. Na ryc. 5.1 przedstawiono przykładowe obrazy z obu kategorii, zarówno te z pęknięciami, jak i bez pęknięć.



Ryc. 5.1. Przykładowe zdjęcia z bazy danych: a, b) bez wad; c, d) z wadami

Źródło: Baza danych „casting product image data for quality inspection” [176]

Wybór architektury sieci neuronowej w badaniach dotyczących klasyfikacji obrazów odlewów został podyktowany specyficznymi wymaganiami zadania, a mianowicie koniecznością dokładnego rozróżnienia między odlewami bez wad a odlewami z defektami. Zdecydowano się na zastosowanie CNN ze względu na ich wyjątkową skuteczność w analizie i klasyfikacji obrazów. Architektura CNN jest szczególnie dobrze przystosowana do przetwarzania danych wizualnych, ponieważ potrafi automatycznie rozpoznać i wyodrębnić istotne cechy z obrazów, co jest ważne przy zadaniach, takich jak wykrywanie defektów w odlewach. Dodatkowo w celu maksymalizacji efektywności klasyfikacji zastosowano metodologię transfer learning. Transfer learning

szczególnie dobrze sprawdza się w wykrywaniu defektów, ponieważ pozwala na skorzystanie z gotowych, wytrenowanych modeli, które mają już zdolność rozpoznawania złożonych wzorców wizualnych. To sprawia, że proces uczenia przebiega znacznie szybciej, a dokładność klasyfikacji wzrasta. Dzięki temu podejściu możliwe jest uzyskanie wysokiej jakości wyników nawet przy ograniczonej ilości danych specyficznych dla danego zadania, jak to ma miejsce w przypadku klasyfikacji obrazów odlewów.

Tęgo rodzaju podejście nie jest nowe w literaturze naukowej, a jego skuteczność została już wcześniej potwierdzona przez różnych badaczy. Przykładowo Gupta i in. przeprowadzili szczegółową analizę wydajności kilku modeli, w tym EfficientNetB3, ResNet18, VGG16 oraz ResNet50, ostatecznie wybierając model ResNet50 jako najefektywniejszy w zadaniu wykrywania wad odlewniczych. Ich badania pokazały, że ResNet50, jako model o odpowiedniej głębokości i zdolności do generalizacji, jest wyjątkowo skuteczny w identyfikacji różnorodnych defektów [175]. Z kolei Prasad i in. zastosowali w swoim badaniu sieć ShuffleNet V2, która, choć jest mniej złożona i nie posiada tak dużej głębokości jak inne sieci, oferuje znacznie mniejsze zapotrzebowanie na moc obliczeniową. Dzięki temu model ten jest szczególnie użyteczny w środowiskach, gdzie zasoby sprzętowe są ograniczone, choć może to odbywać się kosztem nieco niższej dokładności w porównaniu z głębszymi sieciami neuronowymi [174].

W prowadzonych badaniach zastosowano podejście, które częściowo różni się od dotychczasowych badań, wykorzystując nie tylko sieci neuronowe stosowane przez innych badaczy, ale także te same modele, aby móc je bezpośrednio porównać. W szczególności, podobnie jak Gupta i in. [175], uwzględniono model ResNet50, który służył jako punkt odniesienia dla oceny skuteczności innych sieci. Dodatkowo w badaniach zastosowano sieci, takie jak ResNet101, Darknet53, Xception, InceptionResNet-v2 oraz Densenet201, które różnią się strukturą i złożonością obliczeniową. Dzięki temu możliwa była dokładna analiza, jak te bardziej złożone modele radzą sobie w porównaniu z ResNet50 w zadaniu identyfikacji defektów w odlewach. Modele te zostały uporządkowane według rosnącej złożoności struktury oraz czasu obliczeń, co pozwoliło na dokładną ocenę ich efektywności i zdolności do identyfikacji defektów w odlewach.

Takie podejście jest uzasadnione z kilku powodów. Po pierwsze, zastosowanie modeli o różnej złożoności struktury umożliwia porównanie ich wydajności i znalezienie optymalnego balansu pomiędzy dokładnością klasyfikacji a wymaganiami obliczeniowymi. Modele, takie jak ResNet50 i ResNet101, choć bardziej złożone, oferują wysoką dokładność, podczas gdy sieci, takie

jak Darknet53 czy Xception mogą dostarczyć lepszy kompromis pomiędzy dokładnością a szybkością przetwarzania. Po drugie, wybór głębokich sieci, takich jak Densenet201 i InceptionResNet-v2, pozwala na wykorzystanie ich zdolności do uchwycenia bardziej złożonych wzorców, co może być ważne w przypadku trudnych do wykrycia defektów. Zastosowanie tej zróżnicowanej puli modeli umożliwia uzyskanie wszechstronnych i solidnych wyników w zadaniu wykrywania wad w odlewach, co ma istotne znaczenie w kontekście przemysłowej kontroli jakości.

W przypadku sieci konwolucyjnych, takich jak Inception-ResNet-v2, ResNet50, ResNet101, Darknet53 oraz Xception, proces ten jest znacznie szybszy i mniej zasobochłonny w porównaniu z trenowaniem sieci od podstaw. Na przykład sieć Inception-ResNet-v2, wytrenowana na ponad milionie obrazów z bazy ImageNet, posiada zdolność klasyfikacji obrazów do 1000 kategorii. Przez zamianę warstwy końcowej na warstwę binarną (dwuklasową) i ponowne trenowanie na nowych danych, możliwe jest przystosowanie tej sieci do wykrywania konkretnych defektów w obiektach.

ResNet50 to 50-warstwowa sieć konwolucyjna, która jest częścią rodziny ResNet (Residual Networks) opracowanej przez zespół badawczy He [177]. Sieć ta wykorzystuje technikę residual learning, która polega na wprowadzeniu połączeń przeskakujących umożliwiających propagację sygnału z pominięciem jednej lub więcej warstw. Dzięki temu sieć może efektywnie trenować głębsze modele bez problemu znikającego gradientu. ResNet50 jest powszechnie używana w różnych zastosowaniach związanych z rozpoznawaniem obrazów i klasyfikacją ze względu na swoją wysoką dokładność i efektywność obliczeniową. Listę warstw tej sieci przedstawia tabela 5.1, gdzie przedstawiono dane dotyczące 22 warstw. Zgrupowano w niej główne typy warstw i bloki resztkowe jako pojedyncze pozycje. W rzeczywistości sieć ResNet50 składa się z 50 warstw, które obejmują wszystkie warstwy konwolucyjne, normalizacyjne, aktywacyjne oraz dodatkowe warstwy w powtarzających się blokach resztkowych. Każdy blok resztkowy składa się z kilku warstw, co w sumie daje pełne 50 warstw w architekturze ResNet50.

Tabela 5.1. Lista warstw sieci ResNET50

Lp.	Typ warstwy	Opis
1	Conv2D	64 filtry, 7×7 , krok 2
2	Batch Normalization	Normalizacja
3	ReLU	Warstwa aktywacji z funkcją ReLU
4	MaxPooling2D	3×3 , krok 2
5	Conv2D	64 filtry, 1×1 , krok 1
6	Batch Normalization	Normalizacja
7	ReLU	Warstwa aktywacji z funkcją ReLU
8	Conv2D	64 filtry, 3×3 , krok 1
9	Batch Normalization	Normalizacja
10	ReLU	Warstwa aktywacji z funkcją ReLU
11	Conv2D	256 filtry, 1×1 , krok 1
12	Batch Normalization	Normalizacja
13	Conv2D	256 filtry, 1×1 , krok 2 (ścieżka skrótu)
14	Batch Normalization	Normalizacja (ścieżka skrótu)
15	Add	Dodanie wyników ze ścieżki głównej i ścieżki skrótu
16	ReLU	Warstwa aktywacji z funkcją ReLU
17	Blok resztkowy $\times 3$	Powtórzenie bloków: 64 filtry, 64 filtry, 256 filtry
18	Blok resztkowy $\times 4$	Powtórzenie bloków: 128 filtry, 128 filtry, 512 filtry
19	Blok resztkowy $\times 6$	Powtórzenie bloków: 256 filtry, 256 filtry, 1024 filtry
20	Blok resztkowy $\times 3$	Powtórzenie bloków: 512 filtry, 512 filtry, 2048 filtry
21	AveragePooling2D	7×7
22	Fully Connected	1000 neuronów (dla klasyfikacji ImageNet)

ResNet101 to rozszerzona wersja sieci ResNet50, składająca się ze 101 warstw [177]. Podobnie jak ResNet50, wykorzystuje technikę residual learning, co pozwala na trenowanie jeszcze głębszych modeli. ResNet101 oferuje wyższą dokładność klasyfikacji w porównaniu z ResNet50, jednak kosztem zwiększonego czasu obliczeń. Sieć ta jest również szeroko stosowana w badaniach i aplikacjach przemysłowych ze względu na swoją zdolność do skutecznego rozpoznawania wzorców w skomplikowanych zbiorach danych. Listę warstw tej sieci przedstawia tabela 5.2, gdzie znajdują się dane dotyczące 22 warstw. Zgrupowano w niej główne typy warstw i bloki resztkowe jako pojedyncze pozycje, analogicznie jak to zostało przedstawione dla sieci ResNet50 powyżej.

Tabela 5.2. Lista warstw sieci ResNET101

Lp.	Typ warstwy	Opis
1	Conv2D	64 filtry, 7×7 , krok 2
2	Batch Normalization	Normalizacja
3	ReLU	Warstwa aktywacji z funkcją ReLU
4	MaxPooling2D	3×3 , krok 2
5	Conv2D	64 filtry, 1×1 , krok 1
6	Batch Normalization	Normalizacja
7	ReLU	Warstwa aktywacji z funkcją ReLU
8	Conv2D	64 filtry, 3×3 , krok 1
9	Batch Normalization	Normalizacja
10	ReLU	Warstwa aktywacji z funkcją ReLU
11	Conv2D	256 filtry, 1×1 , krok 1
12	Batch Normalization	Normalizacja
13	Conv2D	256 filtry, 1×1 , krok 2 (ścieżka skrótu)
14	Batch Normalization	Normalizacja (ścieżka skrótu)
15	Add	Dodanie wyników ze ścieżki głównej i ścieżki skrótu
16	ReLU	Warstwa aktywacji z funkcją ReLU
17	Blok resztkowy $\times 3$	Powtórzenie bloków: 64 filtry, 64 filtry, 256 filtry
18	Blok resztkowy $\times 4$	Powtórzenie bloków: 128 filtry, 128 filtry, 512 filtry
19	Blok resztkowy $\times 23$	Powtórzenie bloków: 256 filtry, 256 filtry, 1024 filtry
20	Blok resztkowy $\times 3$	Powtórzenie bloków: 512 filtry, 512 filtry, 2048 filtry
21	AveragePooling2D	7×7
22	Fully Connected	1000 neuronów (dla klasyfikacji ImageNet)

Darknet53 to 71-warstwowa sieć konwolucyjna, która jest częścią platformy programistycznej Darknet używanej w systemie YOLO do detekcji obiektów w czasie rzeczywistym. Darknet53 charakteryzuje się wysoką dokładnością klasyfikacji i jest zoptymalizowana pod kątem szybkich obliczeń, co czyni ją idealną do zastosowań wymagających przetwarzania obrazu w czasie rzeczywistym. Tabela 5.3 przedstawia strukturę tej sieci, która składa się z 23 warstw. Liczba warstw wynika z charakterystycznej architektury tej sieci, w której występują powtarzające się bloki resztkowe, mające na celu głębsze przetwarzanie danych wejściowych. W przeciwieństwie do sieci ResNet50 czy ResNet101 ostatnia warstwa Darknet53 nie jest warstwą klasyfikacyjną. Wynika to z faktu, że Darknet53 stosuje się głównie jako część większych systemów, takich jak YOLO do detekcji obiektów, gdzie dalsze warstwy są odpowiedzialne za lokalizację i klasyfikację obiektów w obrazie, a nie za klasyfikację całego obrazu do jednej z określonych klas.

Tabela 5.3. Lista warstw sieci Darknet53

Lp.	Typ warstwy	Opis
1	Conv2D	32 filtry, 3×3 , krok 1
2	Batch Normalization	Normalizacja
3	Leaky ReLU	Warstwa aktywacji z funkcją Leaky ReLU
4	Conv2D	64 filtry, 3×3 , krok 2
5	Batch Normalization	Normalizacja
6	Leaky ReLU	Warstwa aktywacji z funkcją Leaky ReLU
7	Blok resztkowy $\times 1$	Powtórzenie bloków: 32 filtry, 32 filtry
8	Conv2D	128 filtry, 3×3 , krok 2
9	Batch Normalization	Normalizacja
10	Leaky ReLU	Warstwa aktywacji z funkcją Leaky ReLU
11	Blok resztkowy $\times 2$	Powtórzenie bloków: 64 filtry, 64 filtry
12	Conv2D	256 filtry, 3×3 , krok 2
13	Batch Normalization	Normalizacja
14	Leaky ReLU	Warstwa aktywacji z funkcją Leaky ReLU
15	Blok resztkowy $\times 8$	Powtórzenie bloków: 128 filtry, 128 filtry
16	Conv2D	512 filtry, 3×3 , krok 2
17	Batch Normalization	Normalizacja
18	Leaky ReLU	Warstwa aktywacji z funkcją Leaky ReLU
19	Blok resztkowy $\times 8$	Powtórzenie bloków: 256 filtry, 256 filtry
20	Conv2D	1024 filtry, 3×3 , krok 2
21	Batch Normalization	Normalizacja
22	Leaky ReLU	Warstwa aktywacji z funkcją Leaky ReLU
23	Blok resztkowy $\times 4$	Powtórzenie bloków: 512 filtry, 512 filtry

Xception to 53-warstwowa sieć konwolucyjna zaprojektowana przez Francois Chollet [178]. Sieć ta opiera się na architekturze Inception, ale wprowadza głębokie separowalne sploty, co pozwala na bardziej efektywną ekstrakcję cech z obrazów. Xception oferuje wysoką dokładność klasyfikacji przy stosunkowo niskim obciążeniu obliczeniowym, co sprawia, że jest atrakcyjna dla różnych zastosowań związanych z analizą obrazów. Tabela 5.4 przedstawiającą listę warstw dla sieci Xception zawiera 43 warstwy, ponieważ jest ona zbudowana z podstawowych bloków zawierających konwolucje rozdzielne głębokie (SeparableConv2D), warstwy normalizacji partii (Batch Normalization) oraz funkcje aktywacji *ReLU*. Struktura sieci obejmuje również warstwy konwolucyjne (Conv2D) na początku, warstwy MaxPooling2D do redukcji wymiarów oraz globalne średnie próbkowanie (Global Average

Pooling2D) na końcu, przed warstwą w pełni połączoną (Fully Connected) służącą do klasyfikacji.

Tabela 5.4. Lista warstw sieci Xception

Lp.	Typ warstwy	Opis
1	Conv2D	32 filtry, 3×3 , krok 2
2	Batch Normalization	Normalizacja
3	ReLU	Warstwa aktywacji z funkcją ReLU
4	Conv2D	64 filtry, 3×3 , krok 1
5	Batch Normalization	Normalizacja
6	ReLU	Warstwa aktywacji z funkcją ReLU
7	SeparableConv2D	128 filtry, 3×3 , krok 1
8	Batch Normalization	Normalizacja
9	ReLU	Warstwa aktywacji z funkcją ReLU
10	SeparableConv2D	128 filtry, 3×3 , krok 1
11	Batch Normalization	Normalizacja
12	MaxPooling2D	3×3 , krok 2
13	Conv2D	256 filtry, 1×1 , krok 2 (ścieżka skrótu)
14	Batch Normalization	Normalizacja (ścieżka skrótu)
15	SeparableConv2D	256 filtry, 3×3 , krok 1
16	Batch Normalization	Normalizacja
17	ReLU	Warstwa aktywacji z funkcją ReLU
18	SeparableConv2D	256 filtry, 3×3 , krok 1
19	Batch Normalization	Normalizacja
20	MaxPooling2D	3×3 , krok 2
21	Conv2D	728 filtry, 1×1 , krok 2 (ścieżka skrótu)
22	Batch Normalization	Normalizacja (ścieżka skrótu)
23	SeparableConv2D	728 filtry, 3×3 , krok 1
24	Batch Normalization	Normalizacja
25	ReLU	Warstwa aktywacji z funkcją ReLU
26	SeparableConv2D	728 filtry, 3×3 , krok 1
27	Batch Normalization	Normalizacja
28	MaxPooling2D	3×3 , krok 2
29	Blok resztkowy x8	Powtórzenie bloków: 728 filtry, 728 filtry
30	SeparableConv2D	1024 filtry, 3×3 , krok 1
31	Batch Normalization	Normalizacja
32	ReLU	Warstwa aktywacji z funkcją ReLU
33	SeparableConv2D	1024 filtry, 3×3 , krok 1
34	Batch Normalization	Normalizacja

Tabela 5.4. Lista warstw sieci Xception, cd.

Lp.	Typ warstwy	Opis
35	MaxPooling2D	3×3 , krok 2
36	SeparableConv2D	1536 filtry, 3×3 , krok 1
37	Batch Normalization	Normalizacja
38	ReLU	Warstwa aktywacji z funkcją ReLU
39	SeparableConv2D	2048 filtry, 3×3 , krok 1
40	Batch Normalization	Normalizacja
41	ReLU	Warstwa aktywacji z funkcją ReLU
42	Global Average Pooling2D	Globalne średnie próbkowanie 2D
43	Fully Connected	1000 neuronów (dla klasyfikacji ImageNet)

InceptionResNet-v2 to kombinacja architektury Inception i ResNet składająca się ze 164 warstw [179]. Sieć ta łączy zalety obu podejść, oferując zaawansowane możliwości ekstrakcji cech oraz stabilność uczenia głębokich sieci. InceptionResNet-v2 służy do klasyfikacji obrazów i charakteryzuje się zdolnością klasyfikacji do 1000 kategorii. W badaniach nad wykrywaniem defektów InceptionResNet-v2 wykazała wysoką efektywność dzięki zaawansowanej architekturze. Listę warstw tej sieci przedstawia tabela 5.5. W tabeli dla sieci InceptionResNet-v2 znajduje się 26 pozycji, które odpowiadają różnym warstwom sieci, analogicznie do poprzednich.

Tabela 5.5. Lista warstw sieci InceptionResNet-v2

Lp.	Typ warstwy	Opis
1	Conv2D	32 filtry, 3×3 , krok 2
2	Batch Normalization	Normalizacja
3	ReLU	Warstwa aktywacji z funkcją ReLU
4	Conv2D	32 filtry, 3×3 , krok 1
5	Batch Normalization	Normalizacja
6	ReLU	Warstwa aktywacji z funkcją ReLU
7	Conv2D	64 filtry, 3×3 , krok 1
8	Batch Normalization	Normalizacja
9	ReLU	Warstwa aktywacji z funkcją ReLU
10	MaxPooling2D	3×3 , krok 2
11	Conv2D	80 filtry, 1×1 , krok 1
12	Batch Normalization	Normalizacja
13	ReLU	Warstwa aktywacji z funkcją ReLU
14	Conv2D	192 filtry, 3×3 , krok 1

Tabela 5.5. Lista warstw sieci InceptionResNet-v2, cd.

Lp.	Typ warstwy	Opis
15	Batch Normalization	Normalizacja
16	ReLU	Warstwa aktywacji z funkcją ReLU
17	MaxPooling2D	3×3 , krok 2
18	Inception-A blok x5	Powtórzenie bloków: różne konfiguracje filtrów
19	Reduction-A	Redukcja wymiarów za pomocą max pooling'u i konwolucji
20	Inception-ResNet-B blok $\times 10$	Powtórzenie bloków: różne konfiguracje filtrów
21	Reduction-B	Redukcja wymiarów za pomocą max pooling'u i konwolucji
22	Inception-ResNet-C blok x5	Powtórzenie bloków: różne konfiguracje filtrów
23	Global Average Pooling2D	Globalne średnie próbkowanie 2D
24	Dropout	50% deaktywowanych neuronów
25	Fully Connected	1000 neuronów (dla klasyfikacji ImageNet)
26	Softmax	Warstwa aktywacji z funkcją Softmax

DenseNet201 to sieć konwolucyjna składająca się z 201 warstw, będąca częścią rodziny DenseNet (densely connected convolutional networks) opracowanej przez Huang i jego współpracowników [180]. Sieć ta charakteryzuje się gęstymi połączeniami, gdzie każda warstwa jest połączona z każdą poprzednią. Dzięki temu sygnały są efektywnie propagowane przez sieć, co pozwala na lepsze wykorzystanie informacji i bardziej skuteczne trenowanie. DenseNet201 minimalizuje problem znikającego gradientu i umożliwia trenowanie bardzo głębokich sieci bez utraty dokładności. Jest szeroko stosowana w zadaniach związanych z klasyfikacją obrazów i analizą danych medycznych. Listę warstw sieci DenseNet201 przedstawia tabela 5.6, która zawiera 16 pozycji, ponieważ uwzględnia wszystkie główne etapy przetwarzania danych, analogicznie jak w poprzednich sieciach. Zawiera warstwy konwolucyjne, normalizacyjne, aktywacyjne, poolingowe, bloki gęste, warstwy przejściowe oraz końcowe warstwy klasyfikacyjne, co zapewnia kompletną strukturę modelu od wejścia do wyjścia.

Tabela 5.6. Lista warstw sieci DenseNet201

Lp.	Typ warstwy	Opis
1	Conv2D	64 filtry, 7×7 , krok 2
2	Batch Normalization	Normalizacja
3	ReLU	Warstwa aktywacji z funkcją ReLU
4	MaxPooling2D	3×3 , krok 2
5	Blok gęsty x6	Powtórzenie bloków: różne konfiguracje filtrów

Tabela 5.6. Lista warstw sieci DenseNet201, cd.

Lp.	Typ warstwy	Opis
6	Transition Layer	Conv2D: 128 filtry, 1×1 , krok 1, AveragePooling2D: 2×2 , krok 2
7	Blok gęsty $\times 12$	Powtórzenie bloków: różne konfiguracje filtrów
8	Transition Layer	Conv2D: 256 filtry, 1×1 , krok 1, AveragePooling2D: 2×2 , krok 2
9	Blok gęsty $\times 48$	Powtórzenie bloków: różne konfiguracje filtrów
10	Transition Layer	Conv2D: 512 filtry, 1×1 , krok 1, AveragePooling2D: 2×2 , krok 2
11	Blok gęsty $\times 32$	Powtórzenie bloków: różne konfiguracje filtrów
12	Batch Normalization	Normalizacja
13	ReLU	Warstwa aktywacji z funkcją ReLU
14	Global Average Pooling2D	Globalne średnie próbkowanie 2D
15	Fully Connected	1000 neuronów (dla klasyfikacji ImageNet)
16	Softmax	Warstwa aktywacji z funkcją Softmax

Każda z wymienionych sieci została dostosowana do klasyfikacji dwuklasowej poprzez zastąpienie oryginalnej w pełni połączonej warstwy (ang. *fully connected layer*) nową z dwoma neuronami wyjściowymi oraz dodanie warstwy softmax w celu uzyskania prawdopodobieństw dla każdej z dwóch kategorii.

W analizie wykorzystano SVM (ang. *support vector machine* – maszynę wektorów nośnych) – algorytm uczenia maszynowego, który jest szeroko stosowany w zadaniach klasyfikacji i regresji. SVM należy do rodziny tzw. algorytmów nadzorowanych, co oznacza, że do jego trenowania potrzebny jest zestaw danych uczących z przypisanymi etykietami.

Celem algorytmu SVM jest znalezienie hiperpowierzchni, która w jak najlepszy sposób oddziela punkty danych jednej klasy od punktów danych innej klasy. „Najlepszy” oznacza hiperpowierzchnię z największym marginesem między dwiema klasami. Margines to maksymalna szerokość pasa równoległego do hiperpowierzchni, który nie zawiera żadnych punktów danych wewnątrz. Hiperpowierzchnia to granica decyzyjna oddzielająca różne klasy, a wektory nośne to podzbiór obserwacji uczących, które określają położenie hiperpowierzchni. Margines to kryterium, które SVM optymalizuje, maksymalizując odległość między klasami.

W praktyce wiele problemów nie jest liniowo separowalnych. W takich przypadkach SVM używa funkcji jądra, aby przekształcić dane wejściowe do przestrzeni o wyższym wymiarze, co ułatwia ich separację. Matlab obsługuje kilka typów funkcji jądra.

- Funkcja jądra liniowego jest używana do klasyfikacji dwuklasowej i można ją przedstawić za pomocą wzoru:

$$K(a, b) = a^T b \quad (5.1)$$

gdzie: a, b – wektory danych.

- Funkcja jądra wielomianowego to:

$$K(a, b) = (a^T b + 1)^c \quad (5.2)$$

gdzie: c – stopień wielomianu.

- Funkcja RBF (Radial Basis Function) jest zdefiniowana jako:

$$K(a, b) = \exp \left(-\gamma \|a - b\|^2 \right) \quad (5.3)$$

gdzie: γ – parametr skalujący.

- Funkcja sigmoidalna opisana jest wzorem:

$$K(a, b) = \tanh(\beta_0 a^T b + \beta_1) \quad (5.4)$$

i jest jądrem Mercera dla pewnych wartości β_0 i β_1 .

Trenowanie maszyny wektorów nośnych polega na rozwiązaniu problemu optymalizacji kwadratowej, aby dopasować hiperpowierzchnię minimalizującą miękki margines między klasami. Liczba przekształconych cech jest określana przez liczbę wektorów nośnych.

W prowadzonych badaniach wykorzystano liniowy SVM, który został wytrenowany z użyciem cech wyekstrahowanych przez sieci CNN. Proces obejmował ekstrakcję cech z jednej z warstw sieci CNN dla każdej próbki obrazu, a następnie użycie tych cech jako wejściowych danych do trenowania klasyfikatora SVM. Dane zostały podzielone na zestawy uczące i testowe w proporcji 30% : 70%, co zapewniło odpowiednie rozdzielenie danych na potrzeby uczenia i testowania modelu. W celu zbalansowania zbioru danych dokonano redukcji liczby obrazów tak, aby każda kategoria miała równą liczbę obiektów (po 3137 zdjęć), co zapewniło, że model nie był uprzedzony na korzyść którejkolwiek z klas. Dane wejściowe zostały przetworzone, aby dopasować je do rozmiaru wejściowego sieci. Proces ten obejmował konwersję obrazów do skali szarości i ich augmentację. W celu ekstrakcji cech z jednej z ostatnich warstw sieci wykorzystano partię danych o wielkości 32. Następnie cechy te posłużyły do trenowania klasyfikatora SVM z użyciem funkcji jądra liniowego.

W odniesieniu do transfer learningu z użyciem wymienionych wcześniej modeli sieć była inicjalizowana z wstępnie wytrenowanymi wagami,

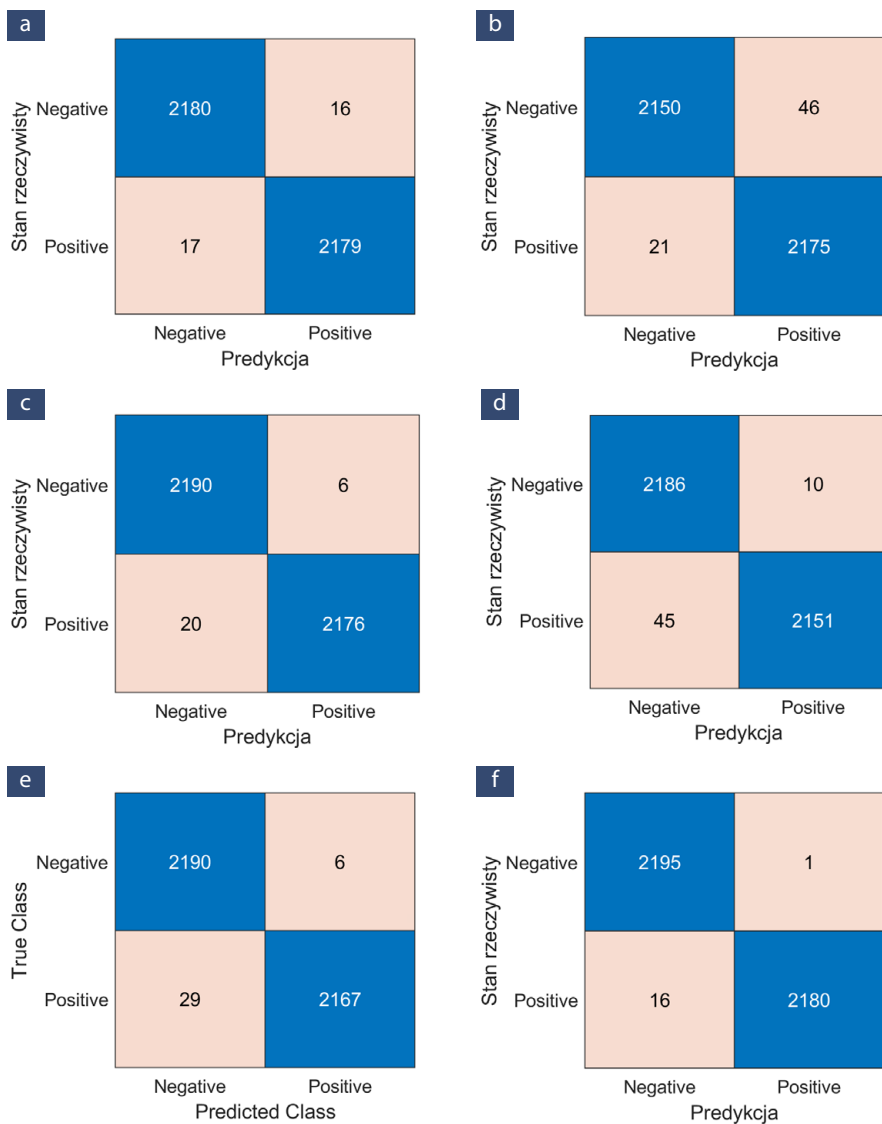
a następnie dostosowywana do specyficznego zadania klasyfikacji obrazów odlewów. Domyślna liczba epok została ustawiona na 30, co pozwala na stopniowe dostosowywanie wag bez ryzyka przeuczenia modelu, które mogłoby wystąpić przy zbyt dużej liczbie iteracji. Współczynnik uczenia, domyślnie ustawiony na 0,01, został wykorzystany do kontrolowania tempa, z jakim sieć modyfikuje swoje wagi podczas procesu trenowania. Dodatkowo została zastosowana regularyzacja L2 z domyślnym współczynnikiem 0,0001, aby przeciwdziałać nadmiernemu dopasowaniu, zwłaszcza podczas pracy z małymi zestawami danych.

Uczenie modeli odbywało się w następujących etapach: inicjalizacja sieci CNN z wagami wytrenowanymi na zadanym zbiorze, zastąpienie ostatniej warstwy w pełni połączonej nową warstwą dwuklasową, trenowanie nowych warstw sieci z użyciem danych uczących.

Modele były oceniane przy użyciu danych testowych (2196 obrazów każdej kategorii). Ocena skuteczności modeli obejmowała obliczenie miar jakości, takich jak dokładność, precyzja, czułość oraz F1-score. Dodatkowo przeprowadzono analizę krzywej ROC oraz AUC, aby ocenić zdolności modeli do rozróżniania między dwoma klasami. Wyniki różnych architektur sieci CNN zostały porównane w celu identyfikacji najsukcesowniejszego modelu.

5.1.2. Wyniki działania klasyfikatorów

Porównanie algorytmów klasyfikacji opartych na transfer learning z głębokimi sieciami neuronowymi (Resnet50, Resnet101, Darknet53, Xception, Inceptionresnetv2 i Densenet201) zostało przeprowadzone na podstawie macierzy pomyłek, które przedstawia ryc. 5.2. Do testowania klasyfikatorów wykorzystano kolejne 2196 obrazów z każdej kategorii z analizowanego zbioru danych („casting product image data for quality inspection” [176]), które nie były wcześniej używane w procesie uczenia tych klasyfikatorów.

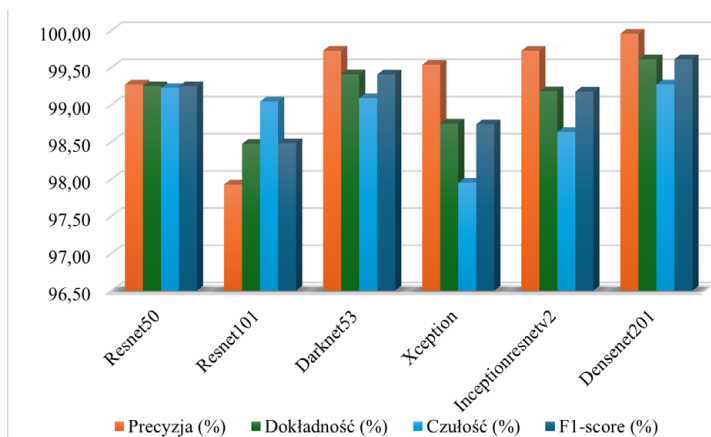


Ryc. 5.2. Macierze pomyłek dla analizowanych modeli klasyfikacyjnych:
a) Resnet50, b) Resnet101, c) Darknet53, d) Xception, e) Inceptionresnetv2, f) Densenet201

Tabela 5.7 przedstawia wyniki wskaźników jakościowych: dokładność, precyzja, czułość i F1-score dla poszczególnych algorytmów klasyfikacji z wykorzystaniem CNN. Wyniki te również przedstawiono w formie wykresu na ryc. 5.3.

Tabela 5.7. Ocena wydajności poszczególnych algorytmów klasyfikacji z wykorzystaniem CNN

Sieć CNN	Dokładność (%)	Precyzja (%)	Czułość (%)	F1-score (%)
Resnet50	99,25	99,27	99,23	99,25
Resnet101	98,47	97,93	99,04	98,48
Darknet53	99,41	99,73	99,09	99,41
Xception	98,75	99,54	97,95	98,74
Inceptionresnetv2	99,18	99,72	98,63	99,18
Densenet201	99,61	99,95	99,27	99,61



Ryc. 5.3. Ocena wydajności poszczególnych algorytmów klasyfikacji z wykorzystaniem CNN

Na podstawie przeprowadzonej analizy wyników działania różnych sieci CNN można stwierdzić, że każda z badanych architektur charakteryzuje się odmiennymi parametrami wydajności, które mają istotne znaczenie w kontekście efektywności klasyfikacji obrazów. Sieć ResNet50 uzyskała bardzo wysoką dokładność na poziomie 99,25%, przy jednocześnie zbliżonych wartościach precyzji i czułości wynoszących odpowiednio 99,27% i 99,23%. Wskaźnik F1-score dla tej sieci wyniósł 99,25%, co wskazuje na zrównoważoną wydajność tej architektury w przewidywaniu obu klas, co czyni ją wyjątkowo skuteczną w tym zadaniu.

Z kolei sieć ResNet101, mimo że osiągnęła wysoką czułość na poziomie 99,04%, charakteryzuje się niższą precyzją (97,93%) oraz dokładnością (98,47%) w porównaniu z innymi modelami. Może to sugerować, że sieć ta

lepiej identyfikuje pozytywne przypadki, jednak kosztem większej liczby fałszywych pozytywów, co obniża jej ogólną efektywność.

Sieć Darknet53 uzyskuje wysoką precyzję, osiągając wynik 99,73%. Jej dokładność wyniosła 99,41%, a czułość 99,09%. Wysoki poziom precyzji wskazuje, że sieć ta jest wyjątkowo skuteczna w minimalizowaniu liczby fałszywych klasyfikacji, co czyni ją bardzo solidnym wyborem w kontekście redukcji błędnych klasyfikacji.

Architektura Xception, pomimo wysokiej precyzji (99,54%), uzyskuje niższą czułość (97,95%), co może sugerować, że sieć jest bardziej rygorystyczna w swoich klasyfikacjach, prowadząc tym samym do pominięcia niektórych pozytywnych przypadków. Dokładność tej sieci wyniosła 98,75%, a F1-score 98,74%, co wskazuje na ograniczoną równowagę tego modelu w porównaniu do innych.

InceptionResNetV2 uzyskuje bardziej stabilne wyniki, z precyzją wynoszącą 99,72% oraz czułością na poziomie 98,63%. Dokładność tej sieci wyniosła 99,18%, a F1-score 99,18%, co wskazuje na jej solidne parametry wydajności. Niemniej jednak nieco niższa czułość w porównaniu do modeli DenseNet201 i ResNet50 może być pewną wadą tej architektury.

Najlepsze wyniki osiągnęła sieć DenseNet201, która uzyskała najwyższą dokładność (99,61%), precyzję (99,95%) oraz F1-score (99,61%). Czułość tej sieci wyniosła 99,27%, co wskazuje na jej doskonałą zrównoważoność i efektywność w identyfikacji zarówno pozytywnych, jak i negatywnych przypadków. W odniesieniu do przeprowadzonej analizy można stwierdzić, że sieć DenseNet201 jest najbardziej wszechstronną i skuteczną spośród wszystkich badanych modeli, co czyni ją najlepszym wyborem do zadań klasyfikacji obrazów.

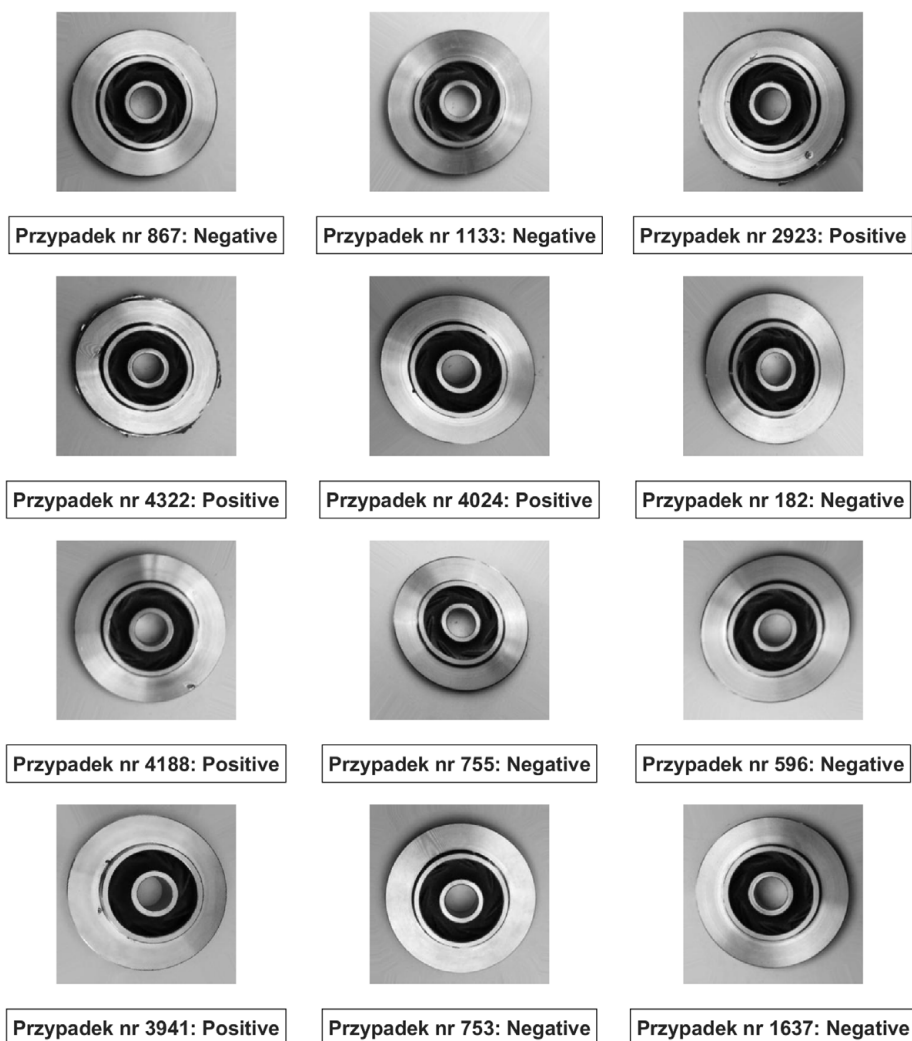
Warto zauważyć, że w badaniach przeprowadzonych przez Gupta i in. [175] model ResNet50 został uznany za najlepszy pod względem identyfikacji defektów. W ramach niniejszej analizy ResNet50 również został zastosowany jako punkt odniesienia. Wyniki sugerują jednak, że bardziej złożone modele, takie jak DenseNet201, mogą przewyższać ResNet50 pod względem dokładności, precyzji i ogólnej skuteczności, co wskazuje na ich potencjalną przewagę w zadaniach wymagających wysokiej precyzji klasyfikacji.

5.1.3. Ocena jakości wyrobów gotowych na podstawie wykrywania wad w odlewach

W wyniku porównania różnych architektur sieci CNN, w tym ResNet50, ResNet101, Darknet53, Xception, InceptionResNetV2 oraz DenseNet201 najlepsze wyniki osiągnięto dla sieci DenseNet201, która okazała się najbardziej

skuteczna w wykrywaniu wad. W związku z powyższym do oceny jakości wyrobów gotowych na podstawie wykrywania wad w odlewach przy użyciu zaawansowanych algorytmów klasyfikacji obrazów wykorzystano sieć DenseNet201.

Wyniki działania algorytmu przedstawiono dla 12 losowo wybranych przypadków, co przedstawia ryc. 5.4. Działanie klasyfikatora zostało zaprezentowane w postaci podpisu pod rysunkiem, dla którego wykonano analizę. Opracowany algorytm 5.1 przedstawia sposób generowania obrazów z wykorzystaniem sieci DenseNet201. Działanie klasyfikatora jest poprawne, co można zaobserwować na przedstawionych poniżej przykładach.



Ryc. 5.4. Wyniki działania algorytmu

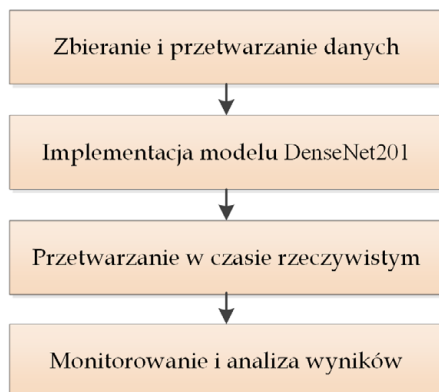
Algorytm 5.1. Generowanie obrazów i ich klasyfikacja

```
% Losowo wybierz 12 obrazów testowych
numImages = 12;
randomIndices = randperm(Ntest, numImages);
% Przygotuj rysunek do wyświetlania wyników
figure;
for i = 1:numImages
    % Wczytaj obraz testowy i jego etykietę
    testImage = readimage(testSet, randomIndices(i));
    testLabel = testSet.Labels(randomIndices(i));
    % Wyświetl obraz testowy
    subplot(4, 3, i); % Rozmieść obrazy w siatce 4x3
    imshow(testImage);
    % Utwórz augmentedImageDatastore, aby automatycznie zmieniać
    rozmiar obrazu podczas wyodrębniania cech obrazu za pomocą aktywacji.
    ds = augmentedImageDatastore(imageSize, testImage,
    'ColorPreprocessing', 'gray2rgb');
    % Wyodrębnij cechy obrazu za pomocą CNN densenet201 (net)
    imageFeatures = activations(net, ds, featureLayer, 'OutputAs',
    'columns');
    % Wykonaj predykcję za pomocą klasyfikatora
    predictedLabel = predict(classifier, imageFeatures,
    'ObservationsIn', 'columns');
    % Dodaj przewidywaną etykietę jako tytuł do podwykresu
    titleStr = sprintf('Przypadek nr %d: %s', randomIndices(i),
    char(predictedLabel));
    % Dodaj tekst poniżej obrazu
    xlabel(titleStr, 'FontSize', 10, 'FontWeight', 'bold',
    'BackgroundColor', 'white', 'EdgeColor', 'black', 'Interpreter',
    'none');
end
```

Na podstawie prowadzonych analiz w tym podrozdziale oceniano skuteczność różnych sieci CNN w klasyfikacji obrazów odlewów na dwie kategorie: bez wad i z wadami. Wykorzystano metodę transfer learning, adaptując wcześniej wytrenowane modele CNN, takie jak ResNet50, Darknet53, i DenseNet201. Najlepsze wyniki uzyskała sieć DenseNet201, która wykazała się najwyższą dokładnością i precyzją w wykrywaniu defektów. Implementacja modelu DenseNet201 w procesie kontroli jakości może znacząco poprawić jakość i efektywność produkcji poprzez automatyczne i szybkie wykrywanie wadliwych odlewów.

