

A G H

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: *dr hab. Bartłomiej Śnieżyński, prof. AGH*

Kraków, 2021

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

Spis treści

1. Wstęp.....	9
1.1. Wprowadzenie	9
1.2. Cel i zakres pracy	9
1.3. Zawartość pracy.....	10
2. Stan badań	11
2.1. Powiązane prace – zarys historyczny	11
2.1.1. Predykcja wartości liczby ferrytowej.....	11
2.1.2. Predykcja własności mechanicznych odlewów	13
2.1.3. Predykcja błędów w produkcji odlewniczej	15
2.2. Powiązane prace – stan aktualny	15
2.2.1. Predykcja cech i jakości odlewów za pomocą metod rozpoznawania obrazów	15
2.2.2. Predykcja cech i jakości odlewów za pomocą metod głębokiego uczenia	16
2.3. Wnioski.....	18
3. Uczenie maszynowe.....	21
3.1. Pojęcia podstawowe.....	21
3.2. Typy uczenia maszynowego	24
3.2.1. Uczenie nadzorowane	24
3.2.2. Uczenie częściowo nadzorowane.....	24
3.2.3. Uczenie nienadzorowane	25
3.2.4. Uczenie przez wzmacnianie.....	25
3.3. Inżynieria cech.....	25
3.4. Augmentacja danych	26
3.5. Uczenie się przez transfer.....	27
3.6. Wykorzystane metody uczenia maszynowego	28
3.6.1. Maszyna wektorów nośnych.....	28
3.6.2. Drzewo decyzyjne.....	29
3.6.3. Las losowy	30

3.6.4. K najbliższych sąsiadów	31
3.6.5. Regresja logistyczna	32
3.6.6. Naiwny klasyfikator bayesowski	33
3.6.7. Wzmacnianie.....	34
3.6.8. Sieci neuronowe.....	37
3.6.9. Pozostałe zagadnienia	40
4. Przygotowanie danych.....	41
4.1. Pozyskanie danych	41
4.2. Zrozumienie danych	41
4.3. Przetwarzanie danych	44
4.3.1. Normalizacja przybliżenia	45
4.3.2. Normalizacja skali	45
4.4. Augmentacja.....	47
5. Eksperymenty i wyniki	49
5.1. Opis podejścia	49
5.2. Klasyfikacja struktur.....	50
5.2.1. Uogólniona transformata Hougha.....	50
5.2.2. Detekcja krawędzi filtrem Canny'ego.....	52
5.2.3. Wycinanie pojedynczych struktur z obrazków	53
5.2.4. Rozpoznawanie wyciętych struktur	56
5.2.5. Wnioski	59
5.3. Ocena jakości odlewów	60
5.3.1. Momenty Hu oraz tekstury Haralicka.....	62
5.3.2. Klasyfikacja za pomocą liczby struktur	65
5.3.3. Sieci neuronowe.....	82
5.3.4. Podejście hybrydowe	82
6. Podsumowanie i wnioski.....	85
Bibliografia	87

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 wskażujących 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 jest przez nią wykonywanych 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 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. Rozwiążanie przedstawione w tej pracy pozwoliłoby zaoszczędzić środki za zużyte materiały, a także czas potrzebny na „ręczne” zidentyfikowanie jakości odlewu.

Niniejsza praca jest kontynuacją poprzednich badań [1], w których operowano na tych samych danych. We wspomnianej pracy głównym aspektem był przegląd literatury oraz przygotowanie danych. Praca ta była współkoordynowana przez opiekuna tej pracy.

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 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. Przedstawione zostały również techniki za pomocą których możliwe było rozszerzenie zbioru danych, a także zbalansowanie nierówności w liczności przykładów pomiędzy różnymi klasami.

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

W ostatnim rozdziale 6 zostały zawarte podsumowania badań oraz wnioski, jakie można było z nich wyciągnąć. Zostały również zaproponowane dalsze prace wymagane w celu udoskonalenia użytych metod i tym samym ulepszenia otrzymanych wyników.

2. Stan badań

W tym rozdziale postaramy się pokazać w jaki sposób badacze podchodzili do problemów powiązanych z tematem niniejszej pracy a także postaramy się rozszerzyć te podejścia.

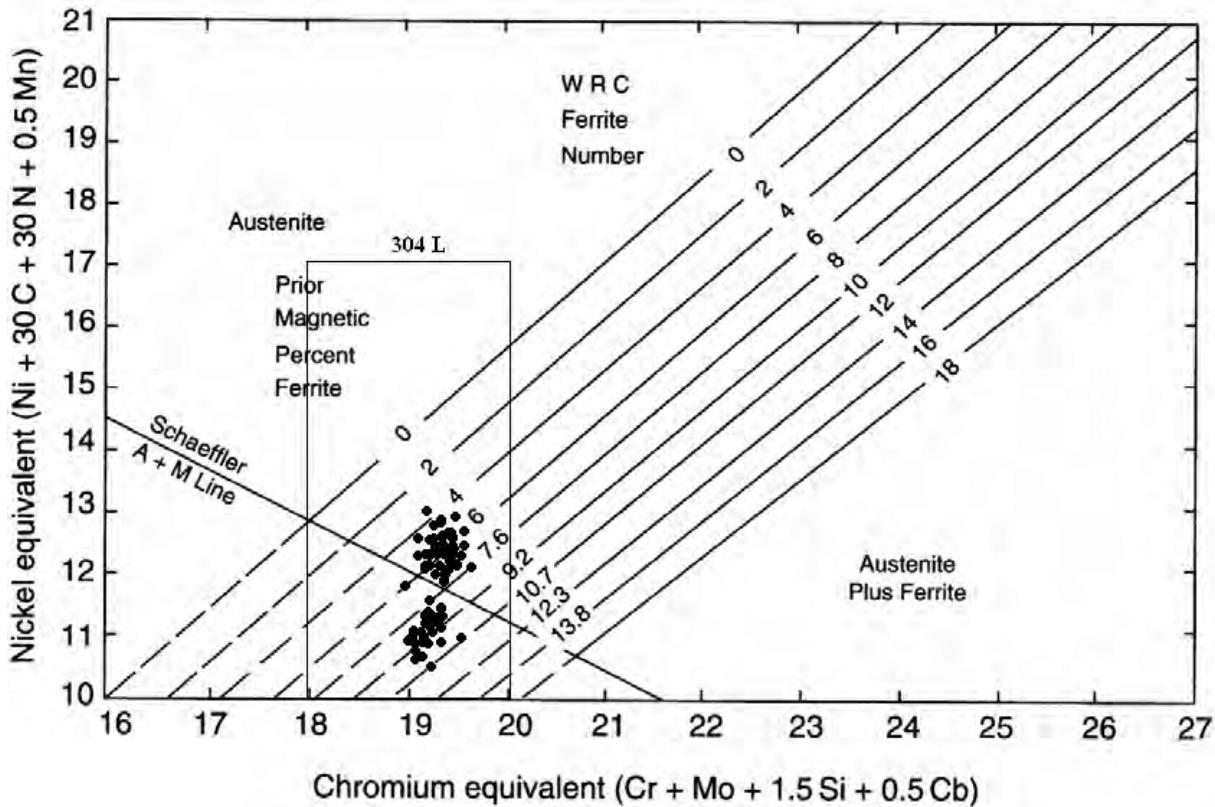
2.1. Powiązane prace – zarys historyczny

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 [2] mówiono o tym, że przyszłe techniki predykcyjne będą opierać się raczej na wyrażenach matematycznych czy sztucznej inteligencji niż na diagramach. Ówczesnie do przewidywania fazy mikrostruktury spoiny metalu wykorzystywano właśnie diagramy (rys. 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 [4] 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 pęknięcia [5, 6], z drugiej strony wysoki poziom ferrytu prowadzi do niższej odporności na korozję [6]).

W pracy Babu i in. [7] 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 [7],



Rys. 2.1. Przykładowy schemat do wyznaczania ferrytu δ z granicami składu [3]

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).

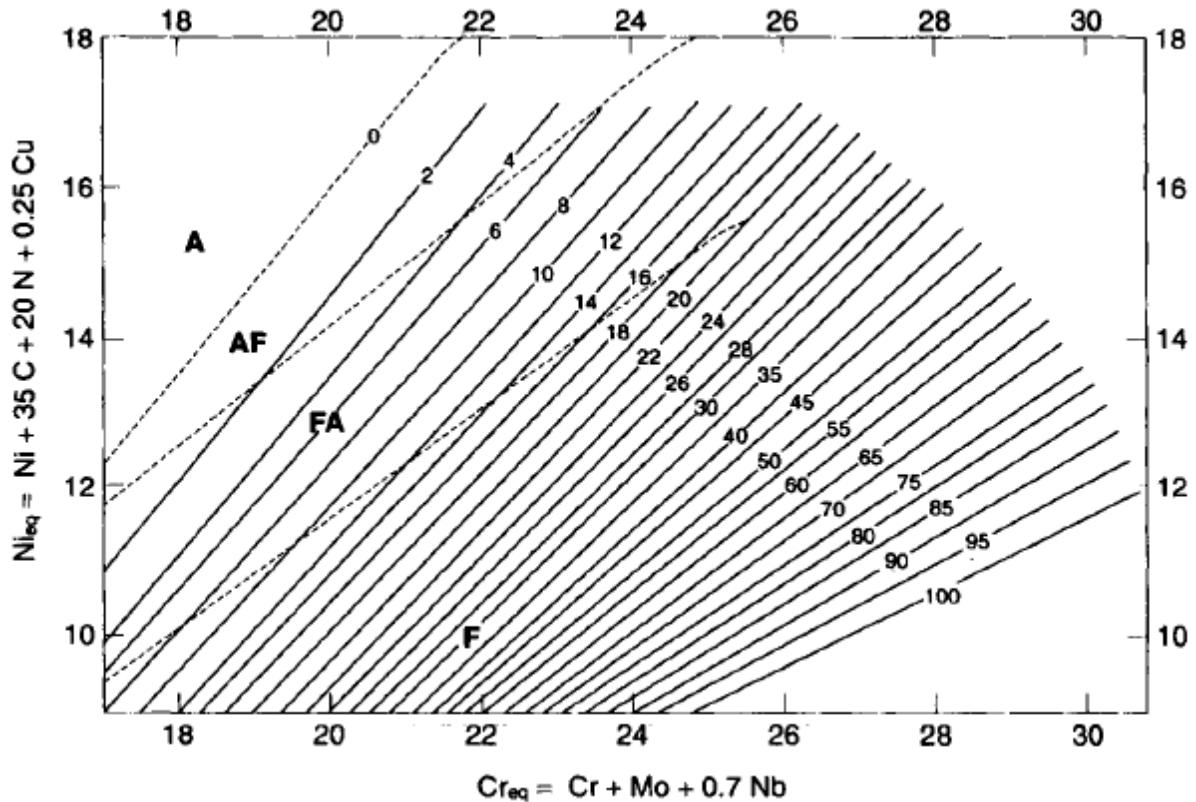
Równania na Cr_{eq} i Ni_{eq} są następujące (zgodnie z rys. 2.2):

$$Cr_{eq} = Cr + Mo + 0.7Nb \quad (2.1)$$

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

gdzie symbole pierwiastków przedstawiają procentowy udział wagi każdego pierwiastka. W kolejnej pracy [4] dotyczącej predykcji FN ci sami autorzy wykorzystali sieć neuronową, która na wejściu przyjmowała procentowy udział wagi 13 pierwiastków (Fe , Cr , Ni , C , N , Mo , Mn , Si , Cu , Ti , Nb , V i Co), czyli posiadała warstwę wejściową z 13 neuronami, następnie warstwę ukrytą z sześcioma neuronami, natomiast na wyjściu był pojedynczy neuron, który zwracał liczbę ferrytową (rys. 2.3).

