AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

INSTYTUT INFORMATYKI

PRACA DYPLOMOWA MAGISTERSKA

Ocena jakości i parametrów mechanicznych odlewów przy użyciu metod uczenia maszynowego

Casting quality and mechanical parameters prediction using machine learning methods

Autor: *Wiktor Reczek*
Kierunek studiów: *Informatyka*
Typ studiów: *Stacjonarne*
Opiekun pracy: *prof. dr hab. Bartłomiej Śnieżyński*

Kraków, 2021

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Jednocześnie Uczelnia informuje, że zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych Uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli Uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony, autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto Uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. – Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem systemu antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych.

Serdecznie dziękuję opiekunowi mojej pracy magisterskiej prof. dr. hab. Bartłomiejowi Śnieżyńskiemu za poświęcony mi czas oraz za cenne uwagi.

Spis treści

Streszczenie pracy

Celem niniejszej pracy jest zbadanie skuteczności różnych algorytmów klasyfikacji i ich rozszerzeń w ocenie jakości odlewów. Pomocne w realizacji projektu może być opracowanie oprogramowania, bądź skorzystanie z gotowego rozwiązania, pozwalającego na kompleksowe przebadanie wszystkich zaimplementowanych algorytmów. Korzystając z tej aplikacji, zostaną przeprowadzone eksperymenty, w których przygotowane algorytmy i metody uczenia maszynowego zostaną przetestowane w różnych warunkach i konfiguracjach, a także dla różnych danych wejściowych, którymi są zdjęcia przekrojów odlewów (zdjęcia mikrostruktury) lub informacje na temat materiału (np. typ, skład).

Praca ma charakter badawczy, gdyż jak wykazał przegląd literatury, prac naukowych na ten temat (tj. pod kątem wykorzystania uczenia maszynowego w celu oceny jakości odlewów) oraz źródeł wskazujących na praktyczne stosowanie oprogramowania o podobnym przeznaczeniu jest niewiele. Dlatego też skuteczność algorytmów uczenia maszynowego w ocenie jakości odlewów zostanie przetestowana z użyciem najbardziej uniwersalnych metod. Wyniki uzyskane przez klasyczne metody uczenia maszynowego oraz przez sieci neuronowe zostaną ze sobą porównane, biorąc pod uwagę takie aspekty, jak interpretowalność rezultatów, łatwość implementacji modelu, prostotę algorytmu, czy czas uczenia.

Abstract of master's thesis

The aim of this research is to see how efficient various categorization algorithms and extensions are at determining casting quality. The development of software or the use of a ready-made solution that allows for extensive testing of all developed algorithms could be beneficial to the project's implementation. Experiments will be conducted using this application, in which the developed algorithms and machine learning methods will be tested in a variety of situations and configurations, as well as for a variety of input data, such as images of casting sections (pictures of microstructure) or material information (e.g., type, composition).

Because there are few scholarly articles on the issue (i.e., using machine learning to assess the quality of castings) and few sources suggesting the actual usage of software for comparable reasons, the work is of a research character. As a consequence, the capabilities of utilizing machine learning to assess the quality of castings will be explored using the most general approaches. The results obtained by classical machine learning methods and by neural networks will be compared with each other, taking into account aspects such as the interpretability of results, ease of model implementation, algorithm simplicity, and learning time.

1. Wstęp

1.1. Wprowadzenie

Człowiek od zawsze starał się maksymalnie upraszczać swoje życie. W czasach pradawnych wiązało się to z konstrukcją coraz to bardziej skomplikowanych przyrządów, początkowo prymitywnych technicznie. W miarę postępu człowiek był już w stanie opracowywać bardziej zaawansowane narzędzia. Obecnie, na bardzo długiej osi rozwoju ludzkości znajdujemy się w miejscu, gdzie większość postępu jest związana z odkryciami naukowymi w takich dziedzinach, jak fizyka, chemia, biologia czy informatyka. W tej ostatniej szczególnie dużo się dzieje, a to za sprawą m.in. uczenia maszynowego, czy szerzej, sztucznej inteligencji (SI). Wbrew pozorom nie jest to dziedzina całkiem nowa, gdyż jej początki sięgają lat 50. XX wieku, natomiast znaczące przyspieszenie rozwoju w tej dziedzinie nastąpiło dopiero w ostatnich kilkunastu latach, a to ze względu na coraz większą ilość produkowanych danych oraz możliwość ich szybkiego przetworzenia (szybsze procesory). Aktualnie postęp w sztucznej inteligencji jest na takim etapie, że wiele zadań przez nią wykonywanych jest lepiej, niż przez ludzi (tzw. osiągnięcia nadludzkie); wiele zadań jest też wykonywanych na poziomie mistrzowskim. Dlatego panuje obecnie trend, aby jak najwięcej czynności zautomatyzować, czy też wykonywać za pomocą sztucznej inteligencji. Stąd zapewne pojawił się pomysł, aby wykorzystać ją do kolejnego zadania, jakim jest klasyfikacja jakości odlewów.

Zdjęcia mikrostruktury metali dla osoby bez specjalistycznej wiedzy wyglądają niemal identycznie. Obecnie wykorzystuje się m.in. badania niszczące w celu określenia parametrów mechanicznych odlewów. Rozwiązanie przedstawione w tej pracy pozwoliłoby zaoszczędzić środki za zużyte materiały, a także czas potrzebny na „ręczne” zidentyfikowanie jakości odlewu.

1.2. Cel i zakres pracy

Celem niniejszej pracy jest pokazanie, że za pomocą metod uczenia maszynowego można skutecznie badać jakość otrzymywanych w produkcji odlewów za pomocą analizy zdjęć mikrostruktury oraz wartości parametrów mechanicznych odlewów.

Aby osiągnąć ten rezultat, wykonano przegląd dostępnej literatury w celu rozeznania się, w jaki sposób podchodzi się do zagadnienia oceny jakości odlewów za pomocą uczenia maszynowego. Jak się

niestety okazało, nie ma zbyt wielu dostępnych źródeł, które w pełni spełniałyby założenia tego projektu. Dlatego po przeglądzie zebrano odpowiednie dane uczące oraz przeprowadzono na nich proces wykrywania i korygowania błędnych instancji. Następnym krokiem było już zastosowanie najbardziej uniwersalnych algorytmów uczenia maszynowego w celu zdiagnozowania, które z nich mają największą skuteczność dla tak postawionego problemu. Ostatnim krokiem było sformułowanie wniosków oraz wskazanie najbardziej skutecznego podejścia w celu oceny jakości odlewów.

1.3. Zawartość pracy

Rozdział 2 zawiera przegląd prac naukowych, które wpłynęły na wybór metod w tej pracy. Zostały w nim przedstawione dosyć szeroko podejścia stosowane na przestrzeni lat w celu oceny jakości odlewów i innych zadaniach blisko powiązanych.

W rozdziale 3 została przedstawiona charakterystyka uczenia maszynowego oraz sieci neuronowych, różne typy reprezentacji wiedzy, metody uczenia się i wiele innych aspektów powiązanych z tą tematyką.

W rozdziale 4 pokazano, w jaki sposób uzyskano dane oraz jak były one przekształcane, aby wydobyć z nich jak najwięcej informacji.

Wszystkie przeprowadzone testy i badania wraz z wynikami zostały omówione w rozdziale 5.

W ostatnim rozdziale **??** zostały zawarte podsumowania badań, wnioski, jakie można było z nich wyciągnąć. Zostały również zaproponowane dalsze wymagane prace w celu udoskonalenia otrzymanych wyników.

2. Stan badań

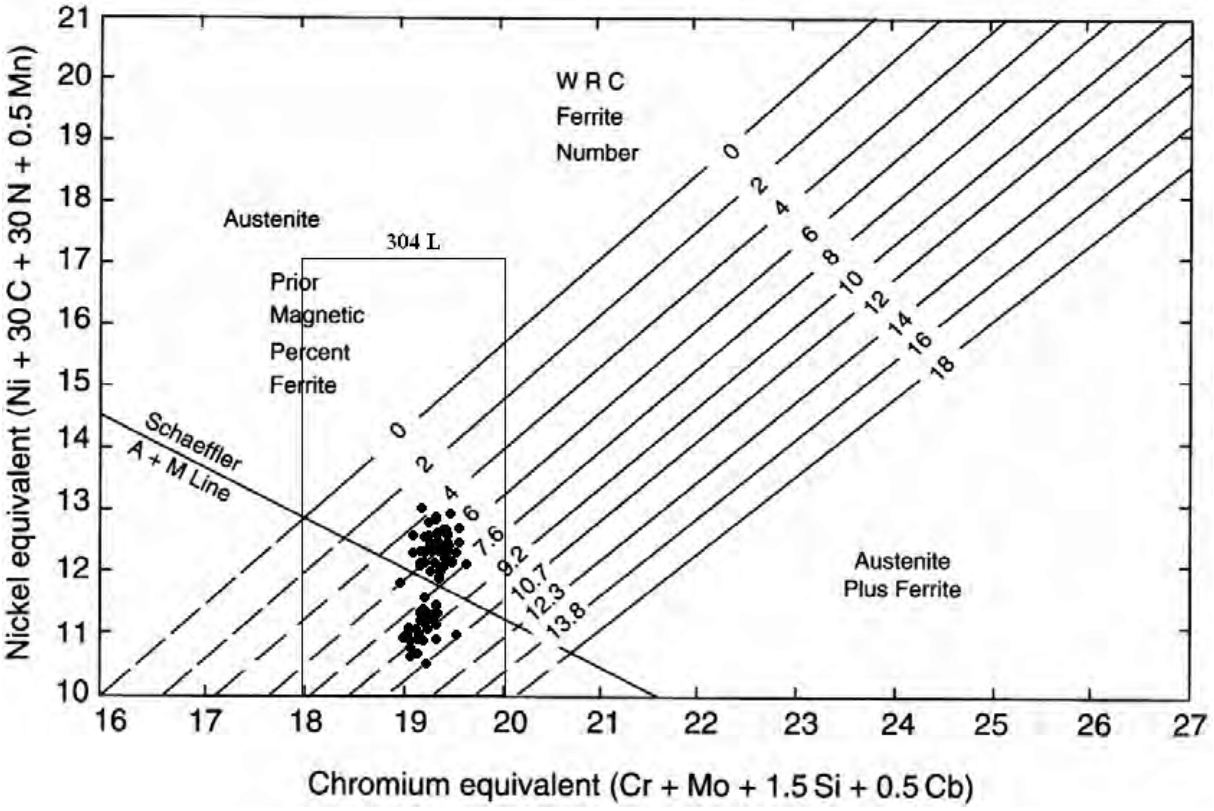
2.1. Powiązane prace – zarys historyczny

2.1.1. Predykcja wartości liczby ferrytowej

Przewidywanie własności mechanicznych produktów odlewniczych za pomocą metod uczenia maszynowego ma niemal tak długą historię, jak sama dziedzina uczenia maszynowego. Już w jednej z pierwszych prac naukowych na ten temat [1] mówiono o tym, że przyszłe techniki predykcyjne będą opierać się raczej na wyrażeniach matematycznych czy sztucznej inteligencji niż na diagramach. Ówcześnie do przewidywania fazy mikrostruktury spoiny metalu wykorzystywano właśnie diagramy (2.1), lecz jak łatwo się domyślić, jest to podejście mało praktyczne. Stąd nacisk na wykorzystywanie jak najbardziej wszechstronnych wyrażeń ilościowych do przewidywania mikrostruktury jako funkcji m.in. składu.

Następnym krokiem było opracowanie modelu półempirycznego w celu powiązania składu metalu spoiny z liczbą ferrytową (zwaną dalej FN), czyli miarą oznaczania zawartości ferrytu w stali nierdzewnej. Dlaczego jest to istotne? Jak wskazano w pracy [2] wielkość FN określa właściwości metalu takie, jak wytrzymałość, twardość, odporność na korozję i inne (jej poziom powinien wynosić $3 - 7\%$, ponieważ niski poziom ferrytu może prowadzić do pęknięć [3, 4], z drugiej strony wysoki poziom ferrytu prowadzi do niższej odporności na korozję [4]).

W pracy Babu i in. [5] przedstawiono wyniki aproksymacji punktowej, która dopasowuje model do prognoz zgodnych z obserwacjami eksperymentalnymi. Stwierdzono w niej, iż ogólna dokładność badanego modelu jest porównywalna do tej z diagramu WRC-1992 (czyli najnowocześniejszej ówcześnie metody – przyp. aut.), który powstał w ten sposób, iż skład stopu jest konwertowany do dwóch czynników – ekwiwalentu chromu (Cr_{eq}, wzór 2.1) oraz ekwiwalentu niklu (Ni_{eq}, wzór 2.2). Wynika to z tego, iż ten pierwszy zawiera elementy, które wpływają na mikrostrukturę w ten sam sposób jak chrom (tj. stabilizatory ferrytu), natomiast ten drugi zawiera elementy, które wpływają na mikrostrukturę w ten sam sposób, jak nikiel (tj. stabilizatory austenitu). Następnie z diagramu można odczytać poziom ferrytu, który jest przedstawiony jako funkcja od ekwiwalentów chromu i niklu. Jak stwierdzają autorzy [5], zaletą aproksymacji punktowej w porównaniu z diagramem WRC-1992 jest jego zdolność do uwzględniania wpływu innych pierwiastków stopowych oraz łatwość ekstrapolacji do wyższych wartości Cr_{eq} i Ni_{eq} (WRC-1992 jest pod tym względem mocno ograniczone, co widać na rysunku 2.2).