5.1.4. Koncepcja implementacji modelu do wykrywania wad w odlewach w kontroli jakości

Automatyczne wykrywanie wad w produktach na liniach produkcyjnych ma istotne znaczenie dla poprawy jakości i efektywności procesów produkcyjnych. W oparciu o wytrenowany klasyfikator DenseNet201, który wykazał najwyższą dokładność (99,61%) w zadaniu wykrywania defektów, możliwe jest stworzenie zaawansowanego systemu automatycznej kontroli jakości. Implementacja takiego systemu wymaga skrupulatnego przygotowania danych, integracji modelu z linią produkcyjną oraz monitorowania wyników w czasie rzeczywistym. Schemat koncepcji implementacji modelu do wykrywania wad w produktach na linii produkcyjnej można przedstawić w postaci głównych etapów, które obejmują zbieranie i przetwarzanie danych, konfigurację modelu, przetwarzanie w czasie rzeczywistym oraz zarządzanie wynikami (ryc. 5.5).



Ryc. 5.5. Schemat implementacji modelu do wykrywania wad w produktach na linii produkcyjnej

Należy podkreślić, że jest to ramowa koncepcja implementacji, która może stanowić podstawę do wdrożenia systemu w rzeczywistych warunkach produkcyjnych. Poszczególne elementy tego rozwiązania mogą zostać dostosowane do specyficznych wymagań i warunków danej linii produkcyjnej, co pozwoli na optymalne wykorzystanie modelu do konkretnych potrzeb przedsiębiorstwa. Warto również zaznaczyć, że przedstawiona koncepcja nie zawiera wszystkich szczegółów technicznych, które mogłyby być potrzebne podczas pełnej implementacji systemu w praktyce.

Pierwszy krok to zbieranie i przetwarzanie danych. Niezbędne jest zainstalowanie wysokiej jakości kamery przemysłowej o wysokiej rozdzielczości,

co najmniej 8 megapikseli. Kamera powinna być zamontowana w strategicznym miejscu linii produkcyjnej, aby umożliwić rejestrowanie szczegółowych obrazów każdego produktu. Ważne jest, aby była umieszczona pod odpowiednim kątem, minimalizując zniekształcenia obrazu.

Równie istotne jest zapewnienie stabilnych warunków oświetleniowych. Wykorzystanie stałych źródeł światła, takich jak lampy LED, zapewni równomierne oświetlenie bez migotania, co minimalizuje refleksy i cienie. Kamera przemysłowa powinna rejestrować obrazy każdego produktu przechodzącego przez linię produkcyjną. Obrazy muszą być zapisywane w formacie zgodnym z wymaganiami modelu.

Kolejnym krokiem jest przygotowanie danych do analizy, w tym zastosowanie technik augmentacji, które mogą zwiększyć różnorodność danych i poprawić wydajność modelu. Techniki, takie jak losowe obracanie, przycinanie, zmiana jasności, kontrastu i kolorów, zwiększają różnorodność danych i poprawiają wydajność modelu. Wykorzystanie bibliotek, takich jak *Keras ImageDataGenerator*, umożliwia automatyczne augmentowanie obrazów podczas uczenia modelu. Przykładowy zapis algorytmu pokazujący augmentowanie danych przedstawia algorytm 5.2.

Algorytm 5.2. Augmentacja obrazów

```
# Instalacja TensorFlow i Keras
!pip install tensorflow keras
# Importowanie potrzebnych modułów
import tensorflow as tf
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
import matplotlib.pyplot as plt
# Definicja parametrów augmentacji
datagen = ImageDataGenerator(
    rotation_range=40,          # Losowe obracanie w zakresie 40 stopni
    width_shift_range=0.2,      # Przesunięcie w poziomie do 20%
                                # szerokości obrazu
    height_shift_range=0.2,     # Przesunięcie w pionie do 20%
                                # wysokości obrazu
    shear_range=0.2,           # Przekształcenie perspektywiczne do 20%
    zoom_range=0.2,            # Zoom do 20%
    horizontal_flip=True,       # Losowe odbijanie w poziomie
    fill_mode='nearest'         # Tryb wypełniania pustych pikseli po
                                # przekształceniach
)
```

Algorytm 5.2. Augmentacja obrazów, cd.

```
# Wczytanie przykładowego obrazu
img = tf.keras.preprocessing.image.load_img('path_to_image.jpg')
x = tf.keras.preprocessing.image.img_to_array(img)
x = x.reshape((1,) + x.shape) # Zmiana rozmiaru na (1, height,
width, channels)
# Generowanie augmentowanych obrazów
i = 0
for batch in datagen.flow(x, batch_size=1):
    plt.figure(i)
    imgplot = plt.imshow(tf.keras.preprocessing.image.
array_to_img(batch[0]))
    i += 1
    if i % 4 == 0: # Generowanie czterech obrazów
        break
plt.show()
```

Zebrane obrazy powinny być przechowywane w dobrze zorganizowanej bazie danych, umożliwiającej szybki dostęp i przetwarzanie. Każdy obraz powinien być opatrzony metadanymi, takimi jak czas rejestracji, identyfikator produktu i etykieta (z wadą lub bez wad). Baza danych może być zrealizowana za pomocą różnych technologii, takich jak SQL, NoSQL lub systemy plików z odpowiednią strukturą katalogów.

W celu implementacji modelu sieci neuronowej należy zainstalować i skonfigurować platformę programistyczną (strukturę aplikacyjną) do głębokiego uczenia, taką jak np. TensorFlow lub PyTorch (to dwa najpopularniejsze frameworki do głębokiego uczenia). Następnie należy zaimportować wytrenowany wcześniej model DenseNet201, przygotowany do klasyfikacji dwóch typów wyjść – obraz z wadami oraz bez wad. Matlab umożliwia eksportowanie wytrenowanych modeli do formatów kompatybilnych z innymi językami programowania, co ułatwia ich integrację.

Przetwarzanie w czasie rzeczywistym wymaga integracji kamery przemysłowej z systemem komputerowym, który będzie odpowiedzialny za odbieranie strumienia wideo. Biblioteki takie jak OpenCV mogą być użyte do przetwarzania strumienia wideo na pojedyncze klatki, które będą analizowane przez model DenseNet201. System powinien automatycznie wyodrębniać klatki ze strumienia wideo w regularnych odstępach czasu lub w odpowiedzi na określone zdarzenia, takie jak wykrycie produktu na linii produkcyjnej. Poniżej znajduje się przykład kodu w Pythonie (algorytm 5.3), który ilustruje,

jak może to zostać zrobione. W danym kodzie sesja MATLABa jest inicjalizowana, a następnie obraz z kamery jest przetwarzany w czasie rzeczywistym. Obrazy są przeskalowywane, konwertowane do formatu RGB i normalizowane. Następnie są przekazywane do funkcji MATLABa *predict_pckniecie*, która wykonuje klasyfikację. Wynik klasyfikacji jest wyświetlany na obrazie.

Algorytm 5.3. Przetwarzanie obrazów w języku Python

```
import cv2
import numpy as np
import matlab.engine
# Inicjalizacja sesji MATLABa
eng = matlab.engine.start_matlab()
# Inicjalizacja kamery
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    # Przeskalowanie obrazu do 227x227 pikseli
    resized_frame = cv2.resize(frame, (227, 227))
    # Konwersja do formatu RGB
    rgb_frame = cv2.cvtColor(resized_frame, cv2.COLOR_BGR2RGB)
    # Normalizacja pikseli
    normalized_frame = rgb_frame / 255.0
    # Konwersja obrazu do formatu odpowiedniego dla MATLABa
    matlab_frame = matlab.double(normalized_frame.tolist())
    # Przewidywanie klasy obrazu za pomocą modelu w MATLABie
    predictions = eng.predict_pckniecie(matlab_frame)
    class_id = np.argmax(predictions)
    # Wyświetlenie wyniku na obrazie
    label = "Positive" if class_id == 1 else "Negative"
    cv2.putText(frame, label, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0, 255, 0), 2)
    # Wyświetlenie obrazu
    cv2.imshow('Detection', frame)
    # Zakończenie pętli po naciśnięciu klawisza 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

System powinien być wyposażony w mechanizmy alarmowe, które powiadają operatorów o wykryciu pęknięć. Powiadomienia mogą mieć formę wizualną, na przykład komunikatów wyświetlanych na ekranie, lub dźwiękową, taką jak alarm. Warto również zintegrować system wykrywania z mechanizmami sortowania, które automatycznie usuwają produkty z wykrytymi defektami z linii produkcyjnej. Może to obejmować ramiona robotyczne lub systemy taśmowe, które przekierowują wadliwe produkty.

Aby skutecznie monitorować wydajność systemu, wyniki klasyfikacji wraz z obrazami powinny być zapisywane do dziennika, umożliwiającego późniejszą analizę i ewaluację. Poniższy przykład pokazuje, jak zapisać wyniki do pliku CSV (algorytm 5.4).

Algorytm 5.4. Zapisywanie danych do pliku CSV w języku Python

```
import csv
from datetime import datetime
# Inicjalizacja pliku CSV
with open('results.csv', mode='w') as file:
    writer = csv.writer(file)
    writer.writerow(['Timestamp', 'Product ID', 'Prediction'])
# Funkcja zapisywania wyników
def save_result(product_id, prediction):
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    with open('results.csv', mode='a') as file:
        writer = csv.writer(file)
        writer.writerow([timestamp, product_id, prediction])
# Przykład użycia
save_result('12345', 'Positive')
```

System powinien regularnie generować raporty dotyczące liczby wykrytych defektów, wskaźników precyzji i czułości oraz innych miar skuteczności. Tworzenie wykresów i tabel przedstawiających wyniki analizy umożliwia łatwe monitorowanie wydajności systemu i identyfikację obszarów do poprawy. Można to osiągnąć za pomocą narzędzi, takich jak *pandas* i *matplotlib*. Przykładowy kod w Pythonie przedstawiono jako algorytm 5.5.

Algorytm 5.5. Tworzenie raportu w języku Python

```
import pandas as pd
import matplotlib.pyplot as plt
# Wczytanie wyników z pliku CSV
df = pd.read_csv('results.csv')
# Generowanie raportów
report = df.groupby('Prediction').size()
print(report)
# Tworzenie wykresu
report.plot(kind='bar')
plt.xlabel('Klasy')
plt.ylabel('Liczba wykryć')
plt.title('Raport wykrytych defektów')
plt.show()
```

Ostatecznie, aby system był skuteczny w dłuższej perspektywie, konieczna jest jego regularna optymalizacja i utrzymanie. Model powinien być regularnie aktualizowany na podstawie nowych danych z linii produkcyjnej, co można osiągnąć poprzez okresowe retrenowanie (ponowne trenowanie) modelu. Gromadzenie informacji zwrotnej od operatorów i inżynierów pozwala na identyfikację potencjalnych problemów i wprowadzenie usprawnień. Ponadto regularna konserwacja kamer oraz systemów komputerowych zapewnia nieprzerwane działanie systemu w optymalnych warunkach. Ciągłe monitorowanie stanu technicznego sprzętu i natychmiastowe reagowanie na wszelkie problemy pomaga uniknąć przestojów i utraty danych.

Implementacja opisanego systemu wykrywania wad w odlewach w kontroli jakości, opartego na wcześniej wytrenowanym modelu DenseNet201, stanowi istotną koncepcję, którą można dostosować do specyficznych potrzeb i warunków produkcyjnych. Dzięki swojej elastyczności model ten może być modyfikowany i optymalizowany w zależności od charakterystyki produkcji, rodzaju wykrywanych wad, dostępnych zasobów technologicznych oraz specyficznych wymagań danego przedsiębiorstwa.

Zastosowanie takiego systemu niesie ze sobą szereg korzyści, które mogą znacząco wpłynąć na poprawę jakości wyrobów, zwiększenie efektywności produkcji oraz wzmocnienie konkurencyjności firmy na rynku. Automatyzacja procesu wykrywania wad pozwala na szybkie i dokładne identyfikowanie defektów, co zmniejsza ryzyko przekazania wadliwych produktów do klienta końcowego. Ponadto zintegrowanie systemu z linią produkcyjną umożliwia bieżące monitorowanie procesów i natychmiastowe

reagowanie na wszelkie nieprawidłowości, co w efekcie redukuje koszty związane z odpadami i przeróbkami.

Koncepcja ta oferuje również możliwość dostosowania do zmieniających się warunków i wymagań produkcyjnych. W miarę rozwoju technologii oraz pojawiania się nowych typów defektów, system można aktualizować, wprowadzając nowe algorytmy i techniki przetwarzania danych, co zapewnia jego długotrwałą skuteczność. Dodatkowo, dzięki skalowalności modelu, możliwe jest jego zastosowanie zarówno w małych zakładach produkcyjnych, jak i na dużych liniach produkcyjnych, gdzie wymaga się analizy dużych wolumenów danych.

Warto jednak zaznaczyć, że przedstawiona koncepcja jest ramowa i nie zawiera wszystkich szczegółów technicznych, które mogłyby być potrzebne podczas pełnej implementacji systemu. Ostateczna realizacja projektu wymaga dokładnego planowania, szczegółowej analizy wymagań oraz współpracy między zespołem technicznym a personelem odpowiedzialnym za procesy produkcyjne. Niemniej jednak ta elastyczna koncepcja stanowi solidną podstawę do wdrożenia zaawansowanego systemu kontroli jakości, który może przyczynić się do znaczącej poprawy wyników produkcyjnych i jakościowych w nowoczesnych zakładach produkcyjnych.

5.2. Wykrywanie błędów kształtu na linii produkcyjnej – problem klasyfikacyjny

Automatyzacja i optymalizacja kontroli jakości na linii produkcyjnej ma bezpośredni wpływ na efektywność produkcji oraz na zadowolenie klientów. Wykrywanie błędów kształtu w procesach produkcyjnych stanowi istotny element w zapewnieniu wysokiej jakości wyrobów gotowych. Problem ten nabiera szczególnego znaczenia w kontekście dynamicznych linii produkcyjnych, gdzie elementy przemieszczają się z dużą prędkością na taśmociągach, co utrudnia precyzyjną inspekcję wizualną. Tradycyjne metody wykrywania wad często nie są wystarczająco skuteczne w takich warunkach, co wymaga zastosowania zaawansowanych technologii, takich jak sztuczne sieci neuronowe.

W odniesieniu do dynamicznych linii produkcyjnych, gdzie czas reakcji jest istotny, wykorzystanie ANN do analizy obrazów pozwala na utrzymanie wysokiego standardu jakości wyrobów bez spowalniania procesu produkcji. Dodatkowo systemy oparte na ANN mogą być stopniowo doskonalone poprzez dalsze uczenie modelu na nowych danych, co sprawia, że są one elastyczne i mogą się dostosowywać do zmieniających się warunków produkcyjnych.

Integracja sztucznych sieci neuronowych w procesach produkcyjnych wpisuje się w szerszy trend automatyzacji i wdrażania nowoczesnych technologii w przemyśle. Takie podejście nie tylko zwiększa konkurencyjność przedsiębiorstw, ale również prowadzi do redukcji kosztów operacyjnych i zwiększenia wydajności. Zastosowanie ANN do wykrywania błędów kształtu na liniach produkcyjnych może zatem stanowić nie tylko innowacyjne rozwiązanie technologiczne, ale również strategiczny krok w kierunku optymalizacji procesów produkcyjnych na skalę przemysłową.

W niniejszym podrozdziale przedstawione zostanie zastosowanie sztucznych sieci neuronowych do wykrywania błędów kształtu na liniach produkcyjnych, gdzie obrazy obiektów są rozmyte z powodu ruchu na taśmociągu. Omówiony zostanie proces tworzenia bazy danych obrazów oraz trenowania klasyfikatora, który identyfikuje obiekty o nieprawidłowym kształcie, oraz metoda lokalizacji błędów w czasie rzeczywistym.

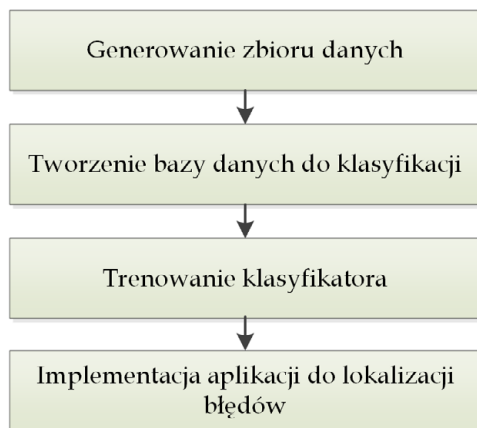
Zastosowanie takiego typu rozwiązania ma istotne znaczenie z kilku powodów. Po pierwsze, automatyzacja procesu wykrywania błędów kształtu znacząco redukuje potrzebę manualnej inspekcji, co przekłada się na wzrost efektywności produkcji. Dzięki ANN możliwe jest dokładne i szybkie wykrywanie defektów, które mogą zostać przeoczone przez tradycyjne metody. Po drugie, technologia ta umożliwia monitorowanie jakości w czasie rzeczywistym, co pozwala na natychmiastowe reagowanie na wykryte nieprawidłowości, minimalizując w ten sposób potencjalne straty produkcyjne.

5.2.1. Metodyka prowadzonych badań oraz przygotowanie danych do modelowania

Celem prowadzonych w tej części badań było opracowanie i wdrożenie systemu opartego na ANN, który umożliwia automatyczne wykrywanie błędów kształtu na linii produkcyjnej. W szczególności skupiono się na identyfikacji obiektów o kształcie koła, które zostały uznane za niepożądane w kontekście procesu produkcyjnego. System miał na celu nie tylko klasyfikację tych obiektów jako błędnych, ale także precyzyjne określenie ich lokalizacji na obrazach, co jest ważne dla poprawy jakości i efektywności produkcji w warunkach rzeczywistych. Poprzez zastosowanie ANN dążono do stworzenia rozwiązania, które pozwoli na szybkie i dokładne wykrywanie defektów, eliminując potrzebę manualnej inspekcji i minimalizując potencjalne straty produkcyjne.

Badania koncentrowały się na wykrywaniu obiektów o kształcie koła na linii produkcyjnej, które uznano za niepożądane. Schemat koncepcji

prorowadzonych badań można przedstawić w postaci głównych etapów, które zostały przedstawione na ryc. 5.6.



Ryc. 5.6. Schemat prowadzonych badań

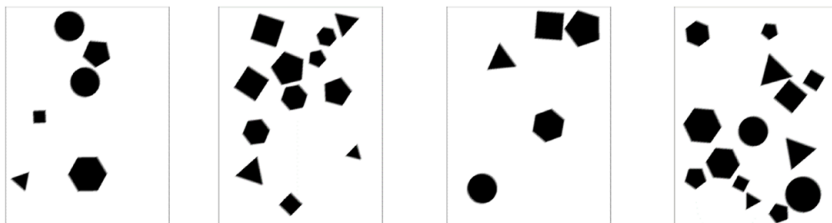
Wygenerowanie zestawu danych odbywało się za pomocą skryptu *sim_shapes* [181], który tworzył obrazy zawierające różne kształty geometryczne. Główne elementy skryptu obejmowały:

- tworzenie obrazów binarnych z wybranymi kształtami;
- dodawanie rozmycia symulującego efekty wynikające z ruchu na taśmociągu;
- konwersja obrazów do formatu RGB w celu przygotowania ich do dalszej obróbki.

Ze względu na ruch przedmiotów poruszających się po taśmociągu, jak również różnych warunków oświetleniowych, obraz na zdjęciu może ulec rozmyciu. Można go porównać z poruszonym zdjęciem. W związku z tym wygenerowane dane przedstawiają rozmyte na obrazie figury. Ten krok pozwolił na uzyskanie realistycznych obrazów, które odzwierciedlały warunki panujące w rzeczywistych systemach produkcyjnych. Na tym etapie wygenerowano 1400 obrazów, każdy z nich o rozdzielczości: 1197×1380. Przykładowe obrazy z otrzymanej bazy danych zostały przedstawione na ryc. 5.7. Obiekty w kształcie koła uznano za błędne, a wygenerowane obrazy posłużyły jako baza danych do dalszych analiz.

W kolejnym etapie stworzono bazę danych obrazów, w której etykiety przypisywano na podstawie obecności lub braku obiektów w kształcie koła. Do tego celu wykorzystano skrypt (algorytm 5.6) z funkcją *Klasyfikacja* (algorytm 5.7), który przetwarzał każdy obraz, identyfikując obiekty o kształcie koła i zapisując odpowiednie etykiety w pliku *.csv. Obrazy z wykrytymi obiektami

w kształcie koła oznaczano jako klasa pozytywna (1), a pozostałe jako klasa negatywna (0).

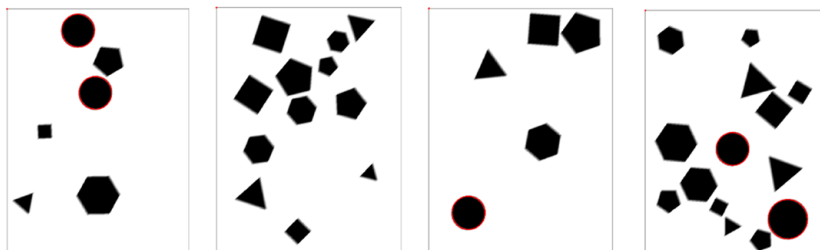


Ryc. 5.7. Przykładowe obrazy z otrzymanej bazy danych

Skrypt *Klasyfikacja* działa w kilku krokach:

- wczytywanie i wyświetlanie obrazu;
- konwersja obrazu do skali szarości oraz jego binaryzacja;
- inwersja binarnego obrazu oraz wykrywanie konturów obiektów;
- analiza właściwości obiektów i klasyfikacja ich kształtu na podstawie kryteriów geometrycznych, takich jak stosunek boków prostokąta opisującego obiekt, jego okrągłość oraz inne cechy;
- oznaczanie czerwonym okręgiem wykrytych obiektów w kształcie koła na obrazie oraz zapisanie wyników.

Przykłady działania skryptu na obrazach z ryc. 5.7 przedstawiono na ryc. 5.8.



Ryc. 5.8. Przykładowe wyniki działania funkcji *Klasyfikacja*

Algorytm 5.6. Tworzenie nowej bazy danych z etykietami

```
% Ścieżka do katalogu z obrazami
inputDir = 'baza_danych';
outputDir = 'new_images';
labelFile = 'labels.csv';
% Upewnij się, że katalog wyjściowy istnieje
if ~exist(outputDir, 'dir')
    mkdir(outputDir);
end
% Otwórz plik do zapisu etykiet
fid = fopen(labelFile, 'w');
fprintf(fid, 'Image,Label\n'); % Nagłówek pliku CSV
% Iteracja przez wszystkie pliki w katalogu „dane”
for i = 1:1400
    % Nazwa pliku
    imageFileName = sprintf('Obraz_%d.png', i); % Zakładamy, że
    obrazy są w formacie PNG
    imageFilePath = fullfile(inputDir, imageFileName);
    % Sprawdź, czy plik istnieje
    if exist(imageFilePath, 'file')
        % Przetwórz obraz i zapisz wynik
        W = Klasyfikacja(imageFilePath);
        % Wyciągnij nazwę pliku bez rozszerzenia
        [~, fileName, ~] = fileparts(imageFileName);
        % Zapisz przetworzony obraz
        outputFilePath = fullfile(outputDir, [fileName, '_new.png']);
        saveas(gcf, outputFilePath);
        % Zapisz etykietę: 1 jeśli są koła, 0 jeśli nie
        if any(W == 1)
            label = 1;
        else
            label = 0;
        end
        fprintf(fid, '%s,%d\n', imageFileName, label);
        % Zamknij wszystkie otwarte figury, aby nie zapełnić pamięci
        close all;
    else
        warning('Plik %s nie istnieje.', imageFilePath);
    end
end
% Zamknij plik z etykietami
fclose(fid);
disp('Przetwarzanie zakończone!');
```


Algorytm 5.7. Funkcja Klasyfikacja

```
function W = Klasyfikacja(ImageFile)
% Krok 1: Wczytanie obrazu
RGB = imread(ImageFile);
figure,
imshow(RGB),
title('Oryginalny Obraz');
% Krok 2: Konwersja obrazu z RGB na skalę szarości
GRAY = rgb2gray(RGB);
figure,
imshow(GRAY),
title('Obraz w Skali Szarości');
% Krok 3: Progowanie obrazu - Konwersja obrazu do postaci
czarno-białej
% w celu przygotowania do śledzenia konturów przy użyciu
bwboundaries.
threshold = graythresh(GRAY);
BW = imbinarize(GRAY, threshold);
figure,
imshow(BW),
title('Obraz Binarny');
% Krok 4: Odwrócenie obrazu binarnego
BW = ~BW;
figure,
imshow(BW),
title('Odwrócony Obraz Binarny');
% Krok 5: Znajdzenie konturów - Skoncentrowanie się wyłącznie na
zewnątrznych konturach.
% Opcja ,noholes' przyspiesza przetwarzanie, zapobiegając
przeszukiwaniu przez bwboundaries konturów wewnętrznych.
[B,L] = bwboundaries(BW, 'noholes');
% Krok 6: Określenie właściwości obiektów
STATS = regionprops(L, 'BoundingBox', 'Extent', 'Centroid', 'Area',
'Perimeter');
% Krok 7: Klasyfikacja kształtów według ich właściwości (tylko koła)
figure,
imshow(RGB),
title('Wykryte Koła');
hold on
W = zeros(1, length(STATS)); % Inicjalizacja tablicy W
for i = 1:length(STATS)
```

Algorytm 5.7. Funkcja Klasyfikacja, cd.

```

% Sprawdzenie, czy obiekt jest kołem
boundingBoxRatio = abs(STATS(i).BoundingBox(3) - STATS(i).
BoundingBox(4)) / STATS(i).BoundingBox(3);
circularity = (4 * pi * STATS(i).Area) / (STATS(i).Perimeter^2);

isCircle = (boundingBoxRatio < 0.1) && (STATS(i).Extent < 1) &&
(circularity > 0.85);
if isCircle
    W(i) = 1; % Wartość 1 dla występowania koła
    centroid = STATS(i).Centroid;
    % Rysowanie czerwonej linii wzdłuż granic koła
    boundary = B{i};
    plot(boundary(:,2), boundary(:,1), 'r', 'LineWidth', 2);
end
end
hold off
return

```

Po utworzeniu bazy danych przystąpiono do trenowania klasyfikatora na przygotowanych obrazach. Wykorzystano do tego wstępnie wytrenowaną sieć AlexNet, której końcowe warstwy zostały dostosowane do liczby klas w zbiorze danych. Opis tej sieci został przedstawiony wcześniej w podrozdziale 3.4. Ponieważ zadaniem było rozpoznanie obiektów z różnych klas, liczba neuronów w warstwie wyjściowej została dostosowana do liczby unikalnych etykiet w zestawie danych, co wynosiło odpowiednią liczbę klas (2). Oryginalne warstwy końcowe sieci AlexNet zostały zastąpione nową warstwą w pełni połączoną z liczbą neuronów równą liczbie klas oraz nową warstwą klasyfikacyjną, odpowiedzialną za przypisanie etykiety do każdego obrazu. W kodzie zastosowano optymalizator *sgdm* (ang. *stochastic gradient descent with momentum*), który jest domyślnym wyborem dla trenowania sieci AlexNet w MATLAB-ie. Optymalizator ten korzysta z gradientu stochastycznego z momentem, co pozwala na stabilizację procesu uczenia i przyspieszenie konwergencji w porównaniu ze standardowym gradientem stochastycznym.

Następnie zdefiniowano parametry trenowania sieci. Ustalono wielkość partii danych (ang. *MiniBatchSize*) na 32 obrazy, co oznacza, że podczas jednej iteracji algorytmu uczenia model przetwarzał 32 obrazy. Proces trenowania był zaplanowany na maksymalnie 100 epok. Współczynnik uczenia ustawiono na 0,0001, co pozwala na delikatne dostosowanie wag i zminimalizowanie

ryzyka nadmiernego dopasowania. Dane były mieszane przed każdą epoką, aby zapewnić losowy rozkład obrazów, co mogło poprawić generalizację modelu.

Dodatkowo co 50 iteracji przeprowadzano walidację modelu na zbiorze walidacyjnym, aby monitorować jego postępy w trakcie uczenia. Ustalono również mechanizm wczesnego zatrzymania, który kończył trenowanie, jeśli dokładność modelu na zbiorze walidacyjnym przestawała się poprawiać przez 8 kolejnych epok. To zapobiegało przeuczeniu modelu, gdy dalsze uczenie nie przynosiło już korzyści.

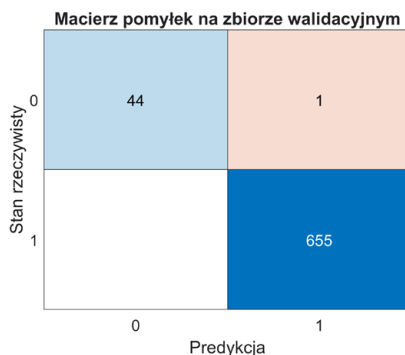
Obrazy wykorzystywane w procesie trenowania były przeskalowane do rozmiarów 227×227 pikseli, co było wymagane przez architekturę AlexNet. Podzielono dane na zestawy do trenowania i walidacji w proporcjach 50 : 50, gdzie obrazy były losowo przypisywane do obu zbiorów.

Model został oceniony przy użyciu danych walidacyjnych (700 obrazów). Ocena skuteczności modelu obejmowała obliczenie miar skuteczności, takich jak dokładność, precyzja, czułość oraz F1-score.

W ostatnim etapie wytrenowany klasyfikator został zaimplementowany w aplikacji, której zadaniem było automatyczne wykrywanie i lokalizowanie obiektów o niepożądanym kształcie na obrazach z linii produkcyjnej. Po sklasyfikowaniu obrazu i wykryciu obiektów w kształcie koła (wynik 1), funkcja *Klasyfikacja* była ponownie wykorzystywana do precyzyjnej identyfikacji i oznaczania miejsce występowania tych obiektów.

5.2.2. Wyniki modelowania klasyfikatora

Do testowania otrzymanego klasyfikatora wykorzystano kolejne 700 obrazów, które nie były wcześniej używane w procesie uczenia. Wskaźnikiem skuteczności dopasowania sieci neuronowej do danych jest macierz pomyłek (ryc. 5.9).

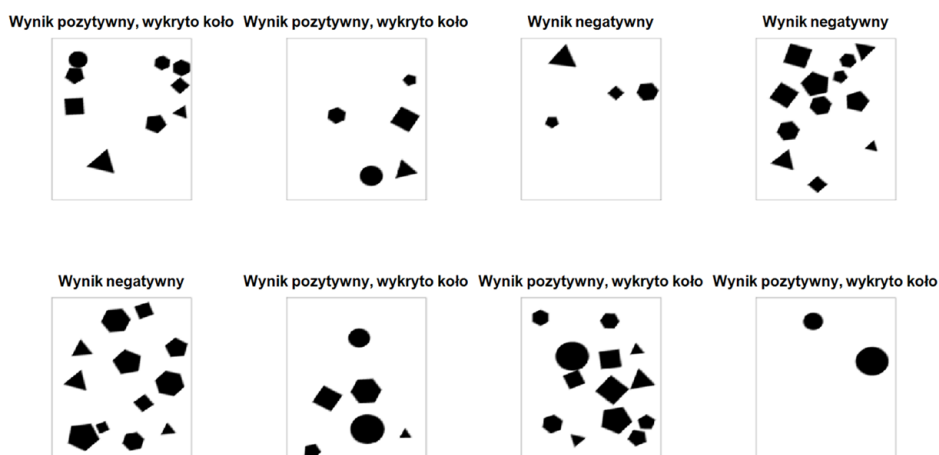


Ryc. 5.9. Macierz pomyłek na zbiorze walidacyjnych klasyfikatora wykrywającego błędy kształtu

Analiza przedstawionej macierzy pomyłek pozwala na ocenę skuteczności klasyfikatora w zadaniu wykrywania błędów kształtu. Macierz wskazuje, że spośród 700 przetestowanych obrazów 655 zostało poprawnie zaklasyfikowanych jako zawierające błędy, podczas gdy 44 obrazy, które były poprawne, zostały również właściwie rozpoznane. Wystąpiła jedna sytuacja, w której prawidłowy obraz został błędnie sklasyfikowany jako zawierający błąd, co oznacza pojedynczy przypadek fałszywie pozytywny. Nie zanotowano żadnych fałszywie negatywnych przypadków, co świadczy o tym, że wszystkie obrazy zawierające błędy zostały poprawnie wykryte.

Na podstawie tych wyników dokładność klasyfikatora wynosi 99,86%, co oznacza, że model bardzo dobrze radzi sobie z poprawnym klasyfikowaniem obrazów. Precyzja modelu, czyli miara tego, jak wiele spośród przewidywanych przez model błędów rzeczywiście jest błędami, wynosi 99,85%. Wysoka czułość na poziomie 100% wskazuje, że model nie przegapił żadnego obrazu z błędem, skutecznie identyfikując wszystkie przypadki. F1-score, będący średnią harmoniczną precyzji i czułości, również osiąga bardzo wysoką wartość 99,92%, co potwierdza zrównoważoną skuteczność modelu w kontekście zarówno wykrywania błędów, jak i unikania fałszywie pozytywnych klasyfikacji. Ogólnie rzecz biorąc, model charakteryzuje się doskonałą efektywnością w wykrywaniu błędów kształtu na podstawie testowanych danych.

Wyniki dla przykładowych obrazów działania klasyfikatora przedstawiono na ryc. 5.10. Działanie klasyfikatora zostało zaprezentowane w postaci podpisu nad rysunkiem, dla którego wykonana została analiza. Jak można zaobserwować na tych przykładach, działanie klasyfikatora jest poprawne.



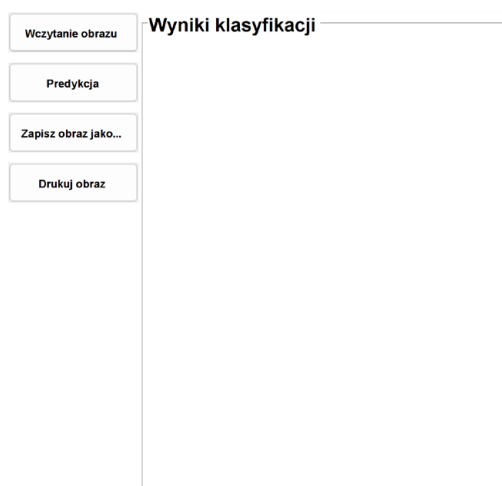
Ryc. 5.10. Wyniki działania klasyfikatora dla wybranych obrazów

5.2.3. Implementacja aplikacji do wykrywania błędów

Po wytrenowaniu klasyfikatora stworzono aplikację, która umożliwia wykrywanie i lokalizację błędnych obiektów na obrazach produkcyjnych. Aplikacja działała na zasadzie:

- klasyfikacji obrazów przy użyciu wytrenowanej sieci, aby zidentyfikować obecność błędów;
- lokalizacji wykrytych obiektów przy użyciu funkcji *Klasyfikacja*, która precyzyjnie oznacza miejsca występowania obiektów w kształcie koła.

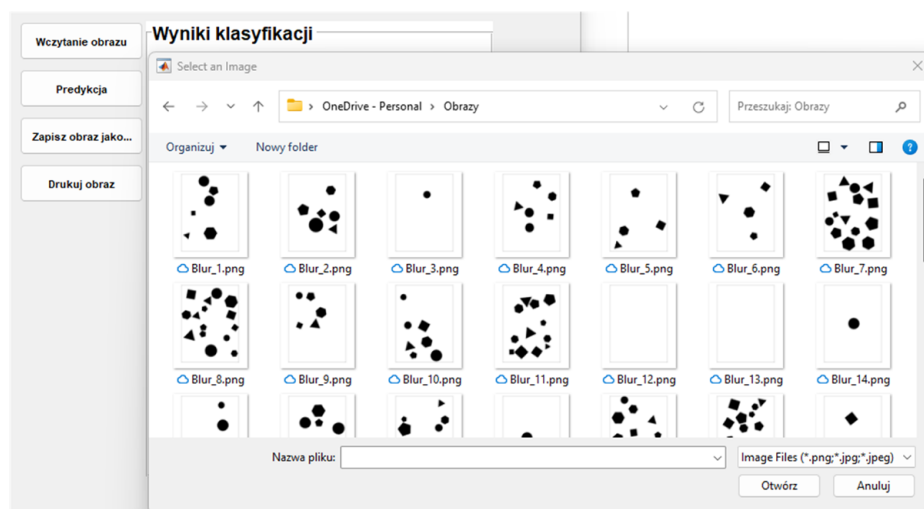
Interfejs aplikacji został zaprojektowany w celu uproszczenia procesu wykrywania błędów kształtu (kształt koła) na obrazach z linii produkcyjnej. Podstawowym celem tej aplikacji było stworzenie narzędzia, które umożliwia użytkownikom, nawet bez specjalistycznej wiedzy technicznej, łatwe wykonywanie analizy obrazów przy użyciu zaawansowanych algorytmów sztucznej inteligencji. Aplikacja ta integruje proces klasyfikacji, lokalizacji oraz wizualizacji wyników w jednym, intuicyjnym środowisku graficznym (ryc. 5.11).



Ryc. 5.11. Interfejs użytkownika aplikacji do wykrywania obiektów o kształcie koła

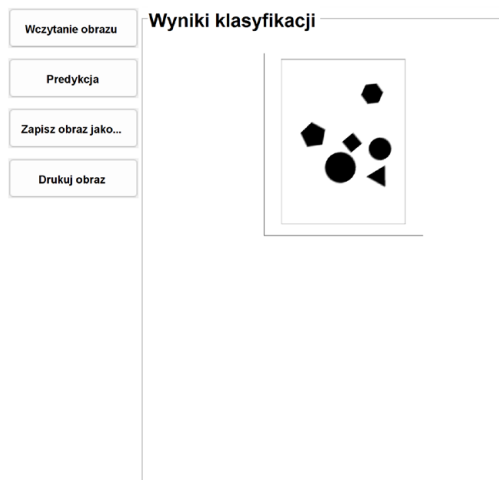
Interfejs użytkownika składa się z kilku głównych elementów, które wspierają proces analizy. Po uruchomieniu aplikacji użytkownik jest kierowany do głównego okna, które oferuje funkcjonalności niezbędne do przetworzenia i analizy obrazów. Na początku użytkownik ma możliwość wczytania obrazu, który pochodzi z lokalnego systemu plików (ryc. 5.12). Obraz ten stanowi dane wejściowe, które będą analizowane pod kątem występowania błędnych

obiektów, takich jak koła, które zgodnie z założeniami są uznawane za niepożądane na linii produkcyjnej. Obrazy, które są wczytywane do aplikacji, mogą być zapisane w formatach graficznych typowych dla plików obrazów. W szczególności aplikacja obsługuje formaty, takie jak: PNG (.png), JPG/JPEG (.jpg, .jpeg). Te formaty są szeroko stosowane w przemyśle oraz w codziennym użytku, co zapewnia, że aplikacja jest kompatybilna z większością plików obrazów generowanych w różnych procesach produkcyjnych.



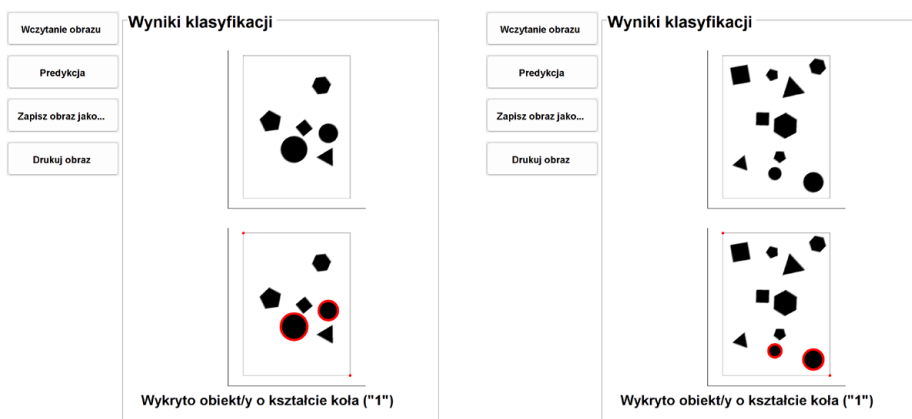
Ryc. 5.12. Interfejs użytkownika – wczytanie danych

Wczytanie pliku w jednym z tych formatów uruchamia proces analizy obrazu, zgodnie z funkcjonalnością aplikacji. Po załadowaniu obrazu (ryc. 5.13) użytkownik ma możliwość uruchomienia procesu klasyfikacji, który jest najważniejszym elementem aplikacji poprzez wciśnięcie przycisku „Predykcja”.



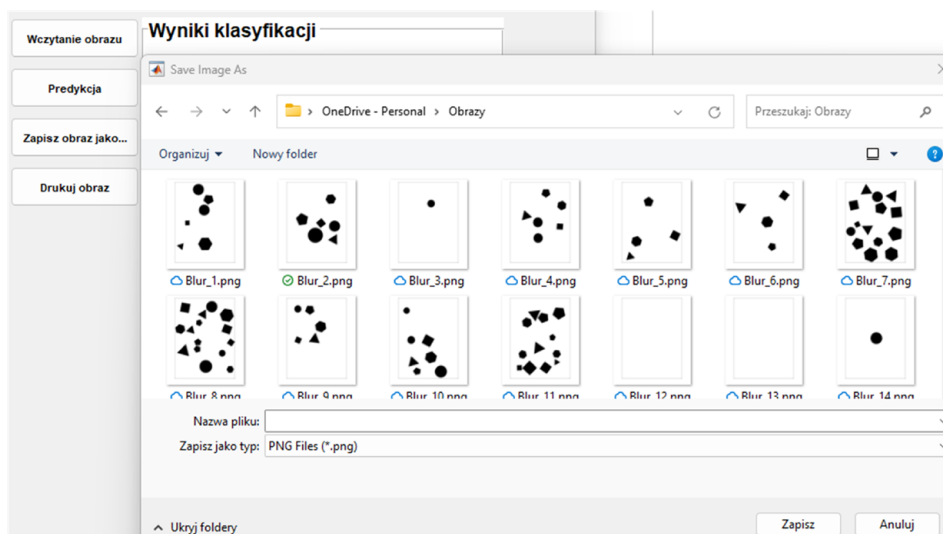
Ryc. 5.13. Interfejs użytkownika – widok po wczytaniu obrazu do analizy

W tle aplikacja korzysta z wcześniej wytrenowanej sieci neuronowej, która analizuje wczytany obraz i ocenia, czy zawiera on obiekty o kształcie koła. Wynik tej klasyfikacji jest natychmiast wyświetlany w interfejsie. Gdy wykryte zostaną obiekty o kształcie koła, użytkownik jest informowany o tym fakcie poprzez odpowiedni komunikat. W przypadku wyniku klasyfikacji 1 wyświetla się komunikat: „Wykryto obiekt/y o kształcie koła („1”)”, w przeciwnym przypadku jest to komunikat: „Nie wykryto obiektów o kształcie koła („0”)”. Dodatkowo aplikacja umożliwia dokładne zlokalizowanie tych obiektów na obrazie. Dzięki zaimplementowanej funkcji lokalizacji *Klasyfikacja* (algorytm 5.7) koła są automatycznie zaznaczane na przetworzonym obrazie, co pozwala na ich łatwą identyfikację. Zaznaczenie to odbywa się poprzez obrysowanie wykrytych obiektów czerwoną linią, co wyróżnia je na tle innych elementów widocznych na obrazie. Przykładowe wyniki działania aplikacji przedstawiono na ryc. 5.14.



Ryc. 5.14. Przykładowe wyniki działania aplikacji do wykrywania obiektów o kształcie koła

W interfejsie użytkownik ma również możliwość zapisania przetworzonego obrazu (ryc. 5.15). Dzięki tej funkcji można zachować wyniki analizy w formacie graficznym na lokalnym dysku, co jest szczególnie użyteczne do dalszej analizy, raportowania lub dokumentacji. Przechowywanie tych danych pozwala na ich późniejsze wykorzystanie, na przykład do porównania z innymi wynikami lub jako dowód poprawności procesu produkcyjnego.



Ryc. 5.15. Interfejs użytkownika – funkcja zapisz jako

Ponadto aplikacja oferuje funkcję bezpośredniego drukowania przetworzonego obrazu. Po dokonaniu analizy i zaznaczeniu błędnych obiektów użytkownik może wydrukować obraz wraz z zaznaczonymi błędami bezpośrednio z poziomu interfejsu. Obraz po kliknięciu w opcję „Drukuj obraz” od razu przesyłany jest do domyślnej drukarki. Ta funkcjonalność jest szczególnie przydatna w środowiskach produkcyjnych, gdzie istnieje konieczność szybkiego przekazywania wyników analizy do odpowiednich działów lub ich fizyczne przechowywanie w dokumentacji. Sposób implementacji opisanego interfejsu graficznego aplikacji do wykrywania błędów przedstawia algorytm 5.8.

Algorytm 5.8. Implementacja interfejsu graficznego aplikacji do wykrywania błędów

```
function simpleUI3(trainedNet)
% Deklaracja zmiennych globalnych
global imageData;
global processedImage;
% Inicjalizacja zmiennych globalnych
imageData = [];
processedImage = [];
load('trainedAlexNet.mat', 'trainedNet'); % Załaduj wytrenowaną sieć
% Tworzenie głównej figury interfejsu użytkownika
f = figure('Name', 'Image Classifier', 'Position', [300, 300, 610,
600]); % Zwiększenie rozmiaru figury
% Dodanie przycisku do załadowania obrazu
uicontrol('Style', 'pushbutton', 'String', 'Wczytanie obrazu', ...
'Position', [20, 500, 130, 40], 'Callback', @loadImage,
FontWeight='bold');
% Dodanie przycisku do uruchomienia klasyfikacji
uicontrol('Style', 'pushbutton', 'String', 'Predykcja', ...
'Position', [20, 450, 130, 40], 'Callback', @classifyImage,
FontWeight='bold');
% Dodanie przycisku do zapisu obrazu
uicontrol('Style', 'pushbutton', 'String', 'Zapisz obraz jako...', ...
'Position', [20, 400, 130, 40], 'Callback', @saveImage,
FontWeight='bold');
% Dodanie przycisku do wydruku obrazu
uicontrol('Style', 'pushbutton', 'String', 'Drukuj obraz', ...
'Position', [20, 350, 130, 40], 'Callback', @printImage,
FontWeight='bold');
```

Algorytm 5.8. Implementacja interfejsu graficznego aplikacji do wykrywania błędów, cd.