Wyniki zostały przedstawione w [9] i jak się okazało, testowana sieć zwracała lepsze wyniki od jakichkolwiek dotychczasowych podejść z błędem RMS (średnia kwadratowa od ang. *root mean square*) mniejszym o 50% od poprzedniej najlepszej metody. W ostatnim przytoczonym artykule dotyczącym predykcji FN [10] również zastosowano sieć neuronową, a konkretnie bayesowską sieć neuronową



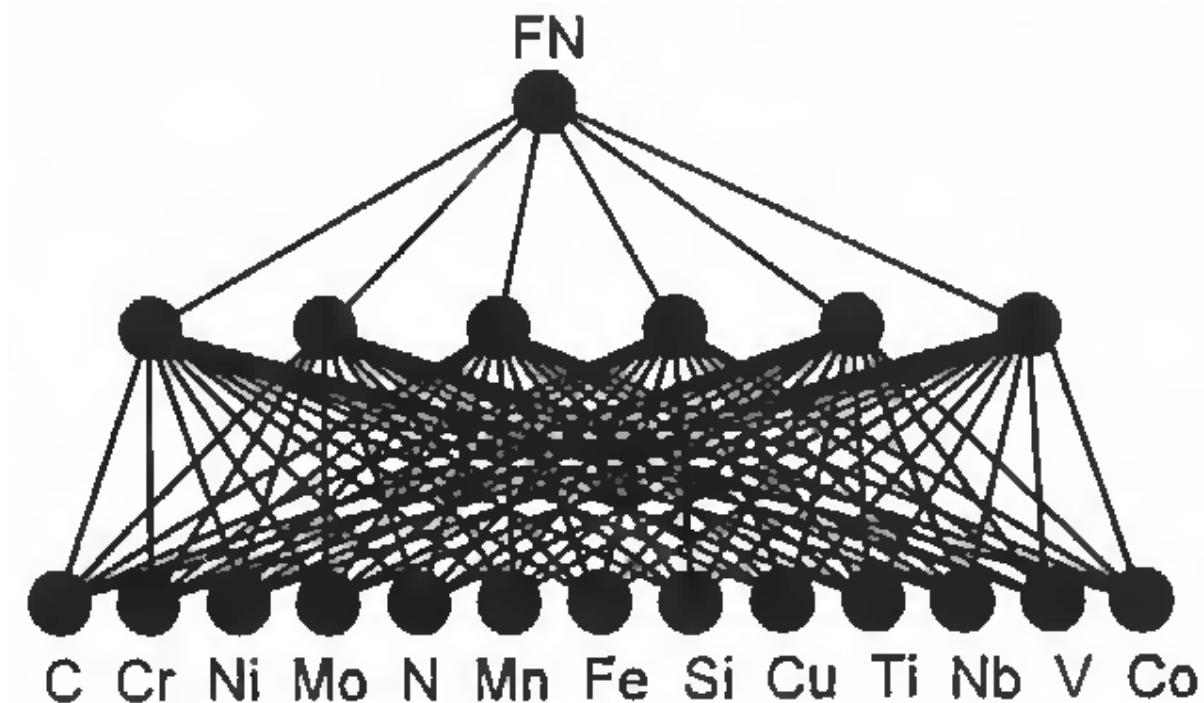
Rys. 2.2. Diagram WRC-1992 [8]

(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 nieużywanych w szkoleniu.

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 [11, 12]. W obydwu tych pracach zastosowano sieci neuronowe oraz uzasadniono, że modele matematyczne nie radzą sobie 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 w zależności od funkcji przyłożonego obciążenia, czasu i wielu innych. Jest ona jednym z ważniejszych czynników do uwzględnienia, gdyż



Rys. 2.3. Model sieci neuronowej ORFN [4, 9]

niewystarczająca wartość wytrzymałości może mieć fatalne skutki (jak np. zawalenie się konstrukcji). Innym powodem może być to, iż jedynym sposobem na zbadanie wartości wytrzymałości jest prowadzenie badań niszczących, co powoduje wzrost kosztów produkcji [13]. Jednym ze sposobów analizy wartości UTS jest predykcja za pomocą wartości różnych właściwości odlewu. W przywołanej wcześniej pracy [13] oraz [14] są to skład chemiczny, rozmiar odlewu, prędkość chłodzenia, obróbka termiczna. Mając dane w postaci wartości rozdzielonych 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 [15]. 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 [16]:

- skład chemiczny metalu oraz zmienne procesu walcania, jak temperatura, czy przebieg;
- dane na temat mikrostruktury.

W pracy tej jako dane wejściowe wykorzystano skład chemiczny, specyfikacje geometryczne oraz zmienne dotyczące procesu rolowania, natomiast modelem użytym 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 (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 [17], gdzie użyto sieci neuronowej, a na jej wejściu podawano 20 zmiennych, takich jak skład chemiczny czy warunki obróbki cieplnej, ale również czynniki typu temperatura badania, gdyż warunki eksplotacji również mają wpływ na właściwości wytrzymałościowe. W ten sposób uzyskano 93% wartości współczynnika R-kwadrat dla granicy plastyczności (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. [18], a mianowicie ponownie zostały wykorzystane bayesowskie sieci neuronowe, lecz tym razem w celu predykcji obecności mikrouruskodzeń w odlewie przed lub w trakcie procesu odlewnictwa. Dzięki takim danym można wpływać 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

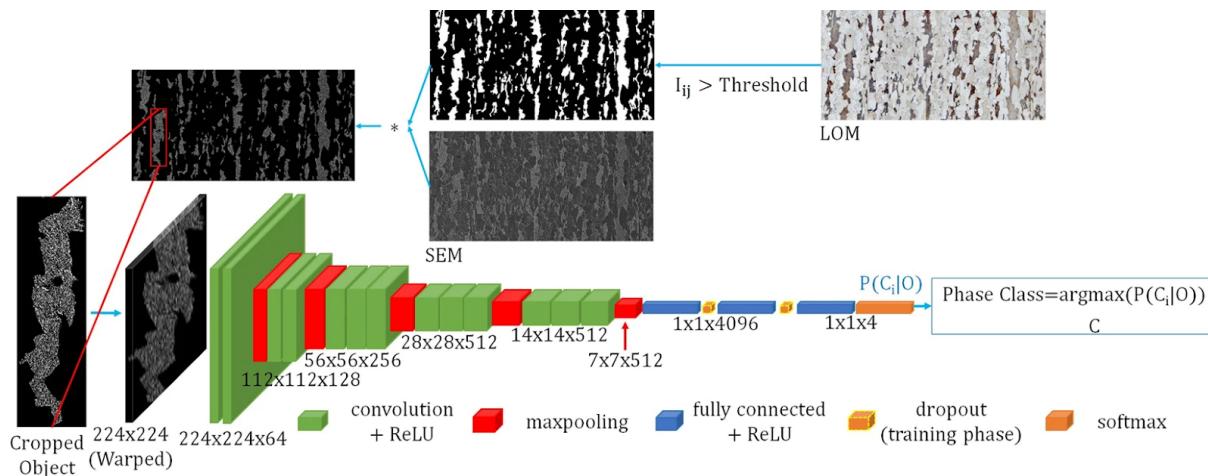
W pierwszej przytoczonej tutaj pracy opracowano nową zautomatyzowaną metodę wykorzystującą dyfrakcję wstecznie rozproszonych elektronów (EBSD, od ang. *electron backscatter diffraction*), aby skutecznie identyfikować i określić ilościowo mikroskładniki ferrytowe w złożonych mikrostrukturach różnych gatunków stali z wykorzystaniem redukcji szumu [19]. 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 [20] 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.

2.2.2. Predykcja cech i jakości odlewów za pomocą metod głębokiego uczenia

Wewnętrzna struktura materiału nazywana jest mikrostrukturą. Określa ona wszystkie jego właściwości fizyczne i chemiczne. Pomimo tego, że charakterystyka mikrostrukturalna jest szeroko rozpoznawiona 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. Wcześniejste artykuły zazwyczaj oddzielały fazę klasyfikacji mikrostruktur od fazy ekstrakcji cech [21]. 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.

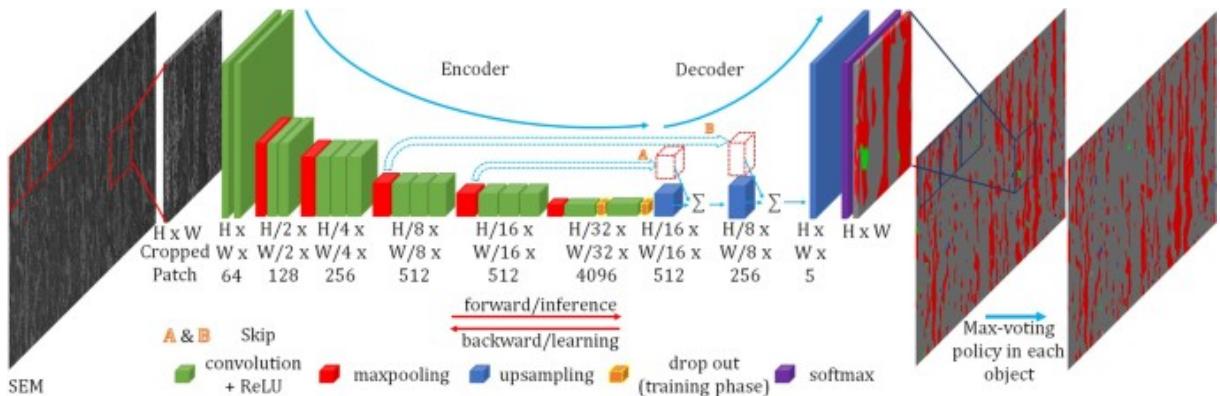
Pierwsza przeanalizowana praca [21] dotyczy klasyfikacji mikrostrukturalnej składników stali niskowęglowej za pomocą metod głębokiego uczenia. Takie podejście może być pomocne przy ocenie jakości danego odlewu jak i predykcji jego właściwości mechanicznych. W artykule wykorzystano w pełni spłotową sieć neuronową (ang. *Fully Convolutional Neural Network*, FCNN), na której wejście podawano segmentowane piksele. System ten osiągnął niemal 94% skuteczność, tym samym przewyższając najlepszą dotychczas metodę. Jak stwierdzają autorzy, nie tylko wynik jest doskonały sam w sobie, ale również wyznacza linię dla przyszłych badań, m.in. związanych z oceną jakości stali (i nie tylko). W pracy tej, oprócz wspomnianego modelu, przetestowano również szereg innych metod oraz podejścia. 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. Jednakże, jak zauważono w pracy, podejście to ma jedną wielką wadę,



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 [21]

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 tekstrą. Po przeprowadzeniu testów, przy użyciu tego podejścia uzyskano wynik na poziomie 49% dokładności (który był dotychczas najlepszy). Drugie podejście, jakie zostało przetestowane w [21] 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. obiekowi jest przypisywana klasa mikrostruktury z maksymalną liczbą sklasyfikowanych pikseli wewnętrz 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



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 dla wynikowego zszytego obrazu [21]

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

Kolejną pracą, z obiecującymi wynikami, która również korzysta z uczenia głębokiego, jest [23]. Celem owej pracy jest segmentacja struktur w stali o złożonej fazie. Autorzy zastosowali w niej sieci

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. [22]	oparte na obiektach	—	48.89%
CIFAR-Net	oparte na obiektach	od zera (ang. <i>from scratch</i>)	57.03%
VGG19 + SVM	oparte na obiektach	—	64.84%
VGG16	oparte na obiektach	strojenie (ang. <i>fine tuning</i>)	66.50%
MVFCNN ^a	oparte na pikselach	strojenie (ang. <i>fine tuning</i>)	93.94%

^a Metoda MVFCNN to metoda badana w przytoczonym artykule [21]. 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%...

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.

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, wiążkość, twardość, kruchosć 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 [24]. 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 odlewów, 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 [21, 22], 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. semantyczną segmentację obrazu [21, 22], 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 obliczać 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 [25]. Algorytmy uczenia maszynowego są budowane na podstawie przykładowych danych, zwanych „danymi szkoleniowymi”, w celu prognozowania lub podejmowania decyzji bez bezpośredniego zaprogramowania do tego [26]. 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 przygotowania 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 [27].

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 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 przeciwnie zjawisko, tzn. gdy model jest zbyt prosty (w stosunku do danych). W tym przypadku istnieje kilka rozwiązań, jak [27]:

- 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 ważyć 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 [27]:

- 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 utrwało 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 [27].

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

natomiaszt zweryfikować jak dobrze sobie radzi dany klasyfikator? Aby to określić w statystyce używa się wielu miar, które pokrótkę zostaną tutaj przedstawione. Zaczniemy 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 [27, 28]:

- 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. Pożądany wynikiem jest uzyskanie wysokich wartości prawdziwie pozytywnej oraz prawdziwie negatywnej miary, co jest równoznaczne z wysoką wartością liczb na głównej przekątnej tablicy pomyłek, natomiast jak najmniejsze (najlepiej zerowe) wartości na pozostałej przekątnej (wszystkie pozostałe wartości poza główną przekątną). Dzięki tej macierzy możemy przeanalizować w jakich przypadkach nasz klasyfikator myli się najczęściej. Dalej można wprowadzić bardziej zwięzłe metryki [27]:

- dokładność (ang. *accuracy*, ACC)

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \quad (3.1)$$

- precyzja (ang. *precision*, PPV)

$$PPV = \frac{TP}{TP + FP} \quad (3.2)$$

- czułość (ang. *recall*, TPR)

$$TPR = \frac{TP}{TP + FN} \quad (3.3)$$

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 [27]. Pod tym względem możemy wyróżnić cztery główne typy uczenia maszynowego, 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 nadzorcy jako pary $< \text{obiekt uczący; etykieta} >$ [29]. Celem takiego systemu jest nauczenie się przewidywania prawidłowej odpowiedzi dla danego obiektu wejściowego oraz generalizacja na przypadki, które nie są obecne w danych uczących [29]. 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. Jednymi z bardziej znanych 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 [27].