Rys. 2.1. Przykładowy schemat do wyznaczania ferrytu δ z granicami składu (DeLong, 1973)

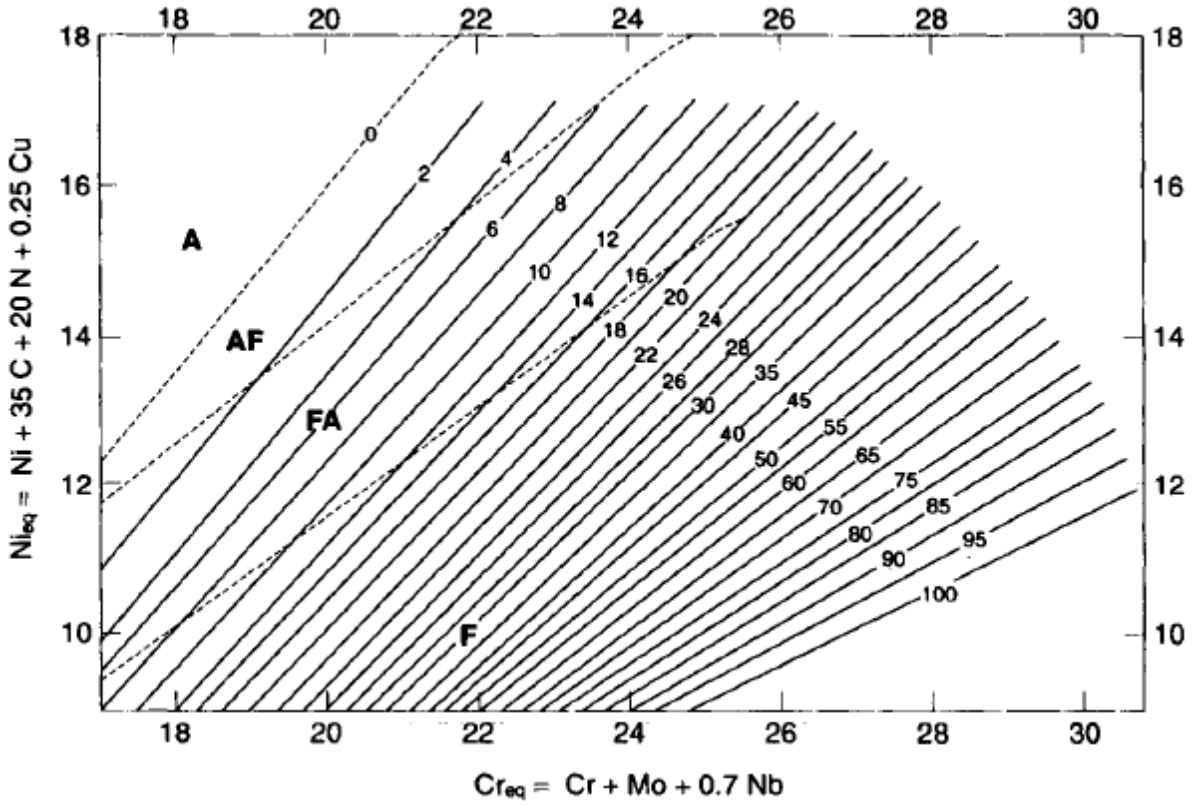
Równania na Cr_{eq} i Ni_{eq} są następujące:

$$Cr_{eq} = Cr + Mo + 0.7Nb \tag{2.1}$$

$$Ni_{eq} = Ni + 35C + 20N + 0.25Cu \tag{2.2}$$

gdzie symbole pierwiastków przedstawiają procent wagi każdego pierwiastka. W kolejnej pracy [2] dotyczącej predykcji FN ci sami autorzy wykorzystali sieć neuronową, która na wejściu przyjmowała procent wagowy 13 pierwiastków ($Fe, Cr, Ni, C, N, Mo, Mn, Si, Cu, Ti, Nb, V$ i Co), czyli warstwa z 13 neuronami, następnie warstwa ukryta z sześcioma neuronami, natomiast na wyjściu był pojedynczy neuron, który zwracał liczbę ferrytową (rys. 2.3).

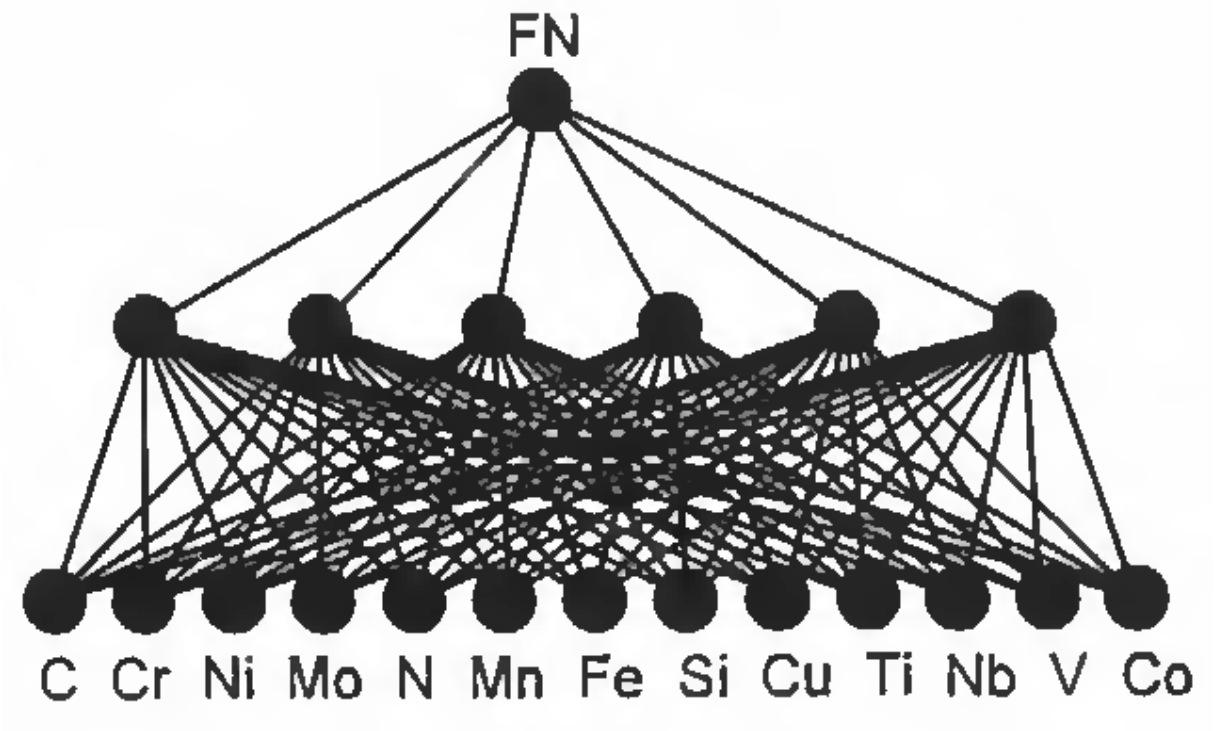
Wyniki zostały przedstawione w [6] i jak się okazało, testowana sieć zwracała lepsze wyniki od jakichkolwiek dotychczasowych podejść z błędem RMS mniejszym o 50% od poprzedniej najlepszej metody. W ostatnim przytoczonym artykule dotyczącym predykcji FN [7] również zastosowano sieć neuronową, a konkretnie bayesowską sieć neuronową (BNN). Na wejściu mamy taką samą warstwę, jak w poprzedniej pracy, natomiast tutaj jest więcej neuronów w warstwie ukrytej. Poprawa wyników względem poprzedniej pracy wynosi około 15% (błąd RMS) testując na niezależnym zbiorze danych nieużywanym w szkoleniu.



Rys. 2.2. Diagram WRC-1992 (Kotecki & Siewert, 1992)

2.1.2. Predykcja własności mechanicznych odlewów

Początki rozwoju i przetwarzania materiałów nie były łatwe. Mimo wielu przeprowadzonych badań naukowych nad materiałami wciąż pozostaje wiele problemów, w przypadku których brakuje metod ilościowych. Tyczy się to głównie predykcji takich parametrów konstrukcyjnych jak wytrzymałość na rozciąganie, trwałość, twardość itp. Pierwsze badania dotyczące własności mechanicznych odlewów przyjmujące podejście ilościowe brały pod uwagę szczegółowy skład chemiczny oraz takie parametry jak rekrystalizacja, proces starzenia, zakres pracy na zimno, temperatura badania czy szybkość odkształcania [8, 9]. W obydwu tych pracach zastosowano sieci neuronowe oraz uzasadniono, że modele matematyczne sobie nie radzą z wymienionymi wyżej parametrami. Standardowo zastosowano sieć, w której na wejściu podawano procent wagowy pierwiastków w badanym materiale. Jedną z głównych własności mechanicznych jest wytrzymałość na rozciąganie (ang. *ultimate tensile strength*, UTS), która jest badana od wielu lat. Po fazie odlewu inżynierowie wykorzystują w swoich obliczeniach tę i inne wartości w celu obliczenia odkształcania się, funkcji przyłożonego obciążenia, czasu i wiele innych. Jest ona jednym z ważniejszych czynników do uwzględnienia, gdyż niewystarczająca wartość wytrzymałości może mieć fatalne skutki (jak np. zawalenie się konstrukcji). Innym powodem może być to, iż jedynym sposobem, aby zbadać wartość tej wytrzymałości trzeba przeprowadzić badania niszczące, co powoduje wzrost kosztów produkcji [10]. Jednym ze sposobów analizy wartości UTS jest predykcja za pomocą wartości różnych właściwości odlewu. W przywołanej wcześniej pracy [10] oraz [11] są to



Rys. 2.3. Model sieci neuronowej ORFN (Vitek, 2003)

skład chemiczny, rozmiar odlewu, prędkość chłodzenia, obróbka termiczna. Mając dane w postaci wartości rozdzielone przecinkiem (CSV, od ang. *comma-separated values*) można skorzystać z klasycznych metod klasyfikacji statystycznej, jak klasyfikacja liniowa, k najbliższych sąsiadów, drzewa decyzyjne, czy sieci bayesowskie [12]. W przytoczonej pracy skupiono się na sieciach bayesowskich. Wyniki są optymistyczne: dokładność na poziomie ponad 82% oraz błędy MAE (średni błąd bezwzględny, od ang. *mean absolute error*) oraz średni błąd kwadratowy (MSE, od ang. *mean square error*) na poziomie odpowiednio 0.2 oraz 0.35. Inne przetestowane metody w tym artykule to k najbliższych sąsiadów (KNN, od ang. *k-Nearest Neighbors*) oraz sztuczne sieci neuronowe (ANN, od ang. *artificial neural networks*), które osiągnęły podobne, aczkolwiek nieco gorsze wyniki. Wytrzymałość na rozciąganie można przewidywać na podstawie dwóch różnych źródeł danych wejściowych [13]:

– skład chemiczny metalu oraz zmienne procesu walcowania, jak temperatura, czy przebieg;

– dane na temat mikrostruktury.

W pracy tej jako dane wejściowe użyto skład chemiczny, specyfikacje geometryczne oraz zmienne dotyczące procesu rolowania, natomiast modelem wykorzystanym do predykcji wartości wytrzymałości na rozciąganie była ponownie bayesowska sieć neuronowa. Sieć ta składała się z jednej warstwy ukrytej, która z kolei składała się z siedmiu neuronów. Jak stwierdzają autorzy, sieć BNN lepiej się sprawdza w tym celu od tradycyjnych sieci neuronowych, a to ze względu na wyższą odporność na nadmierne dopasowanie danych (ang. *overfitting*), szczególnie w przypadku, gdy ilość danych jest znacznie ograniczona i nie mamy możliwości zgromadzenia dużej ilości wysokiej jakości danych.

Bardzo podobne podejście zastosowano w pracy [14], gdzie użyto sieci neuronowej, a na jej wejściu podawano 20 zmiennych, takich jak skład chemiczny, warunki obróbki cieplnej czy temperatura badania. W ten sposób uzyskano 93% wartości współczynnika R-kwadrat dla granicy plastycznoci (YS, od ang. *yield strength*) oraz UTS.

2.1.3. Predykcja błędów w produkcji odlewniczej

Innym, ciekawym podejściem jest to zaprezentowane w pracy Penya et al. [15], a mianowicie ponownie zostały wykorzystane bayesowskie sieci neuronowe, lecz tym razem w celu predykcji obecności mikrouszkodzeń w odlewie przed lub w trakcie procesu odlewnictwa. Dzięki takim danym można wpłynąć na proces odlewniczy w taki sposób, aby zredukować liczbę defektów. W sieci BNN wykorzystano na wejściu osiem zmiennych, takich jak cechy geometryczne (dwie zmienne), jakość metalurgiczna (trzy zmienne), jakość formy (jedna zmienna) oraz dwie zmienne dotyczące samego procesu. Wyniki są obiecujące, gdyż dzięki temu podejściu udało się zredukować liczbę defektów z 5% do 0.075% w przypadku pierwszej odlewni, oraz z 4.7% do 0.19% w przypadku drugiej odlewni.