```
% Dodanie panelu na wyniki klasyfikacji
resultPanel = uipanel('Parent', f, 'Title', 'Wyniki klasyfikacji', ...
    'FontSize', 14, 'Position', [0.25, 0.1, 0.6, 0.8],
'BackgroundColor', 'white', 'FontWeight', 'bold');
% Pole tekstowe do wyświetlenia wyniku klasyfikacji
resultText = uicontrol('Style', 'text', 'Parent', resultPanel, ...
    'Position', [20, 10, 500, 30], 'FontSize', 12, 'FontWeight',
'bold', ...
    'String', ' ', 'HorizontalAlignment', 'left',
'BackgroundColor', 'white');
% Tworzenie osi do wyświetlania obrazów, domyślnie niewidoczne
axOriginal = axes('Parent', resultPanel, 'Position', [0.2, 0.55,
0.7, 0.4], 'Visible', 'off'); % Oryginalny obraz
axProcessed = axes('Parent', resultPanel, 'Position', [0.2, 0.1,
0.7, 0.4], 'Visible', 'off'); % Przetworzony obraz
% Funkcja do załadowania obrazu
function loadImage(~, ~)
    [fileName, filePath] = uigetfile({'*.png;*.jpg;*.jpeg',
'Image Files'}, 'Select an Image');
    if fileName
        imageData = imread(fullfile(filePath, fileName));
        imshow(imageData, 'Parent', axOriginal);
        axis(axOriginal, 'off'); % Ukryj osie
        set(axOriginal, 'XTick', [], 'YTick', [], 'Box', 'off'); %
Ukryj ticki i ramkę
        set(axOriginal, 'Visible', 'on'); % Ustawienie osi na
widoczne
    end
end
% Funkcja do klasyfikacji obrazu
function classifyImage(~, ~)
    if ~isempty(imageData)
        % Przeskalowanie obrazu do rozmiaru akceptowanego przez sieć
        inputImage = imresize(imageData, [227, 227]);
        % Klasyfikacja obrazu przy użyciu wytrenowanej sieci
        result = classify(trainedNet, inputImage);
        % Wyświetlenie odpowiedniego wyniku
        if result == categorical(1)
            resultText.String = 'Wykryto obiekt/y o kształcie koła („1”)';
            % Zapisanie tymczasowego obrazu
            tempFileName = [tempname, '.png'];
            imwrite(imageData, tempFileName);
```

Algorytm 5.8. Implementacja interfejsu graficznego aplikacji do wykrywania błędów, cd.

```
% Przekazanie ścieżki do pliku do Classify2_modified
i wyświetlenie na osi
processedImage = Classify2_modified(tempFileName,
axProcessed);
axis(axProcessed, 'off'); % Ukryj osie
set(axProcessed, 'XTick', [], 'YTick', [], 'Box', 'off');
% Ukryj ticki i ramkę
set(axProcessed, 'Visible', 'on'); % Ustawienie osi na
widoczne

% Usunięcie tymczasowego pliku po przetworzeniu
delete(tempFileName);
else
    resultText.String = 'Nie wykryto obiektów o kształcie koła
(„0”)';
    % Ukryj przetworzony obraz, jeśli wcześniej był widoczny
    set(axProcessed, 'Visible', 'off');
end
else
    msgbox('Please load an image first.');
```

```
end

% Funkcja do zapisu obrazu
function saveImage(~, ~)
    if ~isempty(processedImage)
        [fileName, filePath] = uiputfile({'*.png', 'PNG Files';
'*.jpg', 'JPEG Files'}, 'Save Image As');
        if fileName
            imwrite(processedImage, fullfile(filePath, fileName));
        end
    else
        msgbox('No processed image to save.');
```

```
end

% Funkcja do wydruku obrazu
function printImage(~, ~)
    if ~isempty(processedImage)
        % Przechodzenie na oś przetworzonego obrazu
        axes(axProcessed);
        % Przechwycenie zawartości osi jako obraz
        frame = getframe(axProcessed);
        imgToPrint = frame.cdata;
```

Algorytm 5.8. Implementacja interfejsu graficznego aplikacji do wykrywania błędów, cd.

```
% Tworzenie tymczasowej figury do drukowania obrazu
hFig = figure('Visible', 'off'); % Ukrycie figury
imshow(imgToPrint, 'Border', 'tight'); % Wyświetlenie
obrazu w figurze
% Drukowanie obrazu bezpośrednio na drukarce
print(hFig, '-dwin'); % Użycie sterownika Windows do druku
% Zamknięcie tymczasowej figury
close(hFig);
else
    msgbox('No processed image to print.');
```

end

end

end

Cały proces korzystania z aplikacji został zaprojektowany w sposób przemyślany, aby zapewnić użytkownikowi intuicyjne i uporządkowane działanie. Użytkownik prowadzi analizę krok po kroku, zaczynając od wczytania obrazu, przez klasyfikację, aż po wizualizację wyników i ich zapis lub wydruk. Aplikacja minimalizuje konieczność dogłębnej znajomości technicznych aspektów działania algorytmów sztucznej inteligencji, co sprawia, że jest narzędziem łatwym w użyciu i dostępnym w codziennej pracy na linii produkcyjnej.

Podsumowując prowadzone analizy i rozważania dotyczące wykrywania błędów kształtu na linii produkcyjnej, można stwierdzić, że zastosowanie sztucznych sieci neuronowych w tym kontekście znacząco może usprawnić procesy produkcyjne. Dzięki wdrożeniu ANN możliwe jest automatyczne i precyzyjne wykrywanie błędnych obiektów (w tym przypadku wyrobów o kształcie koła), co ma istotne znaczenie w dynamicznych warunkach produkcyjnych, gdzie tradycyjne metody kontroli jakości mogą być niewystarczające. System ten nie tylko identyfikuje defekty, ale także lokalizuje je na obrazach, co pozwala na szybkie podjęcie działań korygujących. W efekcie integracja ANN może przyczynić się do podniesienia jakości produkcji, minimalizacji ryzyka wystąpienia wadliwych produktów na rynku oraz zwiększenia ogólnej efektywności linii produkcyjnych. Rozwiązanie to stanowi istotny element w doskonaleniu procesów produkcyjnych, oferując zaawansowane narzędzie do utrzymania wysokich standardów jakości w nowoczesnym przemyśle.

5.3. Określenie twardości po odpuszczaniu wyrobów gotowych – problem regresyjny

Współczesny przemysł metalowy nieustannie ewoluuje, wprowadzając coraz bardziej zaawansowane technologie, aby sprostać wymaganiom rynku i zapewnić produkty najwyższej jakości. W szczególności przemysł obróbki cieplnej stali odgrywa istotną rolę w kształtowaniu właściwości mechanicznych wyrobów, takich jak twardość, wytrzymałość, plastyczność oraz odporność na zużycie. Procesy, takie jak hartowanie i odpuszczanie, są fundamentalnymi etapami obróbki cieplnej, które determinują ostateczne właściwości materiału. Precyzyjne sterowanie tymi procesami jest podstawą w osiągnięciu wysokiej jakości produktów, co ma bezpośredni wpływ na ich trwałość, niezawodność i zdolność do spełniania specyficznych funkcji w złożonych aplikacjach inżynierskich.

Jednym z istotnych wskaźników jakości w procesach obróbki cieplnej stali jest twardość materiału po odpuszczaniu. Twardość ta odgrywa fundamentalną rolę w określaniu wytrzymałości, odporności na zużycie oraz innych krytycznych właściwości mechanicznych wyrobów stalowych. Przykłady takich zastosowań obejmują elementy maszyn, wały napędowe, sprężyny oraz różnorodne narzędzia, gdzie nie tylko wytrzymałość, ale również odpowiednia elastyczność i odporność na pękanie są istotne. Tradycyjne metody przewidywania twardości po odpuszczaniu często polegały na empirycznych zależnościach oraz doświadczeniu inżynierów, co, choć skuteczne, nie zawsze pozwalało na uchwycenie pełnego spektrum złożoności procesów obróbki cieplnej. Zastosowanie sztucznej sieci neuronowej do przewidywania twardości stali po procesie odpuszczania pozwoli prognozować, czy produkt końcowy spełni kryteria jakościowe, bazując na parametrach wejściowych, takich jak typ stali, twardość po hartowaniu, czas czy temperatura odpuszczania.

W niniejszym podrozdziale zostanie omówione zastosowanie sztucznych sieci neuronowych MLP do analizy i doskonalenia procesów obróbki cieplnej stali, ze szczególnym naciskiem na przewidywanie twardości po odpuszczaniu jako istotnego wskaźnika jakości wyrobów metalowych. Kluczowym zadaniem jest identyfikacja przypadków, w których przewidywana twardość jest poniżej wymaganych standardów, co może oznaczać, że produkt nie spełni kryteriów jakościowych. Zastosowanie sztucznej sieci neuronowej MLP pozwala na uwzględnienie wielu zmiennych jednocześnie i analizę ich wzajemnych wpływów na końcowy wynik procesu.

5.3.1. Metodyka prowadzonych badań

Celem tej części badań jest przedstawienie metody wykorzystania sztucznej sieci neuronowej MLP do przewidywania twardości stali po odpuszczaniu. W oparciu o parametry wejściowe procesu, takie jak typ stali, twardość po hartowaniu, czas oraz temperatura odpuszczania, model MLP będzie miał za zadanie oszacować twardość końcową materiału. Algorytmy klasyfikacyjne były testowane przy użyciu publicznie dostępnej bazy danych „Tempering data for carbon and low alloy steels”, która została udostępniona w licencji CC BY-NC-ND 4.0 [182].

Bazę danych przygotował zespół Raiipa Technologies. Zestaw danych zawiera informacje dotyczące 1466 prób odpuszczania, obejmujących różne klasy stali, z dodatkowymi danymi na temat składu chemicznego stali, czasu oraz temperatury odpuszczania, a także twardości stali po hartowaniu i po odpuszczaniu.

Na początkowym etapie analizy baza danych została ograniczona do rekordów, w których była dostępna informacja o twardości początkowej materiału po hartowaniu. Twardość ta stanowiła istotny parametr wejściowy do dalszych analiz, a jej brak uniemożliwiałby prawidłowe przewidywanie twardości po odpuszczaniu. Następnie w celu zwiększenia spójności i wiarygodności wyników modelowania baza danych została zawężona do pięciu typów stali, dla których liczba dostępnych próbek była wystarczająca do przeprowadzenia statystycznie istotnych analiz. Wybrane typy stali i liczba próbek dla każdego z nich przedstawiają się następująco:

- typ stali 0: AISI-SAE 4140 (42CrMo4) – 43 próbki;
- typ stali 1: AISI-SAE 4340 (36CrNiMo4) – 51 próbek;
- typ stali 2: AISI-SAE 5160 – 51 próbek;
- typ stali 3: AISI-SAE 6150 (51CrV4) – 51 próbek;
- typ stali 4: AISI-SAE E52100 (100Cr6) – 51 próbek.

Stal AISI-SAE 4140 (42CrMo4) jest szeroko stosowana w branżach, takich jak motoryzacja, lotnictwo i budownictwo ze względu na doskonałą wytrzymałość i odporność na zużycie. Często poddawana jest obróbce cieplnej, na przykład hartowaniu i odpuszczaniu, aby poprawić jej właściwości mechaniczne. Ten drugi proces znacząco wpływa na jej twardość i wytrzymałość [183]. Znany ze swojej wysokiej wytrzymałości AISI 4340 (36CrNiMo4) jest stosowany w wymagających zastosowaniach, takich jak komponenty lotnicze i motoryzacyjne. Wykazano, że techniki inżynierii powierzchniowej, na przykład azotowanie, a następnie powłoki NiP i AlCrN, znacznie poprawiają jego odporność na zużycie [184]. Proces odpuszczania wpływa również na jego właściwości mechaniczne, przy czym wyższe temperatury odpuszczania generalnie zmniejszają twardość, ale zwiększają odporność na uderzenia [185]. Stal AISI-SAE 5160 ze względu na

wysoką sprężystość i wytrzymałość jest szeroko stosowana w produkcji sprężyn i resorów [186]. Podobnie stal AISI-SAE 6150 (51CrV4) stal chromowo-wanadowa również wykorzystywana w produkcji sprężyn i narzędzi, charakteryzuje się wysoką twardością i wytrzymałością [187]. AISI-SAE E52100 (100Cr6) zaś to wysokowęglowa stal chromowana stosowana głównie w zastosowaniach łożyskowych ze względu na wysoką twardość i odporność na zużycie. W trakcie hartowania stal ulega znacznym zniekształceniom sieci, które ulegają zmniejszeniu podczas odpuszczania, co prowadzi do zmniejszenia tetragonalności martenzytu i transformacji zatrzymanego austenitu. Proces ten zwiększa stabilność wymiarową i właściwości mechaniczne stali, dzięki czemu idealnie nadaje się do precyzyjnych elementów, takich jak łożyska [188].

W ramach przeprowadzonej analizy ograniczono się do wybranych zmiennych wejściowych, które najlepiej odzwierciedlają charakter analizowanego problemu:

- typ stali (0-4);
- twardość po hartowaniu;
- czas odpuszczania (w sekundach);
- temperatura odpuszczania (w stopniach Celsjusza).

Istotnym aspektem bazy danych są zmienne docelowe, które określają twardość stali po procesie odpuszczania. Wyjściem z modelu jest wartość twardości wyrażona w skali Rockwella C (HRC) po zakończeniu procesu odpuszczania.

Aby przewidzieć twardość stali po procesie odpuszczania, stworzono model regresyjny, który został zbudowany z wykorzystaniem sztucznych sieci neuronowych MLP przy użyciu Matlab R2023a, uwzględniając specyficzne wymagania wynikające z ograniczonej liczby danych (247 próbek). Warstwa wejściowa składała się z czterech neuronów (zmienne: typ stali, twardość po hartowaniu, czas odpuszczania, temperatura odpuszczania), podczas gdy warstwa wyjściowa zawierała jeden neuron (twardość stali po odpuszczaniu). Tabela 5.8 przedstawia statystyki opisowe danych wejściowych i wyjściowych.

Tabela 5.8. Statystyki opisowe danych wejściowych i wyjściowych

	Typ stali	Twardość po hartowaniu [HRC]	Czas odpuszczania [s]	Temp. odpuszczania [°C]	Twardość po odpuszczaniu [HRC]
średnia	-	60,52	17566,73	360,69	48,06
odchylenie standardowe	-	3,02	32193,50	195,02	12,67
min	0	46,5	10	100	7,3
mediana	-	61,6	900	300	51,5
max	4	63,3	86400	700	64

W niniejszym badaniu zastosowano sieć neuronową MLP. Wybrano ją ze względu na jej zdolność do modelowania nieliniowych zależności przy jednoczesnym zachowaniu prostoty struktury. Model posiadał jedną warstwę ukrytą, której liczba neuronów była iteracyjnie dostosowywana w celu znalezienia optymalnej konfiguracji, w zakresie od 4 do 20 neuronów. W warstwie ukrytej zastosowano funkcję aktywacji typu sigmoidalnego, co umożliwiło sieci uchwycenie złożonych wzorców w danych. Do trenowania modelu wybrano trzy algorytmy uczenia: LM, BR oraz SCG. Proces trenowania rozpoczęto od domyślnej inicjalizacji wag, co zapewniło stabilne warunki początkowe dla optymalizacji. Funkcję kosztu zdefiniowano jako MSE, który służył do oceny jakości dopasowania modelu. Dane podzielono na trzy zestawy: treningowy (70%), walidacyjny (15%) oraz testowy (15%). Zbiór uczący został wykorzystany do dopasowania modelu, zbiór walidacyjny do monitorowania procesu trenowania w celu zapobiegania przeuczeniu, a testowy posłużył do oceny końcowej skuteczności modelu. Mechanizm wczesnego zatrzymania ustawiono na poziomie 6. Po zakończeniu procesu uczenia model został poddany testom, które polegały na porównaniu przewidywanych wartości twardości po odpuszczaniu z rzeczywistymi wartościami z danych wyjściowych. W ten sposób obliczono metryki jakości modelu (R, MSE, RMSE, RIE, MAPE i MAE), co umożliwiło ocenę jego skuteczności.

5.3.2. Wyniki modelowania twardości po procesie odpuszczania

Podczas modelowania analizowano różne konfiguracje liczby neuronów w warstwie ukrytej (4÷20). Ostatecznie zdecydowano się na zastosowanie warstwy ukrytej składającej się z dziewięciu neuronów, co stanowiło optymalny kompromis pomiędzy zdolnością modelu do uchwycenia złożonych zależności w danych a minimalizacją ryzyka przeuczenia. Wybór tej konfiguracji pozwolił na uzyskanie modelu o wysokiej zdolności generalizacji, przy jednoczesnym ograniczeniu ryzyka nadmiernego dopasowania do danych uczących.

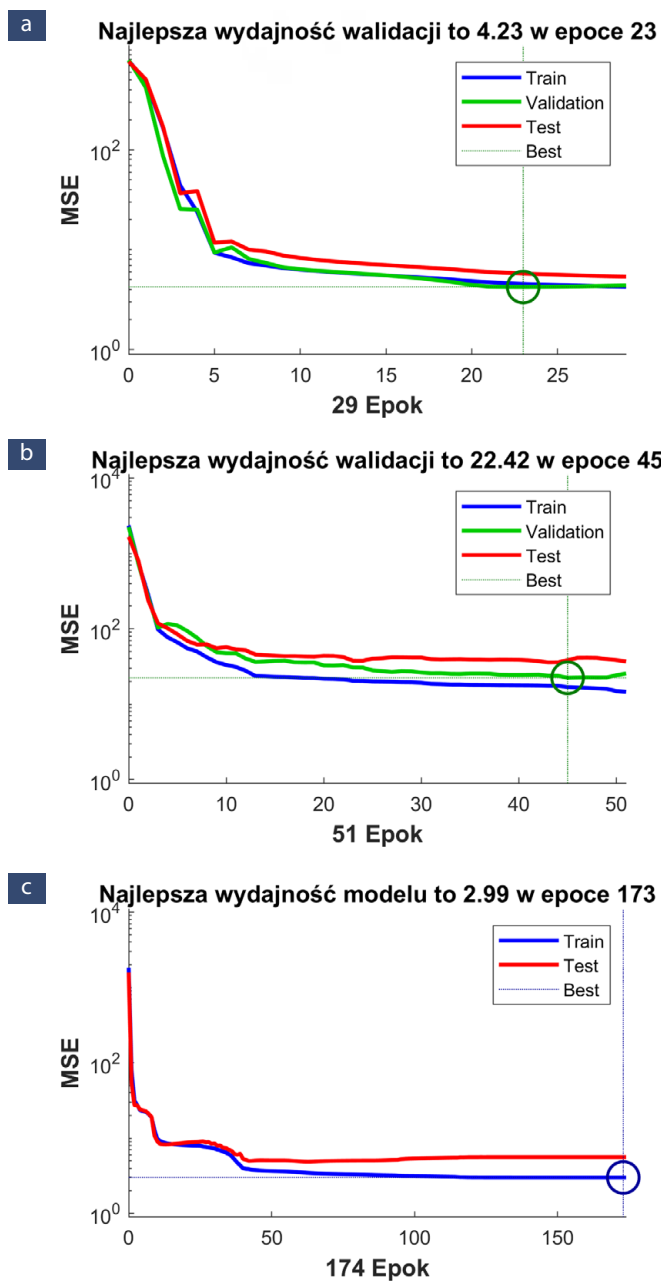
W procesie uczenia na wstępie zastosowano algorytm LM, przy czym najlepsze parametry jakościowe osiągnięto dla sieci neuronowej z dziewięcioma neuronami w warstwie ukrytej, po 29 epokach uczenia. Następnie, wykorzystując konfigurację z dziewięcioma neuronami w warstwie ukrytej, przeprowadzono trenowanie sieci neuronowych z użyciem dwóch dodatkowych algorytmów: SCG oraz BR. Wskaźniki jakości dla różnych modeli sieci neuronowych MLP zestawia tabela 5.9.

Tabela 5.9. Wskaźniki jakości dla różnych algorytmów uczenia sieci modelowania twardości po procesie odpuszczania

Algorytm uczenia sieci MLP	LM	SCG	BR
Epoka	29	51	174
R (cały zbiór danych)	0,9853	0,9335	0,9894
MSE	4,6809	20,9592	3,3918
RMSE	2,1635	4,5781	1,8417
RIE	0,0435	0,0921	0,0371
MAPE	0,0411	0,0913	0,0323
MAE	1,5769	3,4138	1,3451

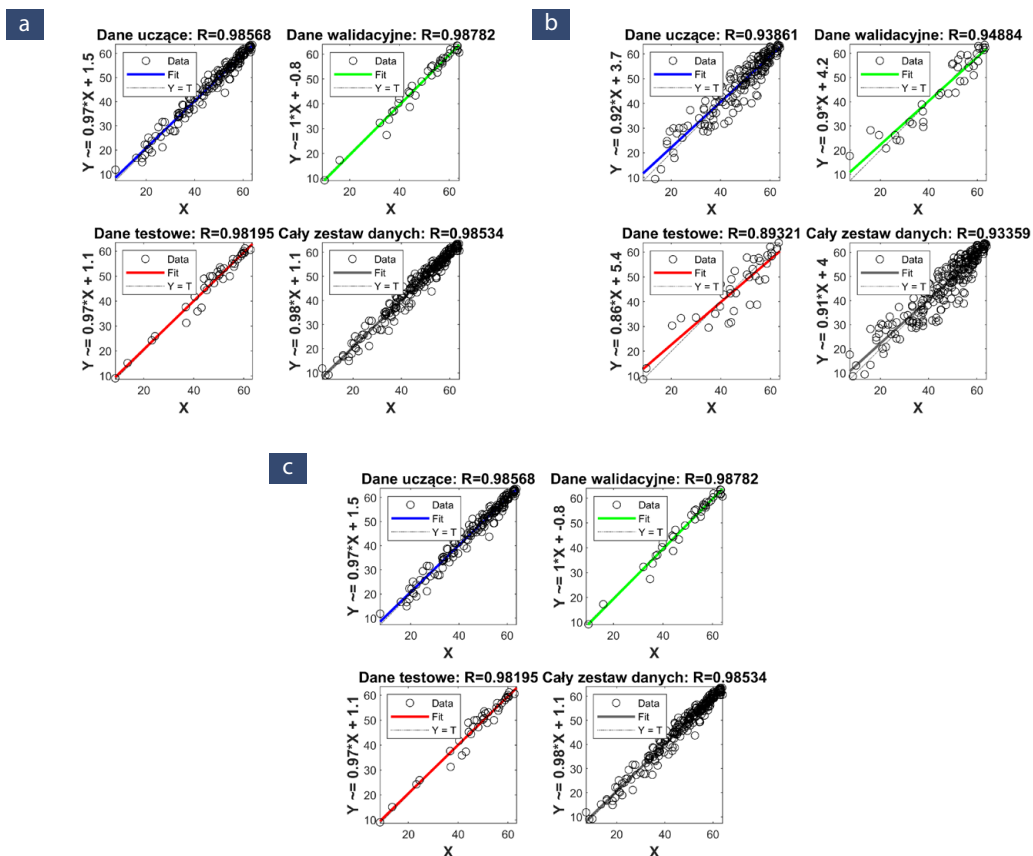
Tabela 5.9 obrazuje zróżnicowanie parametrów i wskaźników jakościowych dla zastosowanych algorytmów LM, SCG i BR w modelowaniu twardości po procesie odpuszczania. Liczba epok różni się znacząco między algorytmami. LM wymaga najmniejszej liczby epok (29), podczas gdy BR potrzebuje ich aż 174, co sugeruje większe zapotrzebowanie na iteracje w przypadku BR, jednak może to prowadzić do lepszej generalizacji modelu. Wartość współczynnika korelacji dla całego zbioru danych jest najwyższa dla BR (0,9894), co wskazuje na najlepsze dopasowanie modelu, podczas gdy SCG osiąga najniższą wartość (0,9335). Analizując wskaźniki błędów, takie jak MSE, RMSE, RIE, MAPE i MAE, można zauważyć różnice w efektywności algorytmów. BR osiąga najniższe wartości dla wszystkich rodzajów błędów, co sugeruje jego wyższą precyzję predykcji. Wyniki sugerują, że BR, mimo większej liczby epok, zapewnia najlepsze dopasowanie modelu i minimalizację błędów.

Na przedstawionych na ryc. 5.16 wykresach można zaobserwować przebieg wartości MSE w zależności od liczby epok dla trzech algorytmów: LM, SCG i BR. Algorytm LM osiąga najlepszą wydajność walidacji na poziomie 4,23 po 23 epokach, co pokazuje szybką zbieżność modelu. W przypadku SCG najlepsza wydajność walidacji wynosi 22,42, osiągnięta po 45 epokach, jednakże dalsze trenowanie skutkuje wolniejszym spadkiem MSE. Algorytm BR wykazuje największą stabilność i najniższą wartość MSE, na poziomie 2,99, osiągniętą dopiero po 173 epokach, co wskazuje na jego zdolność do uzyskiwania bardziej precyzyjnych modeli kosztem dłuższego procesu uczenia.



Ryc. 5.16. Najlepsza wydajność modelu twardości po procesie odpuszczania dla sieci z algorytmem uczenia: a) LM, b) SCG, c) BR

Ryc. 5.17 ilustruje wykresy korelacji dla zbiorów danych uczących, walidacyjnych oraz całego zbioru danych, uzyskane przy zastosowaniu różnych algorytmów uczenia: a) LM, b) SCG, c) BR. Wykresy pokazują, że model oparty na BR osiąga najwyższą zgodność predykcji z danymi rzeczywistymi, co potwierdzają najwyższe wartości współczynnika korelacji R dla danych uczących, testowych i walidacyjnych. Wartość R dla LM również jest wysoka, jednak nieco niższa od tej uzyskanej przez BR, co może wskazywać na mniejszą precyzję modelu. Algorytm SCG, choć osiąga dobre wyniki, wykazuje niższą zgodność między predykcjami a rzeczywistymi wartościami, szczególnie dla danych testowych, co sugeruje, że jest mniej skuteczny w generalizowaniu wyników na nieznanych danych.



Ryc. 5.17. Wykresy korelacji dla modelu twardości po procesie odpuszczania z algorytmem uczenia: a) LM, b) SCG, c) BR

Podsumowując wyniki analizy modeli twardości po procesie odpuszczania opartych na algorytmach LM, SCG i BR, można zauważyć, że każdy z nich wykazuje różne mocne i słabe strony. Algorytm LM charakteryzuje się szybką zbieżnością i niskimi wartościami błędów, co czyni go efektywnym w kontekście minimalizacji błędów bezwzględnych, jednak jego wartość współczynnika korelacji R wskazuje na nieco mniejszą precyzję w porównaniu z BR. Algorytm BR wyróżnia się spośród analizowanych modeli najlepszymi wartościami R, zarówno dla danych uczących, testowych, jak i walidacyjnych. Mimo że wymaga większej liczby epok do zbieżności i ma nieco wyższe wartości błędów niż LM w niektórych przypadkach, jego zdolność do uzyskiwania najniższych błędów przy wysokiej korelacji czyni go najbardziej precyzyjnym i niezawodnym w kontekście przewidywań. Biorąc pod uwagę zarówno wskaźniki wydajności, jak i wyniki korelacji, algorytm BR jest najbardziej odpowiedni do modelowania twardości po procesie odpuszczania, oferując najlepszy balans między precyzją predykcji a ogólną jakością modelu.

Na bazie wytrenowanego modelu twardości po procesie odpuszczania opartego na algorytmie BR opracowano prosty interfejs użytkownika (ryc. 5.18). Przedstawiony interfejs został zaprojektowany w środowisku MATLAB i umożliwia użytkownikowi wprowadzanie danych wejściowych, tj. typ stali, twardość po hartowaniu, czas odpuszczania, temperatura odpuszczania oraz otrzymywanie informacji zwrotnej na twardości po procesie odpuszczania.

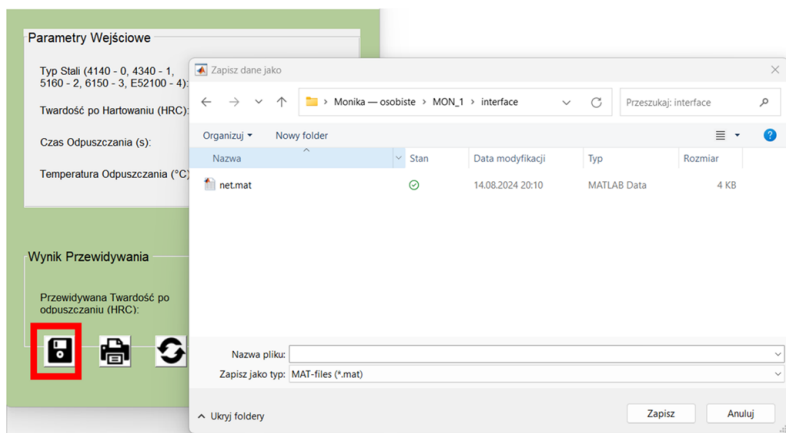
Ryc. 5.18. Interfejs użytkownika aplikacji do predykcji twardości po procesie odpuszczania

Interfejs graficzny, zbudowany za pomocą funkcji *uicontrol*, charakteryzuje się prostotą obsługi oraz czytelnym układem elementów. Główna sekcja „Parametry Wejściowe” umożliwia użytkownikowi wprowadzenie danych, takich jak typ stali (zdefiniowany numerycznie dla różnych rodzajów stali), twardość po hartowaniu (wyrażona w skali HRC), czas odpuszczania w sekundach oraz temperaturę odpuszczania w stopniach Celsjusza. Dane te są niezbędne do prawidłowego funkcjonowania modelu predykcji.

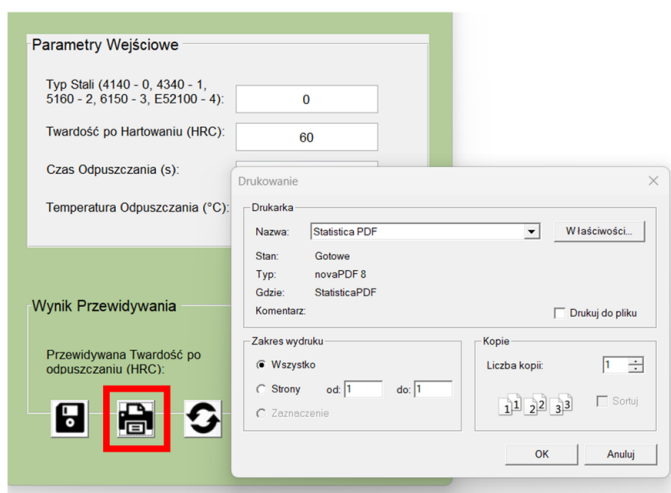
Po wprowadzeniu wszystkich wymaganych wartości użytkownik może skorzystać z przycisku „Predykcja”, który uruchamia proces obliczeniowy w oparciu o wczytaną sieć neuronową. Wynik predykcji, czyli przewidywana twardość stali po procesie odpuszczania, jest wyświetlany w sekcji „Wynik Przewidywania”, zapewniając użytkownikowi bezpośredni wgląd w prognozowaną wartość. Przykład działania aplikacji przedstawia ryc. 5.19.

Ryc. 5.19. Przykład działania aplikacji predykcji twardości po procesie odpuszczania

Interfejs zawiera również zestaw ikon funkcyjnych umieszczonych na dole ekranu. Ikona zapisu umożliwia zapisanie wyników oraz wprowadzonych danych w pliku (ryc. 5.20), ikona drukowania pozwala na bezpośrednie wydrukowanie wyników (ryc. 5.21), ikona odświeżania resetuje pola wprowadzania danych (ryc. 5.22), a ikona zamknięcia służy do zamknięcia aplikacji (ryc. 5.23). Całość została zaprojektowana z myślą o intuicyjnym użytkowaniu, co pozwala na łatwe i efektywne korzystanie z narzędzia do analizy procesów metalurgicznych. Sposób implementacji opisanego interfejsu graficznego został opisany w postaci kodu w oprogramowaniu Matlab przedstawionym jako algorytm 5.9.



Ryc. 5.20. Przykład działania aplikacji predykcji twardości po procesie odpuszczania – funkcja zapisz



Ryc. 5.21. Przykład działania aplikacji predykcji twardości po procesie odpuszczania – funkcja drukuj

Parametry Wejściowe

Typ Stali (4140 - 0, 4340 - 1, 5160 - 2, 6150 - 3, E52100 - 4):

Twardość po Hartowaniu (HRC):

Czas Odpuszczania (s):

Temperatura Odpuszczania (°C):

Predykcja

Wynik Przewidywania

Przewidywana Twardość po odpuszczaniu (HRC):

Ryc. 5.22. Przykład działania aplikacji predykcji twardości po procesie odpuszczania – funkcja odśwież

Parametry Wejściowe

Typ Stali (4140 - 0, 4340 - 1, 5160 - 2, 6150 - 3, E52100 - 4):

Twardość po Hartowaniu (HRC):

Czas Odpuszczania (s):

Temperatura Odpuszczania (°C):

Predykcja

Wynik Przewidywania

Przewidywana Twardość po odpuszczaniu (HRC):

Ryc. 5.23. Przykład działania aplikacji predykcji twardości po procesie odpuszczania – ikona zamknij

Algorytm 5.9. Implementacja interfejsu graficznego z wykorzystaniem modelu predykcji twardości po procesie odpuszczania

```
function twardosc_odpuszczania1()
% Wczytaj zapisaną sieć neuronową 'net'
load('net.mat', 'net'); % Przykład wczytania sieci z pliku
% Tworzenie głównego okna GUI
f = figure('Position', [300, 300, 600, 500], 'Name',
'Przewidywanie Twardości po Odpuszczaniu', 'NumberTitle', 'off',
'Color', [0.7, 0.8, 0.6]);
p1 = uipanel('Title', 'Parametry Wejściowe', 'FontSize', 12,
'Position', [0.05 0.5 0.9 0.45]);
uicontrol('Parent', p1, 'Style', 'text', 'Position', [20, 140,
200, 40], 'String', 'Typ Stali (4140 - 0, 4340 - 1, 5160 - 2, 6150 -
3, E52100 - 4):', 'HorizontalAlignment', 'left', 'FontSize', 10);
steelTypeEdit = uicontrol('Parent', p1, 'Style', 'edit',
'Position', [220, 140, 150, 30], 'String', '0', 'FontSize', 12);
uicontrol('Parent', p1, 'Style', 'text', 'Position', [20,
100, 200, 30], 'String', 'Twardość po Hartowaniu (HRC):',
'HorizontalAlignment', 'left', 'FontSize', 10);
hardnessEdit = uicontrol('Parent', p1, 'Style', 'edit',
'Position', [220, 100, 150, 30], 'String', '60', 'FontSize', 12);
uicontrol('Parent', p1, 'Style', 'text', 'Position', [20, 60, 200,
30], 'String', 'Czas Odpuszczania (s):', 'HorizontalAlignment',
'left', 'FontSize', 10);
timeEdit = uicontrol('Parent', p1, 'Style', 'edit', 'Position',
[220, 60, 150, 30], 'String', '600', 'FontSize', 12);
uicontrol('Parent', p1, 'Style', 'text', 'Position', [20,
20, 200, 30], 'String', 'Temperatura Odpuszczania (°C):',
'HorizontalAlignment', 'left', 'FontSize', 10);
tempEdit = uicontrol('Parent', p1, 'Style', 'edit', 'Position',
[220, 20, 150, 30], 'String', '200', 'FontSize', 12);
% Przycisk do wywołania predykcji
predictButton = uicontrol('Style', 'pushbutton', 'Position',
[250, 200, 120, 30], 'String', 'Predykcja', 'Callback', @(~,~)
predictCallback(net), 'FontSize', 10);
% Panel do wyświetlania wyników
p2 = uipanel('Title', 'Wynik Przewidywania', 'FontSize', 12,
'Position', [0.05 0.15 0.9 0.25], 'BackgroundColor', [0.7, 0.8,
0.6]);
uicontrol('Parent', p2, 'Style', 'text', 'Position', [20, 40,
240, 30], 'String', 'Przewidywana Twardość po odpuszczaniu (HRC):',
'HorizontalAlignment', 'left', 'FontSize', 10, 'BackgroundColor',
[0.7, 0.8, 0.6]);
```


Algorytm 5.9. Implementacja interfejsu graficznego z wykorzystaniem modelu predykcji twardości po procesie odpuszczania, cd.

```
predictedHardnessText = uicontrol('Parent', p2, 'Style', 'text',  
'Position', [260, 40, 150, 30], 'String', '', 'BackgroundColor',  
'white', 'FontSize', 12);  
% Dodanie przycisków funkcyjnych  
addIconButton(50, 50, 'ikona_zapisz.png', @saveFunction); % Zapisz  
addIconButton(120, 50, 'ikona_drukuj.png', @printFunction); % Drukuj  
addIconButton(190, 50, 'ikona_reload.png', @clearFunction); % Wyczyść  
addIconButton(400, 50, 'ikona_wylacz.png', @exitFunction); % Zamknij  
% Funkcja callback dla przycisku predykcji  
function predictCallback(net)  
% Pobieranie wartości z pól edycyjnych  
steel_type = str2double(get(steelTypeEdit, 'String'));  
initial_hardness = str2double(get(hardnessEdit, 'String'));  
tempering_time = str2double(get(timeEdit, 'String'));  
tempering_temperature = str2double(get(tempEdit, 'String'));  
% Tworzenie wektora danych wejściowych  
input_data = [steel_type, initial_hardness, tempering_time,  
tempering_temperature];  
% Predykcja twardości po odpuszczaniu  
predicted_hardness = net(input_data);  
% Wyświetlenie wyniku w GUI  
set(predictedHardnessText, 'String', num2str(predicted_  
hardness, '%.2f'));  
end  
% Funkcja zapisująca wynik i dane wejściowe  
function saveFunction(~, ~)  
steel_type = get(steelTypeEdit, 'String');  
initial_hardness = get(hardnessEdit, 'String');  
tempering_time = get(timeEdit, 'String');  
tempering_temperature = get(tempEdit, 'String');  
result = get(predictedHardnessText, 'String');  
[file, path] = uiputfile('*.mat', 'Zapisz dane jako');  
if ischar(file)  
data = struct('steel_type', steel_type, 'initial_hardness',  
initial_hardness, 'tempering_time', tempering_time, 'tempering_  
temperature', tempering_temperature, 'result', result);  
save(fullfile(path, file), 'data');  
disp(['Dane zapisane w pliku: ', fullfile(path, file)]);  
end  
end
```

Algorytm 5.9. Implementacja interfejsu graficznego z wykorzystaniem modelu predykcji twardości po procesie odpuszczania, cd.