3.2.3. Uczenie nienadzorowane

Uczenie nienadzorowane (ang. *unsupervised learning*) polega na wykrywaniu wzorców, relacji na podstawie nieoznaczonych danych, możliwe 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 (działan). 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 i na tej podstawie wykonywać akcje, następnie odbierać nagrody lub kary. Jego zadaniem jest nauczyć się najlepszej strategii wybierania akcji, zwanej **polityką**, co prowadzi do maksymalizacji nagrody [27]. Ze względu na charakterystykę tego nurtu uczenia maszynowego jest ono często stosowane do uczenia modeli grania w gry [29].

3.3. Inżynieria cech

Inżynieria cech jest procesem wykorzystania wiedzy dziedzinowej w celu ekstrakcji cech z surowych danych [30]. Cecha jest własnością każdej instancji danych i jest ona wykorzystywana przez model w celu predykcji. Odpowiednio przygotowane dane zwiększą skuteczność modeli [30]. Proces tworzenia cech składa się z sześciu głównych etapów [31]:

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 częstek 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 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 [32]. 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 postaci 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 wewnętrz. W przypadku skalowania na zewnątrz otrzymujemy obraz o większym rozmiarze, a więc wycinając odpowiedni obszar możemytrzymać 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 przeskalać 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 [32]. Istnieje też wiele innych, bardziej zaawansowanych metod augmentacji danych [33]. 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 wewnętrz jednocześnie zachowując oryginalny rozmiar obrazu [33].



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 [34]. A więc w najprostszych słowach, uczenie się przez transfer (ang. *transfer learning*) 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 [35]. 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 [35]. 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 [36].

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 i rodzaj zadania.

3.6.1. Maszyna wektorów nośnych

Maszyna wektorów nośnych (ang. *support-vector machine*, SVM) to model nadzorowanego uczenia maszynowego, który wyznacza hiperpłaszczyznę w celu rozdzielenia przykładów należących do dwóch klas z maksymalnym marginesem [29]. SVM to potężny i wszechstronny model uczenia maszynowego, który jest w stanie przeprowadzić klasyfikację liniową, nieliniową oraz regresję [27]. 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 [37].

Jednym z problemów, które można napotkać w trakcie korzystania z SVM jest ich czułość na skalę cech, tzn. otrzymywane wyniki mogą być niepoprawne, gdy wykorzystane cechy będą liczbami o różnych



Rys. 3.2. Działanie SVM. H_1 nie separuje klas, H_2 separuje, lecz z małym marginesem, natomiast H_3 separuje z maksymalnym marginesem (źródło: https://en.wikipedia.org/wiki/Support-vector_machine)

rzędach wielkości. Jednakże łatwo jest to obejść przy pomocy przeskalowania cech (np. standaryzacja). Innym problemem jest konieczność posiadania całkowicie oznaczonych danych [38]. Dodatkowo parametry tego modelu 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*, DT) to algorytm uczenia maszynowego stosowany w celu pozyskiwania wiedzy na podstawie przykładów [39]. 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 [40]. 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, do 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 przyrost 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 [41]:

- obsługa atrybutów ciągłych i dyskretnych,
- obsługa danych z brakującymi wartościami atrybutów,
- przycinanie drzew po ich utworzeniu.

Wśród największych zalet drzew decyzyjnych wymienia się [42]:

- prosty algorytm,
- łatwo interpretowalne wyniki,
- działają nawet dla niewielkiej liczby danych,
- nie wymagają przygotowywania danych (w szczególności skalowania) [27].

Natomiast posiadają też wady, takie jak [42]:

- 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 kolejnym podrozdziale (3.6.3).

3.6.3. Las losowy

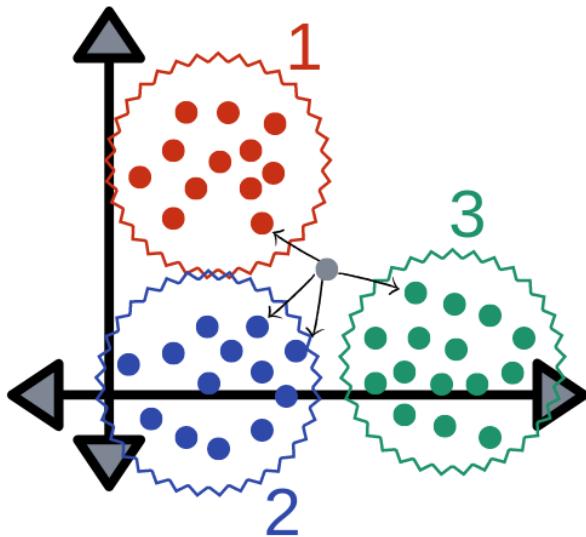
Jak zostało wspomniane wyżej, drzewa decyzyjne mogą zbyt mocno dopasować się do danych treningowych. Jednym z rozwiązań jest technika **lasów losowych** (ang. *random forest*, RFC), która polega na tworzeniu wielu drzew decyzyjnych, a następnie klasyfikacji w drodze głosowania [40], a więc jako odpowiedź jest generowana dominanta klas (klasyfikacja) lub średnia przewidywana wartość (regresja) [43]. 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 [27]. Dzięki temu podejściu zmniejsza się wariancja modelu bez zwiększenia obciążenia [44], 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 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*) [45]. 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 na trudne dane. 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 [40]. 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 [46]. Jeżeli cechy danych znacznie różnią się w skali, bądź reprezentują inne jednostki, wtedy normalizacja może znacząco poprawić dokładność [47]. 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) [48]. k-NN odnotowuje dobre wyniki w zakresie spójności, jest łatwy w implementacji, natomiast może być wymagający obliczeniowo w przypadku dużych zbiorów danych [48]. Jego użyteczność jest największa w przypadku, gdy zależność między zmiennymi objaśniającymi a objaśnianymi jest złożona [48]. Na poniższym rysunku zostało przedstawione schematycznie działanie k-NN (rys. 3.3). Wzmacnianie zostało szczegółowo omówione w rozdziale 3.6.7.



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)

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 [49]. Mechanizm działania tej metody jest podobny do regresji liniowej, tyle że tutaj wyliczamy ważoną sumę cech wejściowych (wzór 3.4)

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1 - p_i}\right) = \beta_1 x_{1,i} + \dots + \beta_k x_{k,i} \quad (3.4)$$

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) [50]. Następnie zwracana jest wartość funkcji logistycznej z tego rezultatu (wzór 3.5) [27], co jest równoznaczne z oszacowaniem prawdopodobieństwa:

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

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

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

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 eksponentów) [27]:

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

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 [50]. 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):

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

gdzie M oznacza liczbę klas, y to wskaźnik binarny (0, gdy predykcja c jest niezgodna z obserwacją c lub 1 w przeciwnym przypadku) 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 klasyfikatorów probabilistycznych, które opierają się na twierdzeniu Bayesa [51]. Istotnym założeniem tego modelu jest wzajemna niezależność predyktorów. Naiwne klasyfikatory bayesowskie są wysoce skalowalne, aczkolwiek wymagają wielu parametrów liniowych w stosunku do liczby zmiennych) [51]. Dużym atutem tego modelu jest zastosowanie metody największej wiarygodności, przez co czas uczenia jest liniowy (w przeciwieństwie do wielu innych typów klasyfikatorów) [51], 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 [51]. Korzystając z twierdzenia Bayesa można wyprowadzić wzór, który stoi za naiwnym modelem probabilistycznym Bayesa (wzór 3.9):

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

gdzie C to dana klasa zmiennej zależnej, n to liczba zmiennych niezależnych, F_1, \dots, F_n to zmienne niezależne, natomiast Z jest współczynnikiem skalowania zależnym od predyktorów [51]. Klasyfikacja odbywa według poniższej funkcji (wzór 3.10):

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

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 [51].

3.6.7. Wzmacnianie

Wzmacnianie (ang. *boosting*) zostało już pokrótko omówione przy okazji lasów losowych (rozdział 3.6.3). W skrócie, 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 [27]. 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 [27]. 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):

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

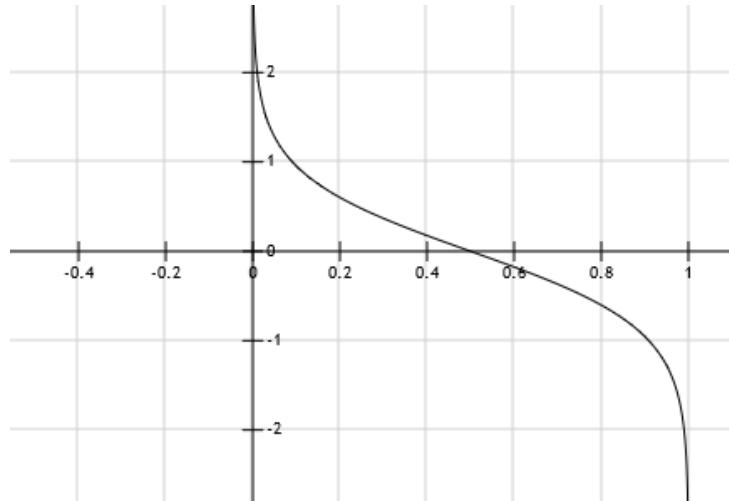
gdzie m to liczba próbek. Należy również pamiętać, że suma wszystkich wag wynosi zawsze jeden. Po wytrenowaniu pierwszego klasyfikatora zostanie mu przypisany ważony współczynnik błędu r_j , gdzie j oznacza liczbę porządkową klasyfikatora (równanie 3.12):

$$r_j = \frac{\sum_{i=1}^m w^{(i)}}{\sum_{i=1}^m w^{(i)}} \quad (3.12)$$

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:

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j} \quad (3.13)$$

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



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)

lejnym etapie następuje aktualizacja wag próbek (równanie 3.14):

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

Cały ten proces jest powtarzany wielokrotnie, iteracyjnie, aż osiągniemy ustaloną liczbę klasyfikatorów bądź też wyniki zwarcane przez model będą na odpowiednim poziomie (zazwyczaj uczenie zatrzymuje się przy 100%) [27]. 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):

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

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 [52]. 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 [52].

3.6.7.2. Wzmacnianie gradientowe

Wzmacnianie gradientowe (ang. *gradient boosting*) to inna bardzo popularna technika wzmacniania [27]. 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 [27]. Zwyczajowo jako słabych predyktorów używa się drzew decyzyjnych, a otrzymany algorytm jest nazywany gradientowo wzmacnionym drzewem (ang. *gradient boosted tree*). Wyniki uzyskiwane za pomocą tej techniki zazwyczaj są lepsze od tych uzyskiwanych za pomocą lasów losowych [53]. Jedno z prostszych wyjaśnień tej metody przedstawił Cheng Li na przykładzie regresji [54]. Założymy, że chcemy nauczyć model F przewidywać wartości postaci (3.16):

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

minimalizując błąd średniokwadratowy (3.17):

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

gdzie:

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

$$h_m(x) = y - F_m(x) \quad (3.18)$$

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):

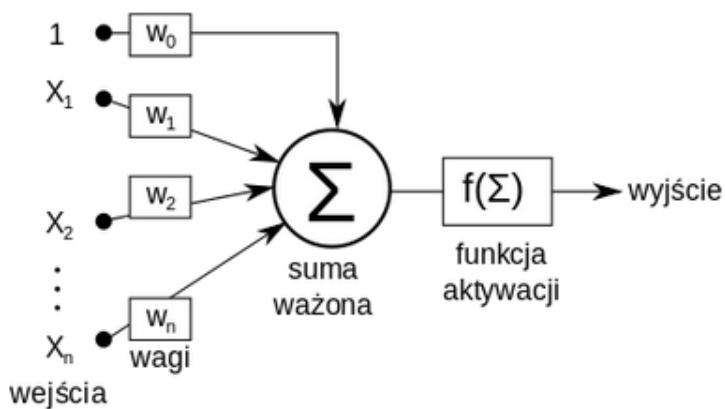
$$L_{MSE} = \frac{1}{2}(y - F(x))^2 \quad (3.19)$$