2.2. Powiązane prace – stan aktualny

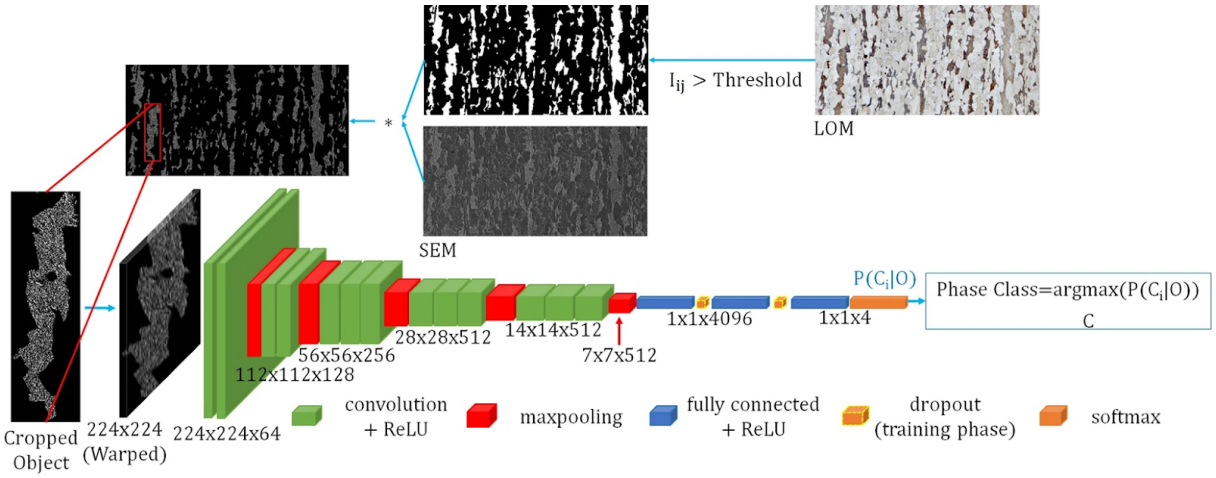
2.2.1. Predykcja cech i jakości odlewów za pomocą metod rozpoznawania obrazów

Wewnętrzna struktura materiału nazywana jest mikrostrukturą. Określa ona wszystkie jego właściwości fizyczne i chemiczne. Pomimo tego, że charakterystyka mikrostrukturalna jest szeroko rozpowszechniona i dobrze znana, klasyfikacja mikrostrukturalna jest zazwyczaj przeprowadzana „ręcznie" przez ekspertów. Ze względu na złożoność mikrostruktur (mogą się one składać z podstruktur) ich klasyfikacja jest bardzo trudna i dotychczas nie powstało zbyt wiele prac naukowych podejmujących się próby klasyfikacji mikrostruktur. Wcześniejsze artykuły zazwyczaj oddzielały fazę klasyfikacji mikrostruktur od fazy ekstrakcji cech [16]. Dzięki postępowi metod głębokiego uczenia (ang. *deep learning*, DL) otworzyły się nowe możliwości. Wiele metod głębokiego uczenia w różnych zadaniach zwraca najlepsze wyniki, jakie udało się dotychczas uzyskać, dlatego warto się nad tymi metodami pochylić.

W pierwszej przytoczonej tutaj pracy opracowano nową zautomatyzowaną metodę wykorzystującą dyfrakcję wstecznie rozproszonych elektronów (EBSD, od ang. *electron backscatter diffraction*), aby skutecznie identyfikować i określać ilościowo mikroskładniki ferrytowe w złożonych mikrostrukturach różnych gatunków stali [17]. Identyfikowano rodzaj ferrytu dla każdego ziarna, co wiązało się z powiązaną z nimi wielkością ziaren. Różne odmiany ferrytów mają różne profile dezorientacji na granicach ziaren, aczkolwiek jest ona zależna od kąta, toteż zostało to wykorzystane w badaniach.

Z kolei w pracy [18] zastosowano korelacyjne podejście oparte na EBSD i mikroskopii optyczno-świetlnej (LOM, od ang. *light-optical microscopy*), zamiast niezależnie korzystać z powszechnych metod. Wykorzystując rozkład orientacji ziarna w EBSD, ręcznie ustalono progi przy pomocy próbki referencyjnej tej samej płytki z mikrostrukturą. Podobnie, w przypadku LOM, próg był ręcznie ustalany i mógł być zweryfikowany krzyżowo.

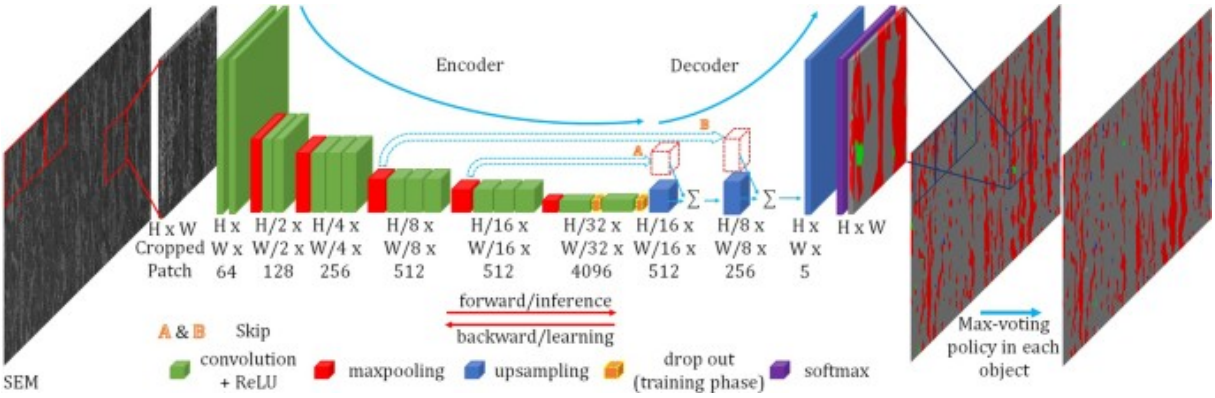
Kolejna przeanalizowana praca [16] dotyczy klasyfikacji mikrostrukturalnej składników stali niskowęglowej za pomocą metod głębokiego uczenia. Takie podejście może być pomocne przy ocenie jakości danego odlewu, jak i predykcji jego właściwości mechanicznych. W artykule wykorzystano w pełni splotową sieć neuronową (ang. *Fully Convolutional Neural Network*, FCNN), na której wejście podawano segmentowane piksele. System ten osiągnął niemal 94% skuteczność, tym samym przewyższając dotychczasową najlepszą metodę. Jak stwierdzają autorzy, nie tylko wynik jest doskonały sam w sobie, ale również wyznacza linię dla przyszłych badań, m.in. związanymi z oceną jakości stali (i nie tylko). W pracy tej, oprócz wspomnianego modelu, przetestowano również szereg innych metod oraz podejść. Jednym z zastosowanych podziałów klasyfikacji jest klasyfikacja mikrostruktur oparta na obiekcie (ang. *object-based microstructural classification*). Polega ona na wycinaniu obiektów z obrazów, a następnie klasyfikacji wyciętych kształtów do jednej z kilku klas (ferryt, cementyt, austenit, perlit, bainit, martenzyt), jak pokazano na rysunku 2.4.



Rys. 2.4. Klasyfikacja oparta na obiektach przy użyciu CNN. Obiekty są wycinane z obrazów a następnie klasyfikowane przez wytrenowane CNN (VGG16). Rozmiar wejściowy jest z góry określony (Azimi et al., 2018)

Jednakże, jak zauważono w pracy, podejście to ma jedną wielką wadę, a mianowicie konieczna jest zmiana rozmiarów wycinanych kształtów tak, aby odpowiadała wejściu sieci neuronowej, które jest z góry ustalone (224 na 224 pikseli). W ten sposób możemy zniszczyć cenne dane związane np. z teksturą. Po przeprowadzeniu testów, przy użyciu tego podejścia uzyskano wynik na poziomie 49% dokładności (który był dotychczas najlepszy). Drugie podejście, jakie zostało przetestowane w [16] polega na klasyfikacji pod względem pikseli (rys. 2.5). Na wejście sieci są podawane obrazki, które otrzymujemy

poprzez wycinanie kawałków oryginalnych obrazów metodą przesuwnego okna. Wykonujemy tę operację tyle razy, aby pokryć cały obraz wejściowy. Wyjściem takiej sieci jest macierz 3D z liczbą kanałów równą liczbie klas. Każdy piksel tej macierzy posiada wartość reprezentującą ufność (bądź prawdopodobieństwo) przynależności do danej klasy mikrostruktury. Następnie przeprowadzany jest etap klasyfikacji według pikseli, wybierając klasę z najwyższym prawdopodobieństwem (pewnością, ufnością) dla każdego piksela. Następnie wszystkie segmenty należące do oryginalnego obrazu wejściowego są scalane razem (rys. 2.5). Wtedy do każdego obiektu stosowana jest zasada maksymalnego głosowania i przypisywana jest mu klasa, jaką posiada większość pikseli (tzn. obiektowi jest przypisywana klasa mikrostruktury z maksymalną liczbą sklasyfikowanych pikseli wewnątrz obiektu). W pracy zastosowano również równoważenie klas oraz rozszerzenie danych (ang. *data augmentation*). Równoważenie klas było konieczne, ponieważ występowały duże rozbieżności pomiędzy liczebnością przykładów w różnych klasach, natomiast oryginalne było samo podejście do tego tematu, a mianowicie w zależności od liczności klasy stosowano różne rozmiary kroku (ang. *stride*), tak, aby uzyskać mniejszą lub większą liczbę wyciętych przykładów. Natomiast rozszerzenie danych odbyło się poprzez rotację obrazków o 90°, 180° oraz 270°, dzięki czemu zbiór obrazków został rozszerzony aż czterokrotnie. Pomimo iż sama augmentacja nie miała znaczącego wpływu na wyniki, to dzięki niej uzyskano wzrost o 2 punkty procentowe. Dzięki takiemu podejściu udało się poprawić najlepszy dotychczas wynik aż o 45 punktów



Rys. 2.5. Klasyfikacja mikrostrukturalna oparta na segmentacji z maksymalną liczbą głosów. Obraz wejściowy jest przycinany, wycinki są przekazywane do sieci FCNN. Następnie segmentowane wycinki są zszywane razem. W ostatnim kroku stosowane jest głosowanie do wynikowego zszytego obrazu (Azimi et al., 2018)

procentowych (sic!). Wyniki wszystkich przetestowanych metod zostały przedstawione w tab. 2.1.

Kolejną pracą, z obiecującymi wynikami, która również korzysta z uczenia głębokiego, jest [20]. Celem owej pracy jest segmentacja struktur w stali o złożonej fazie. Autorzy zastosowali w niej sieci Vanilla U-Net oraz U-Net VGG16, uzyskując najlepsze wyniki skuteczności odpowiednio 91.6% oraz 90.6%. Rezultaty te otrzymano dla danych w postaci obrazów uzyskanych metodą LOM. Wykorzystano również dane w postaci obrazów otrzymywanych metodą SEM, lecz skuteczności były mniejsze o około 10 punktów procentowych.

Tabela 2.1. Wyniki klasyfikacji mikrostrukturalnej przy użyciu metod opartych na obiektach oraz opartych na pikselach (Azimi et al., 2018)

Metoda	**Typ**	**Strategia treningu**	**Dokładność**
Pauly et al. [19]	oparte na obiektach	—	48.89%
CIFAR-Net	oparte na obiektach	od zera (ang. *from scratch*)	57.03%
VGG19 + SVM	oparte na obiektach	—	64.84%
VGG16	oparte na obiektach	strojenie (ang. *fine tuning*)	66.50%
MVFCNN[a]	oparte na pikselach	strojenie (ang. *fine tuning*	**93.94%**

[a] Metoda MVFCNN to metoda badana w przytoczonym artykule [16]. Wynik ten został osiągnięty na obrazkach otrzymanych metodą skaningowej mikroskopii elektronowej (SEM, od ang. *scanning-electron microscopy*), natomiast na obrazkach metodą LOM osiągnięto wynik zaledwie około 70%…

2.3. Wnioski

Jak widzimy, dziedzina, jaką jest inżynieria materiałowa, mimo iż rozwijana już od wielu lat, nadal wymaga wiele pracy, która usprawni proces wytwarzania materiałów, a także zaoszczędzi środki, które obecnie są marnowane na badania niszczące. Właściwości materiału, takie jak wytrzymałość, wiązkość, twardość, kruchość lub ciągliwość, są istotne przy kategoryzacji materiału lub komponentu według ich jakości, lecz testy te są zwyczajowo drogie. Dlatego metody uczenia maszynowego są uznawane za pomocne w przewidywaniu właściwości odlewów [21]. Z drugiej strony problemem może być wciąż niewystarczająca ilość danych uczących – co można zauważyć po tym, iż nie udało się znaleźć żadnych publicznie udostępnionych danych w postaci zdjęć, które można byłoby dołączyć do własnego zbioru danych. Z tego też względu przeprowadzone testy będą się opierały głównie na klasyfikacji metalu ze względu na wysoką bądź niską odporność na rozciąganie, a także ze względu na wysoką bądź niską granicę sprężystości, gdyż takimi właśnie danymi dysponujemy.

Początkowo w celu predykcji właściwości mechanicznych korzystano z dosyć prymitywnych rozwiązań jak np. odczytywanie z wykresu właściwości za pomocą samego składu chemicznego. Następnie, po wielu latach rozwoju w dziedzinie materiałoznawstwa jak i informatyki i uczenia maszynowego zaczęto korzystać z takich zmiennych, jak skład chemiczny, wielkość odlewu, prędkość chłodzenia czy proces obróbki termicznej wykorzystując takie modele, jak sieci bayesowskie, algorytm k-nn czy sieci neuronowe. Z czasem sieci neuronowe zaczęły osiągać najlepsze wyniki, w tym momencie znacznie zostawiając w tyle pozostałe metody. A więc co do wykrywania właściwości mając wspomniane wcześniej dane, sieci neuronowe zostały dosyć dobrze przetestowane i dają gwarantowane, wysokie wyniki. Co natomiast z danymi w postaci obrazów? Tutaj widzimy, że powoli również są adaptowane sieci neuronowe, a w szczególności głębokie sieci neuronowe [16, 19], które dają nadzwyczaj obiecujące wyniki.