```
% Funkcja drukująca wynik
function printFunction(~, ~)
    printdlg(f);
end
% Funkcja czyszcząca pola i wynik
function clearFunction(~, ~)
    set(steelTypeEdit, 'String', '');
    set(hardnessEdit, 'String', '');
    set(timeEdit, 'String', '');
    set(tempEdit, 'String', '');
    set(predictedHardnessText, 'String', '');
end
% Funkcja zamykająca aplikację
function exitFunction(~, ~)
    close(f);
end
% Funkcja dodająca przycisk z ikonką
function addIconButton(x, y, iconPath, callback)
    [iconData, map, alpha] = imread(iconPath); % Wczytaj dane
    % ikonki i mapę kolorów oraz kanał alfa
    if ~isempty(map)
        iconData = ind2rgb(iconData, map); % Konwersja indeksów na
        % wartości RGB
    end
    if size(iconData, 3) == 4 % Sprawdź, czy obraz ma kanał alfa
        iconData = iconData(:, :, 1:3); % Usuń kanał alfa
    end
    % Skalowanie obrazka do wymiarów przycisku
    iconData = imresize(iconData, [40, 40]);
    % Normalizacja wartości pikseli do zakresu [0, 1]
    iconData = double(iconData) / 255;
    % Dodanie axes dla rysowania prostokąta
    ax = axes('Position', [x, y, 40, 40], 'Visible', 'off');
    rectangle('Position', [0, 0, 40, 40], 'EdgeColor', [0.7, 0.8,
    0.6], 'LineWidth', 2, 'Parent', ax);
    % Dodanie przycisku z ikonką
    uicontrol('Style', 'pushbutton', 'Position', [x, y, 40, 40],
    'CData', iconData, 'Callback', callback, 'BackgroundColor', [0.8,
    0.9, 1]);
end
end
```

5.3.3. Ocena jakości wyrobów gotowych na podstawie twardości po procesie odpuszczania

Ocena jakości twardości po odpuszczaniu na podstawie twardości po procesie odpuszczania jest istotnym elementem kontroli jakości, ponieważ pozwala na weryfikację, czy produkt końcowy spełnia wymagania stawiane przez specyfikację techniczną. Brak ogólnie przyjętych norm dla twardości po odpuszczaniu dla każdego rodzaju stali wynika z różnorodności zastosowań, które wymagają specyficznych właściwości materiału. Pożądana twardość zależy nie tylko od rodzaju stali, ale także od kształtu wyrobu (np. wałek czy element płaski), od jego docelowego zastosowania oraz innych czynników, takich jak obciążenia mechaniczne, którym będzie poddawany.

W związku z brakiem uniwersalnych ram dotyczących pożądanej twardości po odpuszczaniu, konieczne było przyjęcie określonych zakresów twardości dla każdego z analizowanych typów stali w kontekście danego – przykładowego zastosowania. Na podstawie tych założeń opracowany został algorytm, który umożliwia ocenę jakości wyrobu gotowego. Algorytm ten, uwzględniając przewidywaną przez model sztucznej sieci neuronowej MLP twardość po odpuszczaniu (model oparty na algorytmie uczenia BR), określa, czy dany wyrób spełnia wymagania jakościowe. Jeżeli przewidywana twardość mieści się w przyjętym przedziale, uznaje się, że wyrób spełnia normy jakościowe, w przeciwnym razie wyrób jest klasyfikowany jako niezgodny z wymaganiami.

Na podstawie danych z analizowanej bazy danych oraz minimalnych i maksymalnych wartości twardości uzyskanych dla różnych typów stali, można zaproponować uniwersalne zakresy twardości, które pozwolą na ocenę jakości wyrobów. Przyjęcie jednolitych przedziałów, takich jak 25-55 HRC, ma na celu pokazanie możliwości stosowania tego typu ograniczeń w ocenie jakości produktów, przy czym w każdym konkretnym zastosowaniu te zakresy mogą być dostosowane do specyficznych wymagań, uwzględniając szczegóły dotyczące finalnego zastosowania wyrobu. Dzięki temu możliwe jest skuteczne zarządzanie procesami produkcyjnymi, minimalizując ryzyko wytwarzania wyrobów niespełniających wymagań jakościowych i zapewniając ich odpowiednią wytrzymałość oraz funkcjonalność. Fragment kodu przedstawiony jako algorytm 5.10 ilustruje sposób implementacji oceny jakości na podstawie zakresu twardości wyrobów gotowych po odpuszczaniu, o który uzupełniono algorytm 5.9.

Algorytm 5.10. Fragment kodu oceny jakości na podstawie zakresu twardości wyrobów gotowych po odpuszczaniu

```
% Ocena jakości na podstawie zakresu twardości
if predicted_hardness >= 25 && predicted_hardness <= 55
    quality_message = ,Wyrób gotowy spełnia wymagania
jakościowe.';
else
    quality_message = ,Wyrób gotowy nie spełnia wymagań
jakościowych.';
end
% Wyświetlenie oceny jakości w GUI
set(qualityAssessmentText, 'String', quality_message);
```

Wyniki implementacji tego fragmentu kodu (algorytm 5.10) do interfejsu użytkownika przedstawiają ryc. 5.24 oraz ryc. 5.25, przy czym przykładowy wynik działania algorytmu dla wyrobu niespełniającego wymagań jakościowych to ryc. 5.24, zaś dla wyrobu spełniającego wymagania jakościowe – ryc. 5.25.

The screenshot shows a software window with a green border. At the top, there is a section titled "Parametry Wejściowe" (Input Parameters) with a light gray background. It contains four input fields: "Typ Stali (4140 - 0, 4340 - 1, 5160 - 2, 6150 - 3, E52100 - 4):" with value "0", "Twardość po Hartowaniu (HRC):" with value "60", "Czas Odpuszczania (s):" with value "600", and "Temperatura Odpuszczania (°C):" with value "200". Below this section is a button labeled "Predykcja". Under the button is another section titled "Wynik Przewidywania" (Prediction Result) with a light green background. It displays "Przewidywana Twardość po odpuszczaniu (HRC):" with value "55.64" and a bold message: "Wyrób gotowy nie spełnia wymagań jakościowych." (The finished product does not meet quality requirements). At the bottom of the window are four icons: a floppy disk, a printer, a circular arrow (refresh), and a power button.

Ryc. 5.24. Wynik działania algorytmu dla wyrobu niespełniającego wymagań jakościowych

6. Predykcja awarii maszyn

Wprowadzenie do tematyki predyktywnego utrzymania ruchu, zwłaszcza z uwzględnieniem wykorzystania sztucznych sieci neuronowych, stanowi ważny element doskonalenia procesów produkcyjnych w przemyśle maszynowym. W obliczu dynamicznie rozwijającej się technologii oraz rosnących wymagań rynkowych wdrażanie nowoczesnych metod zarządzania utrzymaniem maszyn staje się nieodzownym elementem strategii przedsiębiorstw dążących do maksymalizacji efektywności operacyjnej oraz minimalizacji ryzyka awarii.

Predyktywne utrzymanie ruchu to zaawansowana metodologia zarządzania i utrzymania maszyn, która polega na przewidywaniu awarii oraz ocenie stanu technicznego urządzeń. Wykorzystując techniki analizy danych zbieranych z różnorodnych czujników monitorujących parametry pracy maszyn, takie jak temperatura, prędkość obrotowa, moment obrotowy czy zużycie narzędzi, możliwe jest minimalizowanie przestojów produkcyjnych poprzez prognozowanie potencjalnych problemów, zanim jeszcze one wystąpią.

W kontekście przemysłu maszynowego predyktywne utrzymanie ruchu odgrywa ważną rolę, umożliwiając znaczące zmniejszenie przestojów produkcyjnych. Niespodziewane awarie maszyn mogą powodować poważne zakłócenia w procesie produkcyjnym, prowadząc do strat finansowych i opóźnień w realizacji zamówień. Dzięki zastosowaniu predyktywnych modeli przedsiębiorstwa mogą planować konserwacje i naprawy w sposób bardziej efektywny, unikając nieplanowanych przerw w pracy [13,189].

Kolejną istotną korzyścią jest optymalizacja zasobów. Predykcyjne utrzymanie ruchu pozwala na lepsze zarządzanie zapasami części zamiennych oraz planowanie pracy personelu technicznego. Zamiast trzymać duże ilości części zamiennych na magazynie lub angażować liczne zespoły serwisowe w gotowości, firmy mogą dokładnie przewidzieć, kiedy i jakie zasoby będą potrzebne [190,191].

Redukcja kosztów utrzymania to kolejny znaczący aspekt. Tradycyjne podejście do utrzymania ruchu często wiąże się z wysokimi kosztami związanymi z nadmierną konserwacją lub nieoczekiwanymi naprawami. Predykcyjne utrzymanie ruchu umożliwia bardziej precyzyjne i efektywne zarządzanie cyklem życia maszyn, co prowadzi do obniżenia kosztów operacyjnych. Dzięki dokładnej analizie danych i prognozowaniu awarii możliwe jest uniknięcie kosztownych napraw oraz przedłużenie żywotności maszyn [192,193].

Sieci neuronowe, dzięki swojej zdolności do przetwarzania i analizy dużych zbiorów danych, są idealnym narzędziem do przewidywania ryzyka wystąpienia awarii oraz identyfikacji rodzaju awarii na podstawie parametrów operacyjnych [194]. ANN są w stanie analizować dane z monitorowania stanu technicznego maszyn, takie jak temperatura, prędkość obrotowa, moment obrotowy i zużycie narzędzi, w celu prognozowania, czy awaria wystąpi, a w przypadku wystąpienia – określenia jej rodzaju.

Celem zastosowania ANN jest nie tylko przewidywanie, czy awaria wystąpi, ale również dokładne rozpoznanie rodzaju awarii. Pozwala to na bardziej precyzyjne planowanie działań prewencyjnych i naprawczych, co przekłada się na zmniejszenie liczby przestojów oraz optymalizację procesów produkcyjnych. Dzięki funkcji aktywacji *softmax* ANN mogą dostarczać prawdopodobieństwo przynależności do poszczególnych klas awarii, co ułatwia podejmowanie decyzji na podstawie wyników analizy [195].

Predyktywne utrzymanie ruchu jest niezwykle ważną koncepcją dla przemysłu maszynowego, oferującą liczne korzyści, takie jak zmniejszenie przestojów, optymalizacja zasobów oraz redukcja kosztów utrzymania. W dobie rosnącej konkurencji i coraz bardziej zaawansowanych technologii wdrożenie predyktywnych metod zarządzania utrzymaniem ruchu staje się nieodzownym elementem strategii przedsiębiorstw dążących do maksymalizacji efektywności operacyjnej i minimalizacji ryzyka awarii [196,197].

W niniejszym rozdziale omówione zostaną aspekty związane z modelowaniem predykcji awarii maszyn. Na początku zostanie przedstawiona metodyka modelowania predykcji awarii, co pozwoli na zrozumienie podstaw teoretycznych i technologicznych zastosowanych rozwiązań. Następnie szczegółowo omówione zostaną wyniki modelowania klasyfikatora dotyczącego wystąpienia

awarii oraz określania typu awarii. Na zakończenie rozdziału zostanie zaprezentowane praktyczne wykorzystanie predykcji awarii maszyn z zastosowaniem odpowiedniego interfejsu, co pozwoli na pełne zrozumienie wdrożonych metod w kontekście rzeczywistych zastosowań przemysłowych.

6.1. Metodyka modelowania predykcji awarii

Celem prowadzonych analiz była predykcja awarii maszyn na podstawie parametrów określających ich stan techniczny. Parametry były gromadzone podczas monitorowania procesu produkcyjnego. Istotne zmienne wejściowe obejmują temperaturę, prędkość obrotową, moment obrotowy oraz zużycie narzędzi, a zmienną wyjściową jest wskaźnik awarii, czy wystąpi, czy też nie. Algorytmy klasyfikacyjne były testowane przy użyciu publicznej bazy danych „AI4I 2020 Predictive Maintenance Dataset”, która została udostępniona w 2020 roku [198].

Baza danych AI4I 2020 Predictive Maintenance Dataset jest syntetycznym zbiorem danych, który odzwierciedla rzeczywiste dane spotykane w przemyśle dotyczące predykcyjnego utrzymania ruchu. Baza ta zawiera 10 000 próbek z 10 cechami, w tym identyfikatorami produktów, oraz zmiennymi opisującymi warunki operacyjne i zużycie narzędzi. Są to następujące zmienne:

- unikalny identyfikator;
- identyfikator produktu;
- typ produktu (L, M, H);
- temperatura powietrza [K];
- temperatura procesu [K];
- prędkość obrotowa [obr/min];
- moment obrotowy [Nm];
- zużycie narzędzi w [min];
- wskaźnik awarii (0 – brak awarii, 1 – wystąpienie awarii);
- typ awarii.

Istotnym aspektem bazy danych są zmienne docelowe, które określają wystąpienie awarii maszyny oraz typ awarii. Wskaźnik awarii jest binarną zmienną, która określa, czy doszło do awarii (1) lub nie (0). Typ awarii obejmuje pięć niezależnych trybów, takich jak:

- awaria narzędzia – występuje, gdy narzędzie przekroczy wartość zużycia od 200 do 240 minut;
- awaria chłodzenia – występuje, gdy różnica temperatur powietrza i procesu jest mniejsza niż 8,6 K, a prędkość obrotowa spada poniżej 1380 [obr/min];

- awaria mocy – występuje, gdy moc obliczona jako iloczyn momentu obrotowego i prędkości obrotowej [rad/s] jest poza zakresem 3500–9000 [W];
- awaria przeciążenia – występuje, gdy iloczyn zużycia narzędzi i momentu obrotowego przekracza wartości krytyczne (11 000 minNm dla produktów L, 12 000 dla M i 13 000 dla H);
- awarie losowe – niezależne od parametrów procesu, z szansą 0,1% dla każdego punktu danych.

Baza zawiera zmienne rzeczywiste i kategoryczne, co umożliwia zastosowanie różnorodnych metod analitycznych, w tym sztucznych sieci neuronowych. Ze względu na trudność w uzyskaniu rzeczywistych danych predykcyjnego utrzymania ruchu, zbiór ten stanowi wartościowe źródło danych do badań i testowania algorytmów w warunkach symulowanych, które jak najwierniej odzwierciedlają rzeczywiste scenariusze przemysłowe.

Aby przewidzieć, czy awaria wystąpi, czy też nie, stworzono model klasyfikacyjny, który został zbudowany z wykorzystaniem sztucznych sieci neuronowych przy użyciu Matlab R2023a. Warstwa wejściowa składała się z pięciu neuronów (zmienne: temperatura powietrza, temperatura procesu, prędkość obrotowa, moment obrotowy, zużycie narzędzi), podczas gdy warstwa wyjściowa zawierała jeden neuron (klasyfikator z dwoma wartościami 0 lub 1). Wartość klasyfikatora 1 oznacza, że awaria wystąpi, zaś 0, że nie wystąpi. Tabela 6.1 przedstawia statystyki opisowe danych wejściowych.

Tabela 6.1. Statystyka opisowa danych wejściowych

	temperatura powietrza, [K]	temperatura procesu [K]	prędkość obrotowa [obr/min]	moment obrotowy [Nm]	zużycie narzędzi [min]
średnia	300,0049	310,0056	1538,776	39,98691	107,951
odchylenie standardowe	2,000259	1,483734	179,2841	9,968934	63,65415
min	295,3	305,7	1168	3,8	0
mediana	300,1	310,1	1503	40,1	108
max	304,5	313,8	2886	76,6	253

W przypadku tych danych wzięto pod uwagę dwa rodzaje sieci. Pierwszą z nich była sieć neuronowa MLP z jedną warstwą ukrytą, drugą – sieć LSTM. Dane z monitorowania stanu technicznego maszyn często mają charakter sekwencyjny, gdzie kolejne pomiary parametrów, takich jak temperatura, prędkość obrotowa, moment obrotowy i zużycie narzędzi, są zbierane w regularnych odstępach czasu. Sieci LSTM, jako odmiana rekurencyjnych sieci neuronowych, są

zaprojektowane do analizy takich sekwencji danych, dzięki swojej zdolności do zapamiętywania długoterminowych zależności. W celu podkreślenia korzyści wynikających z wykorzystania LSTM oraz różnic pomiędzy sieciami, dokonano również analizy za pomocą tradycyjnej, sieci neuronowej MLP, która nie posiada zdolności do zapamiętywania długoterminowych zależności czasowych.

Do uczenia MLP zastosowano algorytm SCG. Liczba neuronów w warstwie ukrytej, mieszcząca się w zakresie od 2 do 50, została dobrana na drodze eksperymentalnej. Maksymalną liczbę epok uczących ustalono na 1000. Aby uzyskać prawdopodobieństwo przynależności do poszczególnych klas, zastosowano funkcję aktywacji *softmax* w warstwie wyjściowej. Dane zostały podzielone na trzy części: 70% danych przeznaczono na zbiór uczący, 15% na testowy i 15% na walidacyjny. Zbiór uczący był wykorzystywany do trenowania sieci neuronowej. Proces uczenia trwał do momentu, gdy sieć przestawała poprawiać swoje wyniki na zbiorze walidacyjnym. Zastosowano technikę wczesnego zatrzymania na poziomie 6 jako metodę zapobiegania przeuczeniu. Zbiór testowy służył do niezależnej oceny dokładności działania sieci. Jakość działania sieci oceniano przy użyciu funkcji entropii krzyżowej, która była miarą wykorzystywaną do oceny poprawności klasyfikacji dokonywanej przez sieć neuronową.

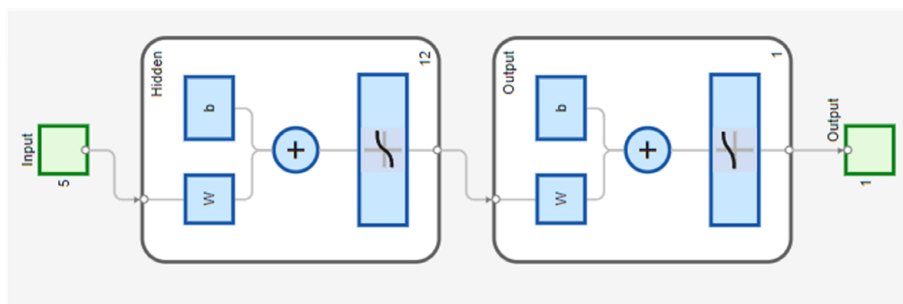
Do uczenia sieci neuronowej typu LSTM zastosowano algorytm Adam. Liczba neuronów (jednostek ukrytych) w warstwach ukrytych była dobierana eksperymentalnie, w zakresie od 256 do 4096 jednostek. Testowano zarówno sieci z jedną warstwą LSTM, jak i z dwiema warstwami LSTM. W przypadku drugiej warstwy LSTM liczba jednostek ukrytych również była zmieniana w zakresie od 256 do 4096. Dodatkowo w celu zapobiegania nadmiernemu dopasowaniu zastosowano warstwę dropout z wartością 0,5. Maksymalna liczba epok uczących została ustalona na 1000, a początkowa szybkość uczenia na 0,001. Rozmiar wsadu (ang. *MiniBatchSize*) został ustalony na 256. Aby uzyskać prawdopodobieństwo przynależności do poszczególnych klas, użyto funkcji aktywacji *softmax* w warstwie wyjściowej. Dane zostały podzielone na trzy części: 70% danych przeznaczono na zbiór uczący, 15% na testowy i 15% na walidacyjny. Wykorzystano te same zestawy, co w przypadku sieci MLP. Proces uczenia był monitorowany przy użyciu zbioru walidacyjnego. Zastosowano technikę wczesnego zatrzymania na poziomie 30.

W celu stworzenia modeli klasyfikacyjnych dotyczących rodzaju awarii użyto tego samego zbioru danych, w tym identyczne dane wejściowe i środowisko Matlab R2023a. Zastosowano takie same parametry, metodykę prowadzonych analiz dla obu rodzajów modeli. W przypadku modelowania tego klasyfikatora zmianie uległa jedynie zmienna wyjściowa, którą był typ awarii, zakodowany jako zmienna kategoryczna.

Porównując modele między sobą, wykorzystywano takie wskaźniki, jak: dokładność, precyzja, czułość oraz F1-score. Dodatkowo przeprowadzono analizę krzywej ROC oraz AUC w celu oceny zdolności modeli do rozróżniania otrzymanych wyników między klasami.

6.2. Wyniki modelowania klasyfikatora dotyczącego wystąpienia awarii

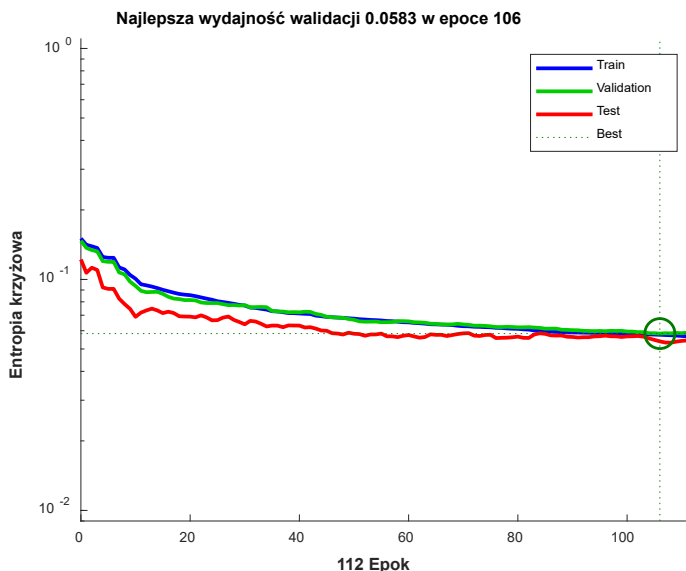
Najlepsze wyniki uczenia sieci neuronowej MLP, która może przewidzieć, czy awaria wystąpi czy też nie, uzyskano dla 12 neuronów w warstwie ukrytej, po 112 epokach uczenia sieci. Struktura uczenia sieci neuronowych została przedstawiona na ryc. 6.1, zaś na ryc. 6.2 pokazano przebieg uczenia się sieci neuronowej, monitorowany za pomocą funkcji entropii krzyżowej jako miary błędu, w zależności od liczby epok uczących. Na osi poziomej (x) znajduje się liczba epok, natomiast na osi pionowej (y) przedstawiono wartość entropii krzyżowej, w skali logarytmicznej.



Ryc. 6.1. Struktura sieci neuronowej MLP klasyfikacyjnej wystąpienia awarii

Proces uczenia sieci neuronowej, która może przewidzieć, czy awaria wystąpi czy też nie, trwał przez 112 epok, a najlepsza wydajność na zbiorze walidacyjnym została osiągnięta po 106 epokach, z wartością entropii krzyżowej wynoszącą 0,0583. Widać, że po tym punkcie (oznaczonym zielonym okręgiem) krzywa walidacyjna przestaje się znacząco poprawiać, co sugeruje zastosowanie techniki wczesnego zatrzymania w celu uniknięcia przeuczenia sieci.

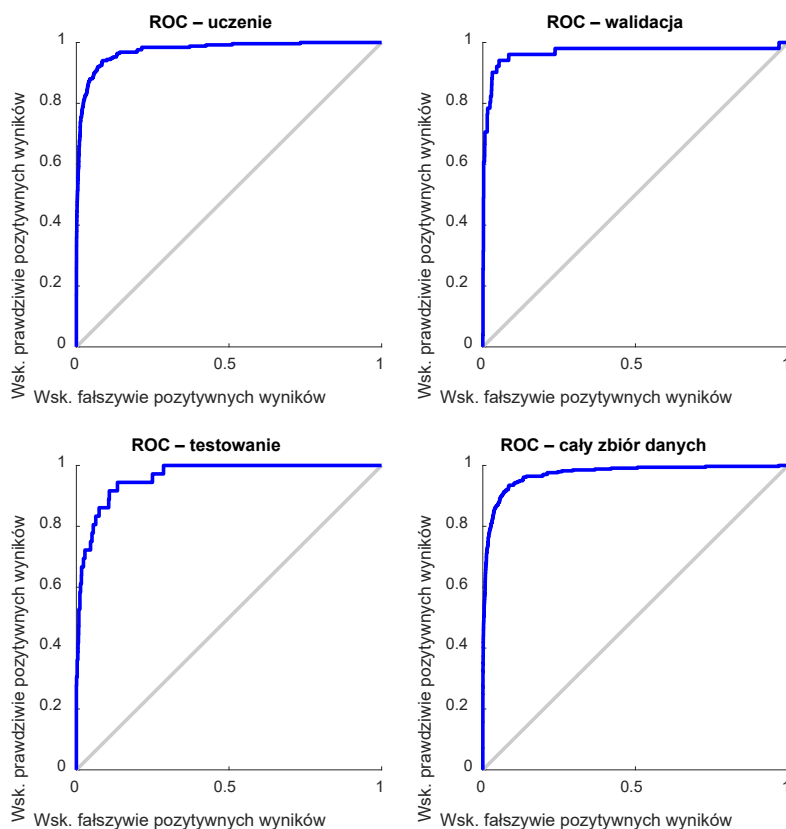
Wartości entropii krzyżowej dla zbiorów uczącego, walidacyjnego i testowego zmniejszają się w miarę postępującego uczenia, co wskazuje na poprawę zdolności sieci do rozpoznawania wzorców. Względnie niewielka różnica między krzywymi walidacyjną a testową świadczy o dobrej zdolności sieci do generalizacji na nowych danych.



Ryc. 6.2. Przebieg uczenia się sieci neuronowej klasyfikacyjnej MLP wystąpienia awarii

Wykresy na ryc. 6.3 przedstawiają krzywe ROC dla analizowanej sieci neuronowej, która została wytrenowana do klasyfikacji danych w celu określenia, czy awaria nastąpi, czy też nie. Wykres w lewym górnym rogu przedstawia krzywą ROC dla danych uczących. Wysokie wartości TPR przy niskich wartościach FPR sugerują, że sieć neuronowa dobrze klasyfikuje dane uczące. Wykres w prawym górnym rogu przedstawia krzywą ROC dla danych walidacyjnych. Krzywa jest zbliżona do wykresu dla danych uczących, co wskazuje na dobrą zdolność sieci do generalizacji na niezależnym zbiorze walidacyjnym. Wykres w lewym dolnym rogu przedstawia krzywą ROC dla danych testowych. Podobnie jak w przypadku danych uczących i walidacyjnych, wysoka wartość TPR przy niskim FPR potwierdza dobrą wydajność klasyfikacyjną sieci. Wykres w prawym dolnym rogu przedstawia zsumowaną krzywą ROC dla wszystkich danych (uczących, walidacyjnych i testowych). Krzywa ta obrazuje ogólną wydajność sieci na całym zbiorze danych.

Z wszystkich wykresów wynika, że sieć neuronowa, która przewiduje, czy awaria wystąpi, czy też nie, ma wysoką skuteczność w klasyfikacji danych, co jest widoczne w wysokich wartościach wskaźnika prawdziwie pozytywnych wyników przy niskich wartościach wskaźnika fałszywie pozytywnych wyników. Zbieżność krzywych ROC dla danych uczących, walidacyjnych i testowych sugeruje, że sieć neuronowa ma dobrą zdolność do generalizacji i nie jest nadmiernie dopasowana do danych uczących.

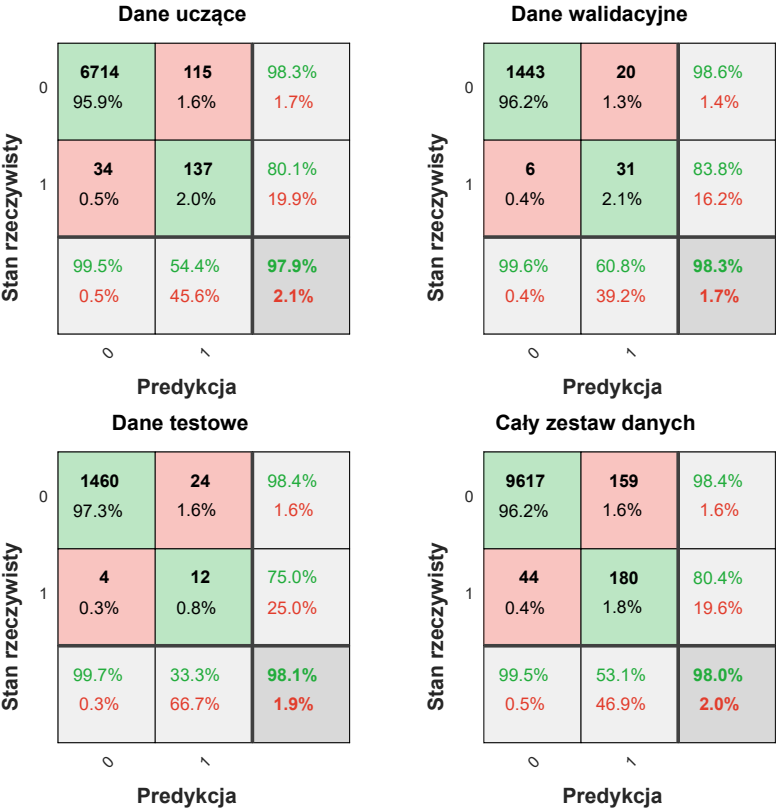


Ryc. 6.3. Krzywe ROC dla sieci neuronowej klasyfikacyjnej MLP wystąpienia awarii

Kolejnym ze wskaźników skuteczności dopasowania sieci neuronowej MLP do danych jest macierz pomyłek (ryc. 6.4), która dla zbioru uczącego pokazuje, że większość próbek została poprawnie sklasyfikowana: 6714 z klasy 0 i 137 z klasy 1. Macierz walidacyjna pokazuje podobne wyniki, z 98,6% poprawnie sklasyfikowanych próbek dla klasy 0 i 83,8% dla klasy 1. Macierz pomyłek dla zbioru testowego również potwierdza dobrą wydajność sieci: 98,4% poprawnych klasyfikacji dla klasy 0 i 75,0% dla klasy 1. Macierz pomyłek dla wszystkich danych przedstawia ogólną wydajność sieci na całym zbiorze danych, z 98,4% poprawnych klasyfikacji dla klasy 0 i 80,4% dla klasy 1. Wysokie wartości procentowe poprawnych klasyfikacji we wszystkich macierzach świadczą o dobrej skuteczności sieci neuronowej w rozpoznawaniu wzorców, czy awaria wystąpi, czy też nie.

Dla modelowania sieci LSTM najlepsze parametry otrzymano dla dwóch warstw LSTM w modelu oraz dla liczby jednostek ukrytych w pierwszej warstwie – 1024, zaś w drugiej – 2048. Strukturę sieci obrazuje tabela 6.2, zaś

kodowanie architektury i procesu uczenia sieci LSTM dla modelu klasyfikacyjnego wystąpienia awarii obrazuje algorytm 6.1.



Ryc. 6.4. Macierz pomyłek dla sieci neuronowej klasyfikacyjnej MLP wystąpienia awarii

Tabela 6.2. Architektura sieci LSTM dla modelu klasyfikacyjnego wystąpienia awarii

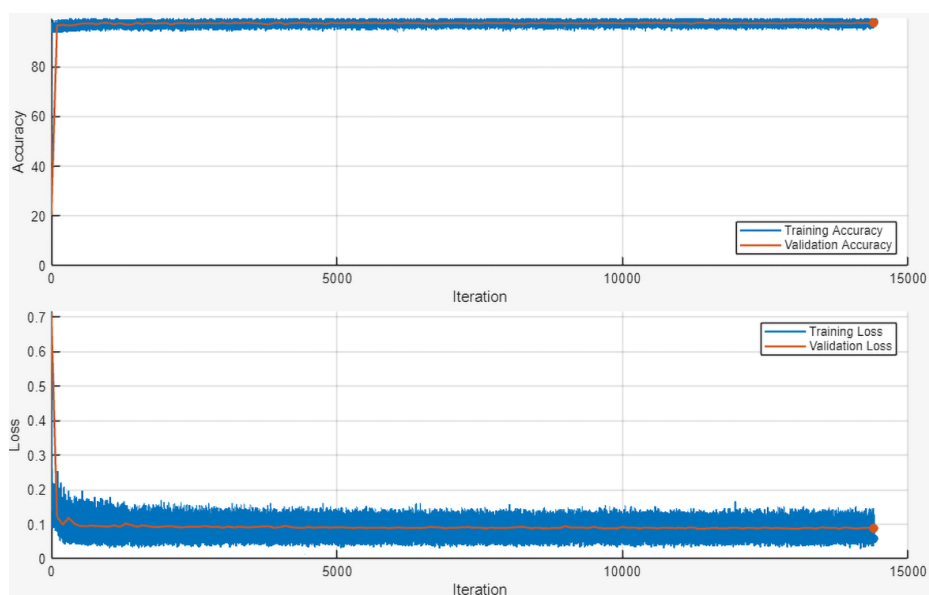
Lp.	Typ warstwy	Opis
1	Sequence Input	5-elementowe wejście sekwencyjne
2	LSTM	1024 jednostek ukrytych
3	Dropout	50% deaktywowanych neuronów
4	LSTM	2048 jednostek ukrytych
5	Dropout	50% deaktywowanych neuronów
6	Fully Connected	2-elementowa warstwa w pełni połączona
7	Softmax	Funkcja aktywacji
8	ClassificationLayer	Warstwa klasyfikacyjna dla dwóch klas

Algorytm 6.1. Kodowanie architektury i procesu uczenia sieci LSTM dla modelu klasyfikacyjnego wystąpienia awarii