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

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

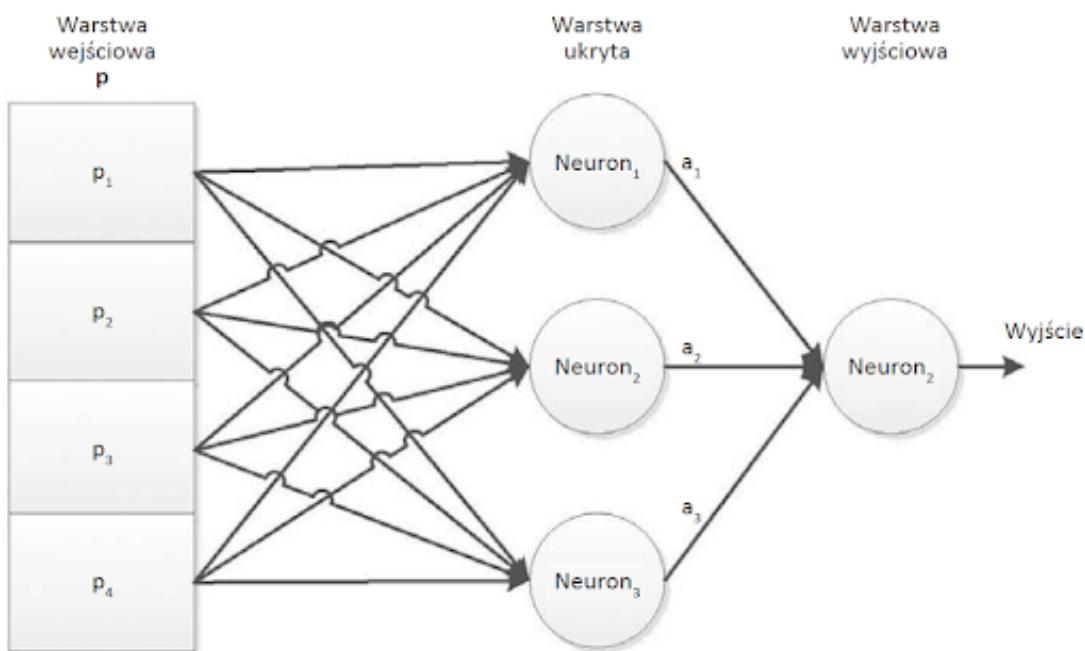
Sztuczne sieci neuronowe (ang. *artificial neural networks*), zwykle nazywane po prostu **sieciami neuronowymi** (ang. *neural networks*), to systemy komputerowe inspirowane biologicznymi sieciami neuronowymi takimi jak mózgi zwierząt [55]. Tak samo jak w przypadku wcześniej omówionych metod, główną 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 przemnożone 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 rys 3.5. Neurony znajdują się w **warstwach** (ang. *layers*).



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

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*). Neurony w obrębie danej warstwy pracują na podobnych danych (przykładowo neurony w warstwie wejściowej sieci neuronowej mogą operować na pikselach obrazu wejściowego w przypadku klasyfikacji obrazów) i nie przekazują między sobą żadnych informacji. Kolejnym elementem są **połączenia** (ang. *connections*) oraz **wagi** (ang. *weights*). Sieć składa 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*). Schemat sieci neuronowej został przedstawiony na poniższym obrazku (rys. 3.6). Uczenie takich sieci polega na modyfikacji wag



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

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 zazwyczaj za pomocą **propagacji wstecznej** (ang. *backpropagation*). Polega ona na obliczeniu gradientu z **funkcji kosztu** (ang. *cost function*), czyli z średniej funkcji straty dla wielu próbek, dla danych próbek wejściowych względem wag.

Każda 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 (głębbsza, dla wielkich i skomplikowanych danych bądź płystsza, 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).

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 trenowań warstw) [56]. Dodatkowo na końcu 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 wejściu przyjmowała obrazki w formacie RGB (ang. *red, green, blue*) o rozmiarze 224×224 (rys. 3.7). Model ten dobrze uogólnia się dla innych zadań i zbiorów danych [56],

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					
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 zastosowana funkcja aktywacji ReLU. Źródło: [56]

i chociaż 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.

3.6.9. Pozostałe zagadnienia

W tej sekcji zostaną omówione różne techniki oraz metody wykorzystane w trakcie badań. Zostaną też wytłumaczone niektóre pojęcia, które obejmują swoim zakresem wiele tematyk i niepraktyczne by-łoby tworzenie dla każdego problemu osobnego rozdziału.

3.6.9.1. Optymalizacja hiperparametrów

Hiperparametry zostały omówione w rozdz. 3.6.8. Przypomnijmy, że są to parametry, które kontrolują proces uczenia się i są ustalane przed uczeniem. W zależności od ich konfiguracji wyniki mogą się znacznie różnić. Stąd takie ważne jest, aby odpowiednio dobrać ich wartości. W jaki sposób się to robi? Osoby dowiadczone mogą od razu ustawić odpowiednie wartości dzięki swojej praktyce. Natomiast aby mieć pewność, można skorzystać z dostępnych metod. Istnieje ich wiele, natomiast tutaj zostanie przedstawiona jedna, która została wykorzystana w niniejszej pracy: mowa o przeszukiwaniu siatki (ang. *grid search*). Metoda ta polega na wyczerpującym przeszukiwaniu (ang. *exhaustive searching*) ręcznie przygotowanego podzbioru przestrzeni hiperparametrów algorytmu uczącego [57]. Następnie dla każdego zestawu hiperparametrów zostaje wytrenowany model na danych uczących, po czym jego skuteczność zostaje zweryfikowana na podstawie rezultatów otrzymanych podczas ewaluacji modelu z wykorzystaniem danych testowych.

3.6.9.2. Sprawdzian krzyżowy

Sprawdzian krzyżowy (ang. *cross-validation*) jest to metoda statystyczna, w której dzieli się próbę statystyczną na podzbiory, po czym przeprowadza się analizy, ale tylko na niektórych z nich. Te, które się wykorzystuje nazywane są zbiorem uczącym, natomiast pozostałe dane służą do weryfikacji poprawności wyników i nazywane są one danymi testowymi. Istnieje wiele rodzajów validacji krzyżowej, aczkolwiek w niniejszej pracy został wykorzystany jeden, a mianowicie sprawdzian k -krotny. Polega on na podziale oryginalnej próby na k podzbiorów, po czym kolejno każdy z nich występuje jako zbiór testowy, podczas gdy pozostałe razem jako zbiór uczący. Następnie wykonuje się analizę (k razy), a uzyskane k rezultaty łączy się, np. uśredniając wyniki, dzięki czemu są one bardziej wiarygodne, a także otrzymujemy jeden wynik.

3.6.9.3. Strata logistyczna

W pracy jako metrykę oceny wydajności klasyfikatora wykorzystano stratę logistyczną (ang. *log-loss*), której wzór podano poniżej (3.21):

$$L_{\log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \quad (3.21)$$

gdzie:

- y oznacza prawdziwą wartość etykiety (0 lub 1 dla klasyfikacji binarnej),
- p to prawdopodobieństwo $P(y = 1)$.

4. Przygotowanie danych

Prace związane z przygotowaniem danych były niezbędne przed rozpoczęciem właściwych badań. Rozdział ten składa się z opisu chronologicznie wykonywanych zadań. Oczywiście, jeszcze przed etapem przygotowywania danych przeprowadzono przegląd literatury, który został opisany w rozdziale 2. Dodatkowo kroki te były również wykorzystane w badaniach nad podobnym problemem, w którym pracowano na tych samych danych [1]. W niektórych miejscach tej pracy wykorzystano metody opracowane również w ramach tamtych badań.

4.1. Pozyskanie danych

Dane zostały pozyskane za pośrednictwem dr hab. inż. Doroty Wilk-Kołodziejczyk z badań naukowych nad nowoczesnymi technologiami materiałowymi [58, 59].

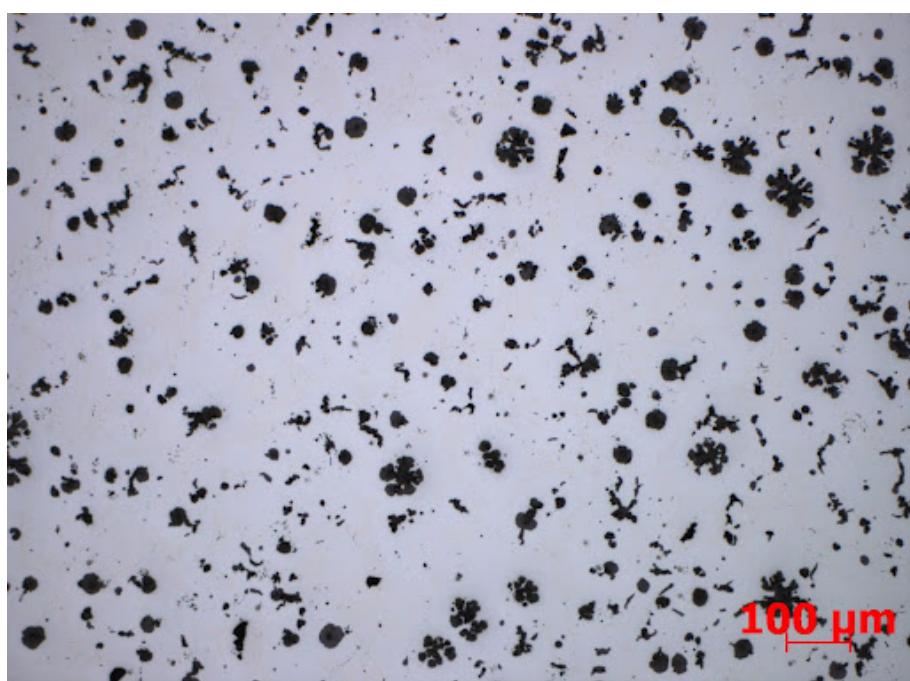
4.2. Zrozumienie danych

Gdy już uzyskano dostęp do danych w postaci zdjęć mikrostruktury odlewów, kolejnym etapem (wg metodologii CRISP-DM będącej schematem ilustrującym ogólny proces eksploracji danych [60]) jest zrozumienie danych, a mówiąc precyzyjniej, ocena przydatności danych. Obrazy, które otrzymaliśmy były zamieszczone w dwóch folderach i dotyczyły dwóch właściwości mechanicznych odlewów, a mianowicie:

- **granicy sprężystości (R_p)** – dane zostały podzielone ze względu na mniejszą oraz większą wartość granicy sprężystości. Podzbiór zawierający obrazy z mniejszą wartością granicy sprężystości składał się z 68 instancji, natomiast podzbiór zawierający obrazy z większą wartością granicy sprężystości składał się z 47 instancji. Wszystkie obrazy w tym zbiorze danych miały wymiary 1388x1040. Łącznie w tym zbiorze znajduje się 115 instancji.
- **wytrzymałości na rozciąganie (R_m)** – dane zostały podzielone ze względu na mniejszą oraz większą wartość wytrzymałości na rozciąganie. Podzbiór zawierający obrazy z mniejszą wartością wytrzymałości na rozciąganie składał się z 65 instancji, natomiast podzbiór zawierający obrazy z większą wartością wytrzymałości na rozciąganie składał się z 158 instancji, a zatem łącznie

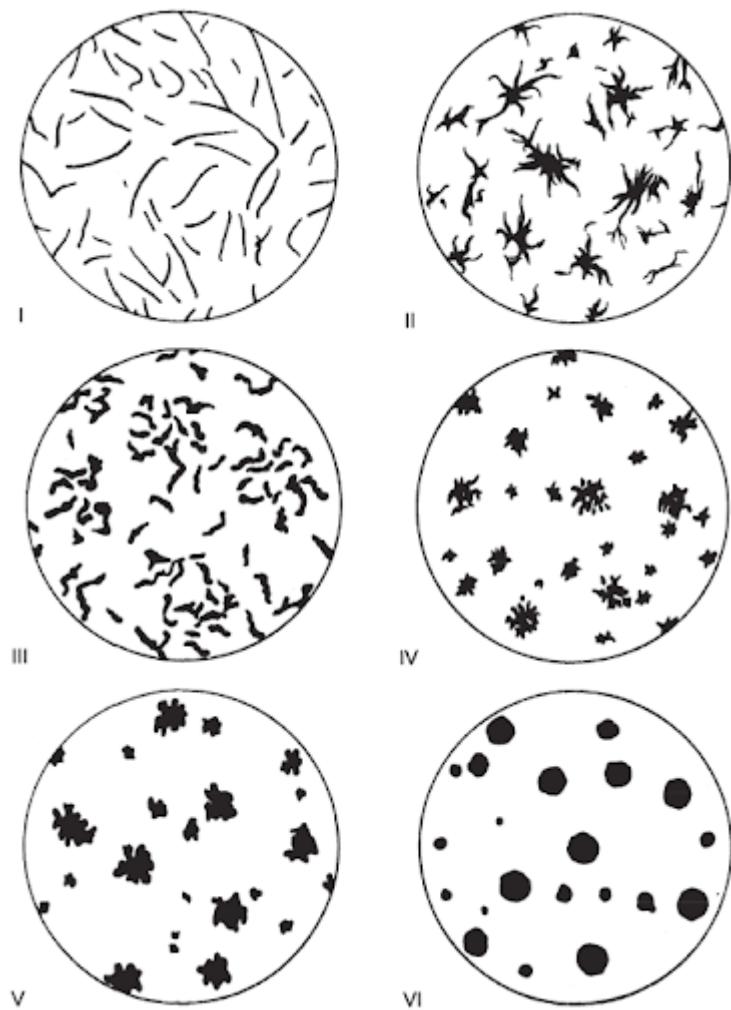
w tym zbiorze znajdują się 223 obrazy. Tym razem jednak 138 instancji ma wymiary 1388x1040, natomiast pozostałe 85 instancji posiada wymiary 2080x1540.