Natomiast prace te były związane z wykrywaniem pojedynczych struktur na obrazkach, a nie z predykcją samych właściwości mechanicznych. Wykorzystano w nich tzw. semantyczna segmentacja obrazu, która wydaje się najbardziej optymistyczna, aczkolwiek nie posiadamy takich danych. Jednak jak najbardziej można to uznać za obiecujący kierunek poszukiwań w celu zwiększenia skuteczności sieci wykrywających i klasyfikujących struktury na obrazkach.

Podsumowując, sieci neuronowe udowodniły swoją przydatność w zakresie predykcji zawartości ferrytu w składzie chemicznym spoiny czy predykcji właściwości mechanicznych za pomocą składu chemicznego i danych dotyczących samego procesu wyrobu takiego materiału. Znaczące postępy też są widoczne w obszarze klasyfikacji struktur na obrazie. Jednak nadal brakuje badań w obszarze predykcji właściwości mechanicznych z obrazów, dlatego autor niniejszej pracy podjął się próby przeprowadzenia takich testów i przeanalizowania możliwości sieci neuronowych (w szczególności głębokich sieci neuronowych) w tym zakresie. Dysponując oznakowanymi danymi w postaci obrazków, które zostały podzielone na dwa podzbiory – ze względu na odporność na rozciąganie oraz ze względu na granicę sprężystości, warto przetestować możliwości predykcyjne sieci neuronowych dla tak postawionego problemu. Dodatkowo można skorzystać z pomysłów (przedstawionych w powyższych podrozdziałach) dotyczących klasyfikacji struktur na obrazach, aby mieć dodatkowe informacje, które być może będą pomocne w docelowej klasyfikacji. Oprócz tego, jako inne źródła informacji można z obrazów wyciągać momenty, które zostały opracowane wiele lat temu i dotychczas były wykorzystywane z sukcesami. Ponadto, głównie jako odniesienie, można wykorzystać klasyczne klasyfikatory, typu las losowy, maszyna wektorów nośnych, czy inne. Oczekuje się wyników na poziomie zbliżonym do wyników przedstawionych w powyższych pracach, które korzystały z sieci neuronowych dla danych zawierających m.in. skład chemiczny.

3. Uczenie maszynowe

Uczenie maszynowe (ML, od ang. *machine learning*) to nauka o algorytmach komputerowych, które automatycznie ulepszają się dzięki doświadczeniu i wykorzystaniu danych [22]. Algorytmy uczenia maszynowego są budowane na podstawie przykładowych danych, zwanych „danymi szkoleniowymi", w celu prognozowania lub podejmowania decyzji bez bezpośredniego zaprogramowania do tego *Koza96*. Natomiast nieco bardziej techniczną definicję uczenia maszynowego przedstawił Tom Mitchell w 1997 roku: *Mówimy, że program komputerowy uczy się na podstawie doświadczenia E w odniesieniu do jakiegoś zadania T i pewnej miary wydajności P, jeśli jego wydajność (mierzona przez P) wobec zadania T wzrasta wraz z nabywaniem doświadczenia E.*

Dyscyplina uczenia maszynowego wykorzystuje różne podejścia do uczenia modeli wykonywania zadań, w przypadku których nie jest dostępny w pełni zadowalający algorytm. Wstęp do uczenia maszynowego wraz z wyjaśnieniem pojęć podstawowych został przedstawiony w rozdziale 3.1. Główne nurty zostały zaprezentowane w rozdziale 3.2. Natomiast w kolejnych podrozdziałach zostały zaprezentowane metody dotyczące przygotowannia danych. Ostatnie podrozdziały dotyczą metod uczenia maszynowego, które zostały wykorzystane w trakcie badań.

3.1. Pojęcia podstawowe

Jednym z podstawowych terminów w uczeniu maszynowym jest **atrybut** (ang. *attribute*). Oznacza on konkretny typ danych (np. wiek). Innym terminem jest **cecha** (ang. *feature*), która oznacza atrybut wraz z jego wartością. Mają one zastosowanie między innymi w przypadku danych w postaci plików csv. Aby móc skorzystać z mocy uczenia maszynowego, musimy wybrać jakiś model, który wytrenujemy. Modele zostały opisane w rozdziale 3.6, natomiast ogólne typy uczenia maszynowego w rozdziale 3.2. Jednakże, zanim ostatecznie zdecydujemy się który model powinien być wykorzystany, należy porównać istniejące modele. W tym celu zazwyczaj wykorzystuje się **funkcję kosztu**, która mówi nam jak duży błąd popełnia model podczas predykcji [23].

Innym aspektem związanym z modelami uczenia maszynowego jest to, jakie wyniki osiąga dany model (również względem posiadanych danych). Jednym z problemów z tym związanych jest tzw. **przetrenowanie**, bądź też inaczej, **nadmierne dopasowanie** (ang. *overfitting*). Oznacza to, iż model bardzo dobrze sobie radzi na danych uczących, natomiast dla danych testowych wyniki są już o wiele gorsze. Jest to równoznaczne z tym, iż model nie uogólnia danych zbyt dobrze. Zdarza się to najczęściej, gdy

mamy zbyt małą ilość danych i dodatkowo użyjemy zbyt skomplikowanego modelu. Jest to dosyć znany problem, który występuje w literaturze pod nazwą kompromis między obciążeniem a wariancją (ang. *bias-variance tradeoff*). Istnieje wiele rozwiązań, które zapobiegają przetrenowaniu. Jednym z nich jest **regularyzacja**. To, jak bardzo chcemy regularyzować proces uczenia zależy od tego, jakie wartości przypiszemy **hiperparametrom** (ang. *hyperparameters*).

Analogicznym problemem jest **niedotrenowanie** (ang. *underfitting*), jednakże w tym przypadku chodzi o dokładnie przeciwne zjawisko, tzn. gdy model jest zbyt prosty (w stosunku do danych). W tym przypadku istnieje kilka rozwiązań, jak [23]:

– wybór mocniejszego modelu (który posiada większą liczbę parametrów),

– zmodyfikowanie danych w taki sposób, aby posiadały większą liczbę cech (patrz rozdział 3.3),

– redukcja regularyzacji (bądź innych ograniczeń modelu).

Kolejnym problemem związanym ze słabymi wynikami modelu jest aspekt dotyczący danych. Pierwsze utrudnienie to zbyt mała liczba danych. W zależności od zagadnienia wymagana liczba przykładów uczących może się wahać między setkami (np. predykcja obecności choroby u pacjenta na podstawie danych zdrowotnych) a milionami (np. rozpoznawanie mowy, klasyfikacja tekstu). Kolejnym problemem związanym z danymi jest zaszumienie danych. Innymi słowy, mamy z taką sytuacją do czynienia, gdy dane nie odzwierciedlają rzeczywistości (a przynajmniej nie całkowicie poprawnie). Wtedy, niezależnie od zastosowanego modelu nie ma możliwości, aby system, mając do dyspozycji niereprezentatywne dane, radził sobie dobrze na danych testowych. Dotyczy to również przypadków, gdy dane są błędne. Rozwiązaniem tego problemu może być przykładowo [23]:

– odrzucenie elementów odstających,

– odrzucenie, bądź uzupełnienie brakujących danych,

– rozpoznanie i odrzucenie błędnych danych.

Ostatnim poruszonym zagadnieniem związanym z danymi są nieistotne cechy. Mogą one działać jak szum i obniżać jakość posiadanych danych. W tym przypadku rozwiązaniem może być odrzucenie nadmiarowych cech i uproszczenie danych. Kwestia ta zazwyczaj jest poruszana w trakcie wykonywania inżynierii cech (patrz rozdział 3.3). Gdy już posiadamy wyuczony model, dobrze byłoby upewnić się, iż działa on tak, jak powinien. Podejście, które się utrwaliło jest podział posiadanych danych na zbiory **danych uczących** oraz **danych testowych** (czasami również odkłada się jeszcze jeden zbiór danych walidacyjnych, aby we wczesnej fazie wyszukiwania odpowiedniego modelu sprawdzać wyniki właśnie na tych danych, a po wybraniu ostatecznego modelu, przetestować jego skuteczność na danych testowych). Zazwyczaj 80% wszystkich danych przeznacza się na dane uczące [23].

Docelowym zadaniem, które będzie badane w tej pracy jest klasyfikacja wytrzymałości odlewów na podstawie mikrostruktur. Dane, jakimi dysponujemy posiadają dwie etykiety: niska oraz wysoka wytrzymałość (bądź też inny wskaźnik). A więc trzeba będzie skorzystać z **klasyfikatora binarnego**. Jak

natomiast zweryfikować jak dobrze sobie radzi dany klasyfikator? Aby to stwierdzić w statystyce używa się wielu miar, które pokrótce zostaną tutaj przedstawione. Zacznijmy od wyjaśnienia takiego pojęcia jak **tablica pomyłek**. W klasyfikacji binarnej dane są oznaczone dwiema etykietami, dla uproszczenia nazwijmy je **pozytywną** i **negatywną**. Podczas klasyfikacji są im przypisywane przewidywane etykiety i istnieje możliwość, że etykieta zostanie źle przypisana. Ilustruje to poniższa tabela (tab. 3.1).

Tabela 3.1. Schemat tablicy pomyłek (wikipedia)

		Klasa rzeczywista	
		pozytywna	**negatywna**
Klasa predykowana	**pozytywna**	prawdziwie pozytywna (TP)	fałszywie pozytywna (FP)
	negatywna	fałszywie negatywna (FN)	prawdziwie negatywna (TN)

Następnie, w celu ułatwienia oceny klasyfikatora, można wprowadzić poniższe miary wydajności [23, 24]:

– prawdziwie pozytywna (ang. true positive, TP),

– prawdziwie negatywna (ang. true negative, TN),

– fałszywie pozytywna (ang. false positive, FP), tzw. błąd pierwszego rodzaju,

– fałszywie negatywna (ang. false negative, FN), tzw. błąd drugiego rodzaju.

Są to podstawowe miary wydajności za pomocą których dalej zdefiniujemy bardziej rozbudowane miary. Pożądanym wynikiem jest uzyskanie wysokich wartoci prawdziwie pozytywnej oraz prawdziwie negatywnej miary, co jest równoznaczne z wysoką wartocią liczb na głównej przekątnej tablicy pomyłek, natomiast jak najmniejsze (najlepiej zerowe) wartości na pozostałej przekątnej (wszystkie pozostałe wartoci poza główną przekątną). Dzięki tej macierzy możemy przeanalizować w jakich przypadkach nasz klasyfikator myli się najczęściej. Dalej można wprowadzić bardziej zwięzłe metryki [23]:

– dokładność (ang. *accuracy*, ACC)

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.1}$$

– precyzja (ang. *precision*, PPV)

$$PPV = \frac{TP}{TP + FP} \tag{3.2}$$

– czułość (ang *recall*, TPR)

$$TPR = \frac{TP}{TP + FN} \tag{3.3}$$

Dokładność mówi nam jaki odsetek predykcji stanowią poprawne predykcje (patrz wzór 3.1). Kolejną miarą jest precyzja, która mówi nam jaka jest dokładność pozytywnych prognoz (wzór 3.2). Ostatnią

przedstawioną tutaj miarą jest czułość i jest to odsetek pozytywnych przykładów, które zostały poprawnie zaklasyfikowane (wzór 3.3). Macierz pomyłek można również rozszerzyć do klasyfikacji wieloklasowej, co zostało uczynione w badaniach (rozdział 5). Bardziej zaawansowane pojęcia i metody będą wyjaśniane w dalszych rozdziałach niniejszej pracy (w miejscu ich zastosowania).

3.2. Typy uczenia maszynowego

Jednym z kryteriów podziału systemów uczenia maszynowego może być stopień i rodzaj nadzorowania procesu uczenia [23]. Pod tym względem możemy wyróżnić cztery główne rodzaje, przedstawione w poniższych podrozdziałach.

3.2.1. Uczenie nadzorowane

Uczenie nadzorowane (ang. *supervised learning*) polega na trenowaniu modelu za pomocą danych, które zostały przygotowane przez ludzkiego nadzorcę jako pary $< obiekt\ uczcy; etykieta >$ [25]. Celem takiego systemu jest nauczenie się przewidywania prawidłowej odpowiedzi dla danego obiektu wejściowego oraz generalizacja na przypadki, które nie są obecne w danych uczących [25]. Uczenie nadzorowane można podzielić na klasyfikację oraz regresję, co jest determinowane przez etykietę danych. W przypadku, gdy każda etykieta należy do skończonego zbioru, mówimy o klasyfikacji. Jeżeli zaś etykiety mogą przyjmować np. dowolną wartość liczby rzeczywistej, wtedy mówimy o regresji. Jedne z ważniejszych przykładowych algorytmów tego rodzaju są:

- regresja liniowa (ang. *linear regression*),
- algorytm k najbliższych sąsiadów (ang. *k-nearest neighbors algorithm*),
- regresja logistyczna (ang. *logistic regression*),
- drzewa decyzyjne (ang. *decision tree*),
- lasy losowe (ang. *random decision forest*),
- maszyny wektorów nośnych (ang. *support-vector machine*),
- sieci neuronowe.

Te i inne metody zostały opisane w podrozdziale 3.6.