```
% Definiowanie warstw sieci LSTM
inputSize = 5; % liczba cech (5)
numHiddenUnits = 1024;
numClasses = 2; % liczba cech (5)
layers = [
    sequenceInputLayer(inputSize,"Name","sequence")
    lstmLayer(numHiddenUnits,"Name","lstm","OutputMode","last")
    dropoutLayer(0.5,"Name","dropout")
    lstmLayer(2048,"Name","lstm1","OutputMode","last")
    dropoutLayer(0.5,"Name","dropout1")
    fullyConnectedLayer(numClasses)
    softmaxLayer];
net = dlnetwork(layers);
net.Initialized
%% Trenowanie Modelu
% Konwersja danych wyjściowych do kategoriycznych
Y_train = categorical(Y_train);
Y_test = categorical(Y_test);
Y_val = categorical(Y_val);
initialLearnRate = 0.001; % Zmniejszona początkowa szybkość uczenia
maxEpochs = 1000; % Maksymalna liczba epok
miniBatchSize = 256; % rozmiar wsadu
options = trainingOptions('adam', ...
    'MaxEpochs', maxEpochs, ...
    'MiniBatchSize', miniBatchSize, ...
    'ValidationData', {X_val_cell, Y_val}, ...
    'ValidationFrequency', 100, ...
    'ValidationPatience', 30, ...
    'Verbose', 0, ...
    'Plots', 'training-progress', ...
    'Metrics', 'accuracy');
% Trenowanie sieci LSTM
[netLSTM,info] = trainnet(X_train_cell, Y_train_balanced,
net,"crossentropy", options);
```

Ryc. 6.5 przedstawia wykresy uczenia sieci LSTM dla modelu klasyfikacyjnego wystąpienia awarii. Proces uczenia został zakończony przed osiągnięciem maksymalnej liczby iteracji, w 534. epoce po 14400 iteracjach. Wykres górny przedstawia zmianę dokładności modelu LSTM w funkcji iteracji podczas procesu uczenia. Oś pozioma (X) reprezentuje liczbę

iteracji, natomiast oś pionowa (Y) przedstawia dokładność w procentach (%). Niebieskie linie wskazują dokładność modelu na zbiorze uczącym, natomiast czerwone oznaczają dokładność modelu na zbiorze walidacyjnym. Dokładność modelu na zbiorze uczącym i walidacyjnym wydaje się stabilna i waha się wokół pewnej wartości przez większość iteracji. Podsumowując, dokładność na zbiorze walidacyjnym jest wysoka, co sugeruje, że model dobrze generalizuje na nowych danych.



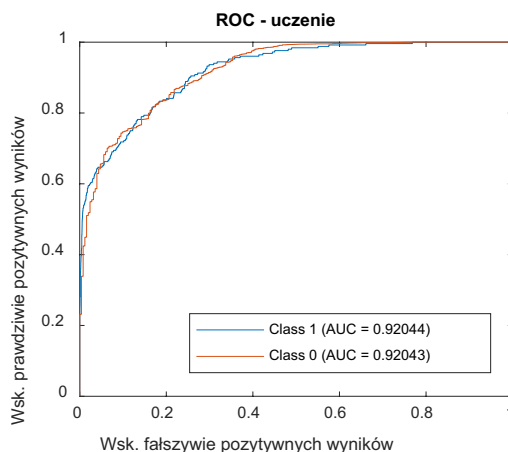
Ryc. 6.5. Wykresy uczenia sieci LSTM dla modelu klasyfikacyjnego wystąpienia awarii

Wykres dolny przedstawia zmianę wartości funkcji straty modelu LSTM w funkcji iteracji podczas procesu uczenia. Wartość funkcji straty na zbiorze uczącym i walidacyjnym jest stabilna przez większość iteracji, z lekkimi wahaniami. Strata na zbiorze walidacyjnym utrzymuje się na niskim poziomie, co sugeruje skuteczność modelu w minimalizacji błędu predykcji.

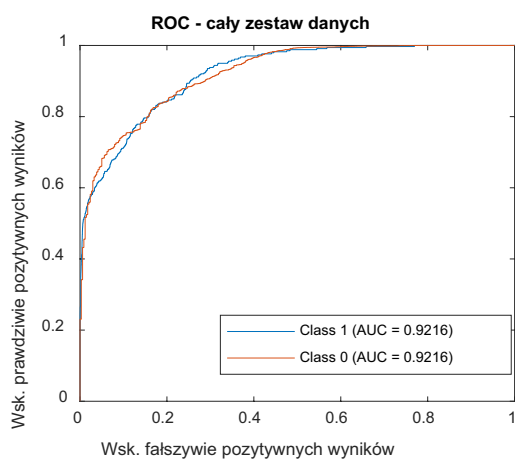
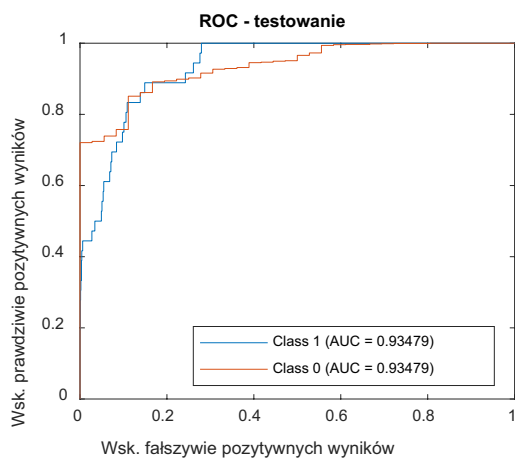
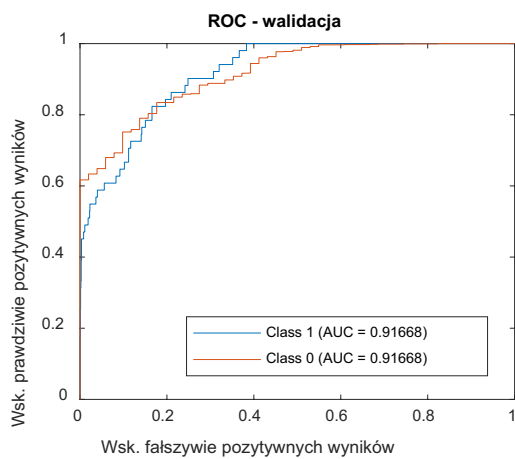
Dokładność oraz wartość funkcji straty pokazują, że proces uczenia modelu LSTM jest stabilny, a model nie wykazuje oznak nadmiernego dopasowania ani niedopasowania. Wysoka dokładność na zbiorze walidacyjnym podkreśla dobrą zdolność modelu do generalizacji na nowych danych. Niskie wartości straty na zbiorze walidacyjnym sugerują, że model skutecznie minimalizuje błąd predykcji, co jest istotne dla dokładnych prognoz. Te wyniki sugerują, że model jest dobrze wytrenowany i efektywnie przewiduje wystąpienie awarii na podstawie danych sekwencyjnych.

Ryc. 6.6 przedstawia krzywe ROC dla sieci LSTM wystąpienia awarii dla różnych zestawów danych: uczących, walidacyjnych, testowych oraz całego zestawu danych. Dla etapu uczenia krzywe ROC obu klas (Class 1 i Class 0) są bardzo zbliżone, co sugeruje, że model skutecznie rozróżnia te klasy podczas procesu uczenia. Wartości AUC (ang. *area under curve*) wynoszą 0,92044 dla klasy 1 oraz 0,92043 dla klasy 0, co wskazuje na wysoką skuteczność modelu w trakcie tego etapu. W przypadku walidacji krzywe ROC również pokazują wysoką skuteczność modelu w rozróżnianiu obu klas. Wartości AUC dla obu klas wynoszą 0,91668, co sugeruje, że model zachowuje stabilność i dobrze generalizuje na danych walidacyjnych. Na etapie testowania krzywe ROC dla danych testowych ukazują jeszcze lepsze dopasowanie modelu. Wartości AUC wynoszą 0,93479 dla obu klas, co jest najwyższym wynikiem spośród wszystkich etapów. Oznacza to, że model bardzo dobrze radzi sobie na nowych, nieznanych danych. Analizując cały zestaw danych, krzywe ROC także pokazują wysoką skuteczność modelu w klasyfikacji. Wartości AUC dla obu klas wynoszą 0,9216, co potwierdza ogólną dobrą wydajność modelu na całym zbiorze danych.

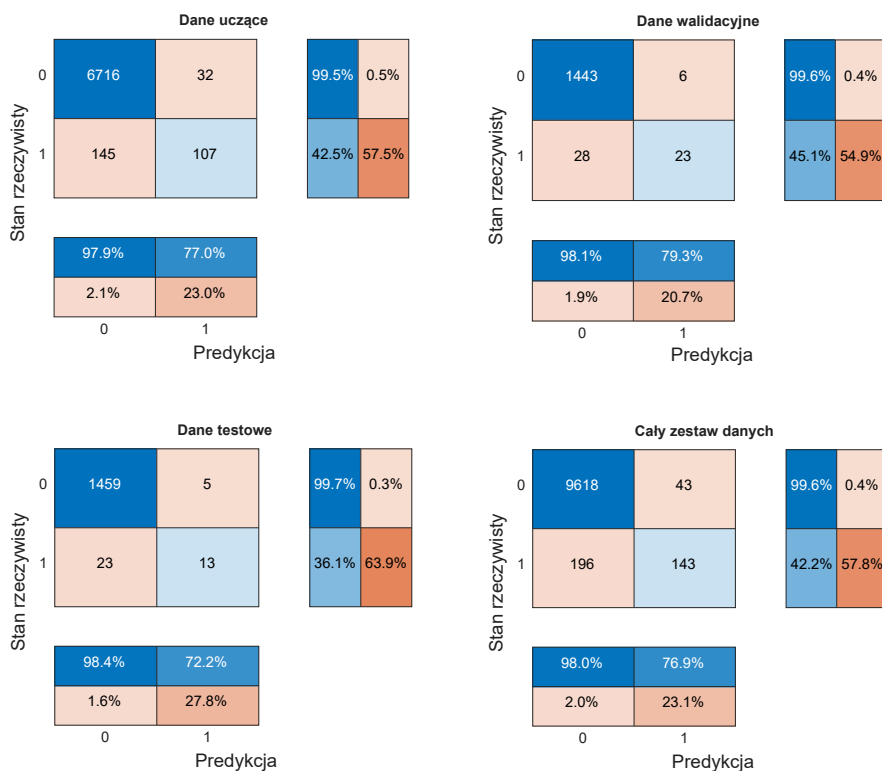
Model LSTM użyty do wykrywania awarii charakteryzuje się wysoką skutecznością na każdym etapie procesu modelowania. Wysokie wartości AUC (powyżej 0,9) dla obu klas na etapach uczenia, walidacji, testowania oraz dla całego zestawu danych wskazują na zdolność modelu do skutecznego rozróżniania klas. Sugeruje to, że model jest dobrze dopasowany i wykazuje zdolność do generalizacji na nowych danych.



Ryc. 6.6. Krzywe ROC dla sieci LSTM wystąpienia awarii



Ryc. 6.6. Krzywe ROC dla sieci LSTM wystąpienia awarii, cd.



Ryc. 6.7. Macierz pomyłek dla sieci LSTM wystąpienia awarii

Ryc. 6.7 przedstawia macierze pomyłek dla modelu LSTM, który został zastosowany do klasyfikacji awarii maszyn dla różnych zestawów danych: uczących, walidacyjnych, testowych oraz całego zestawu danych. W przypadku danych uczących model klasyfikuje prawidłowo 6716 przypadków klasy 0 oraz 107 przypadków klasy 1. Błędne klasyfikacje obejmują 32 przypadki klasy 0 i 145 przypadków klasy 1. Dokładność dla klasy 0 wynosi 99,5%, natomiast dla klasy 1 – 42,5%. Wysoka dokładność klasy 0 wskazuje na skuteczne rozpoznawanie tej klasy przez model w trakcie uczenia. Niemniej jednak niższa dokładność dla klasy 1 sugeruje, że model może mieć trudności z identyfikacją tej klasy. W analizie danych walidacyjnych model osiąga dokładność 99,6% dla klasy 0 oraz 45,1% dla klasy 1. Poprawnie klasyfikuje 1443 przypadki klasy 0 i 23 przypadki klasy 1, przy czym błędne klasyfikacje obejmują 6 przypadków klasy 0 i 28 przypadków klasy 1. Te wyniki wskazują na dobrą ogólną wydajność modelu klasyfikacji klasy 0, jednak zauważalne są pewne problemy z dokładnością dla klasy 1, co może wpływać na stabilność predykcji w praktycznych zastosowaniach. Dane testowe pokazują, że model klasyfikuje

poprawnie 1459 przypadków klasy 0 i 13 przypadków klasy 1, przy błędnych klasyfikacjach 5 przypadków klasy 0 i 23 przypadków klasy 1. Dokładność dla klasy 0 wynosi 99,7%, natomiast dla klasy 1 – 36,1%. Analiza całego zestawu danych ujawnia, że model osiąga dokładność 99,6% dla klasy 0 oraz 42,2% dla klasy 1. Model poprawnie klasyfikuje 9618 przypadków klasy 0 oraz 143 przypadki klasy 1, z błędami w 43 przypadkach klasy 0 i 196 przypadkach klasy 1. Wyniki te potwierdzają ogólną wysoką skuteczność modelu klasyfikacji klasy 0, jednak wyzwanie stanowi utrzymanie wysokiej dokładności dla klasy 1.

Po wykonaniu obu modeli do klasyfikowania, czy usterki wystąpią, czy też nie (MLP oraz LSTM), dokonano ich porównania. Porównanie modeli na danych testowych jest istotne, ponieważ te dane nie były używane podczas trenowania modelu, a więc najbardziej przypominają rzeczywiste warunki pracy modelu. Dane testowe pozwalają na ocenę zdolności generalizacji modelu, co jest niezbędne dla oceny jego wydajności w praktycznych zastosowaniach. Istotne jest również to, że zastosowano w przypadku obu modeli ten sam zestaw danych. Porównanie wskaźników jakościowych dla danych testowych dla obu modeli zestawia tabela 6.3.

Tabela 6.3. Porównanie modeli MLP oraz LSTM modelu wystąpienia awarii (wskaźniki jakości)

Model	TN	FP	FN	TP	Dokładność [%]	Precyzja [%]	Czułość [%]	F1-score [%]
MLP	1460	24	4	12	98,1	33,3	75,0	46,1
LSTM	1459	5	23	13	98,1	72,2	36,1	48,1

Analiza wyników na danych testowych wykazała, że oba modele osiągają dokładność na poziomie 98,1%. Jednakże precyzja, czyli miara liczby prawdziwie pozytywnych wyników w stosunku do wszystkich pozytywnych przewidzianych przez model, wynosi 33,3% dla sieci neuronowej MLP i 72,2% dla modelu LSTM. Wyższa precyzja modelu LSTM oznacza rzadsze występowanie fałszywie pozytywnych klasyfikacji, co jest istotne w kontekście predykcji awarii, gdzie fałszywe alarmy mogą prowadzić do niepotrzebnych interwencji. Czułość, mierząca zdolność modelu do identyfikacji wszystkich rzeczywistych pozytywnych przypadków, wynosi 75,0% dla modelu MLP i 36,1% dla modelu LSTM. Znacznie wyższa czułość modelu MLP wskazuje na jego większą efektywność w wykrywaniu rzeczywistych przypadków awarii. Jest to kluczowy wskaźnik, ponieważ niski poziom czułości oznacza, że wiele rzeczywistych awarii może zostać przeoczonych. F1-score, będący średnią harmoniczną precyzji i czułości, wynosi 46,1% dla modelu MLP i 48,1% dla modelu

LSTM. Wyższy F1-score modelu LSTM sugeruje, że jest on bardziej wyważony w swoim działaniu, łącząc wysoką precyzję i czułość, co czyni go bardziej niezawodnym modelem do predykcji awarii w porównaniu z MLP.

Biorąc pod uwagę wyższy F1-score (48,1%) i precyzję (72,2%) modelu LSTM w porównaniu z modelem MLP (F1-score: 46,1%, precyzja: 33,3%), model LSTM wydaje się bardziej niezawodny, redukując liczbę fałszywych alarmów. Pomimo wyższej czułości modelu MLP (75,0%) w porównaniu z LSTM (36,1%), co oznacza lepsze wykrywanie rzeczywistych przypadków awarii, niższa precyzja modelu MLP wskazuje na większą liczbę fałszywych alarmów.

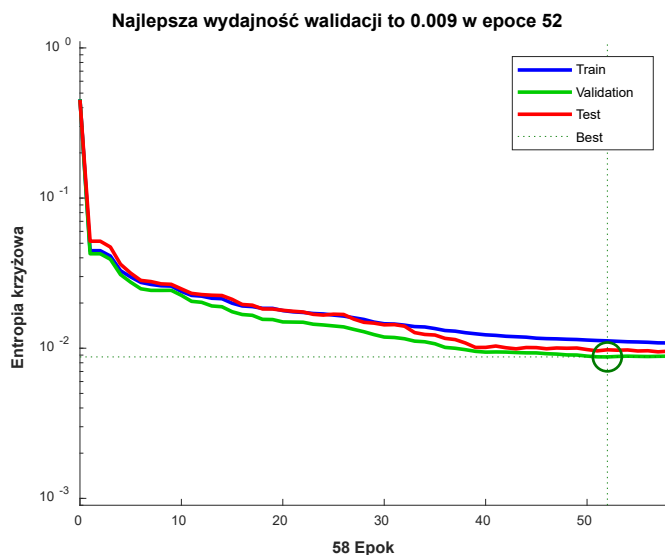
W kontekście danych historycznych dla analizowanego zestawu danych, gdzie liczba awarii była niska, ważne jest, aby model skutecznie radził sobie z niewielką liczbą pozytywnych przykładów (awarii) oraz minimalizował fałszywie negatywne wyniki. W scenariuszach z nie zrównoważonymi klasami wysoka czułość jest istotna, ponieważ odzwierciedla zdolność modelu do prawidłowego wykrywania awarii. W takich przypadkach model o wyższej czułości, mimo większej liczby fałszywych alarmów, może być bardziej korzystny, gdyż każde przeoczenie awarii może prowadzić do poważnych konsekwencji.

Biorąc pod uwagę te aspekty, model MLP z wyższą czułością (75,0%) wydaje się bardziej odpowiedni do zastosowań w predykcji w czasie rzeczywistym, gdzie krytyczne jest wykrywanie wszystkich rzeczywistych przypadków awarii. Pomimo niższej precyzji i większej liczby fałszywych alarmów zdolność modelu MLP do wykrywania większej liczby rzeczywistych awarii czyni go bardziej niezawodnym w kontekście zapobiegania potencjalnym krytycznym awariom.

6.3. Wyniki modelowania klasyfikatora dotyczącego określania typu awarii

W przypadku modelowania typu awarii na wyjściu można wyróżnić 6 klas: 1 – brak awarii, 2 – awaria narzędzia, 3 – awaria chłodzenia, 4 – awaria mocy, 5 – awaria przeciążenia oraz 6 – awarie losowe.

W przypadku modelowania klasyfikatora dla określania typu awarii z wykorzystaniem sieci neuronowej MLP najlepsze wyniki otrzymano dla sieci z 12 neuronami w warstwie ukrytej. Ryc. 6.8 prezentuje przebieg uczenia się sieci neuronowej, monitorowany za pomocą funkcji entropii krzyżowej w zależności od liczby epok uczących.



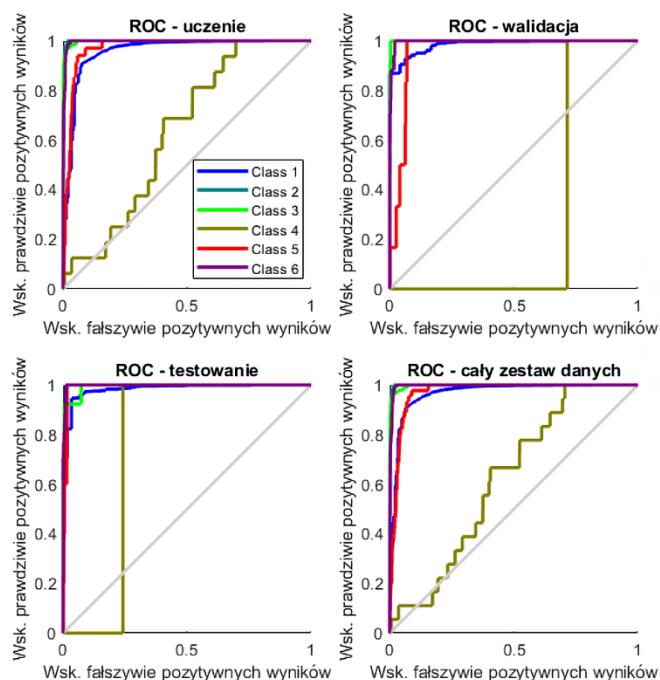
Ryc. 6.8. Przebieg uczenia się sieci neuronowej MLP klasyfikującej typ usterek

Proces uczenia MLP trwał przez 58 epok, a najlepsza wydajność na zbiorze walidacyjnym została osiągnięta po 52 epokach, z wartością entropii krzyżowej wynoszącą 0,009. Po tym punkcie, oznaczonym zielonym okręgiem, krzywa walidacyjna przestaje się znacząco poprawiać, co sugeruje zastosowanie techniki wczesnego zatrzymania w celu uniknięcia przeuczenia.

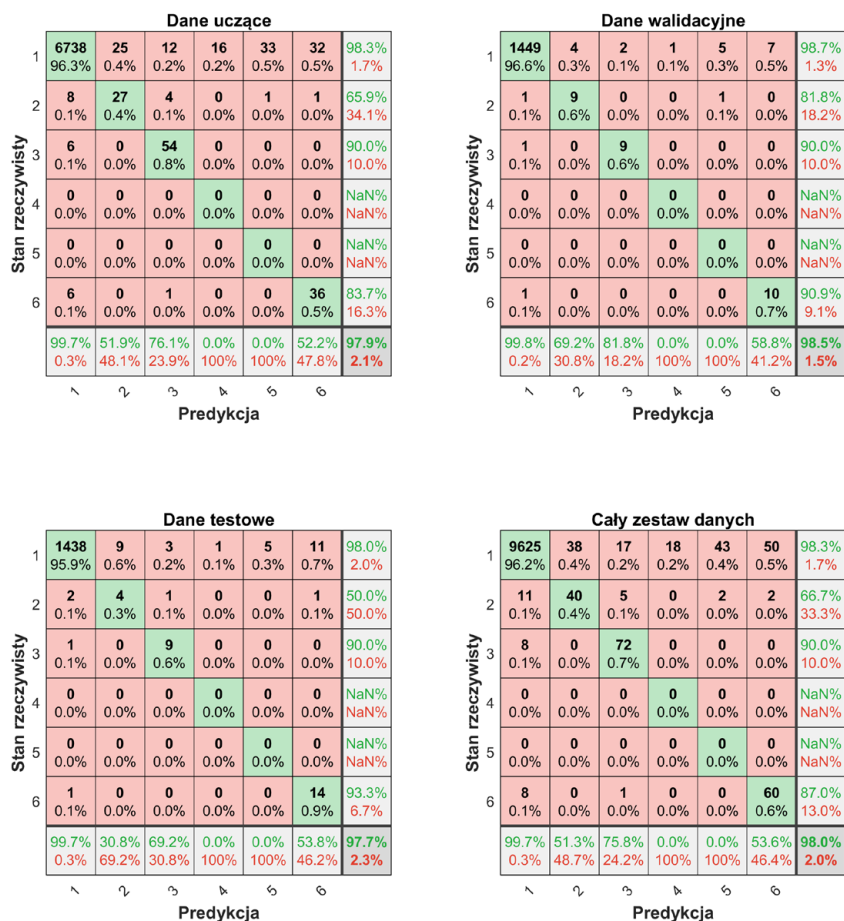
Wartości entropii krzyżowej dla zbiorów uczącego, walidacyjnego i testowego systematycznie maleją wraz z postępem procesu uczenia, co sugeruje, że sieć neuronowa staje się coraz lepsza w rozpoznawaniu wzorców. Dodatkowo stosunkowo mała różnica pomiędzy krzywymi walidacyjną a testową wskazuje na solidną zdolność sieci do generalizacji, czyli poprawnego działania na nowych, niewidzianych wcześniej danych.

Wykresy (ryc. 6.9) przedstawiają krzywe ROC dla różnych klas w problemie wieloklasowej klasyfikacji typu usterek. Na wykresie ROC dla zestawu uczącego model wykazuje bardzo dobrą zdolność do rozróżniania większości klas. Wysokie wartości dla klas 1, 2, 3, 5 i 6 wskazują na wysoką dokładność klasyfikacji tych klas. Klasa 4, reprezentowana przez żółtą krzywą, wykazuje jednak znacznie gorszą wydajność, co sugeruje trudności modelu w prawidłowej klasyfikacji tej klasy. W zbiorze walidacyjnym wartości dla klas 1, 2, 3, 5 i 6 pozostają wysokie, co wskazuje na dobrą generalizację modelu na nowe dane. Klasa 4 ponownie wykazuje znacznie gorszą wydajność, co jest widoczne na krzywej ROC, która jest bliska przekątnej, sugerując, że model

ma trudności z poprawnym rozróżnieniem tej klasy od innych. W zestawie testowym wyniki są podobne do tych z zestawu walidacyjnego. Analizując krzywą ROC dla całego zestawu danych, można zauważyć, że ogólna wydajność modelu utrzymuje się na wysokim poziomie dla klas 1, 2, 3, 5 i 6, co jest zgodne z wynikami dla poszczególnych zestawów danych. Klasa 4 wykazuje najniższą wartość, co sugeruje, że model ma największe problemy z klasyfikacją tej klasy w całym zbiorze danych.



Ryc. 6.9. Krzywe ROC dla sieci neuronowej MLP klasyfikującej typ usterek

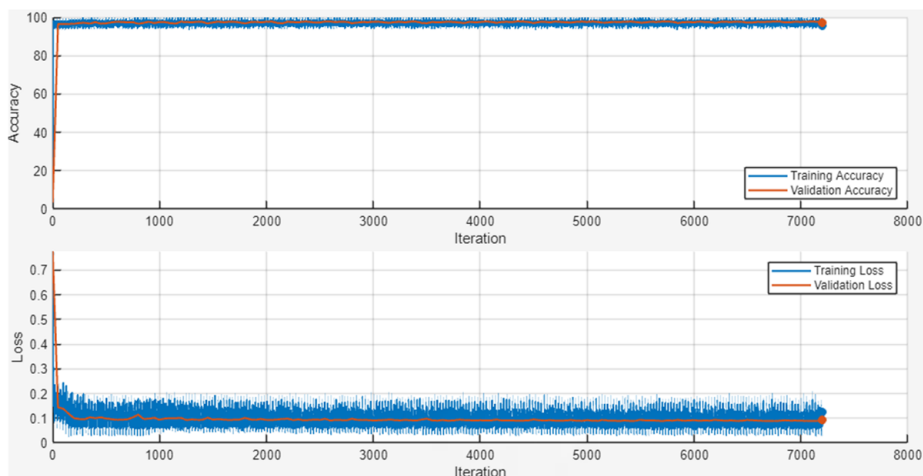


Ryc. 6.10. Macierz pomyłek dla sieci neuronowej MLP klasyfikującej typ usterek

Kolejnym ze wskaźników skuteczności dopasowania sieci neuronowej MLP do danych jest macierz pomyłek. Na ryc. 6.10 przedstawiono macierze pomyłek dla sieci neuronowej wytrenowanej do klasyfikacji wieloklasowej na różnych podzbiorach danych: uczących, walidacyjnych, testowych oraz na całym zestawie danych. Zielone komórki, które oznaczają poprawne klasyfikacje, mają wysokie wartości procentowe. Model klasyfikuje klasę 1 z bardzo wysoką dokładnością we wszystkich zbiorach danych (ponad 98%). Klasa 3 również jest dobrze klasyfikowana, szczególnie w danych uczących (54 przypadki), walidacyjnych (9 przypadków) i testowych (9 przypadków), co wskazuje na solidność modelu dla tej klasy. Dla tej klasy dokładność modelu osiągnięto na poziomie 90%. Brak poprawnych klasyfikacji dla klasy 4 i 5 we wszystkich zbiorach danych. Może to sugerować, że są one trudne do rozpoznania przez model lub są bardzo

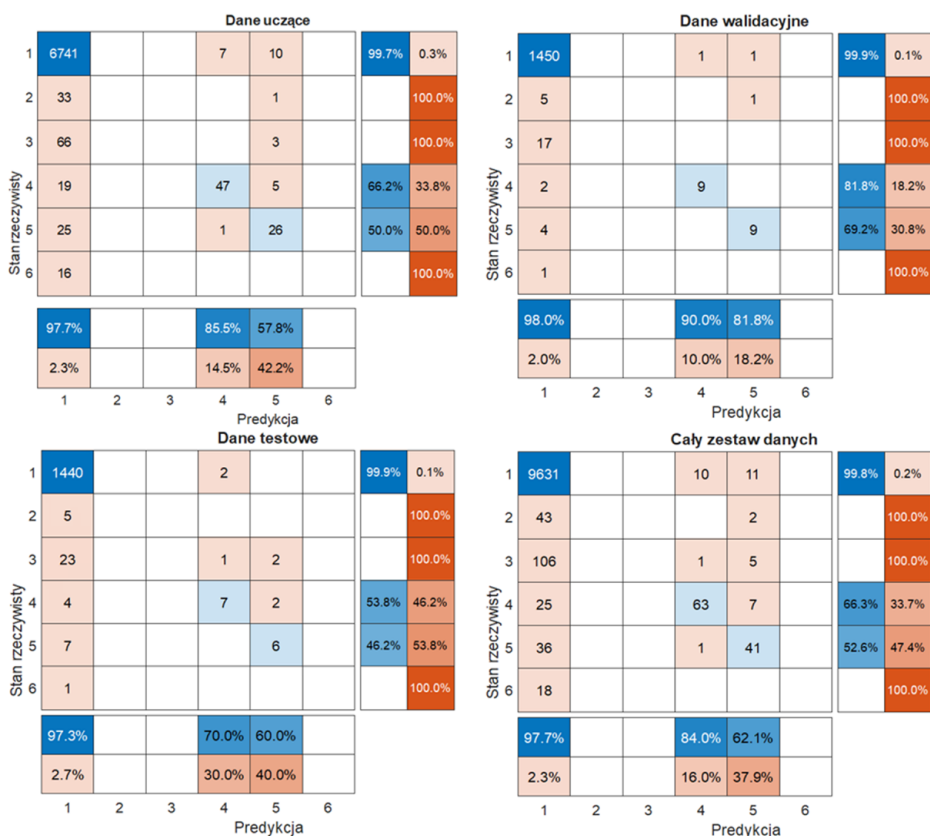
rzadkie. Model ma umiarkowaną skuteczność dla klasy 6, szczególnie w danych uczących (36 przypadków – 83,7%), walidacyjnych (10 przypadków – 90,9%) i testowych (14 przypadków – 93,3%). Klasa 2 ma niską skuteczność klasyfikacji we wszystkich zbiorach danych. Chociaż model potrafi poprawnie zaklasyfikować kilka przypadków klasy 2, liczba poprawnych klasyfikacji jest stosunkowo niska w porównaniu z innymi klasami. Niska liczba poprawnych klasyfikacji dla klasy 2 we wszystkich zbiorach danych sugeruje, że model ma trudności z jej rozpoznaniem. Może to wynikać z niewystarczającej liczby przykładów klasy 2 w danych uczących lub z niewystarczającej rozróżnialności cech dla tej klasy.

Dla modelowania sieci LSTM dla modelu klasyfikującego typ usterek najlepsze parametry otrzymano dla dwóch warstw LSTM w modelu oraz dla liczby jednostek ukrytych w pierwszej warstwie – 1024, zaś w drugiej – 2048. Dla takiej samej struktury jak w przypadku modelu klasyfikującego wystąpienie awarii. Architekturę tej sieci prezentuje tabela 6.2 (podrozdział 6.2).



Ryc. 6.11. Wykresy uczenia sieci LSTM dla modelu klasyfikacyjnego typ usterek

Ryc. 6.11 przedstawia wykresy uczenia sieci LSTM dla modelu klasyfikacyjnego typ usterek. Proces uczenia zakończył się wcześniej, w 267. epoce, po 7200 iteracjach. Dokładność oraz wartość funkcji straty wskazują na stabilny proces uczenia modelu LSTM, bez oznak przeuczenia lub niedouczenia. Wysoka dokładność na zbiorze walidacyjnym świadczy o dobrej zdolności modelu do generalizacji na nowych danych. Niskie wartości straty na zbiorze walidacyjnym sugerują, że model skutecznie redukuje błąd predykcji, co jest istotne dla precyzyjnych prognoz. Niskie wartości straty i wysoka dokładność na danych walidacyjnych wskazują, że model nie ma problemu z niedouczeniem.



Ryc. 6.12. Macierz pomyłek dla sieci LSTM dla modelu klasyfikacyjnego typ usterek

Macierze pomyłek dla sieci LSTM (ryc. 6.12) pokazują wydajność modelu na zestawach danych uczących, walidacyjnych, testowych oraz dla całego zestawu danych. Dla danych uczących macierz pomyłek wskazuje, że model osiągnął bardzo wysoką dokładność (99,7%) w klasyfikacji przypadków braku awarii. Jednakże dokładność dla innych klas awarii, szczególnie dla klas 4 i 5, jest zauważalnie niższa (odpowiednio 66,2% i 50,0%). Wysoka liczba błędnie sklasyfikowanych przypadków w tych klasach sugeruje, że model może mieć trudności z rozróżnianiem między pewnymi typami awarii. W przypadku klasy 4, znaczący odsetek rzeczywistych przypadków został błędnie przypisany do klasy 1, co może wynikać z podobieństwa cech między tymi dwoma klasami. Co więcej, model nie sklasyfikował żadnego przypadku jako klasy 2, 3 i 6, co wskazuje na poważne niedouczenie dla tych klas i sugeruje, że model może mieć problem z wykrywaniem tych specyficznych typów awarii.

Analiza danych walidacyjnych pokazuje, że model utrzymuje wysoką dokładność klasyfikacji dla 1 klasy. Wyniki te są zgodne z obserwacjami z danych

uczących, co sugeruje, że problemy z rozróżnianiem klas 4 i 5 są utrwalone również na etapie walidacji. W przypadku danych testowych model nadal wykazuje wysoką dokładność dla klasy 1 (99,9%) i umiarkowaną dla klas 4 i 5 (odpowiednio 53,8% i 46,2%). Co więcej, model ponownie nie sklasyfikował żadnego przypadku jako klasy 2, 3 i 6, co jest bardzo istotnym problemem, sugerującym, że w ogóle nie nauczył się rozpoznawać tych typów awarii.

Dla całego zestawu danych ogólne wzorce są podobne do tych obserwowanych w danych uczących, walidacyjnych i testowych. Model osiąga wysoką dokładność dla klasy 1 (99,8%) i umiarkowaną dla klasy 4 (66,3%) oraz niską dla klasy 5 (52,6%). Pomimo ogólnej wysokiej dokładności problemy z klasyfikacją mniej reprezentatywnych klas awarii są nadal obecne. Średnia precyzja i czułość dla tych klas pozostają niższe, co może wpłynąć na ogólną wydajność modelu w rzeczywistych zastosowaniach. Co istotne, model nadal nie sklasyfikował żadnego przypadku jako klasy 2, 3 i 6, co wskazuje na strukturalny problem z rozpoznawaniem tych awarii.

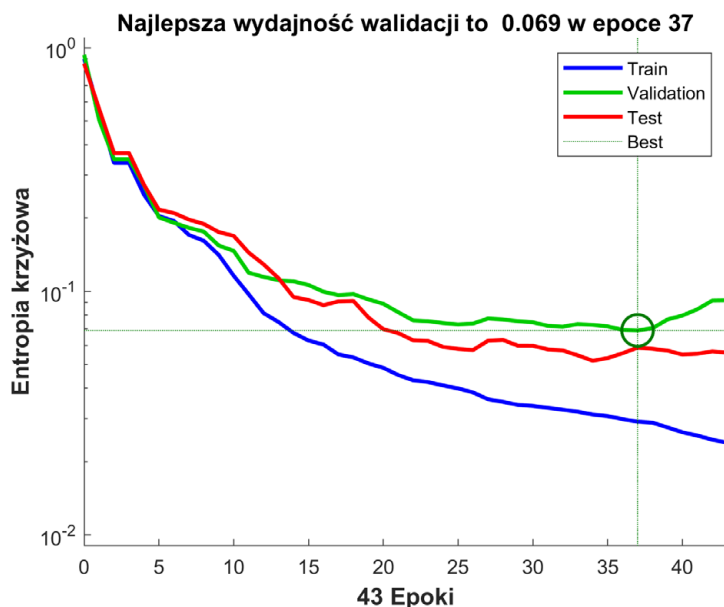
Model LSTM osiąga wysoką ogólną dokładność, szczególnie w klasyfikacji przypadków braku awarii. Jednakże wykazuje trudności w prawidłowym klasyfikowaniu pewnych typów awarii, co jest konsekwentne we wszystkich zestawach danych. Szczególnie niepokoi fakt, że model nie jest w stanie sklasyfikować klas 2, 3 i 6, co wskazuje na potrzebę gruntownej rewizji procesu trenowania, być może poprzez np. przeprojektowanie architektury modelu. Dane te były analizowane dla różnych architektur LSTM, jednakże wyniki wskazują, że sieci LSTM nie sprawdzają się w tym konkretnym przypadku.

Ze względu na analizę macierzy pomyłek, która nie klasyfikuje w ogóle klas 2, 3 i 6, co było zaprezentowane dla najlepszej otrzymanej sieci LSTM, pominięto dalszą analizę ROC oraz wskaźników jakościowych, takich jak dokładność, precyzja, czułość i F1-score.

W wyniku analizowania wydajności modeli klasyfikacyjnych stosowanych do określania typów awarii zauważono, że dotychczasowe podejście, zarówno przy użyciu sieci neuronowej MLP, jak i sieci LSTM, miały ograniczenia w skutecznej klasyfikacji niektórych klas awarii. Modele miały szczególne trudności z rozpoznawaniem klas 2, 3 i 6, co było widoczne na podstawie macierzy pomyłek. W efekcie tych wyników zdecydowano się na zmianę koncepcji modelowania. Nowe podejście polegało na ograniczeniu zbioru danych jedynie do rekordów zawierających usterki, a następnie ponownym przeprowadzeniu modelowania na takim zredukowanym zestawie. Ten zredukowany zbiór danych obejmował 348 rekordów.

6.4. Wyniki modelowania klasyfikatora dotyczącego określania typu awarii na ograniczonym zbiorze danych

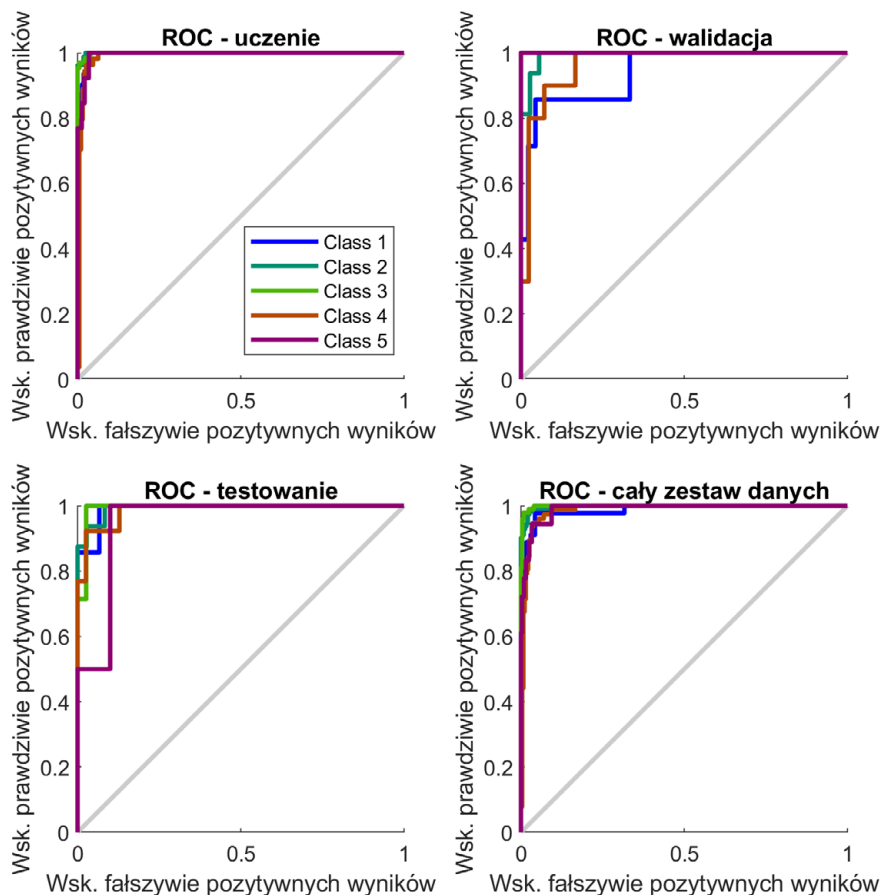
Zmiana koncepcji modelowania była konieczna ze względu na niską jakość wyników osiąganych przez dotychczasowe modele klasyfikacyjne, zarówno MLP, jak i LSTM. Modele te miały trudności z prawidłowym rozpoznawaniem niektórych klas awarii, w szczególności klas 2, 3 i 6, co wykazano na podstawie analizy macierzy pomyłek. W odpowiedzi na te problemy postanowiono ograniczyć zbiór danych jedynie do rekordów zawierających usterki i przeprowadzić ponowne modelowanie na takim zredukowanym zestawie, który obejmował 348 rekordów. W przypadku nowego zbioru danych wyróżniono 5 klas: 1 – awaria narzędzia, 2 – awaria chłodzenia, 3 – awaria mocy, 4 – awaria przeciążenia oraz 5 – awarie losowe.



Ryc. 6.13. Przebieg uczenia się sieci neuronowej MLP klasyfikującej typ usterek na zmniejszonym zestawie danych

W procesie modelowania klasyfikatora do identyfikacji typów awarii na zmniejszonym zestawie danych, przy zastosowaniu sieci neuronowej MLP, najkorzystniejsze rezultaty uzyskano dla sieci z 15 neuronami w warstwie ukrytej. Na ryc. 6.13 przedstawiono przebieg uczenia się tej sieci neuronowej MLP, monitorowany za pomocą funkcji entropii krzyżowej w odniesieniu do liczby

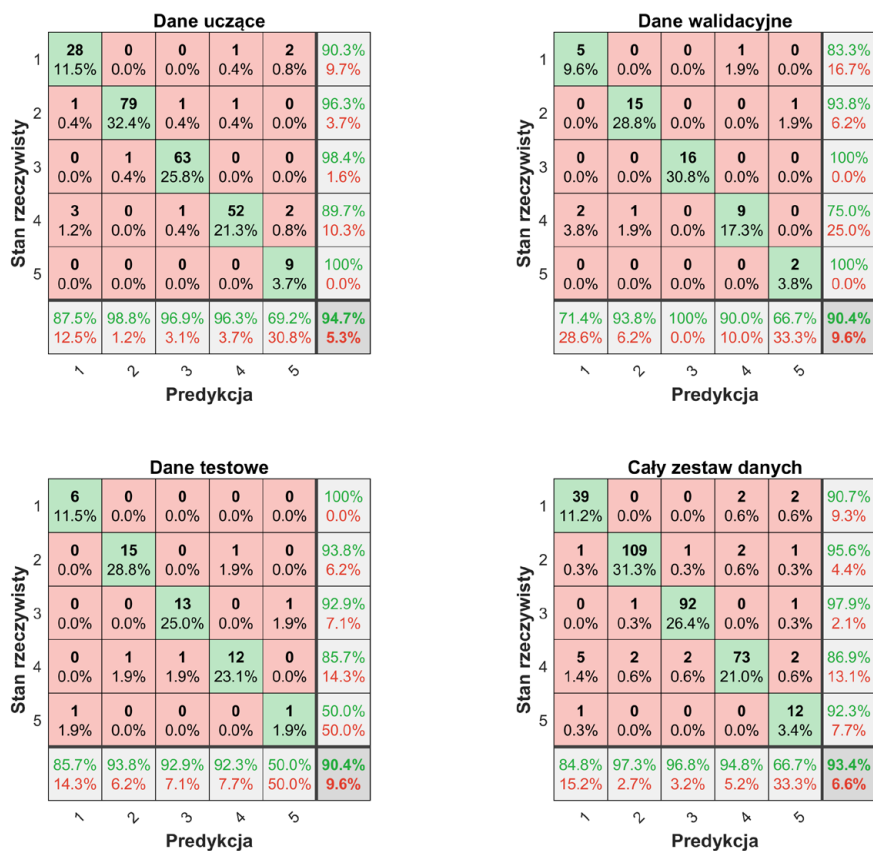
epok uczenia. Na podstawie analizy procesu uczenia można stwierdzić, że proces uczenia sieci neuronowej trwał 43 epoki i przebiegał efektywnie do momentu osiągnięcia najlepszej wydajności na zbiorze walidacyjnym w 37. epoce.



Ryc. 6.14. Krzywe ROC dla sieci neuronowej MLP klasyfikującej typ usterek na zmniejszonym zestawie danych

Wykresy ROC (ryc. 6.14) przedstawiają zdolność modelu do klasyfikacji pięciu różnych klas awarii w zestawach danych uczących, walidacyjnych, testowych oraz całym zbiorze danych. Na wykresie dla zbioru uczącego widać, że model wykazuje bardzo wysoką zdolność rozróżniania klas. Krzywe dla wszystkich klas znajdują się blisko lewego górnego rogu wykresu, co wskazuje na wysoką skuteczność modelu w przypadku danych do trenowania. Dla zbioru walidacyjnego krzywe ROC również pokazują wysoką wydajność modelu, chociaż widać

niewielkie różnice w porównaniu ze zbiorem uczącym Krzywe dla niektórych klas są nieco dalej od lewego górnego rogu, co sugeruje, że model może mieć niewielkie trudności z generalizacją na nowe dane, ale ogólnie nadal jest skuteczny. Analizując wykres ROC dla zbioru testowego, można zauważyć, że krzywe są podobne do tych z zestawu walidacyjnego. Model utrzymuje wysoką zdolność do rozróżniania klas, jednak podobnie jak w przypadku danych walidacyjnych, niektóre krzywe nie są tak blisko idealnego rozróżnienia jak w zestawie uczącym.



Ryc. 6.15. Macierz pomyłek dla sieci neuronowej MLP klasyfikującej typ usterek na zmniejszonym zestawie danych

Ryc. 6.15 przedstawia macierze pomyłek dla zbiorów danych uczących, walidacyjnych, testowych oraz całego zestawu danych. Obrazują one dokładność modelu w klasyfikacji różnych typów awarii na zmniejszonym zestawie danych. Dla zbioru uczącego model wykazuje wysoką skuteczność klasyfikacji w przypadku klas 1, 2 i 3. Klasa 1 została poprawnie zaklasyfikowana

w 90,3% przypadków, klasa 2 w 96,3%, a klasa 3 w 98,4% przypadków. Klasa 4 osiągnęła dokładność na poziomie 89,7%, co również jest wynikiem dobrym, chociaż nieco niższym w porównaniu z poprzednimi klasami. Klasa 5 została poprawnie zaklasyfikowana w 100% przypadków, co sugeruje, że model jest w stanie skutecznie rozpoznawać tę klasę na danych uczących, chociaż należy zauważyć, że jest to klasa z mniejszą liczbą przypadków.

W zbiorze walidacyjnym widoczne są pewne różnice w porównaniu ze zbiorem uczącym. Klasa 1 została poprawnie zaklasyfikowana w 83,3% przypadków, co wskazuje na nieco niższą skuteczność w porównaniu z danymi uczącymi. Klasa 2 zachowała wysoką dokładność na poziomie 93,8%, podobnie jak w przypadku danych uczących. Klasa 4 została poprawnie zaklasyfikowana w 75% przypadków, co jest również niższym wynikiem w porównaniu ze zbiorem uczącym. Klasa 5, podobnie jak w przypadku danych uczących, została poprawnie zaklasyfikowana w 100% przypadków, jednakże ponownie należy zauważyć, że jest to klasa z mniejszą liczbą przypadków. Skuteczność modelu w przypadku klasy 3 wyniosła tak jak dla klasy 5 – 100%.

Dla zbioru testowego dla klasy 1 model poprawnie sklasyfikował 6 przypadków (100%), zaś w przypadku klasy 2 model osiągnął dokładność na poziomie 93,8%, poprawnie klasyfikując 15 przypadków. Jest to wynik bardzo dobry, wskazujący na wysoką zdolność modelu do identyfikacji tej klasy w zbiorze testowym. Nieco niższą dokładność osiągnął dla klasy 3 na poziomie 92,9% (13 poprawnie sklasyfikowanych przypadków, 1 niepoprawnie). Klasa 4 wykazuje, że model właściwie sklasyfikował 12 przypadków. Chociaż dokładność dla tej klasy wynosi 85,7%, istnieją również przypadki błędnej klasyfikacji, co może wskazywać na pewne trudności modelu w odróżnianiu jej od innych. Klasa 5 ma niższą dokładność, na poziomie 50,0%, jeden przypadek został sklasyfikowany poprawnie, zaś drugi błędnie, w tej klasie były tylko dwa rekordy, co ma znaczący wpływ na taki wynik.

Dla całego zestawu danych model wykazuje wysoką skuteczność w klasyfikacji klas 1, 2, 3 i 4, z wartościami powyżej 90%. Klasa 4 osiągnęła dokładność na poziomie 86,9%, co jest nieco słabszym wynikiem od pozostałych. Wysokie ogółem wartości procentowe poprawnych klasyfikacji powyżej 85% świadczą o dobrej dokładności modelu.

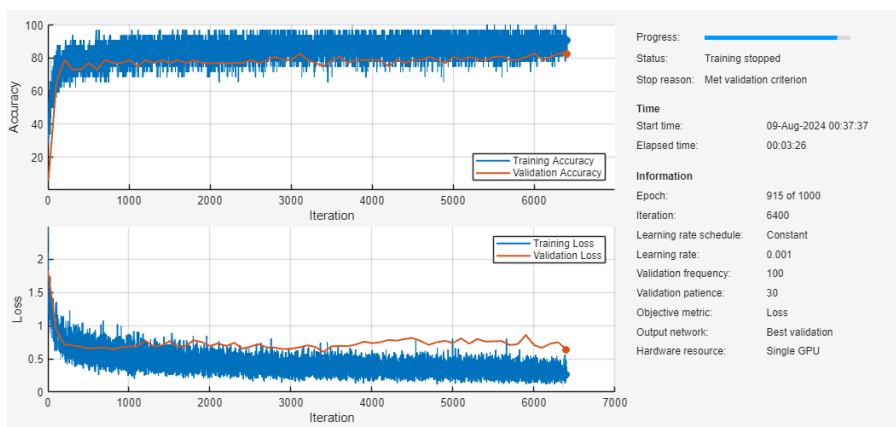
Dla modelowania sieci LSTM dla modelu klasyfikującej typ usterek na zmniejszonym zestawie danych najlepsze parametry otrzymano dla jednej warstwy LSTM w modelu oraz dla liczby jednostek ukrytych 256. Dodatkowo dodano warstwę normalizacyjną. Architekturę tej sieci prezentuje tabela 6.4. Kodowanie architektury i procesu uczenia sieci LSTM dla modelu klasyfikacyjnego wystąpienia awarii na zmniejszonym zestawie danych obrazuje algorytm 6.2.

Tabela 6.4. Architektura sieci LSTM dla modelu klasyfikacyjnego typ usterek na zmniejszonym zestawie danych

Lp.	Typ warstwy	Opis
1	Sequence Input	5-elementowe wejście sekwencyjne
2	LSTM	1024 jednostek ukrytych
3	Batch Normalization	Normalizacja partii
4	Dropout	50% deaktywowanych neuronów
5	Fully Connected	2-elementowa warstwa w pełni połączona
6	Softmax	Funkcja aktywacji
7	ClassificationLayer	Warstwa klasyfikacyjna dla dwóch klas

Algorytm 6.2. Kodowanie architektury i procesu uczenia sieci LSTM dla modelu klasyfikacyjnego wystąpienia awarii na zmniejszonym zestawie danych