Wszystkie zdjęcia zostały wykonane techniką LOM. Dodatkowo wszystkie zdjęcia są w formacie RGB, a więc posiadają trzy kanały. Ażkolwiek po konwersji do skali szarości praktycznie nie widać różnicy. Zdecydowana większość z nich posiada przybliżenie stokrotne bądź też pięćsetkrotne, natomiast zdają się również zdjęcia o przybliżeniu dwunastoipółkrotnym, czy też pięćdziesięciokrotnym (choć są to pojedyncze przypadki). Przybliżenie zdjęcia można odczytać z nazwy jego pliku, ażkolwiek zdjęcia o tym samym przybliżeniu wciąż mogą się lekko różnić w swojej skali. Biblioteka *PIL* (ang. *Python Imaging Package*) została wykorzystana do załadowania wszystkich obrazów użytych w badaniu. Natomiast wyniki zliczania struktur (rozdz. 5.2) zostały zapisane jako tablice liczb całkowitych przy użyciu biblioteki *numpy*. Dzięki wykorzystaniu tych bibliotek manipulowanie obrazkami (a co za tym idzie tablicami liczb) było bezproblemowe. Dodatkowo obie biblioteki umożliwiają zapisywanie tych informacji do plików. Jest to szczególnie korzystne w późniejszych fazach, kiedy sieć neuronowa określa liczbę poszczególnych klas struktur na obrazach i każdorazowe wyznaczanie tych liczb dla wszystkich zebranych zdjęć zajmuje w przybliżeniu godzinę. Natomiast odczytanie tych danych po zapisaniu do pliku zajmuje tylko kilka minut. Pakiet *pickle*, który pozwala na zapisanie dowolnego obiektu Pythona do pliku, może być również przydatny do tego celu. W tym przypadku zapisywane są listy, których elementy są tablicami z biblioteki *numpy* oraz . Na rysunku 4.1 widać przykładową mikrostrukturę. W



Rys. 4.1. Przykładowe zdjęcie mikrostruktury. Przedstawia ono instancję ze zbioru R_m , gdzie zdjęcia są pogrupowane ze względu na wytrzymałość na rozciąganie. W tym przypadku jest to mikrostruktura o małej odporności. Źródło: [58]

prawym dolnym rogu każdego ujęcia znajduje się skala zaznaczona na czerwono, która pokazuje rzeczywisty rozmiar określonego fragmentu ze zdjęcia. Czarne struktury na szarym tle to stały motyw na tych fotografiach. Są to składniki strukturalne stopów żelazo-węgiel, które dzielimy na sześć głównych kategorii na podstawie ich kształtu. Klasy te przedstawiono na rysunku 4.2. Będą one wykorzystywane jako dane wejściowe (tj. liczba struktur każdej z kategorii) dla tradycyjnych modeli kategoryzacji w badaniach. Oznacza to, że podobnie jak specjalisci w tej dziedzinie, spróbujemy prognozować właściwości mechaniczne odlewów na podstawie liczby różnych form w mikrostrukturach. Porównanie wiedzy specjalistycznej z wiedzą wydobytą przez klasyfikatory będzie proste, dzięki czemu modele będą wysoce interpretowalne. Na końcu będzie można porównać te wyniki z wynikami sieci neuronowej. W tym celu zostanie wykorzystane biblioteka *numpy*, aby załadować i przechowywać dane. Klasyfikatory zostały zainstalowane z pakietu *scikit-learn*, który dostarcza wszystkie wymagane przez nas metody. Ponieważ



Rys. 4.2. Obrazy referencyjne dla głównych form grafitowych w żeliwie. Różne kształty występują w rozmaitych rodzajach żeliwa przez co mogą mieć niejednakowy wpływ na wytrzymałość na rozciąganie. Źródło: [61]

dane w postaci kształtów nie są dostępne, wytworzenie takich danych, jak również infrastruktury do ich wyodrębniania, musiało być wykonane oddzielnie, co szczegółowo opisano w rozdziale 4.3.

4.3. Przetwarzanie danych

Postępując zgodnie z metodologią CRISP-DM, kolejnym etapem jest przygotowanie danych, które składa się z następujących podetapów:

- czyszczenie danych – między innymi usunięcie zdjęć, które zbytnio odstawały od reszty instancji (bądź też były zaszumione, nie zawierały informacji umożliwiających ich zidentyfikowanie),
- wykonanie przekształceń – normalizacja danych, sprowadzenie zdjęć do tych samych rozmiarów (rozdzielczości), a także tej samej skali.

Postępując zgodnie z podaną kolejnością, na samym początku usunięto wszystkie zdjęcia najsłabszej jakości oraz te, które nie były wykonane techniką LOM bądź też reprezentowały odlewy innego typu. Przykładowa taka instancja jest przedstawiona na rysunku 4.3. Natomiast druga faza, a więc normaliza-



Rys. 4.3. Jeden z kilku odrzuconych obrazów. Porównując go z rysunkiem 4.1 widać, że wielką trudność sprawiłoby wyodrębnienie czarnych struktur (patrz rys. 4.2).
 Źródło: [58]

cja danych, jest procesem znacznie bardziej istotnym i trudniejszym. Praca nad tym aspektem projektu trwała wiele dni i została podzielona na kilka etapów:

1. normalizacja przybliżenia – jak wcześniej wspomniano, fotografie miały różne przybliżenia, dla tego pierwszym krokiem było ich sprowadzenie do tego samego przybliżenia. Zdecydowano, że najlepszą opcją jest powiększenie wszystkich zdjęć do przybliżenia 500x, co ma kilka zalet, w tym prostotę i rozszerzenie zbioru danych.

2. normalizacja skali – mimo że obrazy były w przybliżeniu podobne, zdarzało się, że długość podziałki dla jednego obrazu różniła się od długości podziałki dla innego obrazu.

Poniższe sekcje zawierają wyjaśnienie tych etapów.

4.3.1. Normalizacja przybliżenia

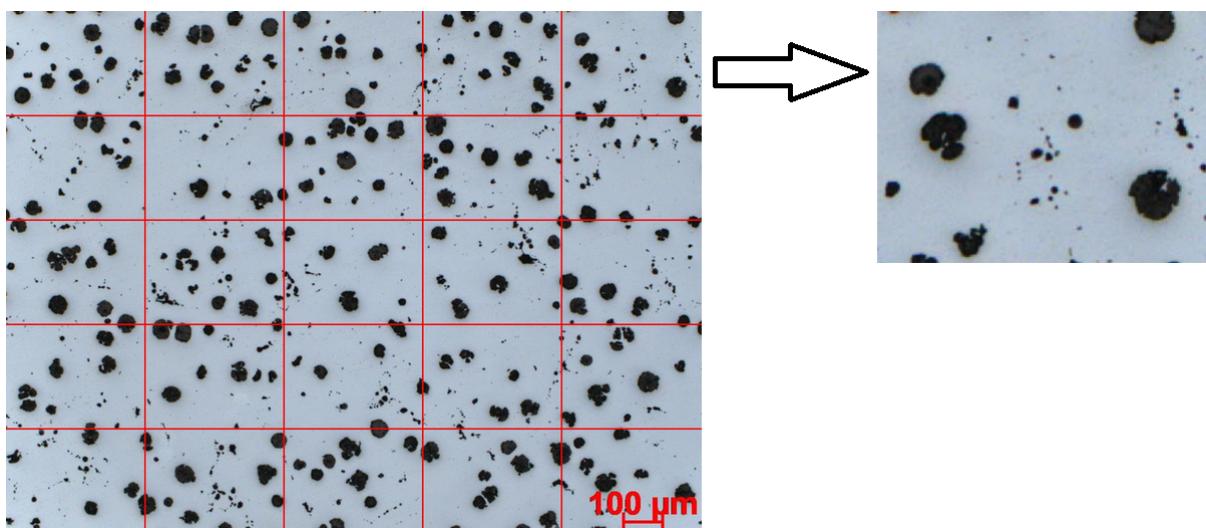
W tym kroku wszystkie zdjęcia zostały wczytane, a następnie wszystkie fotografie o przybliżeniu różnym niż 500 razy zostały pocięte na mniejsze. Obrazy o powiększeniu 500x są zachowywane, natomiast obrazy w powiększeniu około 100x są dzielone na 25 mniejszych części (rys. 4.4) i zapisywane w nowej bazie danych. Ponadto rozmiar ramki dla zdjęć wynosił 335 x 251 (szerokość x wysokość). Po tej fazie uzyskaliśmy 2400 fotografii z początkowej liczby 115 zdjęć mikrostruktur dla kategorii R_p oraz 4757 zdjęć z początkowej liczby 223 zdjęć mikrostruktur dla kategorii R_m . Podsumowując, statystyki dla obu baz danych po tej fazie są następujące:

- R_m (wytrzymałość na rozciąganie):
 - liczba zdjęć reprezentujących niską wytrzymałość wzrosła z 65 do 1337,
 - liczba zdjęć reprezentujących wysoką wytrzymałość wzrosła ze 158 do 3358,
 - łączna liczba zdjęć wzrosła z 223 do 4757,
 - 24 zdjęcia miały przybliżenie 500x, 199 zdjęć miało przybliżenie 100x;
- R_p (granica sprężystości):
 - liczba zdjęć w tym przypadku również średnio wzrosła aż dwudziestokrotnie

Dodatkowo w trakcie badania wyeliminowano zdjęcia ze znaczną ilością czerwonego odcienia (czyli takie, na których skala mocno zniekształca obraz). Udało się to osiągnąć, wykonując analizę histograficzną kanału czerwonego w zakresie 220-256 i usuwając zdjęcia, które zawierały ponad 3000 pikseli w tym przedziale. W rezultacie liczba obrazków została nieznacznie zredukowana (całkowita liczba zdjęć spała z 4999 do 4695). Manipulacja zdjęciami była możliwa między innymi dzięki bibliotece *PIL*.