3.2.2. Uczenie częściowo nadzorowane

Uczenie częściowo nadzorowane (ang. *semi-supervised learning*) polega na trenowaniu modelu za pomocą danych zarówno oznakowanych, jak i nieoznakowanych. Wykorzystuje się go wtedy, gdy liczba danych jest ogromna i system sam może zaproponować odpowiedzi. Często algorytmy tego rodzaju stanowią kombinację algorytmów uczenia nadzorowanego i nienadzorowanego [23].

3.2.3. Uczenie nienadzorowane

Uczenie nienadzorowane (ang. *unsupervised learning*) polega na wykrywaniu wzorców, relacji na podstawie nieoznaczonych danych, możliwie maksymalnie bez ingerencji człowieka. Im większa liczba danych, tym bardziej precyzyjne wyniki. Jedne z ważniejszych algorytmów uczenia nienadzorowanego, to:

- metoda k-średnich (ang. *k-means clustering*),
- analiza głównych składowych (ang. *principal component analysis*, PCA),
- stochastyczne osadzanie sąsiadów przy użyciu rozkładu t (ang. *t-distributed stochastic neighbor embedding*, t-SNE).

Jednym z przykładów użycia uczenia nienadzorowanego może być wizualizacja danych, m.in. przy pomocy algorytmów PCA bądź t-SNE. Ten typ uczenia maszynowego nie został wykorzystany w niniejszej pracy, dlatego nie będzie głębiej analizowany.

3.2.4. Uczenie przez wzmacnianie

Uczenie przez wzmacnianie (ang. *reinforcement learning*, RL) polega na interakcji ze środowiskiem za pomocą polityki, mając do dyspozycji zestaw dozwolonych akcji (działań). Model dokonuje analizy środowiska i automatycznie zbiera z niego dane. Celem jest maksymalizacja nagrody. W uczeniu przez wzmacnianie wyróżnia się trzy główne elementy jak **środowisko**, **agenta** oraz **bufor**. System uczący, czyli agent, może obserwować środowisko, na tej podstawie wykonywać pewne czynności, następnie odbierać nagrody, lub kary. W następnym zaś kroku musi nauczyć się najlepszej strategii, zwanej **polityką**, co prowadzi do maksymalizacji nagrody [23]. Ze względu na charakterystykę tego nurtu uczenia maszynowego jest ono często stosowanie do uczenia modeli grania w gry [25].

3.3. Inżynieria cech

Inżynieria cech jest procesem wykorzystania wiedzy dziedzinowej w celu ekstrakcji cech z surowych danych [26]. Cecha jest własnością każdej instancji danych i jest ona wykorzystywana przez model w celu predykcji. Odpowiednio przygotowane dane zwiększają skuteczność modeli [26]. Proces tworzenia cech składa się z sześciu głównych etapów [27]:

1. Burza mózgów – ma na celu zebranie grupy ekspertów w celu zweryfikowania danych lub ustalenia sposobu przeprowadzenia ekstrakcji cech.
2. Wybór cech – w przypadku, gdy mamy wiele cech, możemy wybrać te, które niosą za sobą najwięcej informacji. Można tego dokonać z wykorzystaniem wiedzy dziedzinowej bądź za pomocą różnych technik wyboru podzbioru cech (np. symulowane wyżarzanie, optymalizacja za pomocą

roju cząstek i.in.). Ten krok jest ważny, gdyż dzięki niemu potencjalnie uzyskamy prostszy model, a prostsze modele mają większą zdolność do generalizacji poprzez redukcję wariancji (tzw. kompromis między obciążeniem a wariancją). Inne korzyści, to:

   – krótszy czas treningu,
   – uniknięcie przekleństwa wymiarowości,
   – lepsza interpretowalność.

3. Tworzenie nowych cech – możemy tworzyć nowe cechy za pomocą tych istniejących, np. za pomocą średniej arytmetycznej, minimum, czy innych statystyk. Można też wykorzystać normalizację (np. standaryzacja).

4. Testowanie wpływu cech na model – warto zautomatyzować ten proces, aby wybrać jak taki zestaw cech, który najlepiej się sprawdza na danych treningowych.

5. Poprawa cech w razie konieczności – dalsza modyfikacja cech wraz z obserwacją wpływu na działanie modelu.

6. Powtórzenie powyższych czynności – powtarzamy powyższe czynności do momentu, gdy przestaniemy uzyskiwać coraz lepsze rezultaty, bądź gdy dojdziemy do wniosku, iż dane są słabej jakości, bądź dysponujemy zbyt małą ich liczbą.

Cechy mogą się różnić pod względem znaczenia. Dodatkowo odpowiednio skomponowany zbiór cech może zapobiec nadmiernemu dopasowaniu się modelu do danych uczących.

3.4. Augmentacja danych

Augmentacja danych to zbiór technik służących zwiększaniu ilości danych poprzez dodanie do zbioru danych zmodyfikowanych kopii istniejących danych bądź nowo utworzonych danych syntetycznych z istniejących danych [28]. Dzięki temu zabiegowi możemy zapobiec nadmiernemu dopasowaniu się modelu do danych treningowych. Jest to szczególnie przydatne kiedy nie dysponujemy zbiorem danych rzędu dziesiątek tysięcy przykładów. Ponieważ nasze dane to zdjęcia mikrostruktur, dlatego w tym rozdziale zajmiemy się tylko rozszerzaniem danych w celu klasyfikacji obrazów. Możemy wymienić kilka głównych strategii:

1. Odwrócenie – możemy odwracać obrazy w pionie i w poziomie, zachowując przy okazji oryginalne rozmiary oryginalnego zdjęcia.

2. Rotacja – możemy obracać obrazy o 90° w każdym kierunku, otrzymując tym samym trzy nowe obrazy. W tym przypadku natomiast otrzymane obrazy mogą mieć inne rozmiary niż oryginalny obraz, w przypadku gdy nie jest on kwadratem.

3. Skalowanie – obraz może być przeskalowany na zewnątrz lub do wewnątrz. W przypadku skalowania na zewnątrz otrzymujemy obraz o większym rozmiarze, a więc wycinając odpowiedni obszar możemy otrzymać obraz o takim samym rozmiarze jak oryginalny obraz.

4. Wycinanie – inną możliwością jest wycinanie losowych fragmentów z obrazu. Gdy chcemy zachować oryginalny rozmiar, trzeba jeszcze przeskalować zmodyfikowany obrazek.

5. Translacja – polega na przesuwaniu obrazu wzdłuż osi. Działa szczególnie dobrze, gdy mamy do czynienia z obrazami, które posiadają jednolite tło.

6. Nałożenie szumu – dzięki tej technice możemy zapobiec nadmiernemu dopasowaniu się modelu do danych.

Wśród innych metod, które również mogą się sprawdzić, można wymienić przekształcenia geometryczne, modyfikację kolorów czy losowe wymazywanie [28]. Istnieje też wiele innych, bardziej zaawansowanych metod augmentacji danych [29]. Przykładowo można zastosować tzw. generatywne sieci współzawodniczące (GAN od ang. *generative adversarial network*). Mogą one m.in. zmieniać domenę obrazu, jak pokazano na rysunku 3.1. Inną zaawansowaną techniką jest interpolacja. Może ona zostać wykorzystana w przypadku, gdy chcemy skorzystać z translacji – możemy wtedy brakujący fragment obrazu interpolować. Przyda się również, gdy chcemy skorzystać ze skalowania do wewnątrz jednocześnie zachowując oryginalny rozmiar obrazu [29].

Rys. 3.1. Przykład zmiany domeny za pomocą CycleGAN (tutaj zmiana pór roku).
Źródło: *https://junyanz.github.io/CycleGAN/*

3.5. Uczenie się przez transfer

Ludzie mają wrodzoną zdolność wykorzystywania wiedzy, którą zdobyli w trakcie wykonywania innego zadania. To znaczy, wiedzę, którą zdobywamy, ucząc się pewnej rzeczy, możemy wykorzystać, wykonując inne, aczkolwiek powiązane zadanie. Im bardziej powiązane są te zadania, tym łatwiej jest nam

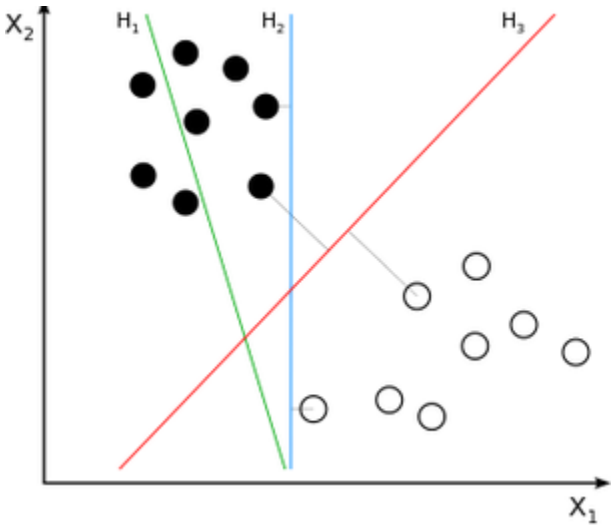
przenieść naszą wiedzę [30]. A więc w najprostszych słowach, uczenie się przez transfer to idea wykorzystania wiedzy zdobytej w ramach jednego zadania do rozwiązywania zadań pokrewnych. W ostatnich latach uczenie się przez transfer również zaczęto stosować w uczeniu maszynowym, czy też w głębokim uczeniu [31]. W tym przypadku najczęściej polega ono na wykorzystaniu już istniejącego modelu do nowego problemu. Jakie wynikają z tego korzyści? Przede wszystkim to podejście pozwala nam na osiągnięcie zamierzonych rezultatów (np. wysoka skuteczność klasyfikacji) przy użyciu mniejszej ilości danych niż w przypadku trenowania całkowicie nowego modelu. Uczenie się przez transfer dobrze można zilustrować na przykładzie rozpoznawania obrazów (tym lepiej, że właśnie w tym celu wykorzystano tę metodologię w niniejszej pracy), a mianowicie sieci neuronowe w początkowych warstwach wykrywają krawędzie, w kolejnych warstwach wykrywają kształty, natomiast w ostatnich warstwach wykrywają specyficzne zależności dla danego zadania [31]. A więc w celu klasyfikacji danych obiektów można skorzystać z ogólnie znanych, dobrze wytrenowanych, na wielkich zbiorach danych, głębokich sieci neuronowych (np. *VGG19*) w ten sposób, że budujemy nowy model, który kopiuje początkowe warstwy wcześniej wytrenowanej sieci, a następnie przyłączamy nowe warstwy, które zostaną dotrenowane w tym konkretnym celu. Oprócz wykorzystania uczenia transferowego do klasyfikacji obrazów skutecznie stosowano to podejście również w takich zadaniach, jak klasyfikacja tekstów czy filtrowanie spamu [32].

3.6. Wykorzystane metody uczenia maszynowego

W tym rozdziale zostaną przedstawione metody uczenia maszynowego, które zostały wykorzystane w trakcie realizacji pracy. Większość z nich to algorytmy uczenia nadzorowanego ze względu na charakterystykę danych.

3.6.1. Maszyna wektorów nośnych

Maszyna wektorów nośnych (ang. *Support Vector Machine*, SVM) to model nadzorowanego uczenia maszynowego, która wyznacza hiperpłaszczyznę w celu rozdzielenia przykładów należących do dwóch klas z maksymalnym marginesem [25]. SVM to potężny i wszechstronny model uczenia maszynowego, który jest w stanie przeprowadzić klasyfikację liniową, nieliniową oraz regresję [23]. Rysunek 3.2 przedstawia koncepcję działania SVM. Model ten mapuje przykłady uczące do punktów w przestrzeni tak, aby maksymalizować odległość między dwiema kategoriami. Następnie nowe przykłady są mapowane do tej samej przestrzeni i w zależności, po której stronie marginesu się znajdują, tak są klasyfikowane. Tak jak wspomniano wyżej, za pomocą SVM również można skutecznie przeprowadzać klasyfikację nieliniową, co jest możliwe dzięki tzw. sztuczce z funkcją jądra (ang. *kernel trick*). Polega ona na tym, iż dane wejściowe są niejawnie odwzorowywane do przestrzeni cech o wyższym wymiarze [33].
Jednym z problemów, które można napotkać w trakcie korzystania z SVM jest ich czułość na skalę cech. Jednakże łatwo jest to obejść przy pomocy przeskalowania cech (np. standaryzacja). Innym problemem jest konieczność posiadania całkowicie oznaczonych danych [34]. Dodatkowo parametry tego modelu



Rys. 3.2. Działanie SVM. H1 nie separuje klas, H2 separuje, lecz z małym marginesem, natomiast H3 separuje z maksymalnym marginesem (źródło: https://en.wikipedia.org/wiki/Support-vector_machine)

są trudne do zinterpretowania. Mimo wszystko wady te nie są zbyt uciążliwe, a nawet istnieją rozwiązania, które je obchodzą. Istnieje taki model, jak SVC (ang. *support-vector clustering*), który można wykorzystać w celu uczenia nienadzorowanego). Dodatkowo istnieją rozszerzenia modelu SVM, dzięki którym można wykorzystać ten model do klasyfikacji wieloklasowej. Polegają one m.in. na takich strategiach, jak rozpatrywanie danej etykiety przeciwko wszystkim pozostałym (i tak dla każdej etykiety). Algorytm SVM jest jednym z najczęściej stosowanych w przemyśle, dlatego jego efektywność zostanie przetestowana na tle innych algorytmów.