```
% Definiowanie warstw sieci LSTM
inputSize = 5; % liczba cech (5)
numHiddenUnits = 256;
numClasses = 5;
layers = [
    sequenceInputLayer(inputSize,"Name","sequence")
    lstmLayer(numHiddenUnits,"Name","lstm","OutputMode","last")
    batchNormalizationLayer("Name","batchnorm_1")
    dropoutLayer(0.5,"Name","dropout")
    fullyConnectedLayer(numClasses)
    softmaxLayer];
net = dlnetwork(layers);
net.Initialized
initialLearnRate = 0.001;
maxEpochs = 1000;
miniBatchSize = 16;
options = trainingOptions('adam', ...
    'MaxEpochs', maxEpochs, ...
    'MiniBatchSize', miniBatchSize, ...
    'ValidationData', {X_val_cell, Y_val}, ...
    'ValidationFrequency', 100, ...
    'ValidationPatience', 30, ...
    'Verbose', 0, ...
    'Plots', 'training-progress', ...
    'Metrics','accuracy');
% Trenowanie sieci LSTM
[netLSTM,info] = trainnet(X_train_cell, Y_train_balanced,
    net,"crossentropy", options);
```



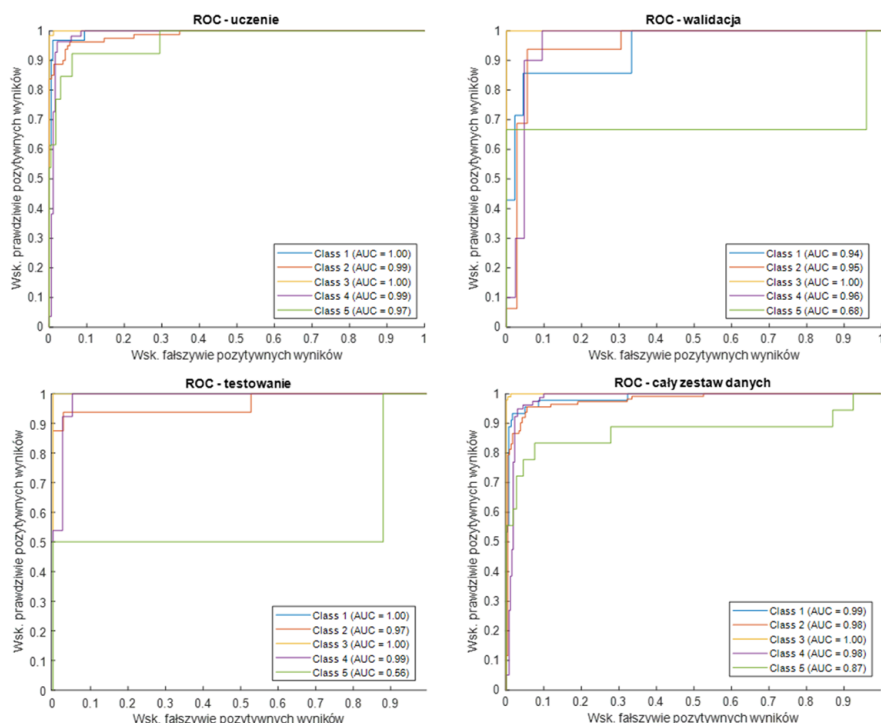
Ryc. 6.16. Wykresy uczenia sieci LSTM dla modelu klasyfikacyjnego typ usterek na zmniejszonym zestawie danych

Na ryc. 6.16 przedstawiono wykresy procesu uczenia sieci LSTM dla modelu klasyfikacji rodzaju awarii. Proces ten zakończono przed osiągnięciem maksymalnej liczby iteracji, dokładnie w 915. epoce, po przeprowadzeniu 6400 iteracji.

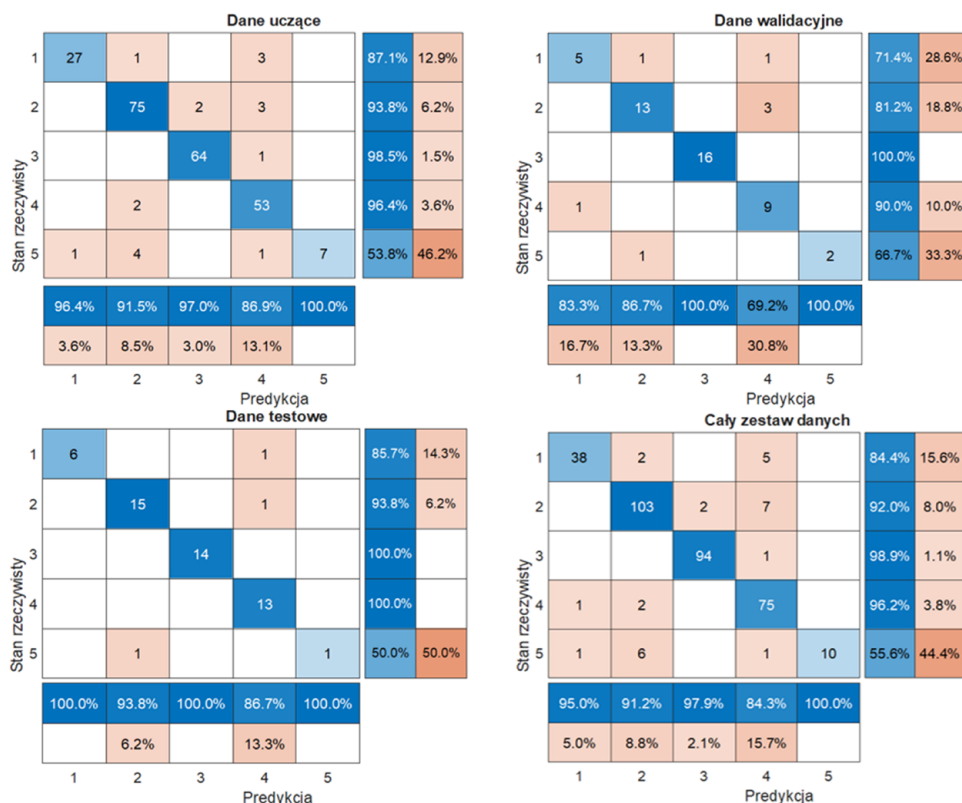
Na przedstawionych wykresach ROC (ryc. 6.17) dla modelu LSTM klasyfikacyjnego typ usterek na zmniejszonym zestawie danych widoczne są wysokie wartości AUC dla większości klas w różnych zestawach danych, co wskazuje na dobrą zdolność modelu do rozróżniania typów usterek. Klasa 5 zarówno w zbiorze walidacyjnym, jak i testowym, wykazuje najniższe wartości AUC, odpowiednio 0,68 i 0,56, co sugeruje, że model ma trudności z poprawnym klasyfikowaniem tej klasy usterek. Wysokie wartości AUC dla klas 1, 2, 3 i 4, które oscylują wokół 0,99 lub 1,00, świadczą o silnej zdolności modelu do precyzyjnej klasyfikacji tych typów usterek, co potwierdza jego skuteczność w większości przypadków.

Macierze pomyłek przedstawione na wykresach dla danych uczących, walidacyjnych, testowych oraz całego zestawu danych pozwalają na ocenę dokładności modelu w klasyfikacji pięciu różnych klas (ryc. 6.18). W przypadku danych uczących model osiąga wysoką dokładność dla większości klas, zwłaszcza dla klasy 3, gdzie poprawnie sklasyfikowano 64 z 65 przypadków (98,5% dokładność). Klasa 5 jednak wykazuje najniższą dokładność, gdzie poprawnie sklasyfikowano tylko 7 z 13 przypadków, co daje dokładność na poziomie 53,8%. Model dobrze rozpoznaje klasy 1, 2 i 4 z dokładnością odpowiednio 87,1%, 93,8% i 96,4%. W danych walidacyjnych model wykazuje solidną zdolność do klasyfikacji dla większości klas, zwłaszcza dla klasy 3, gdzie osiągnięto 100% poprawnych klasyfikacji. Jednak klasa 5 charakteryzuje się słabszą wydajnością, gdzie

tylko 2 z 3 przypadków zostały poprawnie sklasyfikowane, co daje dokładność na poziomie 66,7%. Dane testowe wskazują, że model zachowuje swoją wysoką dokładność dla klasy 3 i 4 (100% poprawnych klasyfikacji). Klasy 1 i 2 również osiągają wysoką dokładność, odpowiednio 85,7% i 93,8%. Klasa 5 charakteryzuje się niższą dokładnością na poziomie 50,0%, co wynika z faktu, że na dwa przypadki w tej klasie tylko jeden został poprawnie sklasyfikowany, a drugi przypisano błędnie. Taki rezultat jest istotny, zważywszy na to, że liczba rekordów w tej klasie była bardzo ograniczona, co znacząco wpływa na ostateczną ocenę dokładności modelu dla tej kategorii.



Ryc. 6.17. Krzywe ROC dla sieci LSTM dla modelu klasyfikacyjnego typ usterek na zmniejszonym zestawie danych



Ryc. 6.18. Macierz pomyłek dla sieci LSTM dla modelu klasyfikacyjnego typ usterek na zmniejszonym zestawie danych

Po wykonaniu obu modeli do klasyfikowania typ usterek na zmniejszonym zestawie danych (MLP oraz LSTM) dokonano ich porównania na podstawie danych testowych. Oba modele zostały ocenione na identycznym zestawie danych, co pozwala na bezpośrednie porównanie ich wydajności. Wskaźniki jakościowe dla danych testowych dla obu modeli przedstawia tabela 6.5.

Tabela 6.5. Porównanie modeli MLP oraz LSTM modelu wystąpienia awarii (wskaźniki jakości)

Model	Dokładność [%]	Precyzja [%]	Czułość [%]	F1-score [%]
MLP	90,6	91,6	89,8	90,6
LSTM	93,8	94,8	93,8	93,8

Kiedy analizuje się dane testowe, widać, że sieć LSTM wykazuje lepsze wyniki we wszystkich wskaźnikach jakości, w tym dokładności, precyzji, czułości oraz F1-score, w porównaniu z siecią neuronową MLP. Model LSTM osiągnął dokładność na poziomie 93,8%, co jest wyraźnie wyższe w porównaniu

z 90,6% osiągniętymi przez MLP. Ponadto sieć LSTM charakteryzuje się lepszą precyzją i czułością, co oznacza, że ma wyższą zdolność do prawidłowego rozpoznawania zarówno pozytywnych, jak i negatywnych przypadków, jednocześnie minimalizując liczbę fałszywie pozytywnych i fałszywie negatywnych klasyfikacji. Macierze pomyłek wskazują, że model LSTM skuteczniej rozpoznaje klasy 1, 2 i 3, osiągając lepsze wskaźniki prawdziwie pozytywnych wyników w tych klasach. Choć w przypadku klasy 5 oba modele wykazują trudności, LSTM wciąż osiąga lepsze wyniki niż sieć neuronowa MLP, co ma znaczenie w zastosowaniach praktycznych.

Na podstawie powyższej analizy model LSTM wydaje się bardziej odpowiedni do zastosowania w warunkach rzeczywistych. Jego wyższa dokładność, lepsza precyzja oraz czułość sugerują, że model ten będzie bardziej niezawodny w praktycznym wykorzystaniu, szczególnie w zadaniach wymagających wysokiej jakości klasyfikacji.

Zmiana koncepcji modelowania przyniosła oczekiwane rezultaty. Skupienie się na przypadkach zawierających usterki pozwala na bardziej efektywne wykorzystanie dostępnych danych, eliminując przypadki bez usterek i umożliwiając lepsze zrozumienie oraz modelowanie charakterystyk rzeczywistych awarii. Dzięki temu podejściu modele mogą poprawić swoją dokładność i precyzję w identyfikacji konkretnych typów awarii, co było problemem w przypadku pełnych zestawów danych. Mniejszy, bardziej ukierunkowany zbiór danych, niweluje również ryzyko przeuczenia modeli, co pozwala im lepiej generalizować na nowe dane. Skoncentrowanie się na rzeczywistych przypadkach awarii prowadzi do lepszego rozróżniania i klasyfikacji różnych typów usterek, co jest istotne dla poprawy wydajności systemu detekcji awarii. Zmiana koncepcji modelowania była zatem uzasadniona i niezbędna, aby osiągnąć lepsze wyniki i poprawić skuteczność wykrywania awarii.

6.5. Predykcja awarii maszyn z wykorzystaniem odpowiedniego interfejsu

Na bazie wytrenowanych w poprzednich podrozdziałach sieci neuronowych opracowano prosty interfejs użytkownika (ryc. 6.19). Przedstawiony interfejs został zaprojektowany w środowisku MATLAB i umożliwia użytkownikowi wprowadzanie istotnych parametrów operacyjnych maszyny oraz otrzymywanie informacji zwrotnej na temat potencjalnych awarii.

Interfejs graficzny, zbudowany za pomocą funkcji *uicontrol*, charakteryzuje się prostotą obsługi oraz czytelnym układem elementów. Na wstępie użytkownikowi prezentowany jest nagłówek „Przewidywanie awarii maszyn”, który znajduje się na górze okna interfejsu, co zapewnia przejrzystość i natychmiastową orientację w funkcjonalności programu.

Ryc. 6.19. Interfejs użytkownika aplikacji do predykcji awarii maszyn

Pod nagłówkiem znajduje się zestaw pól tekstowych, w których użytkownik może wprowadzić wartości pięciu parametrów operacyjnych: temperatury powietrza [K], temperatury procesu [K], prędkości obrotowej [obr/min], momentu obrotowego [Nm] oraz zużycia narzędzi [min]. Każde pole jest odpowiednio oznaczone etykietą, co ułatwia wprowadzanie danych i minimalizuje ryzyko popełnienia błędów.

Po wprowadzeniu wymaganych danych użytkownik ma możliwość uruchomienia procesu predykcji poprzez naciśnięcie przycisku „Predykcja”. W odpowiedzi na to działanie w tle uruchamiana jest funkcja *callback*, która odczytuje wprowadzone wartości, przekształca je w odpowiedni wektor cech, a następnie przekazuje do wcześniej wytrenowanych sieci neuronowych, które są załadowane z plików *net1.mat* oraz *net2.mat*.

Pierwsza sieć neuronowa (*net1*), sieć neuronowa MLP, jest odpowiedzialna za klasyfikację binarną, decydującą o wystąpieniu awarii. W przypadku wykrycia potencjalnej awarii druga sieć neuronowa (*net2*) przeprowadza bardziej szczegółową klasyfikację, identyfikując typ awarii, jest to sieć LSTM.

Wynik predykcji w zależności od jej rezultatu jest wyświetlany w dolnej części interfejsu w postaci tekstowej informacji. Jeśli nie zostanie wykryta awaria, użytkownik otrzymuje komunikat „Brak awarii”. W przeciwnym przypadku wyświetlana jest informacja o wykrytej awarii wraz ze wskazaniem jej typu, na przykład „awaria narzędzia”, „awaria chłodzenia”, „awaria mocy”, „awaria

przeciążenia” lub „awarie losowe”. Przedstawiony poniżej kod ilustruje sposób implementacji opisanego interfejsu graficznego (algorytm 6.3).

Algorytm 6.3. Implementacja interfejsu graficznego z wykorzystaniem wcześniej wytrenowanych sieci

```
function Przewidywanie_awarii_maszyn2()
% Tworzenie interfejsu graficznego
fig = figure('Position', [100, 100, 600, 400], 'Name',
'Przewidywanie_awarii_maszyn', 'Color', [0.8, 0.9, 1]);
% Nagłówek
uicontrol('Style', 'text', 'Position', [100, 350, 400, 40],
'String', 'Przewidywanie awarii maszyn', 'FontSize', 16, 'FontWeight',
'bold', 'BackgroundColor', [0.8, 0.9, 1], 'HorizontalAlignment',
'center');
% Ramka wokół pól wprowadzania danych
uicontrol('Style', 'frame', 'Position', [25, 130, 550, 220],
'BackgroundColor', [0.9, 0.9, 0.9]);
% Pola wprowadzania danych
uicontrol('Style', 'text', 'Position', [50, 320, 200, 20], 'String',
'Temperatura powietrza [K]', 'BackgroundColor', [0.9, 0.9, 0.9],
'FontSize', 12, 'HorizontalAlignment', 'left');
airTempInput = uicontrol('Style', 'edit', 'Position', [300, 320,
200, 20], 'FontSize', 12);
uicontrol('Style', 'text', 'Position', [50, 280, 200, 20], 'String',
'Temperatura procesu [K]', 'BackgroundColor', [0.9, 0.9, 0.9],
'FontSize', 12, 'HorizontalAlignment', 'left');
procTempInput = uicontrol('Style', 'edit', 'Position', [300, 280,
200, 20], 'FontSize', 12);
uicontrol('Style', 'text', 'Position', [50, 240, 200, 20], 'String',
'Prędkość obrotowa [obr/min]', 'BackgroundColor', [0.9, 0.9, 0.9],
'FontSize', 12, 'HorizontalAlignment', 'left');
rotSpeedInput = uicontrol('Style', 'edit', 'Position', [300, 240,
200, 20], 'FontSize', 12);
uicontrol('Style', 'text', 'Position', [50, 200, 200, 20], 'String',
'Moment obrotowy [Nm]', 'BackgroundColor', [0.9, 0.9, 0.9], 'FontSize',
12, 'HorizontalAlignment', 'left');
torqueInput = uicontrol('Style', 'edit', 'Position', [300, 200, 200,
20], 'FontSize', 12);
uicontrol('Style', 'text', 'Position', [50, 160, 200, 20], 'String',
'Zużycie narzędzi [min]', 'BackgroundColor', [0.9, 0.9, 0.9],
'FontSize', 12, 'HorizontalAlignment', 'left');
toolWearInput = uicontrol('Style', 'edit', 'Position', [300, 160,
200, 20], 'FontSize', 12);
```

Algorytm 6.3. Implementacja interfejsu graficznego z wykorzystaniem wcześniej wytrenowanych sieci, cd.

```
% Przycisk do predykcji
predictButton = uicontrol('Style', 'pushbutton', 'Position', [250,
110, 100, 40], 'String', 'Predykcja', 'BackgroundColor', [0.2, 0.6,
0.8], 'ForegroundColor', 'white', 'FontWeight', 'bold', 'FontSize', 12,
'Callback', @predictCallback);
% Pole do wyświetlania wyników
resultText = uicontrol('Style', 'text', 'Position', [50, 50, 500,
50], 'String', '', 'FontSize', 14, 'BackgroundColor', [0.8, 0.9, 1],
'HorizontalAlignment', 'center');
% Callback do przycisku predykcji
function predictCallback(~, ~)
    % Pobieranie danych wejściowych
    airTemp = str2double(get(airTempInput, 'String'));
    procTemp = str2double(get(procTempInput, 'String'));
    rotSpeed = str2double(get(rotSpeedInput, 'String'));
    torque = str2double(get(torqueInput, 'String'));
    toolWear = str2double(get(toolWearInput, 'String'));
    % Sprawdzenie poprawności danych wejściowych
    if isnan(airTemp) || isnan(procTemp) || isnan(rotSpeed) ||
isnan(torque) || isnan(toolWear)
        set(resultText, 'String', 'Błąd: Proszę wprowadzić poprawne
dane wejściowe.');
```

```
        return;
    end
    % Tworzenie wektora cech
    features = single([airTemp, procTemp, rotSpeed, torque, toolWear]);
    % Przekształcenie do odpowiedniego formatu
    featuresMatrix = features'; % Transpozycja do formatu [5x1]
    % Konwersja na komórkę (zgodnie z wymaganiami modelu LSTM)
    featuresCell = {reshape(featuresMatrix, [5, 1])};
    % Ładowanie wytrenowanych sieci
    load('net1.mat', 'net1'); % Sieć do wykrywania awarii
    load('net2.mat', 'net2'); % Sieć do klasyfikacji typu awarii
    % Predykcja awarii
    faultPrediction = net1(featuresMatrix);
    % Klasyfikacja na podstawie progu 0,5
    if faultPrediction < 0.5
        result = 'Brak awarii';
    else
        % Definiowanie klas (zgodne z klasami używanymi w sieci)
        classNames = [1, 2, 3, 4, 5];
```


Algorytm 6.3. Implementacja interfejsu graficznego z wykorzystaniem wcześniej wytrenowanych sieci, cd.