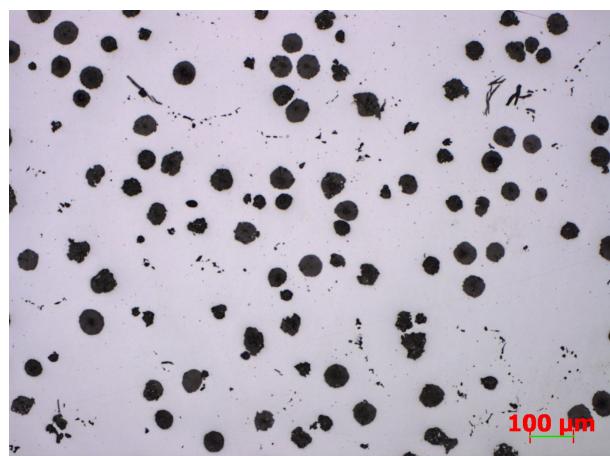
4.3.2. Normalizacja skali

Jak już wspomniano wcześniej, zdarzało się, iż mimo tego samego przybliżenia, dwa zdjęcia nieznacznie różniły się pod względem skali. Można to było zauważać na skali, która znajduje się na każdej fotografii w prawym dolnym rogu. Mimo, iż zdjęcia miały to samo przybliżenie, to ta sama odległość na dwóch różnych zdjęciach mogła odpowiadać innej długości w rzeczywistości. Dlatego zdecydowano, aby znormalizować dane również pod tym względem. Rozwiązanie zostało zaczerpnięte z pracy [1], w której operowano na tym samym zestawie danych, a polegało ono na redukcji wszystkich kolorów oprócz tego, którego użyto do oznaczenia podziałki. Dzięki zastosowaniu tego podejścia możliwe było zeskalowanie wszystkich zdjęć, dzięki czemu mieliśmy pewność, że wszystkie zdjęcia są w tej samej



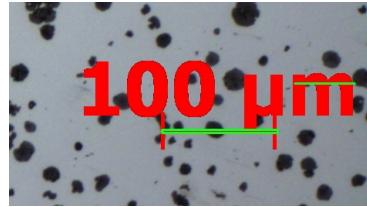
Rys. 4.4. Schemat rozcinania obrazków o przybliżeniu 100x na 25 mniejszych fragmentów. Źródło: opracowanie własne z użyciem danych z [58]

skali. Następnie, za pomocą operacji morfologicznych takich jak erozja czy rozszerzanie (w pythonie `cv.erode` oraz `cv.dilate`) wykrywano poziomą linię, dzięki czemu można było wyznaczyć jej końce, a tym samym obliczyć jej długość. Na rys. 4.5 przedstawiono wizualny wynik tych operacji. Wynik liczbowy natomiast był zapisywany w nazwie pliku. Dzięki takiej wizualnej reprezentacji można było



Rys. 4.5. Przykładowy obrazek z poprawnie wykrytą na nim podziałką skali. Źródło: opracowanie własne przy wykorzystaniu [1, 58]

zweryfikować poprawność w ten sposób wyznaczonej długości podziałki. Ze względu na to, iż tekst reprezentujący rzeczywistą długość podziałki był w tym samym kolorze zdarzały się błędy, jak na rysunku 4.6. Aczkolwiek po przeanalizowaniu wyników okazuje się, że skuteczność tej metody wynosi aż 99%, a więc bez problemu można ją wykorzystać w dalszych badaniach. Ewaluację przeprowadzono na 223 oryginalnych obrazkach. Po zmierzeniu długości podziałki obliczano jak bardzo należy przyciąć zdjęcie, które po użyciu skalowania wracało do tych samych rozmiarów (natomiast długość podziałki w ten



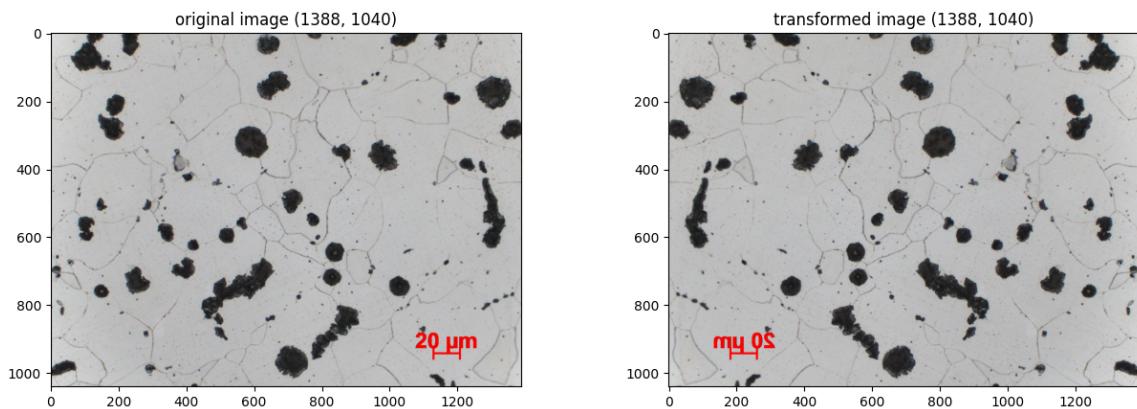
Rys. 4.6. Przykładowy obrazek z niepoprawnie wykrytą na nim podziałką skali. Źródło: opracowanie własne przy wykorzystaniu [1, 58]

sposób osiągała oczekiwana długość). Oczekwaną długość podziałki wyznaczono w ten sposób, aby konieczność zastosowania tych operacji występowała w jak najmniejszej liczbie zdjęć.

4.4. Augmentacja

Augmentacja, a więc rozszerzenie danych to techniki służące do zwiększenia ilości danych poprzez modyfikację kopii istniejących danych (patrz 3.4). Proces ten można było opisać w poprzednim punkcie, czyli przygotowanie danych, a w szczególności wykonanie przekształceń. Jednakże jest to obszerny temat, który zawiera w sobie wiele różnych podejść, dlatego powięcono mu cały osobny podrozdział.

W badaniach wykorzystano kilka rodzajów augmentacji. Najpierw zastosowano technikę zwaną przycinaniem (ang. *cropping*). Polegała ona na pocięciu obrazów o przybliżeniu 100x na 25 mniejszych części. Proces ten został szczegółowo opisany w rozdziale 4.3.1. Nastecną techniką wykorzystaną w badaniach było odbicie (ang. *flipping*). Dzięki tej technice możliwe jest nawet czterokrotne powiększenie



Rys. 4.7. Wynik zastosowania techniki odbicia na przykładowym zdjęciu (po prawej). Widzimy, że rozmiary zdjęcia się nie zmieniają. Źródło: opracowanie własne z użyciem danych z [58]

zestawu danych! Dodatkową, bardzo dużą zaletą tej metody jest fakt, że nie zmienia ona wymiarów obrazów. Jest to szczególnie korzystne, ponieważ posiadane dane nie mają tej samej szerokości i długości, co uniemożliwia wykorzystanie technik modyfikujących wymiary. Ze względu na dużą nierównowagę klas

metoda ta jest wykorzystywana głównie do kompensowania liczności klas rzadkich. Przykład odbicia horyzontalnego został przedstawiony na rysunku 4.7.

5. Eksperymenty i wyniki

W tym rozdziale zostały przedstawione wszystkie wykonane badania wraz z ich wynikami. W badaniach zostały uwzględnione wszystkie metody uczenia maszynowego przedstawione w rozdziale 3 rozszerzając podejścia przytoczone w przeglądzie literatury (rozdział 2). Badania te stanowią uzupełnienie i rozszerzenie prac z [1].

5.1. Opis podejścia

Badania te zostały podzielone na dwa duże etapy. Pierwszy z nich polegał na rozpoznawaniu pojedynczych struktur obecnych na zdjęciach mikrostruktur. Wiązało się to z wyznaczeniem konturów tych obiektów, następnie ich wycięcie i utworzenie z nich bazy danych, co wymagało „ręcznego” przyporządkowania obrazów do siedmiu klas. Następnie, za pomocą tego zestawu danych możliwe byłoby wyuczenie sieci neuronowej rozpoznającej te kształty. Eksperymenty rozpoczęto od przetestowania momentów Hu oraz tekstu Haralicka w celu sklasyfikowania całych zdjęć mikrostruktur i stwierdzenia których kształtów jest najwięcej. Następnym krokiem było przetestowanie uogólnionej transformaty Hougha. Po zakończeniu tych testów rozpoczęto badania nad bardziej skomplikowanymi metodami, które można byłoby zastosować bardziej automatycznie. Szczegóły dotyczące wycinania, tworzenia bazy danych i rozpoznawania tych struktur zostały przedstawione w rozdziałach 5.2.3 oraz 5.2.4, które to zostały w całości poświęcone temu etapowi.

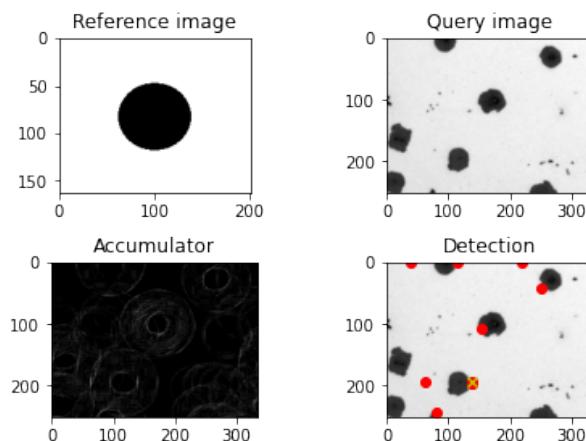
Drugi etap polegał na ocenie jakości odlewów za pomocą technik wypracowanych w pierwszej fazie, wykorzystując klasyczne klasyfikatory, ale również posługując się sieciami neuronowymi. W pierwszej kolejności wykorzystano momenty Hu w celu wyłuskania z obrazów cech, za pomocą których można byłoby przeprowadzić klasyfikację. Analogicznie przetestowano tekstury Haralicka. Cechy te podawano na wejście klasycznych klasyfikatorów, które zostały przedstawione w rozdziale 3.6. Następnie przeprowadzono kompleksowe badania, w których wykorzystywano liczbę struktur konkretnych klas znajdujących się na zdjęciach mikrostruktur, a także klasyczne klasyfikatory. Wszystkie otrzymane wyniki porównano uwzględniając wiele aspektów, jak interpretowalność wyników, łatwość implementacji, prostotę algorytmu czy czas uczenia. Aby mieć ogół widoku całej sytuacji przetestowano również najskuteczniejsze obecnie architektury sieci neuronowych wykorzystywanych w celach klasyfikacji obrazów i również zostały one porównane wielopłaszczyznowo z pozostałymi strategiami.

5.2. Klasyfikacja struktur

Jest to najszerzy obszar badań, ponieważ dzięki danym w postaci zliczonych struktur można sprawdzić, czy kształty tych struktur mają wpływ na właściwości mechaniczne odlewów, a jeśli tak, to w jakim stopniu. Jak dobrze wiadomo, sieci neuronowe, które otrzymują obrazy w postaci pikseli jako dane wejściowe i zwracają wynik w postaci „tak” lub „nie” odpowiadając na pytanie, czy wytrzymałość na rozciąganie danej mikrostruktury jest duża (lub mała) są modelami „czarnej skrzynki”, co oznacza, że trudno zweryfikować, dlaczego model podjął jedną decyzję nad drugą. Wykorzystując jednak dane w postaci liczby struktur i ich klas, można pokusić się o zbudowanie interpretowalnego modelu. W tym celu zostanie przetestowany szereg metod, z pomocą których będziemy w stanie wyodrębnić poszczególne obiekty na zdjęciach mikrostruktur, a następnie, za pomocą modelu sieci neuronowej będziemy mogli je rozpoznawać.

5.2.1. Uogólniona transformata Hougha

Jako pierwsze podejście zostały przetestowane momenty Hu oraz tekstury Haralicka. Metody te zostały również wykorzystane do oceny wytrzymałości odlewów, których zasada działania była podobna, aczkolwiek była ona bardziej rozbudowana, stąd odsyłamy do rozdz. 5.3.1, gdzie została szczegółowo opisana. W tym rozdziale zajmiemy się uogólnioną transformatą Hougha (ang. *generalised Hough transform, GHT*), która została przebadana jako następna. Stąd wykorzystano otwartoźródłową bibliotekę *general-hough* [62]. Ta strategia jest bardziej skoncentrowana na pojedynczych strukturach i ich detekcji.



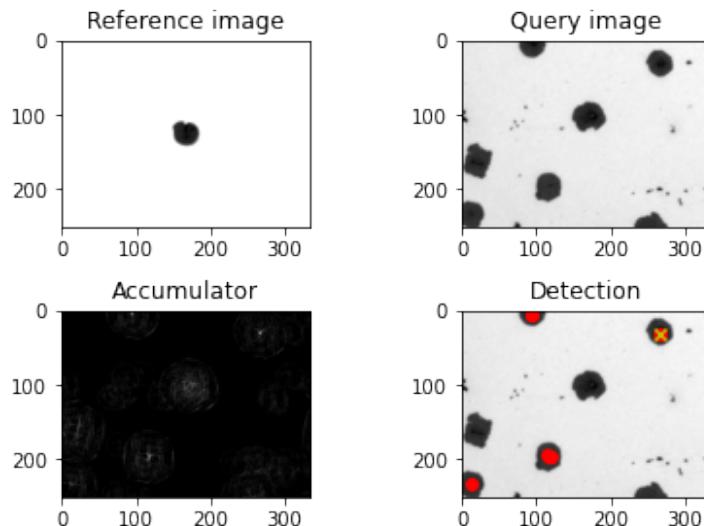
Rys. 5.1. Wykrywanie kształtu przedstawionego na obrazie referencyjnym (ang. *reference image*) w obrazie zapytania (ang. *query image*). W prawym dolnym rogu na czerwono zaznaczono wykryte kształty. Żółty krzyżyk oznacza punkt, który został wybrany za najbardziej podobny do referencyjnego, którym w tym przypadku jest czarne koło. Źródło: opracowanie własne przy wykorzystaniu biblioteki *general-hough*

Początkowe zmagania pokazały, że znajdowanie pojedynczych struktur danej klasy z pomocą tej metody

jest możliwe, aczkolwiek nie wszystkie kształty są wykrywane, a niekiedy zdarzają się nawet większe pomyłki, co przedstawiono na powyższym rysunku (rys. 5.1). Natomiast na rysunkach 5.2 i 5.3 pokazano wyniki dla zmodyfikowanych danych wejściowych oraz obrazów referencyjnych. Chociaż uważa się, że GHT ma zalety takie jak:

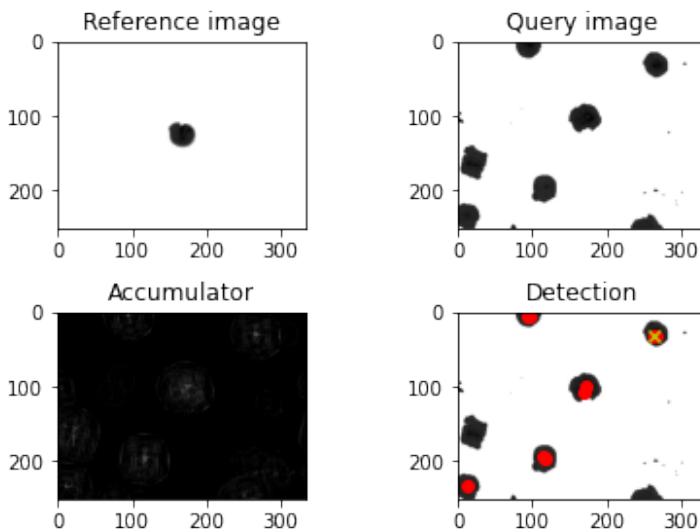
- odporność na częściowe lub nieznaczne zniekształcenie formy
- odporność na obecność innych struktur na obrazie,
- odporność na szum,
- możliwość znajdowania wielu wystąpień danego kształtu na obrazie wejściowym.