3.6.2. Drzewo decyzyjne

Drzewo decyzyjne (ang. *decision tree*) to algorytm uczenia maszynowego stosowany w celu pozyskiwania wiedzy na podstawie przykładów [35]. Mają wiele zastosowań i służą zarówno do zadań klasyfikacji, jak i regresji. Jest to struktura, która kształtem przypomina drzewo (stąd nazwa), a każda ścieżka w tym drzewie przedstawia możliwą **ścieżkę decyzyjną** wraz z jej skutkami [36]. Składa się ono z **węzłów** (które jednocześnie oznaczają jakąś decyzję, stan) i **gałęzi** (wybór, możliwość). Drzewo konstruowane jest od korzenia i z każdą kolejną decyzją do podjęcia jest budowany w dół, to tzw. **liści** (ang. *leaf*), które oznaczają etykietę klasy. Istnieje wiele algorytmów generowania drzew decyzyjnych. Najbardziej znane z nich, to *ID3* (*Iterative Dichotomiser 3*), czy też *C4.5*. Algorytm *ID3* polega na tym, że w każdej iteracji jest wybierany niewykorzystany atrybut, który daje największy wzrost informacji (bądź np. najmniejszą miarę zanieczyszczenia), a zbiór danych jest dzielony według wartości tego atrybutu. Natomiast algorytm *C4.5* jest rozszerzeniem wcześniejszego algorytmu i wprowadza takie usprawnienia, jak [37]:

– obsługa atrybutów ciągłych i dyskretnych,

– obsługa danych z brakującymi wartościami atrybutów,

– przycinanie drzew po ich utworzeniu.

Wśród największych zalet drzew decyzyjnych wymienia się [38]:

– prosty algorytm,

– atwo interpretowalne wyniki,

– działają nawet dla niewielkiej liczby danych,

– nie wymagają przygotowywania danych (w szczególności skalowania) [23].

Natomiast posiadają też wady, takie jak [38]:

– niestabilność (niewielka zmiana danych może prowadzić do dużej zmiany struktury drzewa),

– często są stosunkowo niedokładne (inne klasyfikatory zazwyczaj radzą sobie lepiej),

– może wystąpić problem nadmiernego dopasowania się do danych.

Pomimo tych wad klasyfikator ten zostanie wykorzystany w badaniach, głównie ze względu właśnie na jego bardzo łatwą interpretowalność. Jako ciekawostkę można dodać, że drzewa decyzyjne składają się na algorytm lasu losowego (ang. *random forest*), który zostanie omówiony w podrozdziale 3.6.3.

3.6.3. Las losowy

Jak zostało wspomniane wyżej, drzewa decyzyjne mogą zbyt mocno dopasować się do danych treningowych. Jednym z rozwiązań jest technika **lasów losowych** (ang. *random forest*), która polega na tworzeniu wielu drzew decyzyjnych, a następnie klasyfikacji w drodze głosowania [36], a więc jako odpowiedź jest generowana dominanta klas (klasyfikacja) lub średnia przewidywana wartość (regresja) [39]. Istnieje kilka różnych algorytmów tworzenia lasów losowych. Jednym z nich jest **agregacja** (ang. *bagging*, *bootstrap aggregating*). Polega on na tym, iż wielokrotnie wybiera się losowe podzbiory zestawu uczącego ze zwracaniem (ang. *sampling with replacement*) w celu dopasowania do każdego takiego zbioru nowego drzewa [23]. Dzięki temu podejściu zmniejsza się wariancja modelu, bez zwiększania obciążenia [40], gdyż pojedyncze drzewa decyzyjne nie są ze sobą skorelowane (dzięki uczeniu ich na różnych zbiorach danych). Następnie, w celu klasyfikacji, wszystkie pojedyncze drzewa zwracają wynik, który jest uśredniony. Dodatkowo, aby jeszcze mocniej zmniejszyć korelację między drzewami, w przypadku lasów losowych korzysta się ze zmodyfikowanego algorytmu trenowania drzewa, gdzie przy każdym kolejnym węźle jest wybierany losowy podzbiór cech. Dzięki temu, jeśli w danych występuje cecha, która jest silnym predyktorem dla zmiennej przewidywanej, jej wykorzystanie w różnych drzewach zostanie ograniczone, tym samym dalej zmniejszając korelację między nimi.

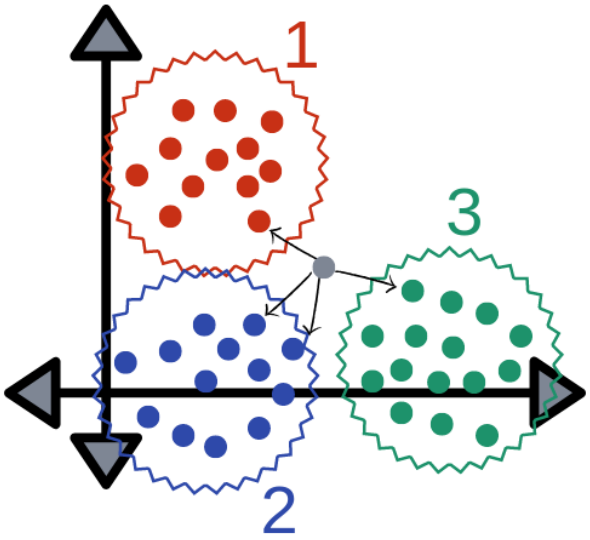
Innym podejściem jest **wzmacnianie** (ang. *boosting*) [41]. Główną różnicą między agregacją a wzmacnianiem jest to, iż w przypadku tego pierwszego każde drzewo jest uczone niezależnie od pozostałych i ten etap może być przeprowadzony równolegle. Natomiast w przypadku drugiego algorytmu uczenie pojedynczych drzew następuje sekwencyjnie i każdy kolejny klasyfikator bierze pod uwagę wyniki poprzedniego klasyfikatora. Danym błędnie sklasyfikowanym przypisuje się większą wagę, tym samym kolejne modele zwracają większą uwagę trudnym danym. Jednakże w przypadku wzmacniania klasyfikacja odbywa się nieco inaczej, gdyż tutaj również mamy wagi, które są przypisywane klasyfikatorom w trakcie uczenia – im lepszy klasyfikator, tym większe wagi otrzymuje. Dzięki zastosowaniu jednego z tych dwóch podejść zwiększa się stabilność modelu. Jednakże jeżeli mamy do czynienia z danymi, dla których pojedyncze drzewo osiąga słabe wyniki, wtedy rozwiązaniem może być zastosowanie wzmacniania. Jeżeli zaś problemem jest nadmierne dopasowanie się drzewa do danych, wtedy pomocne może się okazać zastosowanie agregacji.

3.6.4. K najbliższych sąsiadów

Algorytm k najbliższych sąsiadów (ang. *k-nearest neighbors algorithm*, k-NN) jest jednym z najprostszych modeli predyktywnych [36]. Można go użyć zarówno do klasyfikacji, jak i regresji. Jedynym wymaganiem tej metody jest wybór jakiejś miary odległości, a więc jest metodą nieparametryczną. Jego działanie opiera się na założeniu, że im bliżej siebie znajdują się punkty, tym są bardziej do siebie podobne. Tak więc, dla przykładowej obserwacji liczona jest odległość (według z góry ustalonej metryki) od pozostałych punktów i wybieranych jest k najbliższych (ustalona z góry liczba). Następnie wybór najczęściej występującej (bądź uśrednienie wartości zmiennej objaśnianej) jako wynik klasyfikacji. Najczęściej stosuje się metrykę euklidesową, bądź też metrykę Mahalanobisa [42]. Jeżeli cechy danych znacznie różnią się w skali, bądź reprezentują inne jednostki, wtedy normalizacja może znacząco poprawić dokładność [43]. Innym ciekawym usprawnieniem może być przypisanie wag sąsiadom w taki sposób, aby najbliżsi sąsiedzi mieli największy wpływ na wynik klasyfikacji (często stosowana jest odwrotność odległości) [44]. k-NN odnotowuje dobre wyniki w zakresie spójności, jest łatwy w implementacji, natomiast może być wymagający obliczeniowo w przypadku dużych zbiorów danych [44]. Jego użyteczność jest największa w przypadku, gdy zależność między zmiennymi objaśniającymi a objaśnianymi jest złożona [44]. Na poniższym rysunku zostało przedstawione schematycznie działanie k-NN (rys. 3.3). Wzmacnianie zostało szegółowo omówione w rozdziale 3.6.7.

3.6.5. Regresja logistyczna

Regresja logistyczna (ang. *logistic regression*) to model używany w statystyce do przewidywania prawdopodobieństwa pewnej klasy, bądź zdarzenia, które może przyjmować tylko dwie wartości [45]. Mechanizm działania tej metody jest podobny do regresji liniowej, tyle że tutaj wyliczamy ważoną sumę cech wejściowych (wzór 3.4)



Rys. 3.3. Schemat działania algorytmu k-NN. Szary punkt, w zależności od wartości k i odległości od najbliższych punktów, będzie miał przypisaną jedną z trzech dostępnych klas (źródło medium.com, Madison Scott)

$$logit(p_i) = ln(\frac{p_i}{1-p_i}) = \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i} \tag{3.4}$$

gdzie p_i to nieznane prawdopodobieństwo sukcesu w próbie i, $x_{k,i}$ to wartość k-tego predyktora próby i, natomiast β_k to nieznany parametr k-tego predyktora, który jest optymalizowany (najczęściej za pomocą metody największej wiarygodności) [46]. Następnie zwracana jest wartość funkcji logistycznej z tego rezultatu (wzór 3.5) [23], co jest równoznaczne z oszacowaniem prawdopodobieństwa:

$$\hat{p} = \sigma(t) = \frac{1}{1 + exp(-t)} \tag{3.5}$$

Następnie prognoza modelu regresji logistycznej jest wyliczana za pomocą wzoru 3.6.

$$\hat{y} = \begin{cases} 0 & \text{jeśli } \hat{p} < 0.5 \\ 1 & \text{wpp.} \end{cases} \tag{3.6}$$

Model ten może zostać rozszerzony tak, aby przewidywał kilka klas zdarzeń w taki sposób, iż każdemu zdarzeniu jest przypisywane prawdopodobieństwo, a suma wszystkich tych prawdopodobieństw wynosi jeden. W tym celu wykorzystuje się funkcję *softmax* (wzór 3.7), przekazując jej prawdopodobieństwa wystąpienia każdej klasy (nie muszą się sumować do jedynki), następnie dla każdej jest liczona eksponenta, po czym następuje normalizacja (dzieląc wyniki przez sumę wszystkich eksponent) [23].

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.7}$$

gdzie z to wektor zawierający wyniki każdej klasy, i to ustalona klasa. Jak wspomniano wcześniej, współczynniki są estymowane za pomocą metody największej wiarygodności, gdzie w sposób iteracyjny (np. metoda Newtona) modyfikowane są wartości współczynników [46]. Ostatnim elementem jest uczenie modelu. Jego celem jest uzyskanie modelu, który zwraca wysokie prawdopodobieństwo dla klasy docelowej. W tym celu minimalizuje się funkcję kosztu zwaną entropią krzyżową (ang. *cross entropy*), przedstawioną poniżej (wzór 3.8).

$$H = -\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) \tag{3.8}$$

gdzie M oznacza liczbę klas, y to wskaźnik binarny (0 lub 1 w zależności czy etykieta c jest zgodna z klasą obserwacji o) oraz p to przewidywane prawdopodobieństwo przypisania klasy c obserwacji o.

Regresja logistyczna jest bardzo skuteczna dla prostych zbiorów danych a także w przypadku zbiorów liniowo separowalnych. Dodatkowo jej współczynniki mogą być interpretowane jako wskaźniki ważności cech, dlatego również zostanie uwzględniona w badaniach.

3.6.6. Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski (ang. *naive Bayes classifier*) należy do rodziny prostych probabilistycznych klasyfikatorów, które opierają się na twierdzeniu Bayesa [47]. Istotnym założeniem tego modelu jest wzajemna niezależność predyktorów. Naiwne klasyfikatory bayesowskie są wysoce skalowalne, aczkolwiek wymagają wielu parametrów liniowych w stosunku do liczby zmiennych) [47]. Dużym atutem tego modelu jest zastosowanie metody największej wiarygodności, przez co czas uczenia jest liniowy (w przeciwieństwie do wielu innych typów klasyfikatorów) [47], dodatkowo nie trzeba akceptować prawdopodobieństwa bayesowskiego. Kolejną przewagą nad innymi modelami jest niska wymagana liczba danych uczących, aby model mógł wyestymować parametry potrzebne do klasyfikacji [47]. Korzystając z twierdzenia Bayesa można wyprowadzić wzór, który stoi za naiwnym modelem probabilistycznym Bayesa (wzór 3.9).

$$p(C|F_1, \ldots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C) \tag{3.9}$$

gdzie C to dana klasa zmiennej zależnej, n to liczba zmiennych niezależnych, $F1, \ldots, F_n$ to zmienne niezależne, natomiast Z jest współczynnikiem skalowania zależnym od predyktorów [47]. Klasyfikacja odbywa według poniższej funkcji (wzór 3.10):

$$\hat{y} = \arg\max_{k} p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{3.10}$$

gdzie $\hat{y}$ to przewidywana klasa, n to liczba zmiennych niezależnych. Pomimo faktu, że wymogi niezależności są często łamane, naiwny klasyfikator bayesowski posiada szereg cech, które w rzeczywistości są zaskakująco korzystne. Klasyfikacja jest poprawna, podobnie jak w przypadku wszystkich klasyfikatorów probabilistycznych, które wykorzystują metodę maksimum prawdopodobieństwa *a posteriori*

(MAP), o ile właściwa klasa jest bardziej prawdopodobna niż pozostałe. A więc klasyfikator jest wystarczająco silny, aby zignorować główne wady naiwnego modelu probabilistycznego [47].