```
% Predykcja rodzaju awarii przy użyciu LSTM
YPred = minibatchpredict(net2, featuresCell);
faultType = onehotdecode(YPred, classNames, 2); % Dekodowanie
wyniku
% Konwersja categorical na liczby
faultType = double(faultType);
% Klasyfikacja typu awarii
switch faultType
    case 1
        faultDescription = 'awaria narzędzia';
    case 2
        faultDescription = 'awaria chłodzenia';
    case 3
        faultDescription = 'awaria mocy';
    case 4
        faultDescription = 'awaria przeciążenia';
    case 5
        faultDescription = 'awarie losowe';
    otherwise
        faultDescription = 'nieznany typ awarii';
end
result = sprintf('Awaria wykryta! Typ: %s',
faultDescription);
end
% Wyświetlanie wyniku
set(resultText, 'String', result);
end
end
```

Poniżej przedstawiono kilka przykładów działania interfejsu wraz z wynikami w postaci zrzutów ekranu. Zestawienie analizowanych przykładów wraz z zadanymi parametrami wejściowymi przedstawia tabela 6.6, zaś zrzuty ekranów dla poszczególnych przypadków odpowiednio #1 – ryc. 6.20, #2 – ryc. 6.21, #3 – ryc. 6.22, #4 – ryc. 6.23, #5 – ryc. 6.24, #6 – ryc. 6.25 oraz #7 – ryc. 6.26. Po weryfikacji otrzymanych wyników z danymi źródłowymi można stwierdzić, że stworzona aplikacja działa prawidłowo.

Tabela 6.6. Wyniki wybranych przykładów działania aplikacji dotyczącej predykcji awarii maszyn

Przykład	#1	#2	#3	#4	#5	#6	#7
Temperatura powietrza [K]	298,9	298,4	298,9	298,6	297	303,6	297
Temperatura procesu [K]	308,9	308,2	309,1	309,8	308,3	312	308,3
Prędkość obrotowa [obr/min]	1750	1282	2861	2271	1399	1309	1399
Moment obrotowy [Nm]	29,9	60,7	4,6	16,2	46,4	52,9	46,4
Zużycie narzędzi [min]	50	216	143	218	132	86	30
Czy występuje (0 – brak awarii, 1 – awaria), jeśli tak, to jaki typ	0	1	1	1	0	1	0
Typ awarii (1 – awaria narzędzia, 2 – awaria chłodzenia, 3 – awaria mocy, 4 – awaria przeciążenia, 5 – awarie losowe)	-	4	3	1	-	2	-

Przewidywanie awarii maszyn

Temperatura powietrza [K]

Temperatura procesu [K]

Prędkość obrotowa

Moment obrotowy [Nm]

Zużycie narzędzi [min]

Predykcja

Brak awarii

Ryc. 6.20. Przykład #1 działania aplikacji dotyczącej predykcji awarii maszyn

Przewidywanie awarii maszyn

Temperatura powietrza [K]

Temperatura procesu [K]

Prędkość obrotowa

Moment obrotowy [Nm]

Zużycie narzędzi [min]

Predykcja

Awaria wykryta! Typ: awaria przeciążenia

Ryc. 6.21. Przykład #2 działania aplikacji dotyczącej predykcji awarii maszyn

Przewidywanie awarii maszyn

Temperatura powietrza [K]	298.9
Temperatura procesu [K]	309.1
Prędkość obrotowa	2861
Moment obrotowy [Nm]	4.6
Zużycie narzędzi [min]	143

Predykcja

Awaria wykryta! Typ: awaria mocy

Ryc. 6.22. Przykład #3 działania aplikacji dotyczącej predykcji awarii maszyn

Przewidywanie awarii maszyn

Temperatura powietrza [K]	298.6
Temperatura procesu [K]	309.8
Prędkość obrotowa	2271
Moment obrotowy [Nm]	16.2
Zużycie narzędzi [min]	218

Predykcja

Awaria wykryta! Typ: awaria narzędzia

Ryc. 6.23. Przykład #4 działania aplikacji dotyczącej predykcji awarii maszyn

Przewidywanie awarii maszyn

Temperatura powietrza [K]	297
Temperatura procesu [K]	308.3
Prędkość obrotowa	1399
Moment obrotowy [Nm]	46.4
Zużycie narzędzi [min]	132

Predykcja

Brak awarii

Ryc. 6.24. Przykład #5 działania aplikacji dotyczącej predykcji awarii maszyn

Przewidywanie awarii maszyn

Temperatura powietrza [K]	303.6
Temperatura procesu [K]	312
Prędkość obrotowa	1309
Moment obrotowy [Nm]	52.9
Zużycie narzędzi [min]	86

Predykcja

Awaria wykryta! Typ: awaria chłodzenia

Ryc. 6.25. Przykład #6 działania aplikacji dotyczącej predykcji awarii maszyn

Przewidywanie awarii maszyn

Temperatura powietrza [K]	297
Temperatura procesu [K]	308.3
Prędkość obrotowa	1399
Moment obrotowy [Nm]	46.4
Zużycie narzędzi [min]	30

Predykcja

Brak awarii

Ryc. 6.26. Przykład #7 działania aplikacji dotyczącej predykcji awarii maszyn

Przedstawiony interfejs graficzny nie tylko ułatwia proces predykcji awarii, ale także zwiększa dostępność zaawansowanych metod analizy danych dla szerokiego grona użytkowników, umożliwiając szybkie i intuicyjne przeprowadzenie analizy stanu maszyny. Integracja interfejsu z sieciami neuronowymi MLP i LSTM pozwala na efektywne wykorzystanie technik uczenia maszynowego w rzeczywistych aplikacjach przemysłowych, przyczyniając się do poprawy niezawodności i efektywności operacyjnej maszyn.

W niniejszym rozdziale przeprowadzono szczegółową analizę modelowania predykcji awarii maszyn z wykorzystaniem różnych typów sieci neuronowych. Wykorzystano zarówno sieci neuronowe MLP, jak i bardziej zaawansowane sieci LSTM, aby przewidzieć zarówno wystąpienie awarii, jak i określić jej typ. Wyniki wskazują, że modele te różnią się skutecznością w zależności od specyfiki problemu oraz rodzaju danych. Dodatkowo podjęto próbę

optymalizacji modeli poprzez ograniczenie zbioru danych do przypadków zawierających usterki, co przyniosło pozytywne rezultaty w kontekście poprawy dokładności klasyfikacji. Ostatecznie opracowano także interfejs użytkownika umożliwiający praktyczne zastosowanie wytrenowanych modeli w rzeczywistych warunkach przemysłowych, co podkreśla praktyczny wymiar badań i ich potencjalne zastosowanie w predykcyjnym utrzymaniu ruchu.

7. Metodyka doboru architektury i parametrów procesu uczenia sieci neuronowych w doskonaleniu procesów produkcyjnych

Dobór architektury oraz parametrów procesu uczenia sieci neuronowych jest kluczowym elementem w kontekście skutecznej aplikacji tych modeli w procesach produkcyjnych. W niniejszym rozdziale zebrano i usystematyzowano wnioski z przeprowadzonych analiz dotyczących różnych zastosowań sieci neuronowych w takich obszarach, jak optymalizacja parametrów obróbki, ocena jakości wyrobów oraz predykcja awarii maszyn.

Tabela 7.1. Rodzaje architektur oraz parametrów uczenia sieci rozważane w analizowanych przykładach dla problemów klasyfikacyjnych

Przykład	Wykrywanie wad w odlewanych wyrobach gotowych	Wykrywanie błędów kształtu na linii produkcyjnej	Predykcja wystąpienia awarii
Wielkość zbioru danych	7348	1400	10000
Rodzaj sieci	CNN, transfer learning	CNN, transfer learning	MLP, LSTM
Liczba neuronów	32–2048	96–384	MLP 2–50, LSTM 256–4096
Liczba warstw ukrytych	48–570	5	MLP–1, LSTM 1–2
Funkcje aktywacji	softmax	softmax	softmax
Funkcja kosztu	klasyfikator SVM	entropia krzyżowa	entropia krzyżowa

Tabela 7.1. Rodzaje architektur oraz parametrów uczenia sieci rozważane w analizowanych przykładach dla problemów klasyfikacyjnych, cd.

Metody optymalizacji	BFGS ¹	sgdm	MLP – SCG, LSTM – Adam
Techniki regularyzacji	regularyzacja L2	warstwa dropout (0,5)	MLP – mechanizm wczesnego zatrzymania – 6, LSTM – warstwa dropout (0,5)

W przypadku problemów klasyfikacyjnych dobór architektury sieci neuronowych zależał od charakteru danych oraz złożoności zadania (tab. 7.1). W zadaniach takich jak wykrywanie wad w odlewanych wyrobach gotowych, gdzie analizowano duże zbiory danych, zastosowano CNN z wykorzystaniem techniki transfer learning. Sieci te, charakteryzujące się dużą liczbą neuronów (od 32 do 2048) i głębokimi architekturami (do 570 warstw ukrytych), okazały się efektywne w uchwyceniu skomplikowanych wzorców w obrazach. Wykorzystano funkcję aktywacji *softmax* oraz klasyfikator SVM, co pozwoliło na uzyskanie wysokiej precyzji w klasyfikacji. Podobne podejście zastosowano w wykrywaniu błędów kształtu na linii produkcyjnej, gdzie mniejsza liczba neuronów (96–384) i warstw (5 warstw ukrytych) była odpowiednia dla bardziej specyficznego zadania. Optymalizacja za pomocą stochastycznego gradientu z momentem (sgdm) oraz technika dropout były istotne dla zapewnienia stabilności modelu i zapobiegania przetrenowaniu.

Predykcja awarii maszyn, zrealizowana na podstawie dużego zbioru danych (10 000 próbek), wymagała zastosowania różnych podejść w zależności od charakteru danych. Do klasyfikacji binarnej wykorzystano sieć MLP z funkcją *softmax* i optymalizacją SCG, co zapewniło skuteczne przewidywanie wystąpienia awarii.

Z kolei do przewidywania rodzaju awarii wykorzystano sieć LSTM, przystosowaną do przetwarzania danych sekwencyjnych. Dzięki dużej liczbie neuronów (jednostek ukrytych) (256–4096) i optymalizacji metodą Adam model uzyskał wysoką dokładność w klasyfikacji. Najlepsze pod względem jakościowym modele dla każdego z analizowanych przykładów zostały zaprezentowane w tabeli 7.2.

¹ Funkcją optymalizacyjną jest metoda BFGS (Broyden-Fletcher-Goldfarb-Shanno), jednak sam klasyfikator SVM wykorzystuje wewnętrzną funkcję optymalizacyjną Sequential Minimal Optimization (SMO).

Tabela 7.2. Rodzaje architektur oraz parametrów uczenia sieci z najlepszymi parametrami jakościowymi dla problemów klasyfikacyjnych

Przykład	Rodzaj sieci	Liczba neuronów	Czułość [%]	Dokładność [%]
Wykrywanie wad w odlewanych wyrobach gotowych	CNN, transfer learning sieć Densenet201	64–512	99,27	99,61
Wykrywanie błędów kształtu na linii produkcyjnej	CNN, transfer learning sieć AlexNet	96–384	100	99,86
Predykcja wystąpienia awarii – czy wystąpi	MLP	12	75	98,1
Predykcja wystąpienia awarii – rodzaj awarii	LSTM	1024	93,8	93,8

W przypadku wykrywania błędów kształtu na linii produkcyjnej zastosowano sieć CNN z transfer learning opartą na architekturze AlexNet, która z liczbą neuronów od 96 do 384 oraz pięcioma warstwami ukrytymi jest optymalizowana metodą stochastycznego gradientu z momentem (sgdm). Sieć osiągnęła maksymalną czułość wynoszącą 100% i bardzo wysoką dokładność 99,86%. Podobnie wykrywanie wad w odlewanych wyrobach gotowych również przyniosło bardzo dobre rezultaty, gdzie zastosowano sieć CNN z transferem uczenia, opartą na architekturze DenseNet201. Sieć ta, wyposażona w od 64 do 512 neuronów i 199 warstw ukrytych, osiągnęła czułość na poziomie 99,27% oraz dokładność wynoszącą 99,61%. Dla zadania predykcji wystąpienia awarii najlepsze wyniki uzyskano dla dwóch różnych przypadków. Przy przewidywaniu, czy awaria wystąpi, zastosowanie sieci MLP z 12 neuronami w jednej warstwie ukrytej oraz optymalizacją metodą SCG pozwoliło na uzyskanie współczynnika korelacji na poziomie 75% i dokładności wynoszącej 98,1%. Natomiast dla przewidywania rodzaju awarii najlepsze wyniki osiągnięto przy zastosowaniu sieci LSTM z 1024 jednostkami ukrytymi, jednej warstwie LSTM oraz optymalizacją metodą Adam, co przyniosło dokładność na poziomie 93,8%.

Warto zwrócić uwagę na jeszcze jedną istotną kwestię dotyczącą modelowania klasyfikatora przewidującego rodzaj awarii. Podczas modelowania konieczna była zmiana koncepcji prowadzonych badań ze względu na niską jakość wyników osiąganych przez modele klasyfikacyjne, zarówno MLP, jak

i LSTM. Modele te miały trudności z prawidłowym rozpoznawaniem niektórych klas awarii. W odpowiedzi na te problemy postanowiono ograniczyć zbiór danych jedynie do rekordów zawierających usterki i przeprowadzić ponowne modelowanie na takim zredukowanym zestawie, który obejmował 348 rekordów. Zmiana koncepcji modelowania była konieczna, aby poprawić skuteczność klasyfikatora w rozpoznawaniu awarii oraz dostosować model do bardziej specyficznych i szczegółowych danych. Dzięki takiej modyfikacji możliwe stało się lepsze zrozumienie specyfiki każdej z klas awarii, co wpłynęło na podniesienie ogólnej dokładności i skuteczności nowego modelu klasyfikacyjnego.

Tabela 7.3. Rodzaje architektur oraz parametrów uczenia sieci rozważane w analizowanych przykładach dla problemów regresyjnych

Przykład	Modelowanie wybranych parametrów chropowatości 2D	Modelowanie siły skrawania	Określenie twardości po odpuszczaniu wyrobów gotowych
Wielkość zbioru danych	46	46	247
Liczba neuronów	2–12	2–12	4–20

W problemach regresyjnych, takich jak modelowanie parametrów chropowatości 2D czy siły skrawania, stosowano MLP (tab. 7.3). Przy ograniczonym rozmiarze zbiorów danych (46–247 próbek) sieci te charakteryzowały się jedną warstwą ukrytą i liczbą neuronów w przedziale od 2 do 20. Sigmoidalna funkcja aktywacji oraz optymalizacja za pomocą algorytmu LM, BR i SCG były standardem, co pozwalało na skuteczne modelowanie złożonych zależności przy minimalnym ryzyku przetrenowania. Mechanizm wczesnego zatrzymania dodatkowo zwiększał stabilność modeli, co było szczególnie istotne w kontekście małych zbiorów danych, jak np. w modelowaniu twardości po odpuszczaniu wyrobów gotowych.

Kontynuując analizę problemów regresyjnych, warto zauważyć, że najlepsze parametry modelowania uzyskano przy precyzyjnie dobranej liczbie neuronów oraz odpowiednio dobranej metodzie optymalizacji (tab. 7.4). W modelowaniu parametrów chropowatości najwyższy współczynnik korelacji uzyskano dla parametru Rz (0,9894) przy 9 neuronach i optymalizacji metodą LM. Dla parametru Ra, przy 6 neuronach, współczynnik wyniósł 0,9869. Z kolei dla RSm, mimo zwiększenia liczby neuronów do 10, współczynnik korelacji spadł do 0,911, co sugeruje, że większa liczba neuronów nie zawsze poprawia jakość modelu. W modelowaniu siły skrawania najlepszy wynik osiągnięto przy zastosowaniu 7 neuronów w warstwie ukrytej oraz optymalizacji metodą również LM,

co przyniosło wyjątkowo wysoki współczynnik korelacji na poziomie 0,999. Jest to wynik, który wskazuje na doskonałe dopasowanie modelu do danych. Natomiast w przypadku określenia twardości po odpuszczaniu wyrobów gotowych, przy zastosowaniu 9 neuronów oraz optymalizacji metodą BR, uzyskano współczynnik korelacji wynoszący 0,9853. Ten wynik pokazuje, że metoda BR jest równie efektywna w odniesieniu do mniejszych zbiorów danych, zwłaszcza w zadaniach, które wymagają większej stabilności i odporności na przetrenowanie. Te wyniki jasno pokazują, że istotnym czynnikiem sukcesu w problemach regresyjnych jest nie tylko odpowiedni dobór architektury sieci neuronowej, ale także precyzyjna konfiguracja liczby neuronów i zastosowanie najlepszej metody optymalizacji, dostosowanej do specyfiki danego zadania.

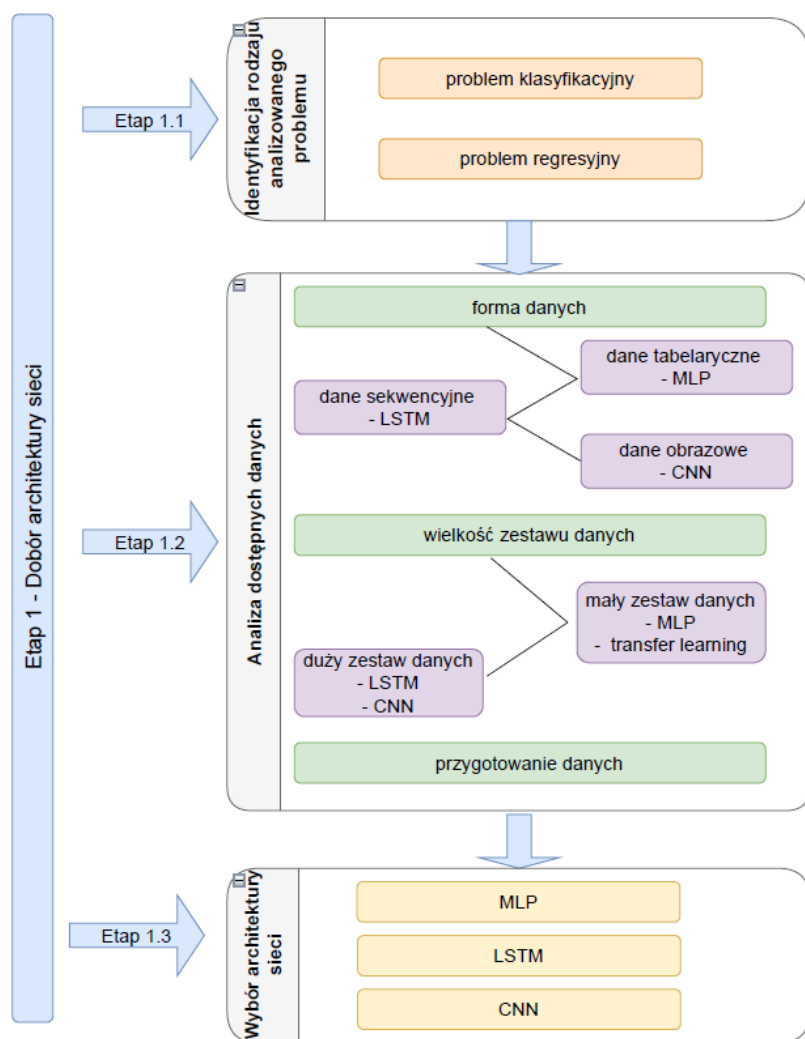
Tabela 7.4. Parametry uczenia sieci z najlepszymi parametrami jakościowymi dla problemów regresyjnych

Przykład	Liczba neuronów	Metoda optymalizacji	Współczynnik korelacji R dla całego zbioru
Modelowanie wybranych parametrów chropowatości – Ra	6	LM	0,9869
Modelowanie wybranych parametrów chropowatości – Rz	9	LM	0,9894
Modelowanie wybranych parametrów chropowatości – RSm	10	LM	0,911
Modelowanie siły skrawania	7	LM	0,999
Określenie twardości po odpuszczaniu wyrobów gotowych	9	BR	0,9853

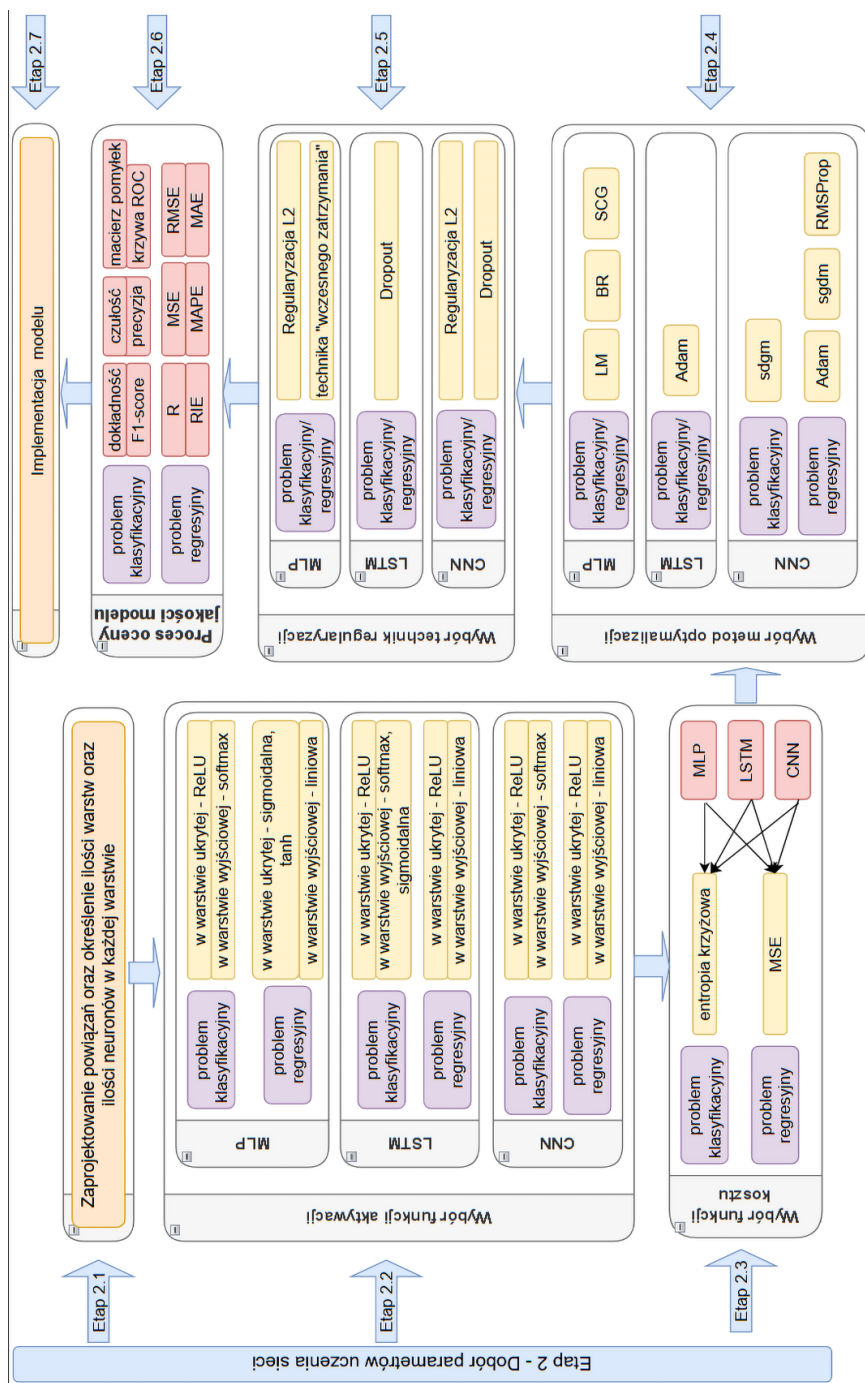
Podsumowując, dobór architektury oraz parametrów sieci neuronowych w analizowanych zadaniach został precyzyjnie dopasowany do specyfiki problemu i dostępnych danych, co pozwoliło na uzyskanie wysokiej skuteczności modeli w aplikacjach produkcyjnych. Przeanalizowane przypadki ilustrują, jak odpowiednie dostosowanie architektury i parametrów uczenia sieci neuronowych do rodzaju problemu, danych i ich wielkości wpływa na efektywność modelu.

Przykłady te zostały przedstawione, aby ukazać, jak można optymalnie dobierać architekturę i parametry uczenia w zależności od konkretnych wymagań zadania. Jednocześnie należy podkreślić, że te modele mogą być z powodzeniem zastosowane w tych konkretnych przypadkach, dla których zostały opracowane. Jednak w wyniku prowadzonych badań zidentyfikowano potrzebę opracowania bardziej ogólnej metodyki doboru architektury i parametrów procesu uczenia sieci neuronowych, która mogłaby znaleźć zastosowanie

w doskonaleniu różnych procesów produkcyjnych. Opracowana metodyka została podzielona na dwa etapy: Etap 1 – Dobór architektury sieci (ryc. 7.1) oraz Etap 2 – Dobór parametrów uczenia sieci (ryc. 7.2).



Ryc. 7.1. Metodyka doboru architektury i parametrów procesu uczenia sieci neuronowych – Etap I



Ryc. 7.2. Metody doboru architektury i parametrów procesu uczenia sieci neuronowych – Etap II

Zaproponowaną metodykę można przedstawić w postaci następujących po sobie kroków.

Etap 1 – Dobór architektury sieci (ryc. 7.1)

Etap 1.1. Identyfikacja rodzaju analizowanego problemu

Etap 1.2. Analiza dostępnych danych

Etap 1.3. Wybór architektury sieci

Etap 2 – Dobór parametrów procesu uczenia sieci (ryc. 7.2)

Etap 2.1. Zaprojektowanie powiązań oraz określenie liczby warstw oraz liczby neuronów w każdej warstwie

Etap 2.2. Wybór funkcji aktywacji

Etap 2.3. Wybór funkcji kosztu

Etap 2.4. Wybór metod optymalizacji

Etap 2.5. Wybór technik regularyzacji

Etap 2.6. Proces oceny jakości modelu

Etap 2.7. Implementacja modelu

Etap 1.1. Identyfikacja rodzaju analizowanego problemu

Zrozumienie specyfiki problemu stanowi ważny etap w procesie doboru architektury sieci neuronowych oraz parametrów ich uczenia. Pierwszym krokiem jest precyzyjne określenie celu badań, czyli identyfikacja parametrów procesu produkcyjnego, które wymagają poprawy. Może to dotyczyć jakości produktu, efektywności produkcji lub predykcji awarii maszyn. Kolejnym krokiem jest ustalenie rodzaju problemu, czy jest to klasyfikacja czy regresja.

Klasyfikacja to jeden z najczęstszych problemów w uczeniu maszynowym, gdzie celem jest przypisanie etykiety (klasy) do danych wejściowych. Sieci neuronowe uczą się rozpoznawać wzorce w danych, co pozwala na klasyfikację każdego przykładu w zbiorze danych. Przykłady zastosowań obejmują wykrywanie wad w produktach, gdzie klasyfikacja jest często używana do automatycznego wykrywania wad na linii produkcyjnej. Na przykład konwolucyjne sieci neuronowe (CNN) analizują obrazy produktów, aby zidentyfikować, czy produkt ma wady. Innym przykładem jest klasyfikacja błędów kształtu w procesach, takich jak formowanie czy odlewanie, gdzie sieci neuronowe analizują dane geometryczne lub obrazy 3D, aby sklasyfikować produkty według rodzaju kształtu. W predykcyjnej konserwacji klasyfikacja jest używana do określenia, czy maszyna działa prawidłowo, czy wymaga interwencji, na przykład poprzez wykrywanie stanów awaryjnych.

Regresja to kolejny rodzaj problemu, gdzie celem jest przewidywanie wartości ciągłej na podstawie danych wejściowych. W produkcji regresja jest używana do modelowania zależności między różnymi parametrami procesów a wynikowymi właściwościami produktów. Przykładem zastosowania regresji może być przewidywanie twardości materiału w przemyśle metalurgicznym, gdzie sieci neuronowe przewidują twardość materiału po procesie obróbki cieplnej na podstawie takich danych wejściowych, jak czas odpuszczania czy temperatura odpuszczania. W procesach obróbki skrawaniem, takich jak frezowanie czy toczenie, sieci neuronowe mogą przewidywać siły działające na narzędzia skrawające w zależności od parametrów procesu.

Warto podkreślić, że do każdego z wymienionych przypadków – klasyfikacji, regresji – można zastosować różne rodzaje sieci neuronowych, takie jak MLP, rekurencyjne sieci LSTM oraz CNN. Każda z tych architektur ma swoje unikalne zalety, które mogą być skutecznie wykorzystane w zależności od specyfiki problemu. Jednak istotne jest zrozumienie specyfiki problemu przed podjęciem decyzji o wyborze architektury sieci neuronowej. Dopiero dokładna analiza charakterystyki danych i zrozumienie, jakie aspekty są najważniejsze w danym problemie, pozwala na świadome dobranie odpowiedniej architektury i parametrów procesu uczenia. Na przykład w przypadku analizy obrazów CNN mogą być pierwszym wyborem ze względu na ich zdolność do automatycznego wykrywania istotnych cech w danych wizualnych. Natomiast w zadaniach związanych z przewidywaniem na podstawie danych sekwencyjnych LSTM będą bardziej adekwatne.

Również parametry procesu uczenia, takie jak liczba warstw, liczba neuronów, funkcje aktywacji oraz metody optymalizacji, powinny być dostosowane do specyficznych wymagań problemu, co pozwoli na uzyskanie najlepszego możliwego modelu. Dlatego, chociaż każda z tych architektur może potencjalnie być zastosowana do różnych typów problemów, ważne jest przeprowadzenie gruntownej analizy problemu, aby wybór architektury i parametrów był optymalny dla osiągnięcia zamierzonych celów w doskonaleniu procesów produkcyjnych.

Etap 1.2. Analiza dostępnych danych

Analiza dostępnych danych jest kolejnym krokiem w procesie projektowania sieci neuronowych i doboru odpowiednich parametrów uczenia. Zrozumienie, jakie dane są dostępne, odgrywa fundamentalną rolę w podejmowaniu decyzji dotyczących architektury modelu oraz metod optymalizacji. Dane mogą

przyjmować różne formy, a każda z nich niesie ze sobą specyficzne wyzwania i możliwości. Przykładowo obrazy są powszechnie stosowane w problemach związanych z rozpoznawaniem obrazów, takich jak wykrywanie wad powierzchniowych w odlewach. W takim przypadku CNN są często najlepszym wyborem ze względu na ich zdolność do automatycznego wykrywania istotnych cech w danych wizualnych. Z kolei dane sekwencyjne, sygnały czasowe, takie jak analiza drgań czy sygnałów akustycznych, są często wykorzystywane w monitorowaniu stanu maszyn i predykcji awarii. Rekurencyjne sieci LSTM są przystosowane do przetwarzania tego typu danych, ponieważ mogą uwzględniać zależności czasowe. Natomiast dane tabelaryczne, typowe dla tradycyjnych procesów produkcyjnych, obejmują takie informacje, jak parametry obróbki, wyniki pomiarów jakości oraz dane dotyczące zużycia surowców. W takich przypadkach MLP mogą być odpowiednim wyborem, ponieważ są wszechstronne i skuteczne w przetwarzaniu danych tabelarycznych.

Wielkość zbioru danych ma ogromny wpływ na wybór architektury sieci neuronowej oraz metodę uczenia. Im większy jest zestaw danych, tym bardziej złożone modele można zastosować. Na przykład duże zbiory danych umożliwiają trenowanie głębokich sieci neuronowych, takich jak CNN czy LSTM, które są zdolne do uchwycenia skomplikowanych zależności w danych. W przypadku CNN, które są szczególnie skuteczne w analizie obrazów, duża ilość danych pozwala na precyzyjne wykrywanie złożonych wzorców, co jest istotne w zadaniach związanych z rozpoznawaniem obrazów. Podobnie LSTM, które są zaprojektowane do przetwarzania sekwencji danych, mogą lepiej uchwycić długoterminowe zależności czasowe, gdy mają do dyspozycji obszerny zbiór danych. Jednak w przypadku mniejszych zbiorów danych istnieje ryzyko nadmiernego dopasowania modelu, czyli sytuacji, w której model zbyt dokładnie uczy się specyficznych cech zbioru uczącego, co prowadzi do słabej generalizacji na nowych danych. MLP, które są wszechstronnymi modelami stosowanymi w zadaniach klasyfikacji i regresji, mogą być bardziej podatne na nadmierne dopasowanie w takich sytuacjach, jeśli są zbyt złożone w stosunku do ilości dostępnych danych. Aby przeciwdziałać problemom związanym z małymi zbiorami danych, można zastosować techniki, takie jak transfer learning, który polega na wykorzystaniu wcześniej wytrenowanych modeli na dużych zbiorach danych do zadań podobnych do tych, które są obecnie analizowane. Transfer learning jest szczególnie przydatny w przypadku CNN, gdzie można wykorzystać modele wytrenowane na dużych zestawach obrazów do analizy mniejszych, specyficznych zbiorów.

Jakość danych jest równie istotnym aspektem w procesie modelowania. Należy ocenić, czy dane są kompletne, czy występują w nich braki oraz czy są obecne szумы, które mogą zakłócić proces uczenia. Dane o niskiej jakości mogą prowadzić do błędnych wniosków i nieskutecznych modeli, dlatego często konieczne jest przeprowadzenie procesów oczyszczania danych. Obejmuje to imputację brakujących wartości, czyli uzupełnianie braków w danych, oraz normalizację, która ma na celu przekształcenie danych do jednolitego formatu, co jest istotne dla prawidłowego działania modeli neuronowych.

Etap 1.3. Wybór architektury sieci

Wybór odpowiedniej architektury sieci neuronowej jest kluczowym etapem w procesie projektowania systemu opartego na sztucznej inteligencji, szczególnie w kontekście doskonalenia procesów produkcyjnych. Architektura sieci decyduje o zdolności modelu do przetwarzania danych, identyfikowania wzorców i dostarczania trafnych prognoz lub klasyfikacji. Wybór architektury powinien być oparty na dokładnej analizie specyfiki problemu oraz charakterystyki dostępnych danych, co zostało omówione w poprzednich etapach.

Etap 2 metodyki projektowania sieci neuronowych skupia się na doborze parametrów procesu uczenia (ryc. 7.2). Ten etap ma istotne znaczenie dla osiągnięcia optymalnych wyników w zależności od specyfiki problemu, jakim może być klasyfikacja lub regresja. W zależności od tego, czy zadaniem jest klasyfikacja, czy regresja, różne parametry mogą być bardziej lub mniej odpowiednie. Poniżej przedstawiono szczegółowy opis tego etapu, uwzględniając różne podejścia w zależności od rodzaju problemu.

Etap 2.1. Zaprojektowanie powiązań oraz określenie liczby warstw oraz liczby neuronów w każdej warstwie

Liczba warstw, liczba neuronów oraz struktura połączeń między nimi odgrywają istotną rolę w projektowaniu sieci neuronowej. Każda warstwa i jej powiązania z innymi warstwami muszą być starannie zaprojektowane, aby sieć mogła skutecznie modelować specyfikę rozwiązywanego problemu, takiego jak klasyfikacja czy regresja.

W przypadku MLP liczba warstw i neuronów zależy od złożoności danych wejściowych. W klasyfikacji przy prostszych problemach, takich jak klasyfikacja na podstawie kilku cech, często wystarczająca może być jedna lub dwie warstwy ukryte z niewielką liczbą neuronów, na przykład 2–100 neuronów na

warstwę. Jednak w bardziej złożonych przypadkach, takich jak klasyfikacja złożonych zbiorów danych tabelarycznych, można zastosować więcej warstw, od 3 do 5, z większą liczbą neuronów, od 100 do 512. Eksperymentowanie z różnymi konfiguracjami jest istotne, aby znaleźć optymalną liczbę warstw i neuronów. W przypadku regresji liczba warstw i neuronów może być podobna do tej stosowanej w klasyfikacji, jednak często regresja wymaga bardziej złożonych modeli, zwłaszcza gdy dane wejściowe mają dużą liczbę cech. W takich przypadkach liczba warstw i neuronów może być większa, aby lepiej uchwycić skomplikowane nieliniowe zależności między cechami a wartościami wyjściowymi. Warto jednak pamiętać, że przy pracy z małymi zbiorami danych liczba neuronów w warstwie ukrytej nie powinna przekraczać liczby obserwacji w zbiorze danych. Jest to zalecane na podstawie wcześniejszych badań eksperymentalnych, które wykazały, że nadmierna liczba neuronów w stosunku do dostępnych danych może prowadzić do nadmiernego dopasowania modelu, co skutkuje gorszą generalizacją na nowych danych.

Rekurencyjne sieci neuronowe, takie jak LSTM, są często stosowane w klasyfikacji sekwencji, na przykład w analizie sygnałów. Typowo stosuje się 1–2 warstwy LSTM z liczbą jednostek ukrytych LSTM w zakresie od 50 do 2048, w zależności od długości sekwencji i złożoności danych. Zbyt duża liczba warstw i jednostek LSTM może prowadzić do nadmiernego dopasowania, szczególnie przy ograniczonym zbiorze danych. W zadaniach regresji na danych sekwencyjnych, takich jak prognozowanie wartości ciągłych w czasie, można zastosować więcej warstw, na przykład 2–3, oraz większą liczbę jednostek LSTM, od 256 do 4096, aby lepiej uchwycić długoterminowe zależności w danych. Warto jednak zachować ostrożność, aby uniknąć ryzyka nadmiernego dopasowania.

CNN znajdują zastosowanie przede wszystkim w klasyfikacji obrazów. Liczba warstw konwolucyjnych oraz liczba filtrów w każdej warstwie zależy od złożoności obrazów i wielkości zbioru danych. Typowe architektury CNN mogą zawierać od 3 do 10 warstw konwolucyjnych z liczbą filtrów rosnącą od 32 do 512 w kolejnych warstwach. W przypadku mniej złożonych problemów, takich jak klasyfikacja prostych wzorców, wystarczy kilka warstw z mniejszą liczbą filtrów. W zadaniach regresji na obrazach takich jak przewidywanie wartości ciągłych na podstawie obrazów, stosuje się podobną liczbę warstw i filtrów jak w klasyfikacji, ale częściej wykorzystuje się dodatkowe warstwy w pełni połączone na końcu sieci, aby przekształcić cechy wykryte przez warstwy konwolucyjne w prognozy ciągłe.

Etap 2.2. Wybór funkcji aktywacji

Wybór funkcji aktywacji jest ważnym aspektem w procesie projektowania sieci neuronowych, a jego dobór zależy od rodzaju problemu (klasyfikacja lub regresja).

W przypadku MLP w zadaniach klasyfikacyjnych najczęściej stosowaną funkcją aktywacji w warstwach ukrytych jest ReLU, która umożliwia efektywne uczenie się złożonych wzorców dzięki właściwościom nieliniowym i zdolności do unikania problemu zanikania gradientu. Na warstwie wyjściowej w problemach klasyfikacji stosuje się funkcję softmax, która przekształca wyniki sieci w prawdopodobieństwa przypisania do poszczególnych klas. W kontekście regresji w MLP często stosuje się funkcję sigmoidalną w warstwach ukrytych, szczególnie w domyślnych ustawieniach narzędzi, takich jak MATLAB, które preferują tę funkcję dla swoich sieci neuronowych. Funkcja sigmoidalna mapuje wejścia na zakres $[0,1]$, co może być korzystne w niektórych aplikacjach. Na warstwie wyjściowej dla regresji stosuje się funkcję liniową, co umożliwia przewidywanie wartości ciągłych. Dodatkowo funkcje, takie jak *tanh*, mogą być stosowane w warstwach ukrytych, jeśli dane cechują się bardziej złożonymi zależnościami nieliniowymi.

Dla LSTM stosowanych często w analizie sekwencji czasowych, w klasyfikacji również wykorzystuje się funkcję ReLU w warstwach ukrytych, co pozwala na efektywne przetwarzanie sekwencji o dużej złożoności. W warstwie wyjściowej dla zadań klasyfikacyjnych, w zależności od liczby klas, stosuje się funkcję softmax lub sigmoidalną (dla klasyfikacji binarnej). W regresji, w przypadku LSTM, na warstwie wyjściowej wykorzystuje się funkcję liniową, co jest standardowym podejściem, umożliwiającym prognozowanie wartości ciągłych w zadaniach takich jak prognozowanie czasowe.

W CNN, które są powszechnie stosowane w zadaniach związanych z przetwarzaniem obrazów, ReLU jest dominującą funkcją aktywacji stosowaną w warstwach konwolucyjnych. Jej nieliniowy charakter oraz prostota obliczeniowa sprawiają, że to idealny wybór do głębokich sieci, umożliwiający efektywne przetwarzanie złożonych wzorców obrazowych. W zadaniach klasyfikacyjnych warstwa wyjściowa CNN często korzysta z funkcji softmax, która przekształca wyjścia sieci w prawdopodobieństwa przypisania do klas. W regresji w CNN, podobnie jak w innych typach sieci, stosuje się funkcję liniową na warstwie wyjściowej, co pozwala na przewidywanie wartości ciągłych na podstawie cech wyodrębnionych przez warstwy konwolucyjne.

Etap 2.3. Wybór funkcji kosztu

Wybór odpowiedniej funkcji kosztu jest istotnym elementem w procesie trenowania sieci neuronowych, ponieważ bezpośrednio wpływa na to, jak dobrze model będzie w stanie się uczyć i generalizować. W przypadku problemów klasyfikacyjnych najczęściej stosowaną funkcją kosztu jest entropia krzyżowa. Sprawdza się ona w optymalizacji modeli klasyfikacyjnych, ponieważ mierzy różnicę między przewidywaną przez model a rzeczywistą klasą w sposób, który uwzględnia wszystkie klasy, co jest szczególnie istotne w zadaniach wieloklasowych.

Z kolei w regresji powszechnie stosowaną funkcją kosztu jest MSE. Oblicza średnią kwadratową różnicę między wartościami przewidywanymi przez model a wartościami rzeczywistymi, co pozwala na dokładną optymalizację modeli regresyjnych. MSE jest szczególnie przydatna w regresji, ponieważ penalizuje duże błędy w prognozach, co sprawia, że model uczy się, jak minimalizować odchylenia od rzeczywistych wartości, prowadząc do bardziej precyzyjnych przewidywań.

Etap 2.4. Wybór metod optymalizacji

Wybór metod optymalizacji jest również ważnym etapem w procesie trenowania sieci neuronowych, ponieważ wpływa bezpośrednio na szybkość i skuteczność uczenia się modelu. Różne rodzaje problemów, takie jak klasyfikacja i regresja, wymagają zastosowania odpowiednich metod optymalizacji, które najlepiej odpowiadają specyfice danych i architektury sieci.

W przypadku sieci MLP zarówno w problemach klasyfikacyjnych, jak i regresyjnych często stosuje się metody, takie jak LM, BR oraz SCG. Te algorytmy są dobrze dopasowane do zadań, które wymagają precyzyjnej optymalizacji przy relatywnie małych zbiorach danych. LM daje szczególnie dobre efekty dla małych i średnich sieci, gdzie zbieżność do optymalnych wag jest priorytetem. BR dodatkowo pomaga w regularyzacji modelu, co ma znaczenie przy pracy z ograniczonymi zbiorami danych, zmniejszając ryzyko nadmiernego dopasowania. SCG natomiast oferuje zbalansowane podejście między szybkością a stabilnością uczenia, co czyni go użytecznym w różnorodnych zastosowaniach.

W przypadku rekurencyjnych sieci neuronowych, takich jak LSTM, które są wykorzystywane zarówno do klasyfikacji, jak i regresji, optymalizatorem z wyboru często jest Adam. Łączy on zalety RMSProp i momentum, oferując szybkie zbieżności przy jednoczesnym dostosowaniu współczynników uczenia dla każdego parametru, co jest niezwykle istotne w zadaniach sekwencyjnych, gdzie dane są często niestacjonarne i złożone.

Dla sieci CNN i transfer learningu, szczególnie w klasyfikacji, powszechnie stosowaną metodą optymalizacji jest sgdm. Preferuje się ją ze względu na jej zdolność do radzenia sobie z dużymi zbiorami danych oraz efektywność w głębokich sieciach neuronowych. Momentum pomaga przyspieszyć zbieżność i uniknąć utknięcia w lokalnych minimach, co jest istotne w skomplikowanych architekturach, takich jak CNN. W zadaniach regresyjnych najczęściej preferowany ze względu na swoją zdolność do efektywnego radzenia sobie z różnorodnymi i złożonymi danymi jest Adam, choć SGDM i RMSProp mogą być również stosowane w zależności od specyfiki problemu i wymagań dotyczących optymalizacji.

Etap 2.5. Wybór technik regularyzacji

Wybór technik regularyzacji jest również istotnym elementem w projektowaniu sieci neuronowych, ponieważ pomaga w zapobieganiu nadmiernemu dopasowaniu modelu do danych treningowych, co ma szczególne znaczenie w przypadku złożonych modeli lub gdy dostępna jest ograniczona ilość danych.

W przypadku sieci MLP zarówno w zadaniach klasyfikacji, jak i regresji, popularnymi technikami regularyzacji są regularyzacja L2 oraz mechanizm wczesnego zatrzymania.

W sieciach LSTM, które są często stosowane do analizy sekwencji czasowych, regularyzacja jest szczególnie ważna, ponieważ modele te mogą być podatne na nadmierne dopasowanie, zwłaszcza gdy mają do czynienia z długimi sekwencjami danych. Dropout to popularna metoda, w której losowo wyłącza się pewien procent neuronów podczas treningu, co zmusza sieć do tworzenia bardziej odpornych reprezentacji danych.

W sieciach CNN, które są powszechnie stosowane do przetwarzania obrazów, regularyzacja ma na celu zapobieganie nadmiernemu dopasowaniu, szczególnie w przypadku głębokich sieci z dużą liczbą parametrów. Regularyzację L2 stosuje się często, zwłaszcza w warstwach pełni połączonych, gdzie liczba parametrów jest największa. Dropout jest również powszechnie stosowany, szczególnie w pełni połączonych warstwach, aby zmusić sieć do tworzenia bardziej odpornych i ogólnych reprezentacji.

Etap 2.6. Proces oceny jakości modelu

Proces oceny jakości modelu w sieciach neuronowych jest istotny dla oceny skuteczności i dokładności modelu oraz wyboru najlepszego rozwiązania do danego problemu. W kontekście regresji ocena jakości modeli opiera się na

wskaźnikach, takich jak R, MSE, RMSE, RIE, MAPE oraz MAE. Umożliwiają one wszechstronną analizę wyników predykcji, zapewniając, że model jest w stanie dokładnie przewidywać wartości ciągle.

W problemach klasyfikacyjnych wskaźniki oceny, takie jak dokładność, precyzja, czułość i F1-score, są ważne dla zrozumienia, jak skutecznie model przypisuje próbki do odpowiednich klas. Analiza tych wskaźników, wsparta narzędziami, takimi jak macierz pomyłek i krzywa ROC, umożliwia dokładną ocenę wydajności modelu, pomagając w identyfikacji jego mocnych i słabych stron oraz w doskonaleniu algorytmów klasyfikacyjnych.

Porównanie różnych modeli oraz architektur sieci neuronowych jest nieodłącznym elementem procesu walidacji. Testowanie różnych konfiguracji sieci, takich jak liczba warstw, neuronów, funkcji aktywacji i metod optymalizacji, umożliwia wybór najlepszego modelu. Wprowadzenie odpowiednich wskaźników oceny jakości prowadzi do bardziej precyzyjnego i świadomego wyboru modelu, co jest kluczowe dla osiągnięcia wysokiej skuteczności w praktycznych zastosowaniach.

Etap 2.7. Implementacja modelu

Implementacja modelu stanowi ważny krok w praktycznym wykorzystaniu sieci neuronowych, zwłaszcza w kontekście doskonalenia procesów produkcyjnych. Po zakończeniu procesu trenowania i walidacji modelu następnym zadaniem jest wdrożenie go w taki sposób, aby mógł być łatwo i efektywnie wykorzystany w środowisku produkcyjnym. Duże znaczenie ma tu przygotowanie odpowiedniego interfejsu użytkownika, który umożliwi pracownikom dostęp do funkcjonalności modelu bez potrzeby głębokiej znajomości technicznych aspektów jego działania.

Metodyka doboru architektury i parametrów procesu uczenia sieci neuronowych, opracowana w celu doskonalenia procesów produkcyjnych, uwzględniała wszystkie etapy prowadzące do stworzenia skutecznego modelu. Ostatecznym krokiem w tej metodyce jest zapewnienie, że wytrenowany model może być z powodzeniem zaimplementowany w rzeczywistym środowisku produkcyjnym. W tym kontekście przygotowanie intuicyjnego i funkcjonalnego interfejsu jest nieodzowne. Interfejs ten powinien być dostosowany do specyficznych wymagań danego przedsiębiorstwa i rodzaju analizowanego problemu, co umożliwia użytkownikom łatwe korzystanie z modelu.

W monografii zawarto przykłady gotowych kodów źródłowych, które można wykorzystać jako podstawę do tworzenia takich interfejsów. Aby dostosować te rozwiązania do konkretnego przypadku, wystarczy zmienić nazwę wytrenowanego

modelu lub zastosować nazwę zaproponowaną przez autora. Takie podejście pozwala na szybką adaptację narzędzi do specyficznych potrzeb danego procesu produkcyjnego, co jest zgodne z metodyką, której celem jest doskonalenie procesów produkcyjnych z zastosowaniem sztucznych sieci neuronowych.

Dzięki wdrożeniu tak przygotowanego modelu przedsiębiorstwo może skutecznie wykorzystać potencjał sieci neuronowych do poprawy jakości, zwiększenia efektywności oraz innowacyjności swoich procesów produkcyjnych, realizując założenia metodyki na każdym etapie, od doboru architektury po praktyczne zastosowanie modelu.

Powyżej przedstawiono metodykę doboru architektury i parametrów procesu uczenia sieci neuronowych, która może być skutecznie zastosowana w doskonaleniu procesów produkcyjnych. Dzięki odpowiedniemu dopasowaniu architektury sieci oraz parametrów uczenia do specyficznych problemów, takich jak klasyfikacja czy regresja, metodyka ta umożliwia znaczną poprawę jakości produktów, efektywności produkcji czy przewidywania awarii maszyn. Przeanalizowane przypadki pokazują, że właściwie dobrana architektura sieci neuronowej oraz optymalne parametry mogą prowadzić do lepszych wyników w różnorodnych zastosowaniach przemysłowych. Metodyka ta stanowi uniwersalne narzędzie, które może być zaadaptowane do różnych procesów produkcyjnych, wspierając innowacyjność i zwiększając konkurencyjność przedsiębiorstw.

8. Zakończenie

W niniejszej monografii zaprezentowano badania dotyczące wykorzystania sztucznych sieci neuronowych w doskonaleniu procesów produkcyjnych, ze szczególnym uwzględnieniem optymalizacji parametrów obróbki skrawaniem, oceny jakości wyrobów gotowych oraz predykcji awarii maszyn. Badania te skupiały się na doborze odpowiednich architektur sieci neuronowych oraz ich parametrów w kontekście specyficznych zastosowań przemysłowych, co pozwoliło na uzyskanie optymalnych rezultatów w każdym z tych obszarów.

Rozdział 1 wprowadza czytelnika w problematykę zastosowania sztucznych sieci neuronowych w przemyśle produkcyjnym, podkreślając znaczenie efektywnej optymalizacji procesów produkcyjnych. Dalej omówiono cel i zakres pracy badawczej, wskazując na obszary, w których zastosowanie sieci neuronowych może przynieść wymierne korzyści. Rozdział 3 przedstawia teoretyczne podstawy sztucznych sieci neuronowych, a także znaczenie wyboru odpowiedniej architektury sieci w kontekście specyfiki danych i zadań produkcyjnych oraz wyzwania związane z procesem uczenia sieci neuronowych. W rozdziale 4 omówiona została metodyka modelowania i optymalizacji parametrów obróbki skrawaniem przy użyciu sztucznych sieci neuronowych, z naciskiem na zastosowanie perceptronów wielowarstwowych w kontekście ograniczonej ilości danych doświadczalnych. Rozdział 5 koncentruje się na zastosowaniu konwolucyjnych sieci neuronowych oraz perceptronu wielowarstwowego do oceny jakości wyrobów gotowych, w tym na klasyfikacji wad i przewidywaniu

twierdności po procesie odpuszczania. W rozdziale 6 opisano podejście do predykcji awarii maszyn z wykorzystaniem perceptronów wielowarstwowych oraz rekurencyjnych sieci neuronowych LSTM, podkreślając ich rolę w optymalizacji działań predykcyjnych. Rozdział 7 przedstawia szczegółową metodykę, która pozwala na skuteczny dobór zarówno architektury sieci neuronowych, jak i parametrów procesu ich uczenia.

8.1. Uwagi końcowe i wnioski

W ramach prowadzonych badań i analiz głównym celem było zaproponowanie sposobu doboru architektury oraz parametrów procesu uczenia sztucznych sieci neuronowych, które mogą być skutecznie zastosowane w doskonaleniu procesów produkcyjnych. Badania koncentrowały się na wykorzystaniu modeli neuronowych w następujących trzech obszarach: optymalizacji parametrów obróbki skrawaniem, ocenie jakości wyrobów gotowych oraz predykcji awarii maszyn. Każdy z tych obszarów został zbadany pod kątem efektywności wykorzystania różnych architektur sieci neuronowych, takich jak perceptrony wielowarstwowe (MLP), rekurencyjne sieci LSTM oraz konwolucyjne sieci neuronowe (CNN).

Opracowanie szczegółowej metodyki doboru odpowiednich architektur sieci neuronowych oraz parametrów ich uczenia stanowiło fundament wszystkich przeprowadzonych analiz. Metodyka ta obejmowała kompleksowy proces przygotowania danych, selekcji odpowiednich modeli oraz ich dostosowania do specyficznych wymagań poszczególnych zastosowań przemysłowych. Kluczowym aspektem tej metodyki było uwzględnienie charakterystyki dostępnych danych oraz specyfiki zadań, co pozwoliło na optymalny dobór architektury sieci neuronowej dla każdego z analizowanych obszarów.

Podczas badań nad optymalizacją parametrów obróbki skrawaniem szczególną uwagę poświęcono doborowi odpowiedniej architektury sieci neuronowej w kontekście dostępnych danych doświadczalnych. Wybór MLP jako głównej architektury do modelowania parametrów obróbki wynikał nie tylko z ich zdolności do modelowania złożonych, nieliniowych relacji, ale również ze specyficznych warunków badawczych, w których dostępna była stosunkowo niewielka ilość danych doświadczalnych. MLP ze względu na swoją strukturę i elastyczność wykazuje dużą odporność na problemy wynikające z ograniczonej liczności prób, co pozwoliło na efektywne wykorzystanie dostępnych danych i uzyskanie wiarygodnych wyników w kontekście optymalizacji procesu obróbki.

W kontekście optymalizacji parametrów obróbki skrawaniem najbardziej efektywne okazały się sieci neuronowe z jedną warstwą ukrytą, przy czym analizowano różne algorytmy uczenia oraz liczbę neuronów w tej warstwie. Spośród badanych algorytmów uczenia: LM, SCG, oraz BR, LM wykazał najwyższą efektywność. Optymalizacja dotyczyła parametrów obróbki, takich jak prędkość skrawania (v_c), posuw na ostrze (f_z) oraz głębokość skrawania (a_p). W przypadku modelowania parametrów chropowatości, takich jak Ra , najlepsze wyniki uzyskano przy wykorzystaniu MLP z 6 neuronami w warstwie ukrytej, co dało współczynnik korelacji R równy 0,9869. Dla parametru Rz optymalną konfiguracją okazała się sieć MLP z 9 neuronami w warstwie ukrytej, z wartością R wynoszącą 0,9894. Natomiast dla RSm zastosowanie 10 neuronów w warstwie ukrytej pozwoliło osiągnąć współczynnik R na poziomie 0,911. W przypadku modelowania składowej siły skrawania wzdłuż osi X (F_x) najlepsze rezultaty uzyskano przy użyciu MLP z 7 neuronami w warstwie ukrytej oraz algorytmu LM, co pozwoliło osiągnąć współczynnik R równy 0,999. Tak dobrane parametry sieci neuronowej umożliwiły skuteczną optymalizację procesu obróbki skrawaniem, nawet przy ograniczonej ilości danych doświadczalnych.

Podobne rozważania dotyczące doboru architektury sieci neuronowej w odniesieniu do ilości i formatu danych były prowadzone w innych obszarach badań. W przypadku oceny jakości wyrobów gotowych dla problemów klasyfikacyjnych dane wykorzystywane do analizy miały format obrazów, co w naturalny sposób kierowało wybór odpowiedniej architektury sieci neuronowej w stronę CNN. Są one specjalnie zaprojektowane do przetwarzania danych obrazowych, ponieważ potrafią automatycznie i efektywnie identyfikować istotne cechy i wzorce bez potrzeby ręcznego definiowania tych cech, co jest charakterystyczne dla tradycyjnych metod przetwarzania obrazów. W przypadku analizy pierwszych dwóch obiektów, takich jak wykrywanie wad w odlewach oraz wykrywanie błędów kształtu na liniach produkcyjnych, zastosowano technikę transfer learning. Transfer learning pozwolił na szybkie zaadaptowanie wstępnie wytrenowanych modeli do nowych zadań, takich jak klasyfikacja defektów w odlewach, gdzie sieć była w stanie skutecznie zidentyfikować różne typy defektów, na przykład pęknięcia, porowatość czy zanieczyszczenia powierzchniowe, bazując na zaledwie kilkuset obrazach uczących. Podobnie w przypadku wykrywania błędów kształtu na liniach produkcyjnych transfer learning umożliwił wykorzystanie wcześniej nabytej wiedzy modelu do precyzyjnej klasyfikacji kształtów. W szczególności na liniach produkcyjnych wykrywane były obiekty w kształcie koła, co wymagało od sieci neuronowej zdolności do identyfikacji subtelnych różnic i anomalii w kształcie oraz

wymiarach tych elementów. Dodatkowo, aby bardziej odwzorować rzeczywiste warunki produkcyjne, obrazy były celowo rozmyte z powodu ruchu na taśmociągu. W obu przypadkach zastosowanie CNN pozwoliło na precyzyjne wykrywanie wad i nieprawidłowości, co jest istotne dla automatyzacji kontroli jakości w czasie rzeczywistym.

W przypadku oceny jakości wyrobów gotowych dotyczących problemu regresyjnego – określenia twardości po odpuszczaniu wyrobów gotowych – również uwzględniono dobór odpowiedniej architektury sieci neuronowej w kontekście ilości i formatu dostępnych danych. W tym przypadku modelowanie twardości po procesie odpuszczania, które jest ważnym parametrem jakościowym wyrobów, wymagało zastosowania sieci neuronowej zdolnej do dokładnego przewidywania wartości ciągłych. Podjęta decyzja o zastosowaniu MLP była uzasadniona specyficznymi warunkami badawczymi. W procesie modelowania twardości po odpuszczaniu dysponowano ograniczoną ilością danych doświadczalnych, co wykluczało możliwość użycia bardziej złożonych modeli, takich jak głębokie sieci neuronowe, które wymagają dużych zbiorów danych do skutecznego uczenia sieci. Poprzez zastosowanie MLP możliwe było także zachowanie stabilności modelu przy niewielkiej liczbie prób, co jest szczególnie ważne w kontekście przemysłowym, gdzie dostęp do dużych ilości danych może być ograniczony. Dzięki precyzyjnemu doborowi liczby neuronów (9) i algorytmu uczenia (BR) uzyskano wysoką dokładność prognoz ($R=0,9894$), co jest istotne dla zapewnienia jakości wyrobów. Dodatkowo opracowano algorytm wraz z interfejsem użytkownika, który umożliwia bieżącą ocenę jakości wyrobu gotowego na podstawie przewidywanej przez model MLP (z algorytmem uczenia BR) twardości po odpuszczaniu. Algorytm ten określa, czy dany wyrób spełnia wymagania jakościowe. Jeżeli przewidywana twardość mieści się w określonym przedziale tolerancji, wyrób jest uznawany za zgodny z normami jakościowymi; w przeciwnym razie klasyfikuje się go jako niezgodny z wymaganiami. Taki system wspomagania decyzji stanowi cenne narzędzie w procesie produkcyjnym, zwiększając precyzję kontroli jakości i poprawiając efektywność zarządzania jakością.

W obszarze predykcji awarii maszyn, gdzie dane miały charakter sekwencyjny, a ich ilość była wystarczająca do trenowania bardziej złożonych modeli, zastosowano zarówno MLP, jak i LSTM, co pozwoliło na skuteczne podejście do zróżnicowanych aspektów problemu. Wybór tych architektur był motywowany specyficznymi właściwościami obu typów sieci. MLP okazały się lepszym wyborem w przypadku uchwycenia podstawowych wzorców i relacji w danych, szczególnie w kontekście klasyfikacji binarnej, czyli określania, czy awaria wystąpi