Pomimo tego, że metoda ta została stworzona właśnie do rozpoznawania form zdefiniowanych matematycznie (takie występują na naszych obrazach), jego działanie nie spełniło naszych oczekiwani. Spo-



Rys. 5.2. Działanie metody GHT na tym samym obrazie zapytania, lecz jako obraz referencyjny została użyta rzeczywista struktura znajdująca się na jednym ze zdjęć. Dodatkowo skala tej struktury jest taka sama, jak skala struktur na obrazie zapytania.
 Źródło: opracowanie własne przy wykorzystaniu biblioteki *general-hough*

glądarki na rys. 5.1 można zauważyć dwa problemy. Po pierwsze, nie wykryto wszystkich kształtów. Po drugie, czerwone kropki oznaczające wykryty kształt niekiedy znajdują się na szarym tle, zamiast na czarnym kształcie. Ale to nie jedyne problemy. Zdarza się również, iż dana struktura jest oznaczona wielokrotnie. Dlatego zdecydowano się poeksperymentować z obrazami referencyjnymi, co potencjalnie może poprawić wyniki. Na kolejnym rysunku 5.2 możemy zauważyć, że obraz zapytania jest wciąż ten sam, natomiast zmienił się obraz referencyjny. Tym razem jest to rzeczywista struktura, a więc nie jest ani idealnie czarna, ani idealnie okrągła. Dodatkowo jej skala jest taka sama, jak skala struktur na obrazie zapytania. Możemy również zaobserwować na obrazie z wykrytymi kształtami (ang. *detection*), iż tym razem czerwone kropki są rozmieszczone o wiele bardziej precyzyjnie. Jednakże wciąż metoda ta nie



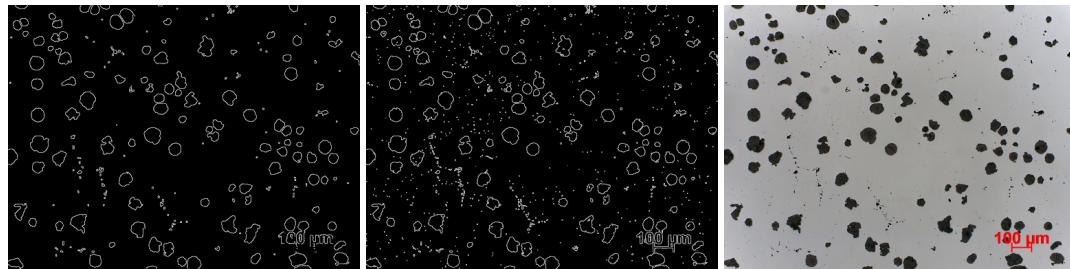
Rys. 5.3. Kolejny test z tym samym obrazem zapytania, a także z tym samym obrazem referencyjnym, co na rys. 5.2. Widzimy, że metoda zadziałała jeszcze lepiej, gdyż wykryła poprawnie jeszcze jedną strukturę. Źródło: opracowanie własne przy wykorzystaniu biblioteki *general-hough*

jest na wystarczająco dokładna. W kolejnych testach usunięto dodatkowo szare tło z obrazów zapytania, przez co szare struktury znalazły się po tych przekształceniach na białym tle. Przykładowe wyniki zostały zaprezentowane na obrazie 5.3. Możemy zauważyć, że ten zabieg przyniósł oczekiwane efekty, a mianowicie wykryto poprawnie o jeden kształt więcej, niż przed tym zabiegiem. Mimo wszystko takie podejście jest mało praktyczne, gdyż dla różnych obrazów należałoby dobierać obrazki referencyjne zgodnie ze skalą struktur na obrazach. Dodatkowo, pomimo tego, iż jest zaledwie sześć klas tych struktur, potrzebowalibyśmy znacznie więcej przykładów, aby pokryć przypadki, w których struktura jest zniekształcona, bądź nakłada się z inną strukturą. A więc wiemy już, że ta metoda nie może zostać wykorzystana w dalszych badaniach, a zatem należy wrócić do poszukiwań tej odpowiedniej metody.

5.2.2. Detekcja krawędzi filtrem Canny'ego

Kolejne techniki były bardziej skoncentrowane na wycinaniu poszczególnych struktur, a następnie kategoryzowaniu każdej z nich niezależnie w kolejnych fazach. W tym celu wykorzystano pakiet *opencv-python* (nazywana dalej *cv*). Jednak w pierwszym kroku opracowano algorytm do przeprowadzania prostego zliczania struktur bez rozróżniania klasy kształtu. Krawędzie struktur wykrywano metodą Canny'ego [63]. Dodatkowo przetestowano różne techniki rozmywania obrazu (ang. *blur*), czy inaczej – wygładzania, m.in. uśrednianie (w pythonie *cv.blur*), rozmycie gaussowskie (ang. *Gaussian blur*, w pythonie *cv.GaussianBlur*), rozmycie środkowe (ang. *median blurring*, w pythonie *cv.medianBlur*) i filtrowanie dwustronne (ang. *bilateral filtering*, w pythonie *cv.bilateralFilter*). W zależności od rozmiaru zastosowanego jądra (ang. *kernel*) otrzymujemy różne efekty przy rozmyciu. Rysunek 5.4 przedstawia trzy zdjęcia, z których jeden jest oryginalny (pierwszy od prawej), a dwa pozostałe to efekt zastosowania

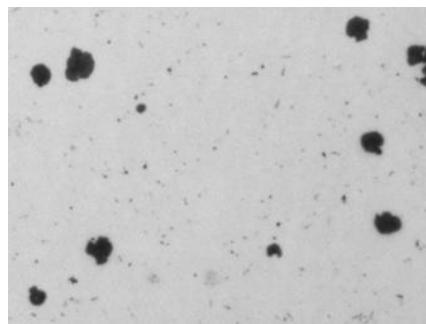
wykrywania krawędzi. W pierwszym od lewej zastosowano antialiasing (rozmiar jądra 5x5), podczas gdy w środkowym obrazie nie. Wykryte struktury zostały następnie zliczone. Są to wszystkie kontury



(a) Wykrywanie krawędzi z wykorzystaniem filtra Canny'ego po wygładzaniu
 (b) Wykrywanie krawędzi na oryginalnym obrazku
 (c) Oryginalne zdjęcie mikrostruktury

Rys. 5.4. Wykrywanie krawędzi z wykorzystaniem filtra Canny'ego. Źródło: opracowanie własne

widoczne na dwóch pierwszych obrazkach na rys. 5.4. Rysunek 5.5 przedstawia przykład obrazu, dla którego algorytm wyliczył dziesięć takich struktur (czy też konturów). Zliczanie konturów przeprowa-



Rys. 5.5. Przykładowe zdjęcie mikrostruktury, dla którego wykryto 10 kształtów

dzono w taki sposób, iż dla każdego obiektu obliczono pole powierzchni i obwód. W ten sposób można odrzucić kontury o małej powierzchni lub małym obwodzie. Próg został dobrany empirycznie, tzn. dla testowanych progów wyświetlane były wyniki wycinania kształtów i następnie wybrano próg, dla którego wyniki były najlepsze pod względem wizualnym (przede wszystkim ważne było, aby najmniejsze kształty, które ciężej sklasyfikować, zostały odrzucone). Wszystkie drobne, nieistotne kształty są eliminowane za pomocą tego podejścia. Podczas prób przebadano również takie metody jak progowanie (ang. *threshold*), dylatacja (ang. *dilation*), erozja (ang. *erosion*) i morfologia (ang. *morphology*), co zostało przedstawione w rozdz. 4.3.2. Mimo wszystko najlepsze wyniki osiągnięto podczas zliczania struktur na oryginalnym zdjęciu.

5.2.3. Wycinanie pojedynczych struktur z obrazków

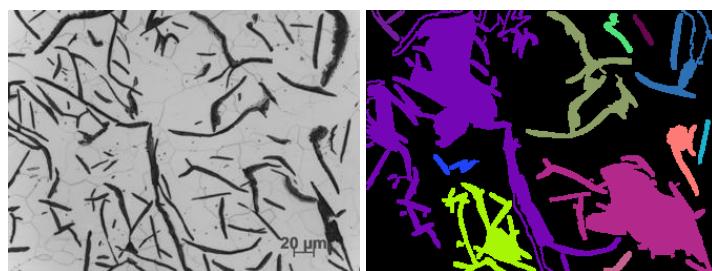
Następnie, posiadając program (przygotowany samodzielnie), z pomocą którego jesteśmy w stanie wyznaczać kontury struktur, możemy spróbować je wycinać. Przyjęto strategię wycinania pojedynczych

obiektów, a następnie nakładania ich na białe tło. Struktury były umieszczane w środku obrazu o białym tle o wymiarach 335×251 (szer. x wys.). Przykładowa wycięta struktura znajduje się na rys. 5.6. Poniższe obrazki (rys. 5.7 i 5.8) przedstawiają działanie tego procesu. Tym samym kolorem zostały



Rys. 5.6. Przykładowe zdjęcie wyciętej struktury (obraz przybliżony). Pozostałe kształty są wycinane analogicznie

zaznaczone kształty, które zostały potraktowane jako jeden obiekt. Natomiast na rys. 5.9 możemy zobaczyć przykładowe struktury, które zostały wycięte ze zdjęcia przedstawionego na rys. 5.7. Struktury na

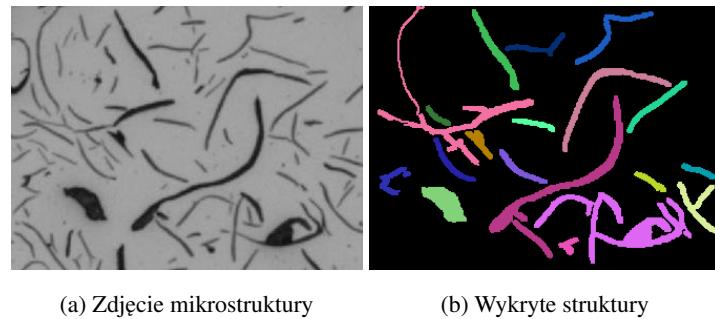


(a) Zdjęcie mikrostruktury

(b) Wykryte struktury

Rys. 5.7. Działanie filtru Canny'ego. Każda wykryta struktura jest zaznaczona innym kolorem. Źródło: opracowanie własne

obrazku oznaczone tym samym kolorem (a zatem uznane jako pojedynczy obiekt) są wycinane i umieszczone na białym tle. Zostaną one wykorzystane do trenowania modeli kategoryzacji w kolejnych fazach. Z drugiej strony, jak widać na powyższym obrazie, wadą tej techniki jest to, że struktury są czasami ze sobą połączone, a następnie traktowane przez algorytm jako jedna całość. Dzieje się tak najczęściej w przypadku klas I i II (rys. 4.2), które są rozcięgnięte na zdjęciach i często się stykają ze sobą. Próbowano tego uniknąć za pomocą takich algorytmów jak erozja i progowanie, ale te podejścia nie przyniosły zamierzonych rezultatów. Powyżej inny przykład działania tego samego algorytmu. Algorytm na rys. 5.8 zadziałał nieco lepiej niż na rys. 5.7. Jak widać, większość elementów została wycięta osobno. Tylko połączone kształty zostały wycięte jako jedna całość. Niestety nie ma na to uniwersalnego rozwiązania, ponieważ gdy ten problem zostanie rozwiązany, pojawia się nowy, jak np. brak wycinania mniej widocznych struktur lub wykrywanie wyblakłych struktur, które powinny zostać pominięte. Wycięte w ten sposób obiekty (rys. 5.9) są wprowadzane na wejście sieci neuronowej (rozdział 5.2.4), która zostanie wytrenowana do kategoryzowania tych struktur. W naszym zbiorze mamy siedem różnych klas. Sześć