3.6.7. Wzmacnianie

Wzmacnianie (ang. *boosting*) zostało już pokrótce omówione przy okazji lasów losowych (rozdział 3.6.3). Pokrótce, polega ono na pracy zespołowej wielu słabych klasyfikatorów, otrzymując w ten sposób zespół będący silnym klasyfikatorem. Istotą wzmacniania jest sekwencyjne uczenie predyktorów w taki sposób, że każdy kolejny próbuje poprawiać błędy poprzednika [23]. W poniższych podrozdziałach zostaną przedstawione dwa najpopularniejsze algorytmy wzmacniania: *AdaBoost* oraz wzmacnianie gradientowe (ang. *gradient boosting*).

3.6.7.1. AdaBoost

Ideą tego modelu jest uczenie sekwencyjne kolejnych klasyfikatorów w taki sposób, że każdy kolejny próbuje korygować błędy swojego poprzednika. To znaczy, każdy kolejny klasyfikator przykłada większą wagę przykładom uczącym, dla których poprzedni algorytm został niedotrenowany [23]. W ten sposób do zespołu są dołączane coraz bardziej dokładne klasyfikatory, tym samym zwiększając jego skuteczność. Ostatecznie prognoza jest średnią ważoną wyników poszczególnych klasyfikatorów. Ogólny schemat jest przedstawiony poniżej.

Początkowo każdej próbce jest przypisywana waga (wzór 3.11:

$$w^{(i)} = \frac{1}{m} \tag{3.11}$$

gdzie m to liczba próbek. Należy również pamiętać, że suma wszystkich wag wynosi zawsze jeden. Po wytrenowaniu pierwszego klasyfikatora zostanie mu przypisany ważony współczynnik błędu r_j, gdzie j oznacza liczbę porządkową klasyfikatora (równanie 3.12):

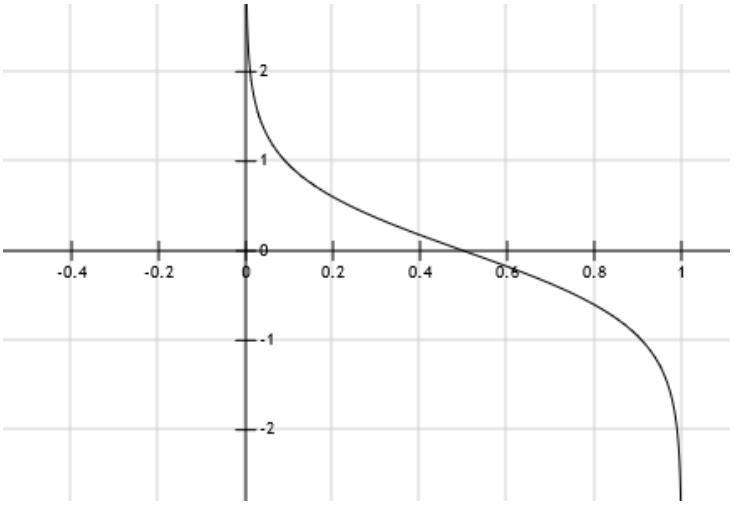
$$r_j = \frac{\sum\limits_{\substack{i=1 \\ \hat{y}_j^{(i)} \neq y^{(i)}}}^{m} w^{(i)}}{\sum\limits_{i=1}^{m} w^{(i)}} \tag{3.12}$$

gdzie $\hat{y}_j^{(i)}$ jest prognozą j-tego klasyfikatora dla i-tego przykładu. Następnie zgodnie z równaniem 3.13 wyliczana jest waga dla j-tego klasyfikatora:

$$\alpha_j = \eta log \frac{1 - r_j}{r_j} \tag{3.13}$$

gdzie η to współczynnik uczenia. W ten sposób dokładniejsze klasyfikatory dostają większe wagi, natomiast te mniej dokładne – mniejsze. Dla zobrazowania wykres tego równania został przedstawiony na rysunku 3.4.

W kolejnym etapie następuje aktualizacja wag próbek (równanie 3.14):



Rys. 3.4. Wykres równania 3.13. Widać, że dla błędów zerowych waga klasyfikatora zbiega się do nieskończoności, natomiast dla błędów dążących do jedynki, waga zbiega się do minus nieskończoności (źródło: opracowanie własne

$$w^{(i+1)} = \begin{cases} w^{(i)} & \text{gdy } \hat{y}_j^{(i)} = y^{(i)} \\ w^{(i)} exp(\alpha_j) & \text{wpp.} \end{cases} \tag{3.14}$$

Cały ten proces jest powtarzany wielokrotnie, iteracyjnie, aż osiągniemy ustaloną liczbę klasyfikatorów, bądź też wyniki zwracane przez model będą na odpowiednim poziomie (zazwyczaj uczenie zatrzymuje się przy 100%) [23]. Ostatecznie, w celu klasyfikacji jest liczona średnia ważona (przy użyciu wag z równania 3.13) z predykcji wszystkich klasyfikatorów składających się na model i wybierana jest odpowiedź, która otrzyma tym sposobem najwięcej głosów (równanie 3.15).

$$\hat{y}(x) = \arg\max_k \sum_{\substack{j=1 \\ \hat{y}_j(x)=k}}^{N} \alpha_j \tag{3.15}$$

gdzie N to liczba klasyfikatorów.

Podsumowując, AdaBoost ma wiele zalet. Wśród nich znajduje się ta, iż AdaBoost nie wymaga modyfikowania parametrów w celu osiągnięcia optymalnego modelu (w przeciwieństwie do np. SVM). Dodatkowo, AdaBoost może być stosowany, gdy chcemy poprawić dokładność słabego klasyfikatora [48]. Jednak trzeba pamiętać o tym, iż metody wzmacniania działają progresywnie, a więc wymagane są dane wysokiej jakości. Dodatkowo należy się wcześniej upewnić, że usunięto wszystkie wartości odstające (ang. *outliers*), na które ten model również jest wrażliwy [48].

3.6.7.2. Wzmacnianie gradientowe

Wzmacnianie gradientowe (ang. *gradient boosting*) to inna bardzo popularna technika wzmacniania [23]. Nadaje się zarówno do klasyfikacji, jak i regresji. Podobnie do poprzedniej metody, polega na dodawaniu kolejnych klasyfikatorów do zespołu w sposób sekwencyjny w taki sposób, że następnik poprawia

poprzednika. Różnicą między tą techniką a AdaBoost jest to, że tutaj nie aktualizujemy wag przykładów po każdym przebiegu, lecz próbujemy dopasować predyktor do błędu resztowego (ang. *residual error*) popełnionego przez poprzedni klasyfikator [23]. Zwyczajowo jako słabych predyktorów używa się drzew decyzyjnych, a otrzymany algorytm jest nazywany gradientowo wzmocnionym drzewem (ang. *gradient boosted tree*). Wyniki uzyskiwane za pomocą tej techniki zazwyczaj są lepsze od tych uzyskiwanych za pomocą lasów losowych [49]. Jedno z prostszych wyjaśnień tej metody przedstawił Cheng Li na przykładzie regresji [50]. Załóżmy, że chcemy nauczyć model F przewidywać wartości postaci (3.16):

$$\hat{y} = F(x) \tag{3.16}$$

minimalizując błąd średniokwadratowy (3.17):

$$L = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{3.17}$$

gdzie

- n liczba próbek w zbiorze uczącym y
- $\hat{y}_i$ wartość przewidywana $F(x)$
- $\hat{y}_i$ obserwowana wartość

Zakładając, iż algorytm składa się z M etapów, w każdym etapie m ($1 < m < M$) wzmacniania istnieje niedoskonały model F_m. W celu poprawienia jakości modelu F_m dodaje się pewien estymator $h_m(x)$ (równanie 3.18):

$$h_m(x) = y - F_m(x) \tag{3.18}$$

W ten sposób h dopasowuje się reszty $y - F_m(x)$ (zwanej błędem resztowym), przyczyniając się do polepszania wyników osiąganych przez kolejne modele. Rozszerzenie tej idei do klasyfikacji polega na obserwacji, iż reszty $h_m(x)$ to tak naprawdę ujemny gradient błędu średniokwadratowego (równania 3.19, 3.20):

$$L_{MSE} = \frac{1}{2}(y - F(x))^2 \tag{3.19}$$

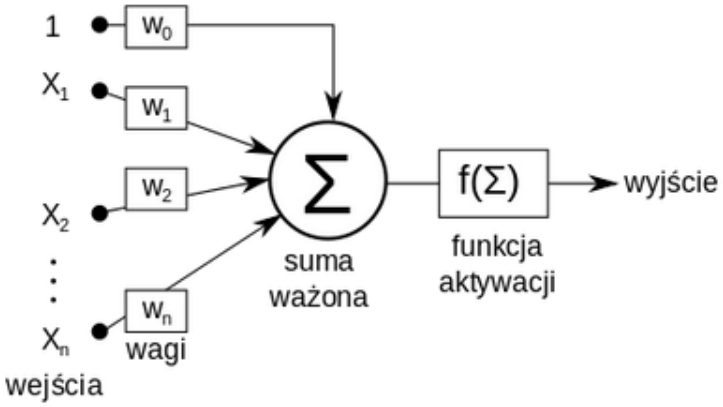
$$h_m(x) = -\frac{\partial L_{MSE}}{\partial F} = y - F(x) \tag{3.20}$$

Wzmacnianie gradientowe to szeroko wykorzystywany algorytm. Jednak trzeba pamiętać, iż mimo że zazwyczaj podnosi dokładność modelu, to jednak tracimy na interpretowalności, dodatkowo rośnie czas uczenia.

3.6.8. Sieci neuronowe

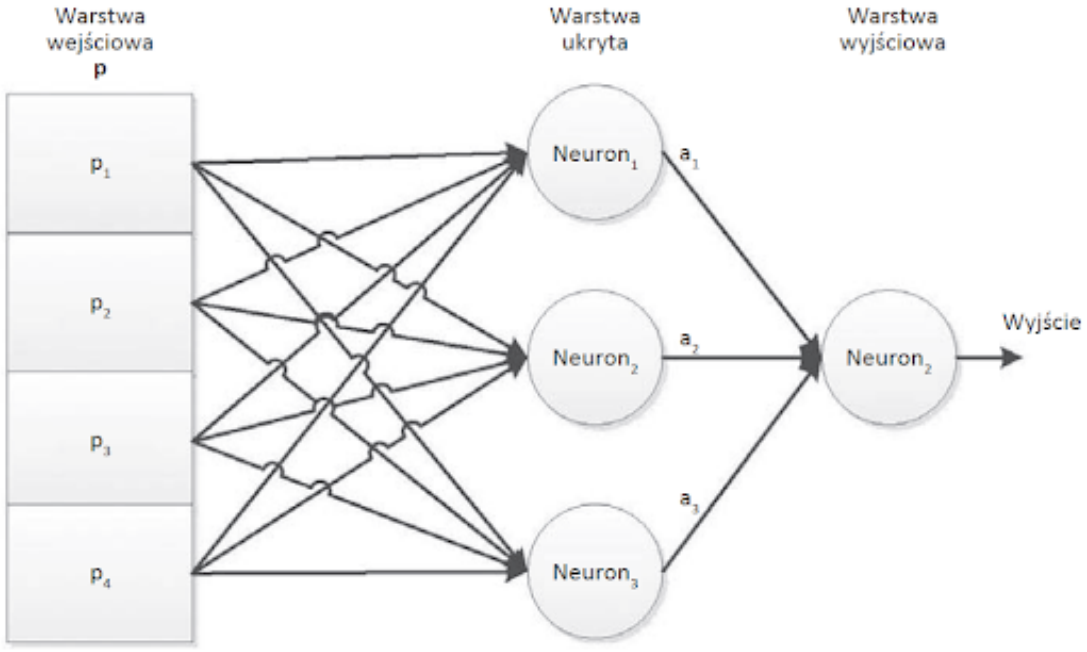
Sztuczne sieci neuronowe (ang. *artificial neural networks*), zwykle nazywane po prostu **sieciami neuronowymi** (ang. *neural networks*), to systemy komputerowe inspirowane biologicznymi sieciami

neuronowymi, na które się składają mózgi zwierząt [51]. Główną zaletą tych systemów jest ich możliwość rozwiązywania problemów, nie będąc bezpośrednio do tego zaprogramowanymi. Najprostszym elementem sieci neuronowych jest sztuczny neuron. Każdy neuron posiada wiele wejść i pojedyncze wyjście, które może być przesyłane do wielu pozostałych neuronów. Wejściem mogą być jakieś cechy (np. wiek, wzrost), natomiast wyjściem ostatnich neuronów, które realizują jakieś zadanie, może być np. wynik klasyfikacji (np. płeć). Aby obliczyć wyjście neuronu, każde jego wejście jest przemnożonenie przez **wagę** połączenia z poprzednim neuronem, następnie dodawana jest wartość zwana **obciążeniem** (ang. *bias*), a na końcu ta wartość jest przekazywana do tzw. **funkcji aktywacji** (ang. *activation function*), której wynik jest jednocześnie wyjściem neuronu. Schemat neuronu został przedstawiony na rysunku 3.5. Kolejnym elementem są **połączenia** (ang. *connections*) oraz **wagi** (ang. *weights*). Sieć składa



Rys. 3.5. Schemat neuronu McCullocha-Pittsa (źródło: wikipedia, Krzysztof Zajączkowski)

się z połączeń, które łączą neurony z poszczególnych warstw, dostarczając wyjście neuronów z warstwy poprzedniej. Jak wcześniej zostało napisane, neuron może mieć tylko jedno wyjście, tzn. jedną wartość wyjścia, natomiast może przekazywać tę wartość wielu neuronom w kolejnej warstwie. Za pomocą wag i połączeń są przekazywane wartości z warstw poprzedzających do warstw następnych. Wykorzystuje się tutaj mechanizm zwany **funkcją propagacji** (ang. *propagation function*). Jak wspomniano wcześniej, neurony znajdują się w **warstwach** (ang. *layers*). Neurony z danej warstwy łączą się tylko z neuronami z warstwy poprzedniej oraz neuronami z warstwy następnej (w standardowych implementacjach). Warstwa, która na wejściu otrzymuje dane jest nazywana **warstwą wejściową** (ang. *input layer*), natomiast warstwa, która zwraca wyniki jest nazywana warstwą wyjściową (ang. *output layer*). Pomiędzy tymi warstwami może być zero lub więcej warstw, które są nazywane **warstwami ukrytymi** (ang. *hidden layers*). Schemat sieci neuronowej został przedstawiony na poniższym obrazku (rys. 3.6). Uczenie takich sieci polega na modyfikacji wag połączeń w taki sposób, aby sieć coraz lepiej dopasowywała się do danych uczących. Wagi modyfikowane są tak, aby zmniejszać błąd zwracany przez tak zwaną **funkcję straty** (ang. *loss function*), która zwraca błąd sieci dla każdej pojedynczej instancji wejściowej (bądź partii wejściowej). Odbywa się to za pomocą **propagacji wstecznej** (ang. *backpropagation*). Polega ona



Rys. 3.6. Schemat sieci neuronowej (źródło: *https://www.controlengineering.pl*, Jimmy W. Key)

na obliczeniu gradientu z **funkcji kosztu** (ang. *cost function*, jest to średnia obliczona z funkcji straty dla wielu próbek) dla danych próbek wejściowych względem wag.