czy też nie. Użycie MLP pozwoliło na szybkie przetworzenie danych wejściowych, takich jak temperatura powietrza, temperatura procesu, prędkość obrotowa, moment obrotowy i zużycie narzędzi, oraz na identyfikację istotnych zależności między tymi parametrami wejściowymi a wynikiem końcowym, czyli wystąpieniem lub brakiem awarii. Natomiast rekurencyjne sieci neuronowe LSTM zostały zastosowane w bardziej zaawansowanym etapie analizy, polegającym na identyfikacji rodzaju awarii, gdy już została ona przewidziana. LSTM dzięki swojej zdolności do przetwarzania i analizy długoterminowych zależności w danych sekwencyjnych okazały się wyjątkowo skuteczne w rozpoznawaniu skomplikowanych wzorców, które są niezbędne do precyzyjnej klasyfikacji typu awarii. Takie dwustopniowe podejście łączące prostotę MLP z zaawansowaną analizą sekwencyjną LSTM zapewniło nie tylko efektywność w wykrywaniu awarii, ale także dokładność w określaniu jej typu, co jest istotne dla planowania działań konserwacyjnych i minimalizacji przestojów produkcyjnych. Wybór tych dwóch różnych architektur sieci neuronowych dostosowanych do specyfiki i ilości danych na każdym etapie analizy przyczynił się do zwiększenia skuteczności i niezawodności całego systemu predykcji awarii.

Przeprowadzone analizy wykazały, że dobór architektury sieci neuronowej i jej parametrów musi być ściśle dostosowany do specyfiki zadania oraz dostępnych danych. Odpowiednie dobranie tych elementów pozwoliło na osiągnięcie optymalnych wyników w każdym z badanych obszarów, potwierdzając, że sztuczne sieci neuronowe stanowią nieodzowne narzędzie w nowoczesnych procesach produkcyjnych i mogą przyczynić się do doskonalenia procesów produkcyjnych.

Opracowanie szczegółowej metodyki doboru odpowiednich architektur sieci neuronowych oraz parametrów ich uczenia stanowi znaczący wkład w rozwój nowoczesnych technologii wytwarzania. Ta metodyka obejmująca kompleksowy proces przygotowania danych, selekcję odpowiednich modeli oraz ich dostosowania do specyficznych wymagań przemysłowych, jest istotna dla optymalizacji różnych aspektów produkcji. Dzięki niej możliwe okazuje się precyzyjne dopasowanie architektury sieci neuronowej do charakterystyki dostępnych danych oraz specyfiki zadania, co pozwala na uzyskanie maksymalnej efektywności w każdym z analizowanych obszarów.

Nowością w prezentowanej pracy jest systematyczne podejście do doboru architektur i parametrów uczenia sieci neuronowych, co umożliwia ich elastyczne i wszechstronne zastosowanie w różnych kontekstach produkcyjnych. Zamiast ograniczać się do jednego modelu czy podejścia, metodyka ta pozwala na dostosowanie narzędzi do specyficznych wyzwań, m.in. takich

jak optymalizacja parametrów obróbki frezowaniem, ocena jakości wyrobów gotowych czy predykcja awarii maszyn.

Pomimo licznych zalet stosowanie sieci neuronowych w procesach produkcyjnych wiąże się z pewnymi ograniczeniami. Wymagają one dużych zbiorów danych do skutecznego uczenia, a bardziej złożone modele, takie jak głębokie sieci neuronowe, mogą być trudne do interpretacji, co utrudnia wyjaśnienie wyników końcowych. Dodatkowo realizacja takich modeli często wymaga znacznych zasobów obliczeniowych, co może stanowić barierę dla ich wdrożenia w środowiskach o ograniczonych zasobach technologicznych. Rozważenie tych ograniczeń jest istotne dla skutecznego wykorzystania sieci neuronowych w przemyśle.

Wyniki prowadzonych badań dostarczają nie tylko nowej wiedzy teoretycznej na temat możliwości i ograniczeń różnych architektur sieci neuronowych, ale również praktycznych narzędzi, które mogą zostać zaimplementowane w rzeczywistych warunkach przemysłowych. Kluczowym elementem tych badań była opracowana metodyka doboru architektury i parametrów procesu uczenia sieci neuronowych, ukierunkowana na doskonalenie procesów produkcyjnych. Aby korzystanie z opracowanych algorytmów optymalizacyjnych i wyuczonych sieci neuronowych było bardziej przyjazne i intuicyjne dla użytkownika, stworzono interfejsy graficzne, które umożliwiają łatwe wprowadzanie danych wejściowych oraz szybkie uzyskiwanie wyników optymalizacji czy predykcji. Dzięki takiemu podejściu użytkownicy mogą skupić się na analizie wyników bez potrzeby zagłębiania się w szczegóły techniczne czy kod źródłowy. Takie podejście nie tylko zwiększa efektywność pracy, ale również umożliwia szersze zastosowanie tych narzędzi w praktyce przemysłowej. Dzięki temu rozwój obszaru inżynierii produkcji może przebiegać szybciej, z większą pewnością w osiąganiu założonych celów jakościowych i wydajnościowych, co stanowi istotny wkład w rozwój nowoczesnych technologii produkcyjnych.

8.2. Kierunki dalszych badań

Zastosowanie sztucznych sieci neuronowych w przemyśle produkcyjnym otwiera nowe perspektywy zarówno w zakresie poprawy efektywności procesów, jak i automatyzacji kontroli jakości oraz predykcji awarii. Mimo że obecne badania dostarczyły istotnych wyników i praktycznych narzędzi, istnieje wiele obszarów, które wymagają dalszych badań i rozwoju, aby w pełni wykorzystać potencjał tych narzędzi w kontekście nowoczesnych technologii wytwarzania.

Jednym z głównych kierunków dalszych badań będzie rozwój i adaptacja hybrydowych modeli sieci neuronowych, które łączą różne architektury w celu uzyskania synergii ich zalet w specyficznych zastosowaniach. Przykładem może być połączenie CNN z rekurencyjnymi sieciami neuronowymi LSTM. Tego rodzaju hybrydowe podejście mogłoby być szczególnie efektywne w analizie danych czasowo-przestrzennych, takich jak dynamiczna kontrola jakości na liniach produkcyjnych, gdzie obiekty są w ruchu, lub w predykcji awarii maszyn na podstawie sygnałów zbieranych w czasie rzeczywistym. Hybrydowe modele mogłyby z jednej strony wykorzystywać zdolność CNN do ekstrakcji cech z obrazów, a z drugiej strony dzięki LSTM analizować zależności czasowe, co w rezultacie mogłoby prowadzić do bardziej precyzyjnych i kompleksowych rozwiązań w procesach produkcyjnych.

Warto również rozważyć zastosowanie innych metod uczenia maszynowego, które mogą być komplementarne do sieci neuronowych lub stanowić alternatywę w specyficznych warunkach. Przykładem są algorytmy zespołowe, takie jak lasy losowe czy *gradient boosting machines* (GBM), które są szczególnie efektywne w analizie danych tabelarycznych i znajdują zastosowanie w klasyfikacji lub regresji tam, gdzie sieci neuronowe mogą być zbyt skomplikowane lub podatne na przetrenowanie. Zespołowe metody uczenia maszynowego mogą być również używane jako elementy większych hybrydowych systemów, gdzie wyniki uzyskane przez sieci neuronowe są dodatkowo wzbogacane lub potwierdzane przez inne algorytmy. Badania nad integracją tych metod z sieciami neuronowymi mogą prowadzić do stworzenia jeszcze bardziej efektywnych systemów produkcyjnych, które potrafią lepiej radzić sobie z różnorodnymi typami danych i zadaniami.

Kolejnym istotnym obszarem badań jest zastosowanie technik transfer learning oraz meta-uczenia w jeszcze szerszym zakresie zastosowań przemysłowych. Transfer learning, które umożliwia adaptację wcześniej wytrenowanych modeli do nowych zadań, ma ogromny potencjał w kontekście skrócenia czasu trenowania i zwiększenia efektywności modeli. Przyszłe badania mogłyby skoncentrować się na zastosowaniu tej techniki do bardziej złożonych zadań produkcyjnych, takich jak adaptacja modeli do specyficznych warunków produkcji lub nowych materiałów. Również meta-uczenie, które polega na szybkim dostosowywaniu modeli do nowych zadań, mogłoby zostać eksplorowane jako metoda poprawy adaptacyjności sieci neuronowych w dynamicznie zmieniającym się środowisku produkcyjnym. Te techniki mogłyby znacząco zwiększyć elastyczność i wszechstronność modeli, co jest istotne w odniesieniu do coraz bardziej wymagających warunków produkcji.

W kontekście nowoczesnych rozwiązań produkcyjnych integracja sieci neuronowych z systemami Internetu Rzeczy (IoT) oraz Przemysłem 4.0 jest kolejnym istotnym kierunkiem badań. Przemysł 4.0 i IoT kładą duży nacisk na zbieranie i analizę danych w czasie rzeczywistym, co stwarza nowe możliwości dla zastosowania sieci neuronowych do dynamicznej optymalizacji procesów produkcyjnych. Badania mogą skoncentrować się na opracowaniu skalowalnych i wydajnych systemów, które integrują analizę danych z czujników z modelami neuronowymi, umożliwiając tym samym bardziej inteligentne zarządzanie produkcją. Taka integracja mogłaby również przyczynić się do zwiększenia automatyzacji i autonomii systemów produkcyjnych, co jest istotne dla podniesienia konkurencyjności przedsiębiorstw na rynku globalnym.

Podsumowując, rozwój hybrydowych modeli sieci neuronowych, stosowanie w szerszym zakresie technik transfer learning i meta-uczenia oraz integracja z systemami IoT to kluczowe kierunki dalszych badań, które mogą przyczynić się do jeszcze bardziej zaawansowanego wykorzystania sztucznej inteligencji i metod uczenia maszynowego w przemyśle produkcyjnym. Realizacja tych badań ma potencjał, aby znacząco wpłynąć na rozwój nowoczesnych technologii wytwarzania, zwiększając efektywność, jakość i konkurencyjność przedsiębiorstw na rynku globalnym.

Literatura

- [1] Ghobadian A, Gallear DN. Total quality management in SMEs. *Omega* (Westport) 1996;24:83–106. [https://doi.org/10.1016/0305-0483\(95\)00055-0](https://doi.org/10.1016/0305-0483(95)00055-0).
- [2] Antony J. Six Sigma in the UK service organisations: results from a pilot survey. *Managerial Auditing Journal* 2004;19:1006–13. <https://doi.org/10.1108/02686900410557908>.
- [3] Antosz K, Pacana A, Stadnicka D, Zielecki W. *Lean Manufacturing: doskonalenie produkcji*. Politechnika Rzeszowska, 2018.
- [4] Patel AS, Patel KM. Critical review of literature on Lean Six Sigma methodology. *International Journal of Lean Six Sigma* 2020;12:627–74. <https://doi.org/10.1108/IJLSS-04-2020-0043/FULL/PDF>.
- [5] Montgomery DC. *Introduction to Statistical Quality Control*, 8th Edition. John Wiley & Sons, Inc, 2019.
- [6] Christopher M. *Logistics and supply chain management*. Pearson/Financial Times Publishing, 2023.
- [7] Jaurker D, Pradhan MK, Jaurker S, Malviya R. Optimization Techniques Used in Machining Processes: A Review. *Lecture Notes in Mechanical Engineering* 2023;93–101. https://doi.org/10.1007/978-981-99-2921-4_10.
- [8] Li L, Li C, Tang Y, Yi Q. Influence factors and operational strategies for energy efficiency improvement of CNC machining. *J Clean Prod* 2017;161:220–38. <https://doi.org/10.1016/J.JCLEPRO.2017.05.084>.
- [9] Singh MP, Sharma P, Dubey Y, Seshagiri Rao GVR, Mohammad Q, Lakhanpal S. Energy-Efficient Manufacturing: Opportunities and Challenges. *E3S Web of Conferences* 2024;505:01032. <https://doi.org/10.1051/E3SCONF/202450501032>.
- [10] Naidoo C, Pillay N, Maharaj S. A Data Driven Approach to Process Control Modelling Using Machine Learning. *International Conference on Electrical, Computer and Energy Technologies, ICECET* 2023;2023. <https://doi.org/10.1109/ICECET58911.2023.10389464>.

- [11] Product quality (PQ) identification in manufacturing companies: the perspective of managers and executives. *Vadyba* 2023;39:15–22.
- [12] Sharma DR, Singh B. Understanding the Relationship Between Customer Satisfaction, Customer Engagement and Repeat Purchase Behaviour. 2021;27:449–57. <https://doi.org/10.1177/0972262921992593>.
- [13] Achouch M, Dimitrova M, Ziane K, Sattarpanah Karganroudi S, Dhoubi R, Ibrahim H, et al. On Predictive Maintenance in Industry 4.0: Overview, Models, and Challenges. *Applied Sciences* 2022;12:8081. <https://doi.org/10.3390/APP12168081>.
- [14] Muganyi P, Mbohwa C. Proactive Maintenance Strategic Application to Advance Equipment Reliability. *Proceedings of the International Conference on Industrial Engineering and Operations Management, IEOM Society International* 2018, Page 3300–9.
- [15] Khoudi A, Masrour T, Hassani E, Mazgualdi E, Khoudi A, Masrour T, et al. A Deep-Reinforcement-Learning-Based Digital Twin for Manufacturing Process Optimization. *Systems* 2024;12:38. <https://doi.org/10.3390/SYSTEMS12020038>.
- [16] Genalti G, Corbo G, Bianchi T, Missaglia M, Negri L, Sala A, et al. Enhancing Manufacturing with AI-powered Process Design. *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence* 2024;9:8665–8. <https://doi.org/10.24963/IJCAI.2024/1004>.
- [17] Li J, Yin W, Yang B, Chen L, Dong R, Chen Y, et al. Modeling of Digital Twin Workshop in Planning via a Graph Neural Network: The Case of an Ocean Engineering Manufacturing Intelligent Workshop. *Applied Sciences* 2023;13:10134. <https://doi.org/10.3390/APP131810134>.
- [18] Merayo D, Rodríguez-Prieto A, Camacho AM. Topological Optimization of Artificial Neural Networks to Estimate Mechanical Properties in Metal Forming Using Machine Learning. *Metals* 2021;11:1289. <https://doi.org/10.3390/MET11081289>.
- [19] Rodríguez-Cabal MA, Marin JGA, Rudas S, Rodríguez-Cabal MA. Prediction of energy consumption in the leadwell v-40 it CNC machining center through artificial neural networks. *Journal of Applied Engineering Science* 2022;20:145–9. <https://doi.org/10.5937/JAES0-30826>.
- [20] Capizzi G, Lo Sciuto G, Wasef Shaheen B, Németh I. Performance Monitoring of Wind Turbines Gearbox Utilising Artificial Neural Networks – Steps toward Successful Implementation of Predictive Maintenance Strategy. *Processes* 2023;11:269. <https://doi.org/10.3390/PR11010269>.
- [21] Tang H, Cui X, Zhu J, Shen L, Cai J, Zhu B. A Bayesian-optimized Hybrid Neural Network Based on CNN and BiLSTM for Predictive Maintenance of Diesel Generator. *Proceedings – 2023 Prognostics and Health Management Conference – Paris, PHM-Paris* 2023;2023:17–22. <https://doi.org/10.1109/PHM58589.2023.00012>.
- [22] Pérez-Gomariz M, López-Gómez A, Cerdán-Cartagena F. Artificial Neural Networks as Artificial Intelligence Technique for Energy Saving in Refrigeration Systems – A Review. *Clean Technologies* 2023;5:116–36. <https://doi.org/10.3390/CLEANTECHNOL5010007>.
- [23] Massaro A, Manfredonia I, Galiano A, Xhahysa B. Advanced Process Defect Monitoring Model and Prediction Improvement by Artificial Neural Network in Kitchen Manufacturing Industry: A Case of Study. *2019 IEEE International Workshop on Metrology for Industry 40 and IoT, MetroInd 40 and IoT* 2019 – *Proceedings* 2019:64–7. <https://doi.org/10.1109/METROI4.2019.8792872>.

- [24] Bakshi K Das, Thilagham KT, Yamini R, Reddy CVB, Gaikwad AS, Gupta A. Modelling and Manufacturing of Abrasive Water Jet Machining of Reinforced AI Alloy Cast Hybrid Nanocomposites with Mechanical and Metallurgical Behavior through Artificial Neural Network. Proceedings of the 4th International Conference on Smart Electronics and Communication, ICOSEC 2023 2023;1387–93. <https://doi.org/10.1109/ICOSEC58147.2023.10276082>.
- [25] Jilak A, Assareh E, Nedaei M. Application of a novel multi-objective optimisation method integrated with the artificial neural networks for optimum design of a plate heat exchanger. Australian Journal of Mechanical Engineering 2020;18:1–15. <https://doi.org/10.1080/14484846.2017.1359897>.
- [26] McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 1943;5:115–33. <https://doi.org/10.1007/BF02478259/METRICS>.
- [27] Hebb DO. The organization of behavior; a neuropsychological theory. Wiley 1949.
- [28] Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychol Rev 1958;65:386–408. <https://doi.org/10.1037/H0042519>.
- [29] Widrow B, Hoff ME. Adaptive switching circuits. Neurocomputing, Vol 1, The MIT Press; 1960, Pages 96–104. <https://doi.org/10.7551/MITPRESS/4943.003.0012>.
- [30] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986 323:6088 1986;323:533–6. <https://doi.org/10.1038/323533a0>.
- [31] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 1998;86:2278–323. <https://doi.org/10.1109/5.726791>.
- [32] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Comput 2006;18:1527–54. <https://doi.org/10.1162/NECO.2006.18.7.1527>.
- [33] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM 2017;60:84–90. <https://doi.org/10.1145/3065386>.
- [34] Cho K, van Merriënboer B, Bahdanau D, Bengio Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. Proceedings of SSST 2014 – 8th Workshop on Syntax, Semantics and Structure in Statistical Translation 2014:103–11. <https://doi.org/10.3115/v1/w14-4012>.
- [35] Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Comput 1997;9:1735–80. <https://doi.org/10.1162/NECO.1997.9.8.1735>.
- [36] Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, et al. A survey on deep learning in medical image analysis. Med Image Anal 2017;42:60–88. <https://doi.org/10.1016/J.MEDIA.2017.07.005>.
- [37] Hireš M, Bugata P, Gazda M, Hreško DJ, Kanász R, Vavrek L, et al. Brief Overview of Neural Networks for Medical Applications. Acta Electrotechnica et Informatica 2022;22:34–44. <https://doi.org/10.2478/AEI-2022-0010>.
- [38] Hamdan BA. Neural Network Principles and its Application. Webology 2022;19:3955–70. <https://doi.org/10.14704/WEB/V19I1/WEB19261>.
- [39] Heaton JB, Polson NG, Witte JH. Deep learning for finance: deep portfolios. Appl Stoch Models Bus Ind 2017;33:3–12. <https://doi.org/10.1002/ASMB.2209>.
- [40] Graves A, Mohamed AR, Hinton G. Speech Recognition with Deep Recurrent Neural Networks. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing – Proceedings 2013:6645–9. <https://doi.org/10.1109/ICASSP.2013.6638947>.

- [41] Saxena V, Kumar A, Arya G, Bhardwaj MK. Jarvis-Virtual Voice Assistant. *Journal of Computer Systems, Virtualization and Languages* 2024;1:24–33. <https://doi.org/10.48001/JOCSVL.2024.1124-33>.
- [42] Yadav P, Tugnait T, Dubey SK. Analysis of Personalized AI Assistant with Facial Recognition and Voice Representation. *Proceedings of the 2023 12th International Conference on System Modeling and Advancement in Research Trends, SMART 2023* 2023:57–64. <https://doi.org/10.1109/SMART59791.2023.10428670>.
- [43] Simon C, Rajeswari M. Voice-based Virtual Assistant with Security. *Proceedings of the 2023 2nd International Conference on Electronics and Renewable Systems, ICEARS 2023* 2023:822–7. <https://doi.org/10.1109/ICEARS56392.2023.10085043>.
- [44] Zhang W, Yang D, Wang H. Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey. *IEEE Syst J* 2019;13:2213–27. <https://doi.org/10.1109/JSYST.2019.2905565>.
- [45] Kulisz M, Kujawska J, Przysucha B, Cel W. Forecasting water quality index in groundwater using artificial neural network. *Energies (Basel)* 2021;14. <https://doi.org/10.3390/en14185875>.
- [46] Kujawska J, Kulisz M, Oleszczuk P, Cel W. Machine Learning Methods to Forecast the Concentration of PM10 in Lublin, Poland. *Energies (Basel)* 2022;15. <https://doi.org/10.3390/en15176428>.
- [47] Kulisz M, Kujawska J, Aubakirova Z, Zhairbaeva G, Warowny T. Prediction of the compressive strength of environmentally friendly concrete using artificial neural network. *Applied Computer Science* 2022;18:68–81. <https://doi.org/10.35784/ACS-2022-29>.
- [48] Rymarczyk T, Kłosowski G. Innovative methods of neural reconstruction for tomographic images in maintenance of tank industrial reactors. *Eksploracja i Niezawodność – Maintenance and Reliability* 2019;21:261–7. <https://doi.org/10.17531/EIN.2019.2.10>.
- [49] Kłosowski G, Rymarczyk T, Niderla K, Kulisz M, Skowron Ł, Soleimani M. Using an LSTM network to monitor industrial reactors using electrical capacitance and impedance tomography – a hybrid approach. *Eksploracja i Niezawodność* 2023;25. <https://doi.org/10.17531/EIN.2023.1.11>.
- [50] Qureshi M, Arbab MA, Rehman S. Deep learning-based forecasting of electricity consumption. *Scientific Reports* 2024;14:1–11. <https://doi.org/10.1038/s41598-024-56602-4>.
- [51] Ramos PVB, Villela SM, Silva WN, Dias BH. Residential energy consumption forecasting using deep learning models. *Appl Energy* 2023;350:121705. <https://doi.org/10.1016/J.APENERGY.2023.121705>.
- [52] Kamilaris A, Prenafeta-Boldú FX. Deep learning in agriculture: A survey. *Comput Electron Agric* 2018;147:70–90. <https://doi.org/10.1016/J.COMPAG.2018.02.016>.
- [53] Younis H, Sundarakani B, Alsharairi M. Applications of artificial intelligence and machine learning within supply chains: systematic review and future research directions. *Journal of Modelling in Management* 2022;17:916–40. <https://doi.org/10.1108/JM2-12-2020-0322/FULL/PDF>.
- [54] Pournader M, Ghaderi H, Hassanzadegan A, Fahimnia B. Artificial intelligence applications in supply chain management. *Int J Prod Econ* 2021;241:108250. <https://doi.org/10.1016/J.IJPE.2021.108250>.
- [55] Lechtenberg S, Hellingrath B. Applications of artificial intelligence in supply chain management: Identification of main research fields and greatest industry interests. *ERCIS Working Papers* 2021.

- [56] Kandel ER, Schwartz JH, Jessell TM, Siegelbaum SA, Hudspeth AJ. Principles of Neural Science, Fifth Edition. McGraw-Hill Education, 2012.
- [57] Bishop CM. Neural Networks for Pattern Recognition. 1st ed. Oxford University Press, 1996.
- [58] Goodfellow I, Bengio Y, Courville A. Deep learning. The MIT Press; 2016.
- [59] Nair V, Hinton GE. Rectified Linear Units Improve Restricted Boltzmann Machines. International Conference on Machine Learning 2010.
- [60] Maas AL. Rectifier Nonlinearities Improve Neural Network Acoustic Models 2013.
- [61] Tadeusiewicz R. Sieci neuronowe. Warszawa: Akademicka Oficyna Wydawnicza, 1993.
- [62] Haykin S. Neural networks: a comprehensive foundation. Prentice Hall 1998.
- [63] Zhu H. Multi-layered perceptron and its applications in biotechnology. Theoretical and Natural Science 2023;20:159–65. <https://doi.org/10.54254/2753-8818/20/20230753>.
- [64] Zhang G, Eddy Patuwo B, Hu MY. Forecasting with artificial neural networks: The state of the art. Int J Forecast 1998;14:35–62. [https://doi.org/10.1016/S0169-2070\(97\)00044-7](https://doi.org/10.1016/S0169-2070(97)00044-7).
- [65] Tsoukalas LH., Uhrig RE. Fuzzy and neural approaches in engineering 1997:587.
- [66] Mohammadi J, Ataei M, Kakaei RK, Mikaeil R, Haghshenas SS. Prediction of the Production Rate of Chain Saw Machine using the Multilayer Perceptron (MLP) Neural Network. Civil Engineering Journal 2018;4:1575–83. <https://doi.org/10.28991/CEJ-0309196>.
- [67] Ahmarofi AA, Kassa FM, Ishak MK. Predicting the Cycle Time at a Production Line Through the Development of the 3-3-1 Multilayer Perceptron Artificial Neural Networks with Formulated Momentum Rate. Lecture Notes in Mechanical Engineering 2021:165–73. https://doi.org/10.1007/978-981-16-0866-7_13.
- [68] Ramezani J, Jassbi J. A hybrid expert decision support system based on artificial neural networks in process control of plaster production – An industry 4.0 perspective. IFIP Adv Inf Commun Technol 2017;499:55–71. https://doi.org/10.1007/978-3-319-56077-9_5/FIGURES/7.
- [69] Hochreiter S, Schmidhuber J. LSTM can Solve Hard Long Time Lag Problems. Neural Information Processing Systems 1996.
- [70] Graves A. Generating Sequences With Recurrent Neural Networks 2013.
- [71] Kłosowski G, Rymarczyk T, Niderla K, Rzemieniak M, Dmowski A, Maj M. Comparison of machine learning methods for image reconstruction using the LSTM classifier in industrial electrical tomography. Energies (Basel) 2021;14. <https://doi.org/10.3390/en14217269>.
- [72] Staudemeyer RC, Morris ER. Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. ArXivOrg 2019.
- [73] Long Short-Term Memory Neural Networks – MATLAB & Simulink n.d. <https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html> (accessed August 10, 2024).
- [74] Sutskever I, Vinyals O, Le Q V. Sequence to Sequence Learning with Neural Networks. Adv Neural Inf Process Syst 2014;4:3104–12.
- [75] Graves A, Mohamed AR, Hinton G. Speech Recognition with Deep Recurrent Neural Networks. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing – Proceedings 2013:6645–9. <https://doi.org/10.1109/ICASSP.2013.6638947>.

- [76] Al-Said S, Findik O, Assanova B, Sharmukhanbet S, Baitemirova N. Enhancing Predictive Maintenance in Manufacturing: A CNN-LSTM Hybrid Approach for Reliable Component Failure Prediction. *Studies in Systems, Decision and Control* 2024;223:137–53. https://doi.org/10.1007/978-3-031-51997-0_11.
- [77] Dehghan Shoorkand H, Nourelfath M, Hajji A. A hybrid CNN-LSTM model for joint optimization of production and imperfect predictive maintenance planning. *Reliab Eng Syst Saf* 2024;241:109707. <https://doi.org/10.1016/J.RESS.2023.109707>.
- [78] Munir A, Kong J, Qureshi M. Accelerators for convolutional neural networks 2024.
- [79] Yamashita R, Nishio M, Do RKG, Togashi K. Convolutional neural networks: an overview and application in radiology. *Insights Imaging* 2018;9:611–29. <https://doi.org/10.1007/S13244-018-0639-9/FIGURES/15>.
- [80] Chen Y-W, Ruan X, Jain RK. Deep Convolutional Neural Networks 2024:1–31. https://doi.org/10.1007/978-3-031-59811-1_1.
- [81] Basha NK, Bhatia S, Kumar A, Mashat A. Deep learning and its applications using Python. Scrivener Publishing, 2023:256. <https://doi.org/10.1002/9781394167791>.
- [82] Zan P, Yao JJ, Fan L, Yang Y, Zhou Z, Wu Z, et al. Efficacy of a Four-Hour Drainage Clamping Technique in the Reduction of Blood Loss Following Total Hip Arthroplasty: A Prospective Cohort Study. *Med Sci Monit* 2017;23:2708–14. <https://doi.org/10.12659/MSM.904864>.
- [83] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 1998;86:2278–323. <https://doi.org/10.1109/5.726791>.
- [84] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. *Adv Neural Inf Process Syst* 2012;25.
- [85] Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017* 2017;2018-January:1–6. <https://doi.org/10.1109/ICENGTECHNOL.2017.8308186>.
- [86] Chi J, On CK, Zhang H, Chai SS. A Review of Deep Convolutional Neural Networks in Mobile Face Recognition. *International Journal of Interactive Mobile Technologies (IJIM)* 2023;17:4–19. <https://doi.org/10.3991/IJIM.V17I23.40867>.
- [87] Day O, Khoshgoftaar TM. A survey on heterogeneous transfer learning. *J Big Data* 2017;4:1–42. <https://doi.org/10.1186/S40537-017-0089-0/FIGURES/5>.
- [88] Dewangan U, Talwekar RH, Bera S. A Systematic Review on Cotton Plant Disease Detection & Classification Using Machine & Deep Learning Approach. 2023 1st DMIHER International Conference on Artificial Intelligence in Education and Industry 40, IDICAIEI 2023 2023. <https://doi.org/10.1109/IDICAIEI58380.2023.10406941>.
- [89] Han L, Wen Z, Pu H, Li W. Research on Mechanical Fault Diagnosis Algorithm Based on Sound Signal and CNN. *Int J Sci Res Sci Eng Technol* 2022;9:153–60. <https://doi.org/10.32628/IJSRSET229141>.
- [90] Meghashree, Mendonca AE, Shetty AS. Automated Quality Assessment of Crops Using CNN - Keras. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* 2021:641–5. <https://doi.org/10.32628/CSEIT2173186>.
- [91] Sucharitha V, Sneha K, Sravani D, Jhade S, Sravani P, Sreeja S. Imaging Description Production by Means of Deeper Neural Networks 2023:1–5. <https://doi.org/10.1109/ACCAI58221.2023.10200618>.

- [92] Chen Y, Goorden MC, Beekman FJ, Li K. Convolutional Neural Networks Combined with Machine Vision for Mechanical Compressor Defect Detection. *J Phys Conf Ser* 2023;2425:012039. <https://doi.org/10.1088/1742-6596/2425/1/012039>.
- [93] Ma J, Li N, Jiang Y, Feng T. Abnormal sound detection of washing machines based on convolution neural network in production environment. *International Journal of Wireless and Mobile Computing* 2021;20:201–11. <https://doi.org/10.1504/IJWMC.2021.115660>.
- [94] Wang C, Jiang W, Yang X, Zhang S. RUL Prediction of Rolling Bearings Based on a DCAE and CNN. *Applied Sciences* 2021;11:11516. <https://doi.org/10.3390/AP112311516>.
- [95] Lee RST. Transfer Learning and Transformer Technology. *Natural Language Processing* 2024;175–97. https://doi.org/10.1007/978-981-99-1999-4_8.
- [96] Malik N, Bzdok D. From YouTube to the brain: Transfer learning can improve brain-imaging predictions with deep learning. *Neural Networks* 2022;153:325–38. <https://doi.org/10.1016/J.NEUNET.2022.06.014>.
- [97] Srikanth M, M B. Deep Learning Approaches for Predictive Modeling and Optimization of Metabolic Fluxes in Engineered Microorganism. *International Journal of Research in Science & Engineering (IJRISE)* ISSN: 2394-8299 2023;3:1–11. <https://doi.org/10.55529/ijrise.35.1.11>.
- [98] Tanyıldız H, Şahin CB. Transfer Learning for Detection of Casting Defects Model in Scope of Industrial 4.0. *Turkish Journal of Nature and Science* 2023;12:45–51. <https://doi.org/10.46810/TDFD.1236584>.
- [99] Manohar R, Aarthi MS, Ancy Jenifer J. Leveraging Deep Learning for Early Stage Diabetic Retinopathy Detection: A Novel CNN and Transfer Learning Comparison. *2nd International Conference on Artificial Intelligence and Machine Learning Applications: Healthcare and Internet of Things, AIMLA 2024* 2024. <https://doi.org/10.1109/AIMLA59606.2024.10531542>.
- [100] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2016;13-17-August-2016:785–94. <https://doi.org/10.1145/2939672.2939785>.
- [101] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. *Nature* 1986;323:533–6. <https://doi.org/10.1038/323533a0>.
- [102] Al-Haddad Y, İbrahim AA, Naeem R. Forecasting Energy Consumption in Smart Grids: A Comparative Analysis of Recurrent Neural Networks. *Iraqi Journal for Computers and Informatics* 2024;50:123–32. <https://doi.org/10.25195/IJCI.V50I1.492>.
- [103] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *EMNLP 2014 – 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* 2014:1724–34. <https://doi.org/10.3115/v1/d14-1179>.
- [104] Alshehri OS, Alshehri OM, Samma H. Blood Glucose Prediction Using RNN, LSTM, and GRU: A Comparative Study. *2024 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET)* 2024:1–5. https://doi.org/10.1109/IC_ASET61847.2024.10596176.
- [105] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Networks. *Sci Robot* 2014;3:2672–80.

- [106] Yorston C, Chen C, Camelio J. Advancing architectural frameworks for vibration signature classification in rotating machinery. 2024. <https://doi.org/10.1177/09544054241260928>.
- [107] Liu K, Lu Y, Bai R, Xu K, Peng T, Tai Y, et al. Ellipse Detection with Applications of Convolutional Neural Network in Industrial Images. *Electronics* 2023;12:3431. <https://doi.org/10.3390/ELECTRONICS12163431>.
- [108] Eldessouky ME, Darwish AM, Mohamed II, Abdelhalem TH, Khalil AK. Development of Two Robust Data-Driven Models Using Machine Learning and Artificial Neural Network Methods to Predict Wells Fluid Rate in a Challenging Offshore Brown Field. *International Petroleum Technology Conference, IPTC 2024* 2024. <https://doi.org/10.2523/IPTC-23669-MS>.
- [109] VencI A, Svoboda P, Klančnik S, But A, Vorkapić M, Harničárová M, et al. Influence of Al₂O₃ Nanoparticles Addition in ZA-27 Alloy-Based Nanocomposites and Soft Computing Prediction. *Lubricants* 2023;11:24. <https://doi.org/10.3390/LUBRICANTS11010024>.
- [110] Brigato L, Iocchi L. A Close Look at Deep Learning with Small Data. *Proceedings – International Conference on Pattern Recognition 2020*:2490–7. <https://doi.org/10.1109/ICPR48806.2021.9412492>.
- [111] Yuan Q, Zuo Z. Research on the Performance of different Convolutional Neural Network Models on small Datasets. *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE) 2022*:57–62. <https://doi.org/10.1109/MLISE57402.2022.00019>.
- [112] Lee W, Choe YR, Ghosh RS. Recurrent Neural Network and Convolutional Neural Network for Detection of Denial of Service Attack in Microservices. *Proceedings – 22nd IEEE International Conference on Machine Learning and Applications, ICMLA 2023* 2023:1451–6. <https://doi.org/10.1109/ICMLA58977.2023.00219>.
- [113] Meyes R, Lu M, de Puiseau CW, Meisen T. *Ablation Studies in Artificial Neural Networks* 2019.
- [114] Data GWP, Ngu K, Murray DW, Prisacariu VA. Interpolating convolutional neural networks using batch normalization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 2018;11217 LNCS:591–606. https://doi.org/10.1007/978-3-030-01261-8_35/FIGURES/4.
- [115] He K, Zhang X, Ren S, Sun J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Proceedings of the IEEE International Conference on Computer Vision, Vol 498, Institute of Electrical and Electronics Engineers Inc.*; 2015, Pages 1026–34.
- [116] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings*; 2010, Pages 249–56.
- [117] West AHL. Adaptive Back-Propagation in On-Line Learning of Multilayer Networks. 1995:323–9.
- [118] Zhang H, Chen W, Liu T-Y. Train Feedfoward Neural Network with Layer-wise Adaptive Rate via Approximating Back-matching Propagation 2018.
- [119] Srinivasan N, Ravichandran V, Chan KL, Vidhya JR, Ramakirishnan S, Krishnan SM. Exponentiated backpropagation algorithm for multilayer feedforward neural networks. *ICONIP 2002 – Proceedings of the 9th International Conference on Neural Information Processing: Computational Intelligence for the E-Age 2002*;1:327–31. <https://doi.org/10.1109/ICONIP.2002.1202187>.

- [120] Gong B. A novel learning algorithm of back-propagation neural network. Proceedings – 2009 IITA International Conference on Control, Automation and Systems Engineering, CASE 2009 2009:411–4. <https://doi.org/10.1109/CASE.2009.146>.
- [121] Bottou L. Large-Scale Machine Learning with Stochastic Gradient Descent. Proceedings of COMPSTAT 2010 – 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers 2010:177–86. https://doi.org/10.1007/978-3-7908-2604-3_16.
- [122] Kingma DP, Ba JL. Adam: A Method for Stochastic Optimization. 3rd International Conference on Learning Representations, ICLR 2015 – Conference Track Proceedings 2014.
- [123] Duchi J, Hazan E, Singer Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research 2011;12:2121–59.
- [124] Dharrao D, MR A, Mital R, Vengali A, Pangavhane M, Rajput S, et al. An efficient method for disaster tweets classification using gradient-based optimized convolutional neural networks with BERT embeddings. MethodsX 2024;13. <https://doi.org/10.1016/J.MEX.2024.102843>.
- [125] Wu H, Cai X, Feng M. The evaluation of course teaching effect based on improved RBF neural network. Systems and Soft Computing 2024;6. <https://doi.org/10.1016/J.SASC.2024.200085>.
- [126] Sari Y. Performance evaluation of the various training algorithms and network topologies in a neural-Network-Based inverse kinematics solution for robots. Int J Adv Robot Syst 2014;11. https://doi.org/10.5772/58562/ASSET/IMAGES/LARGE/10.5772_58562-FIG13.JPEG.
- [127] Kwok TY, Yeung DY. Bayesian regularization in constructive neural networks. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 1996;1112 LNCS:557–62. https://doi.org/10.1007/3-540-61510-5_95.
- [128] Porto VW, Fogel DB, Fogel LJ. Alternative Neural Network Training Methods. IEEE Expert-Intelligent Systems and Their Applications 1995;10:16–22. <https://doi.org/10.1109/64.393138>.
- [129] Ng AY. Feature selection, L1 vs. L2 regularization, and rotational invariance. International Conference on Machine Learning 2004:615–22. <https://doi.org/10.1145/1015330.1015435>.
- [130] Tibshirani R. Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological) 1996;58:267–88. <https://doi.org/10.1111/J.2517-6161.1996.TB02080.X>.
- [131] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 2014;15:1929–58.
- [132] Behnke M, Guo S, Guo W. Comparison of early stopping neural network and random forest for in-situ quality prediction in laser based additive manufacturing. Procedia Manuf 2021;53:656–63. <https://doi.org/10.1016/J.PROMFG.2021.06.065>.
- [133] Prechelt L. Early Stopping – But When? Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524. Springer, Berlin, Heidelberg 1998:55–69. https://doi.org/10.1007/3-540-49430-8_3.

- [134] Botchkarev A. Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology. *Interdisciplinary Journal of Information, Knowledge, and Management* 2018;14:45–76. <https://doi.org/10.28945/4184>.
- [135] Chai T, Draxler RR. Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. *Geosci Model Dev* 2014;7:1247–50. <https://doi.org/10.5194/GMD-7-1247-2014>.
- [136] Yao H, Wu F, Ke J, Tang X, Jia Y, Lu S, et al. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence* 2018;32:2588–95. <https://doi.org/10.1609/AAAI.V32I1.11836>.
- [137] Gilli M, Maringer D, Schumann E. Numerical analysis in a nutshell. *Numerical Methods and Optimization in Finance* 2019:17–30. <https://doi.org/10.1016/B978-0-12-815065-8.00012-1>.
- [138] Hyndman RJ, Koehler AB. Another look at measures of forecast accuracy. *Int J Forecast* 2006;22:679–88. <https://doi.org/10.1016/J.IJFORECAST.2006.03.001>.
- [139] Hossin M, Sulaiman MN. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process* 2015;5:01–11. <https://doi.org/10.5121/IJDKP.2015.5201>.
- [140] Fürnkranz J, Flach PA. An Analysis of Rule Evaluation Metrics. *Proceedings of the Twentieth International Conference on Machine Learning*, Washington DC 2003.
- [141] Gunakala A, Hussain Shahid A. A Comparative Study on Performance of Basic and Ensemble Classifiers with Various Datasets. *Applied Computer Science* 2023;19:107–32. <https://doi.org/10.35784/ACS-2023-08>.
- [142] Fawcett T. An introduction to ROC analysis. *Pattern Recognit Lett* 2006;27:861–74. <https://doi.org/10.1016/J.PATREC.2005.10.010>.
- [143] Powers DMW, Ailab. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation 2020.
- [144] Martínez Pérez JA, Pérez Martin PS. [ROC curve]. *Semergen* 2023;49. <https://doi.org/10.1016/J.SEMERG.2022.101821>.
- [145] Junge MRJ, Dettori JR. ROC Solid: Receiver Operator Characteristic (ROC) Curves as a Foundation for Better Diagnostic Tests. *Global Spine J* 2018;8:424. <https://doi.org/10.1177/2192568218778294>.
- [146] Sivashanmugam N, Harikrishna KL. Influence of Rare Earth Elements in Magnesium Alloy – A Mini Review. *Materials Science Forum* 2020;979:162–6. <https://doi.org/10.4028/WWW.SCIENTIFIC.NET/MSF.979.162>.
- [147] Ahmad IR, Shu DW. Magnesium Alloys a Promising Option as Protection Materials. *Solid State Phenomena* 2012;185:116–8. <https://doi.org/10.4028/WWW.SCIENTIFIC.NET/SSP.185.116>.
- [148] Wilson CB, Claus KG, Earlam MR, Hillis JE. Magnesium and Magnesium Alloys. *Kirk-Othmer Encyclopedia of Chemical Technology* 2000. <https://doi.org/10.1002/0471238961.1301071423091219.A01>.
- [149] Rusz S, Salajka M, Čížek L, Tylšar S, Kedroň J. Metallographic Analysis of ECAP Processed Selected Magnesium Alloys. *Materials Science Forum* 2014;782:404–7. <https://doi.org/10.4028/WWW.SCIENTIFIC.NET/MSF.782.404>.
- [150] Kojima Y, Aizawa T, Kamado S. Magnesium Alloys 2000 2000;350–351. <https://doi.org/10.4028/B-8BNIZF>.

- [151] Wilson CB, Claus KG, Earlam MR, Hillis JE. Magnesium and Magnesium Alloys. Kirk-Othmer Encyclopedia of Chemical Technology 2000. <https://doi.org/10.1002/0471238961.1301071423091219.A01>.
- [152] Zagórski I, Kulisz M, Kłonica M, Matuszak J. Trochoidal milling and neural networks simulation of magnesium alloys. *Materials* 2019;12. <https://doi.org/10.3390/MA12132070>.
- [153] Korpysa J, Zagórski I, Weremczuk A, Habrat W. Process Stability Analysis during Trochoidal Milling of AZ91D Magnesium Alloy Using Different Toolholder Types. *Applied Sciences* 2024;14:3616. <https://doi.org/10.3390/AP14093616>.
- [154] Ritti L, Bhat T. Design and numerical analysis of tool for FSP simulation of magnesium alloys. *Mater Today Proc* 2021;46:2489–97. <https://doi.org/10.1016/J.MATPR.2021.01.414>.
- [155] Vodenčarević A, Bürring HK, Niggemann O, Maier A. Identifying behavior models for process plants. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2011. <https://doi.org/10.1109/ETFA.2011.6059080>.
- [156] Rivera DE, Bhatnagar S. Closed-loop system identification of restricted complexity models using iterative refinement. *American Control Conference* 1993:1993–4. <https://doi.org/10.23919/ACC.1993.4793226>.
- [157] Flakiewicz W. *Podejmowanie decyzji kierowniczych*. Warszawa: Państwowe Wydawnictwo Ekonomiczne, 1973.
- [158] Sumalatha A, Rao AB. Novel method of system identification. *International Conference on Electrical, Electronics, and Optimization Techniques, ICEEOT* 2016:2323–8. <https://doi.org/10.1109/ICEEOT.2016.7755109>.
- [159] Awrejcewicz J. *Matematyczne modelowanie systemów*. Warszawa: Wydawnictwa Naukowo-Techniczne, 2004.
- [160] Zagórski I, Korpysa J. Surface Quality Assessment after Milling AZ91D Magnesium Alloy Using PCD Tool. *Materials* 2020;13:617. <https://doi.org/10.3390/MA13030617>.
- [161] Zagórski I, Kulisz M, Semeniuk A. Artificial neural network modelling of cutting force components in milling. *ITM Web of Conferences* 2017;15:02001. <https://doi.org/10.1051/ITMCONF/20171502001>.
- [162] Qiao Y, Wang S, Guo P, Yang X, Wang Y. Experimental research on surface roughness of milling medical magnesium alloy. *IOP Conf Ser Mater Sci Eng* 2018;397:012114. <https://doi.org/10.1088/1757-899X/397/1/012114>.
- [163] Salahshoor M, Guo YB. Surface integrity of biodegradable Magnesium–Calcium orthopedic implant by burnishing. *J Mech Behav Biomed Mater* 2011;4:1888–904. <https://doi.org/10.1016/J.JMBBM.2011.06.006>.
- [164] Qiao Y, Wang S, Guo P, Yang X, Wang Y. Experimental research on surface roughness of milling medical magnesium alloy. *IOP Conf Ser Mater Sci Eng* 2018;397:012114. <https://doi.org/10.1088/1757-899X/397/1/012114>.
- [165] Kolluru UK, Narala SKR, Dutta S. Optimization of cutting forces and surface roughness in dry turning of AM magnesium alloy using Taguchi method. *Mater Today Proc* 2021;38:2830–8. <https://doi.org/10.1016/J.MATPR.2020.08.774>.
- [166] Grzesik W. *Podstawy skrawania materiałów konstrukcyjnych*. Wydawnictwo WNT 2018.
- [167] Zaleski K, Matuszak J, Zaleski R. *Metrologia warstwy wierzchniej*. Politechnika Lubelska 2018. <https://doi.org/10.2/JQUERY.MIN.JS>.

- [168] Wieczorowski M, Cellary A, Chajda J. Przewodnik po pomiarach nierówności powierzchni, czyli o chropowatości i nie tylko. Politechnika Poznańska 2003.
- [169] Grzesik W. Wpływ topografii powierzchni na właściwości eksploatacyjne części maszyn*. *Mechanik* 2015;587–93. <https://doi.org/10.17814/MECHANIK.2015.8-9.493>.
- [170] Lu X, Chen Y, Cong C, Wang K, Liang SY. Finite element method based modeling of cutting forces and cutting temperature in micro-milling LF21. 2024. <https://doi.org/10.1177/09544054241261095>.
- [171] Babaev A, Kozlov V, Semenov A, Shevchuk A, Ovcharenko V, Sudarev E. Investigation of cutting forces and machinability during milling of corrosion-resistant powder steel produced by laser metal deposition. *Metal Working and Material Science* 2024;26:38–56. <https://doi.org/10.17212/1994-6309-2024-26.2-38-56>.
- [172] Zaleski K, Matuszak J. Podstawy obróbki ubytkowej. Politechnika Lubelska 2016.
- [173] Skoczylas L, Skoczylas K. Catalog performance of solid carbide endmills for roughing milling. *Technologia i Automatyzacja Montażu (Assembly Techniques and Technologies)* 2018;102:53–7.
- [174] Narasimha Prasad LV, Dokku DB, Talasila SL, Tumuluru P. Hyper Parameter Optimization for Transfer Learning of ShuffleNetV2 with Edge Computing for Casting Defect Detection. *J Sci Ind Res (India)* 2023;82:171–7. <https://doi.org/10.56042/JSIR.V82I2.70250>.
- [175] Gupta R, Anand V, Gupta S, Koundal D. Deep learning model for defect analysis in industry using casting images. *Expert Syst Appl* 2023;232:120758. <https://doi.org/10.1016/J.ESWA.2023.120758>.
- [176] casting product image data for quality inspection n.d. <https://www.kaggle.com/datasets/ravirajsinh45/real-life-industrial-dataset-of-casting-product> (accessed August 9, 2024).
- [177] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016;2016-December:770–8. <https://doi.org/10.1109/CVPR.2016.90>.
- [178] Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions. *Proceedings – 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2016;2017-January:1800–7. <https://doi.org/10.1109/CVPR.2017.195>.
- [179] Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *31st AAAI Conference on Artificial Intelligence, AAAI 2017* 2016:4278–84. <https://doi.org/10.1609/aaai.v31i1.11231>.
- [180] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely Connected Convolutional Networks. *Proceedings – 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2016;2017-January:2261–9. <https://doi.org/10.1109/CVPR.2017.243>.
- [181] How to make Geometric shapes detection? – MATLAB Answers – MATLAB Central n.d. <https://www.mathworks.com/matlabcentral/answers/261721-how-to-make-geometric-shapes-detection> (accessed August 15, 2024).
- [182] Raiipa Technologies. Tempering data for carbon and low alloy steels n.d. <https://doi.org/https://doi.org/10.34740/KAGGLE/DSV/8468279>.
- [183] Zhang Y, Zhao H, Liu P, Wei M, Lu Y, Liu Z. Effect of Tempering Temperature on the Microstructure and Mechanical Properties of 42CrMo4 Steel. *J Phys Conf Ser* 2023;2459:012044. <https://doi.org/10.1088/1742-6596/2459/1/012044>.

- [184] Soares ME, He Q, DePaiva JM, de Freitas BM, Soares P, Veldhuis SC, et al. Mechanical and Tribological Behavior of Nitrided AISI/SAE 4340 Steel Coated with NiP and AlCrN. *Lubricants* 2024;12:181. <https://doi.org/10.3390/LUBRICANTS12050181>.
- [185] Saraç E, Özbek NA. Effect of tempering temperature on mechanical properties and microstructure of AISI 4140 and AISI 4340 tempered steels. *Materials Testing* 2022;64:832–41. <https://doi.org/10.1515/MT-2021-2151>.
- [186] Perla S, Kulkarni S, Balachandran G, Balasubramanian V. Influence of Section Size and Grain Size on the Microstructure Evolution and Mechanical Properties in Steel Grade AISI 5160. *Transactions of the Indian Institute of Metals* 2017;70:2449–58. <https://doi.org/10.1007/S12666-017-1106-X/FIGURES/16>.
- [187] Civi C, Yurddaskal M, Atik E, Celik E. Quenching and tempering of 51CrV4 (SAE-AISI 6150) steel via medium and low frequency induction. *Materialpruefung/ Materials Testing* 2018;60:614–8. <https://doi.org/10.3139/120.111196/MACHINEREADABLECITATION/RIS>.
- [188] Hunkel M. Tempering effects of athermal martensite during quenching and reheating of a SAE 52100 bearing steel. *Materials Science and Engineering: A* 2020;790:139601. <https://doi.org/10.1016/J.MSEA.2020.139601>.
- [189] Krenek J, Kuca K, Blazek P, Krejcar O, Jun D. Application of Artificial Neural Networks in Condition Based Predictive Maintenance. *Studies in Computational Intelligence* 2016;642:75–86. https://doi.org/10.1007/978-3-319-31277-4_7.
- [190] Abidi MH, Mohammed MK, Alkhalefah H. Predictive Maintenance Planning for Industry 4.0 Using Machine Learning for Sustainable Manufacturing. *Sustainability* 2022;14:3387. <https://doi.org/10.3390/SU14063387>.
- [191] Karimova N, Ochilov U, Yakhshiev S, Egamberdiev I. Predictive maintenance of cutting tools using artificial neural networks. *E3S Web of Conferences* 2024;471:02021. <https://doi.org/10.1051/E3SCONF/202447102021>.
- [192] Zhong D, Xia Z, Zhu Y, Duan J. Overview of predictive maintenance based on digital twin technology. *Heliyon* 2023;9:e14534. <https://doi.org/10.1016/J.HELIYON.2023.E14534>.
- [193] Aboshosha A, Haggag A, George N, Hamad HA. IoT-based data-driven predictive maintenance relying on fuzzy system and artificial neural networks. *Scientific Reports* 2023;13:1–13. <https://doi.org/10.1038/s41598-023-38887-z>.
- [194] Fordal JM, Schjølberg P, Helgetun H, Skjermo TØ, Wang Y, Wang C. Application of sensor data based predictive maintenance and artificial neural networks to enable Industry 4.0. *Adv Manuf* 2023;11:248–63. <https://doi.org/10.1007/S40436-022-00433-X/FIGURES/10>.
- [195] Capizzi G, Lo Sciuto G, Wasef Shaheen B, Németh I. Performance Monitoring of Wind Turbines Gearbox Utilising Artificial Neural Networks — Steps toward Successful Implementation of Predictive Maintenance Strategy. *Processes* 2023;11:269. <https://doi.org/10.3390/PR11010269>.
- [196] Pech M, Vrchota J, Bednář J. Predictive Maintenance and Intelligent Sensors in Smart Factory: Review. *Sensors* 2021;21:1470. <https://doi.org/10.3390/S21041470>.
- [197] Gama J, Ribeiro RP, Veloso B. Data-Driven Predictive Maintenance. *IEEE Intell Syst* 2022;37:27–9. <https://doi.org/10.1109/MIS.2022.3167561>.
- [198] AI4I 2020 Predictive Maintenance Dataset. UCI Machine Learning Repository 2020. <https://doi.org/10.24432/C5HS5C>.