Każdą sieć definiuje zestaw **hiperparametrów** (ang. *hyperparameters*). Są to stałe wartości ustalane przed procesem uczenia. Odpowiednio manipulując wartościami hiperparametrów możemy uzyskać lepsze modele. Najważniejszymi hiperparametrami są:

– szybkość uczenia się (ang. *learning rate*) – gdy sieć popełnia błąd, modyfikowane są wagi połączeń w taki sposób, aby zbliżyć się do poprawnego wyniku. Ten hiperparametr określa jak duży krok powinna wykonać sieć w kierunku minimalizacji błędu,

– liczba ukrytych warstw (ang. *hidden layers*) – za pomocą tego hiperparametru możemy zdecydować jaka powinna być architektura sieci (głębsza, dla wielkich i skomplikowanych danych, bądź płytsza, dla prostych danych),

– wielkość partii (ang. *batch size*) – określa liczbę próbek, które są propagowane przez sieć. Ma to wpływ na wykorzystywaną pamięć, prędkość uczenia się, a także na finalną dokładność.

Tak jak w przypadku klasycznych metod uczenia maszynowego, tak samo w przypadku sieci neuronowych można skorzystać z różnych typów uczenia (patrz rozdział 3.2).

3.6.8.1. VGG19

W badaniach wykorzystano sieci neuronowe, jednakże mała liczba posiadanych danych spowodowała, że konieczne było użycie techniki zwanej uczeniem się przez transfer (rozdział 3.5). W tym celu

wykorzystano znaną architekturę, a mianowicie VGG19 (ang. *Visual Geometry Group*, 19 to liczba trenowalnych warstw) [52]. Dodatkowo na górze architektury dodano kilka warstw, za pomocą których można wykonać **strojenie** (ang. *fine tuning*), a więc dopasowanie modelu do własnych danych. Sieć ta była trenowana w celu klasyfikacji zdjęć w turnieju *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) w roku 2014. Na wejciu przyjmowała obrazki w formacie RGB (ang. *reg, green, blue*) o rozmiarze 224 224 (rys. 3.7). Model ten dobrze generalizuje się do innych zadań oraz zbiorów danych [52],

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Rys. 3.7. Konfiguracje sieci VGG (w kolumnach). Głębokość architektury rośnie od lewej do prawej. Warstwy konwolucyjne zostały oznaczone jako „conv<rozmiar pola odbiorczego>-<liczba kanałów>". Została również zastosowana funkcja aktywacji ReLU. Źródło: [52]

oraz, mimo iż minęło już kilka lat od momentu jego wydania, wciąż osiąga wysokie wyniki w zadaniach

klasyfikacji obrazów, dlatego jego skuteczność zostanie przetestowana na tle klasycznych algorytmów uczenia maszynowego.

4. Przygotowanie danych

1: [10]).
2: [12].
3: [1].
4: [3].
5: [5]
6: [4]
7: [2]
8: [6]
9: [7]
10: [8]
11: [9]
12: [11]
13: [13]
14: [15]
15: [16]
16: [19]
17: [18]
18: [17]
19: [14]
20: [20]
21: [21]
22: [22]
23: [53]
24: [25]
25: [26]
26: [28]
27: [29]
28: [54]
29: [30]
30: [32]

31: [31]
32: [23]
33: [24]
34: [33]
35: [36]
36: [35]
37: [37]
38: [38]
39: [39]
40: [41]
41: [42]
42: [43]
43: [44]
44: [45]
45: [47]
46: [48]
47: [27]
48: [34]
49: [40]
50: [46]
51: [49]
52: [50]
53: [51]
54: [52]

5. Testy i wyniki

6. Podsumowanie i wnioski

Bibliografia

[1] D. Olson. „Prediction of Austenitic Weld Metal Microstructure and Properties". W: *Weld. J. (Miami); (United States)* (1985).

[2] J. Vitek et al. „Improved Ferrite Number Prediction Model That Accounts for Cooling Rate Effects Part 1: Model Development Details of a Prediction Model Based on a Neural Network System of Analysis Are Described". W: *Semantic Scholar* (2003).

[3] *Measure Ferrite Content of Austenitic and Duplex Steel*. source. Diverse.

[4] R. Saluja. „Formation, Quantification and Significance of Delta Ferrite for 300 Series Stainless Steel Weldments". W: *Academia.edu* (2015).

[5] S. S. Babu et al. „New Model for Prediction of Ferrite Number of Stainless Steel Welds". W: *Taylor & Francis* (2013).

[6] J. Vitek et al. „Improved Ferrite Number Prediction Model That Accounts for Cooling Rate Effects: Part 2: Model Results". W: *Semantic Scholar* (2003).

[7] M. Vasudevan et al. „Prediction of Ferrite Number in Stainless Steel Welds Using Bayesian Neural Network Model". W: *Welding in the World, Springer-Verlag* (2013).

[8] H. K. D. H. Bhadeshia. „Neural Networks in Materials Science". W: *ISIJ International, The Iron and Steel Institute of Japan* (2007).

[9] A. Y. Badmos i H. K. D. H. Bhadeshia. „Tensile Properties of Mechanically Alloyed Oxide Dispersion Strengthened Iron Alloys Part 2 – Physical Interpretation of Yield Strength". W: *Taylor & Francis* (2013).

[10] I. Santos et al. „Machine-learning-based Mechanical Properties Prediction in Foundry Production". W: *IEEE Xplore* (2009).

[11] J. Nieves et al. „Mechanical Properties Prediction in High-Precision Foundry Production". W: *IEEE Xplore* (2009).

[12] *Klasyfikacja Statystyczna*. *https://pl.wikipedia.org/wiki/Klasyfikacja_statystyczna*. 2019.

[13] Y. Y. Yang et al. „Tensile Strength Prediction for Hot Rolled Steels by Bayesian Neural Network Model". W: *IFAC Proceedings Volumes, Elsevier* (2016).

[14] Y. Wang et al. „Prediction and Analysis of Tensile Properties of Austenitic Stainless Steel Using Artificial Neural Network". W: *MDPI, Multidisciplinary Digital Publishing Institute* (2020).

[15] Y. K. Penya i in. „Advanced fault prediction in high-precision foundry production". W: *IEEE Xplore* (2008).

[16] S. M. Azimi et al. „Advanced Steel Microstructural Classification by Deep Learning Methods". W: *Nature News, Nature Publishing Group* (2018).

[17] S. L. Shrestha et al. „An Automated Method of Quantifying Ferrite Microstructures Using Electron Backscatter Diffraction (EBSD) Data". W: *Ultramicroscopy* (2013).

[18] A. Schneider D. Britz J. Webel i F. Mücklich. „Identifying And Quantifying Microstructures in Low-alloyed Steels: A Correlative Approach". W: *Metallurgia Italiana* (2017).

[19] D. Britz J. Pauly i F. Mücklich. „Advanced Microstructure Classification Using Data Mining Methods". W: *Computational Materials Science* (2016).

[20] A. R. Durmaz et al. „A Deep Learning Approach for Complex Microstructure Inference". W: *Research Square* (2021).

[21] A. Stoll i P. Benner. „Machine Learning for Material Characterization with an Application for Predicting Mechanical Properties". W: *Wiley Online Library* (2021).

[22] T. M. Mitchell. „Machine Learning". W: *McGraw-Hill* (1997).

[23] K. Sawka. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow*. Wydawnictwo Helion, 2018.

[24] *Tablica pomyłek*. https://pl.wikipedia.org/wiki/Tablica_pomyłek. 2020.

[25] K. Sawka. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow. Wydanie II*. Wydawnictwo Helion, 2020.

[26] A. Burns, B. Dobbing i T. Vardanega. *Machine Learning and AI via Brain simulations*. Spraw. tech. Stanford University & Google, 2014.

[27] *Big Data: Week 3 Video 3 - Feature Engineering*. Spraw. tech. CCNMTL, 2014.

[28] C. Shorten i T. M. Khoshgoftaar. „A Survey on Image Data Augmentation for Deep Learning". W: *Journal of Big Data, Springer International Publishing* (2019).

[29] A. Handhi. *Data Augmentation | How to use Deep Learning when you have Limited Data — Part 2*. Spraw. tech. Nanonets, 2021.

[30] D. Sarkar. *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning*. Spraw. tech. Medium, Towards Data Science, 2018.

[31] P. Chmiel. *Czym Jest Transfer Learning? Trenowanie Sieci Neuronowych w Deep Learning'u.* Spraw. tech. Greenlogic, 2020.

[32] S. Bickel. „ECML-PKDD Discovery Challenge 2006 Overview". W: *CiteSeerX* (2006).

[33] B. E. Boser et al. „A Training Algorithm for Optimal Margin Classifiers | Proceedings of the Fifth Annual Workshop on Computational Learning Theory". W: *EECS* (1992).

[34] A. B. Hur et al. „Support Vector Clustering". W: *The Journal of Machine Learning Research* (2002).

[35] *Drzewo decyzyjne*. *https://pl.wikipedia.org/wiki/Drzewo_decyzyjne*. 2020.

[36] J. Grus. *Data science od podstaw. Analiza danych w Pythonie*. Wydawnictwo Helion, 2018.

[37] J. R. Quinlan. „Improved Use of Continuous Attributes in C4.5". W: *ArXiv.org* (1996).

[38] *Decision tree*. *https://en.wikipedia.org/wiki/Decision_tree*. 2021.

[39] T. K. Ho. „Random Decision Forests". W: *IEEE Xplore* (1995).

[40] P. Ramchandani. *Random Forests and the Bias-Variance Tradeoff*. Spraw. tech. Medium, Towards Data Science, 2021.

[41] *What is the difference between Bagging and Boosting?* Spraw. tech. QuantDare, 2016.

[42] P. A. Jaskowiak i R.J.G.B. Campello. „Comparing Correlation Coefficients as Dissimilarity Measures for Cancer Classification in Gene Expression Data". W: *ResearchOnline@JCU | Brazilian Computer Society* (1970).

[43] S. M. Piryonesi i T. E. El-Diraby. „Role of Data Analytics in Infrastructure Asset Management: Overcoming Data Size and Quality Problems". W: *Journal of Transportation Engineering* (2020).

[44] *K-Nearest Neighbors Algorithm*. *https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm*. 2021.

[45] *Logistic regression*. *https://en.wikipedia.org/wiki/Logistic_regression*. 2021.

[46] S. Menard. „Applied Logistic Regression Analysis". W: *SAGE Publications* (2002).

[47] *Naive Bayes Classifier*. *https://en.wikipedia.org/wiki/Naive_Bayes_classifier*. 2021.

[48] V. Kurama. *A Guide To Understanding AdaBoost*. Spraw. tech. Paperspace Blog, 2021.

[49] S. M. Piryonesi i T. E. El-Diraby. „Data Analytics in Asset Management: Cost-Effective Prediction of the Pavement Condition Index". W: *Journal of Infrastructure Systems, American Society of Civil Engineers* (2019).

[50] C. Li. *A Gentle Introduction to Gradient Boosting*. Spraw. tech. College of Computer and Information ScienceNortheastern University.

[51] R. Tadeusiewicz i M. Szaleniec. *Leksykon Sieci Neuronowych*. Academia.edu, Projekt Nauka Fundacja na rzecz promocji nauki polskiej, 2015.

[52] K. Simonyan i A. Zisserman. „Very Deep Convolutional Networks for Large-Scale Image Recognition". W: *ArXiv.org* (2015).

[53] J. R. Koza et al. „Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming". W: *Kluwer Academic Publishers* (1996).

[54] *Mikrostruktura żeliwa – Część 1: Klasyfikacja wydzieleń grafitu na podstawie analizy wizualnej*. Spraw. tech. KT 301, Odlewnictwa, Polski Komitet Normalizacyjny, 2018.