Rys. 5.8. Kolejny przykład działania filtru Canny'ego. Każda wykryta struktura jest zaznaczona innym kolorem. Źródło: opracowanie własne



Rys. 5.9. Przykładowe wycięte struktury z rys. 5.7. Jak pokazały eksperymenty, założenie tych kształtów do klasy I lub II nie miało większego wpływu na wynik klasyfikacji. Również stworzenie wspólnej klasy dla tych dwóch typów kształtów nie wpłynęło na wyniki. Źródło: opracowanie własne

z nich (rys. 4.2) to kształty wyróżnione w normie [61], natomiast autor dodał klasę siódmą, aby odróżnić te struktury od obiektów, które zostały źle wycięte lub zawierających skalę, którą można znaleźć na wszystkich zdjęciach mikrostruktur. Wszystkie wycięte elementy były przechowywane lokalnie i arbitralnie przypisywane do klas podczas tworzenia zbioru danych. Liczność tych klas jest następująca:

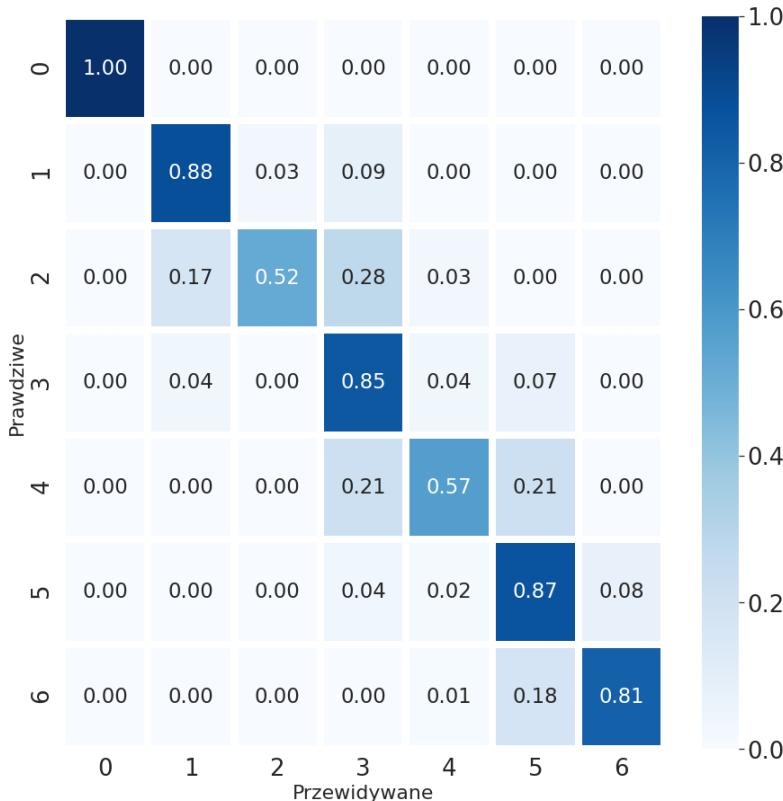
- klasa 0 – 14 przykładów (dodana przez autora),
- klasa I – 278 przykładów,
- klasa II – 122 przykładów,
- klasa III – 292 przykładów,
- klasa IV – 76 przykładów,

- klasa V – 289 przykładów,
- klasa VI – 501 przykładów.

W kolejnym podrozdziale (tj. 5.2.4) zostały zaprezentowane badania związane z klasyfikacją wyciętych struktur.

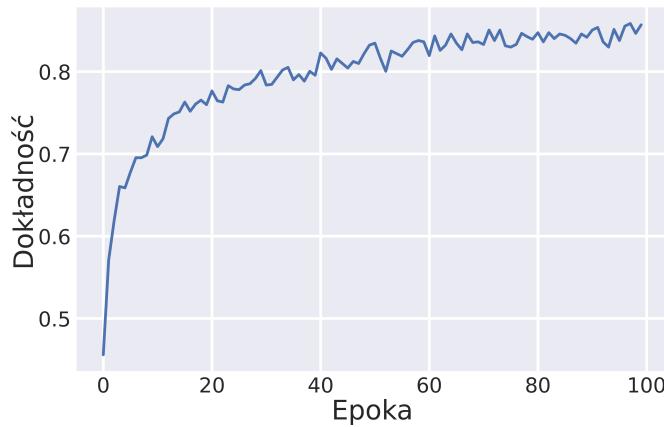
5.2.4. Rozpoznawanie wyciętych struktur

Na tym etapie badań posiadamy już własnoręcznie przygotowane dane dotyczące pojedynczych struktur obecnych na zdjęciach mikrostruktur (rys. 4.2). Ponownie, można by zacząć eksperymenty od takich najprostszych metod jak momenty Hu, momenty Zernike i tekstury Haralicka wraz z klasycznymi klasyfikatorami, aczkolwiek takie podejście zostało już sprawdzone w pracy [1] i pokazano, iż w przypadku takich zestawów algorytmów oraz dostępnych danych, wyniki są porównywalne do tych, które otrzymujemy w przypadku wykorzystania klasyfikatorów klasycznych z wejściem w postaci pikseli. Dlatego tutaj przejdziemy od razu do bardziej zaawansowanych metod. W tym celu zdecydowano się wykorzystać sieci neuronowe (rozdz. 3.6.8). W szczególności zastosowano uczenie transferowe (rozdz. 3.5) wykorzystując dobrze znaną sieć VGG19. Implementacja tej architektury została uzyskana z biblio-



Rys. 5.10. Tablica pomyłek dla klasyfikacji struktur znajdujących się na zdjęciach mikrostruktur dla danych testowych. Źródło: opracowanie własne (z użyciem biblioteki *seaborn*)

teki *Keras*. Podczas szkolenia sieć ta była wyłączona z treningu. Dodatkowo, jej końcówka (tj. ostatnie kilka warstw odpowiadających za klasyfikację) nie została uwzględniona, natomiast dodano kilka dodatkowych warstw, które pozwoliły nam na douszczanie sieci pod nasze dane. Zastosowano optymizator *Adam*, wielkość paczki uczącej (ang. *batch size*) wyniosła jeden, natomiast liczba epok wyniosła 100 (rys. 5.11). Pozostałe parametry zostały dostrojone. Taka sieć średnio osiąga wydajność około 82% dla danych testowych (w przypadku danych treningowych dokładność wahę się od 86% do 95% wydajności, w zależności od konfiguracji), wartość metryki F1 wynosi 0.79, natomiast wartość straty logistycznej (ang. *logistic loss*) wyniosła 0.66. Są to lepsze wyniki niż w przytoczonej wcześniej pracy [1] ze względu na to, że pracujemy na poprawionych danych. Proces korekcji danych został szczegółowo opisany w przytoczonej pracy [1]. Rysunek 5.10 przedstawia tablicę pomyłek znormalizowaną względem wierszy (tj. rzeczywistych wartości). Po przeliczeniu na błąd względny otrzymujemy błędy na poziomie 10%, 52%, 26%, 50%, 49% oraz 1% odpowiednio dla klas od I do VI (brak klasy „0” w danych testowych ze względu na ich niską licznosć). Błędy wynikają jednak głównie z podobieństwa tych form w ramach „sąsiadujących” klas takich jak klasy V i VI (na rysunku są to klasy 4 i 5), gdzie prawie połowa kształtów z klasy V jest sklasyfikowana jako klasa VI. Podobieństwo klas można dostrzec również na rys. 4.2, aczkolwiek jest to bardziej widoczne w przypadku rzeczywistych danych, ponieważ większość tych kształtów zawiera cechy wielu klas (trudno im przypisać jedną konkretną cechę). Co więcej, baza danych została zbudowana arbitralnie przez autora, który nie jest ekspertem w dziedzinie metalurgii. Rozbieżności w dokładności predykcji między klasami są najprawdopodobniej związane z brakiem zbalansowania w ilości danych dostępnych dla każdej klasy. Z drugiej strony wyniki wydają się być



Rys. 5.11. Historia dokładności modelu sieci neuronowej w trakcie uczenia się. Są to rezultaty dla danych treningowych, gdzie najlepsze wyniki sięgają 87%. Źródło: opracowanie własne (z wykorzystaniem biblioteki *seaborn*)

znacznie lepsze przy wykorzystaniu danych treningowych do ewaluacji. Po przeliczeniu błędów względnych otrzymujemy następujące wyniki: 36% (klasa „0”), 0.4%, 20%, 7%, 19%, 38% oraz 3%. Możemy zaobserwować, że dane uczące mają znacznie mniej błędów względnych, co wskazuje, że wyniki można jeszcze poprawić. Dobrym kierunkiem może być zbalansowanie danych, gdyż, jak można zauważyć, najlepsze wyniki otrzymujemy dla klas, których liczba instancji była największa w zbiorze (z wyjątkiem

klasy V, aczkolwiek przykłady tej klasy są łudząco podobne do przykładów z klasy VI). Przyjęto więc strategię, aby rozszerzyć dane z tych klas, które jednocześnie mają najniższą skuteczność i najmniej przykładów. Stąd zostaną rozszerzone klasy: „0”, II, IV oraz V. Ponownie zostanie zastosowana (podobnie jak w 5.3.1) metoda odbicia, która nie zmienia rozmiarów zdjęcia. Również tym razem zostanie zastosowana metoda odbicia horyzontalnego. Oprócz tego przetestowano jeszcze kilka innych podejść, jak nałożenie szumu na rozszerzone przykłady danych, a także usunięcie szarości z obrazów. Wyniki zostały przedstawione w tab. 5.1. A więc, jak możemy zauważyć, manipulacja danymi nie wpłynęła po-

Tabela 5.1. Podsumowanie różnych podejść co do klasyfikacji kształtów. Źródło: opracowanie własne

Augmentacja	Zaszumienie	Usuwanie szarości	Dokładność
Nie	Nie	Nie	82.2%
Tak	Nie	Nie	78%
Tak	Tak	Nie	79.7%
Tak	Tak	Tak	77.1%
Tak (dod. klasę III)	Tak	Tak	74.9%

zytywnie na wynik klasyfikacji. W związku z tym dalsze badania zostaną przeprowadzone na architekturze, dla której osiągnięto najlepsze wyniki. Biorąc pod uwagę niebalansowanie klas oraz to, iż te dane były przygotowywane ręcznie przez autora tej pracy (który nie jest specjalistą w dziedzinie metalurgii), w celu poprawy wyników tej klasyfikacji najprawdopodobniej konieczne jest, aby przygotował je ekspert w tej dziedzinie. Można to zadanie potraktować jako dalszą część prac, które byłyby rozszerzeniem niniejszej pracy. W dalszych badaniach zostanie wykorzystana architektura wraz z danymi, dla których otrzymano najwyższe wyniki, tj. sieć neuronowa VGG19 wraz z pierwotnie przygotowanymi obrazami struktur. Na rys 5.12 została przedstawiona legenda. Każdy z siedmiu kolorów przedstawionych na wykresie kołowym odpowiada jednej z siedmiu klas struktur (zgodnie z opisem na diagramie). Kolory te zostały dobrane za pomocą generatora wizualnie odmiennych kolorów (mokole.com), aby były jak najbardziej wyraziste. Poniżej przedstawiono przykłady klasyfikacji struktur z wykorzystaniem opisanych powyżej metod. Na tych obrazach (rys. 5.13 - 5.17) obowiązuje schemat, iż po prawej stronie znajduje się oryginalne zdjęcie mikrostruktury, natomiast po lewej stronie znajduje się to samo zdjęcie, z tym że obecne tam struktury zostały pokolorowane zgodnie z legendą (rys. 5.12). Są to przykładowe zdjęcia, które dobrze reprezentują cały proces, który wygląda w ten sposób, że dla każdego zdjęcia wejściowego przeprowadzane są operacje opisane w rozdziale 5.2.3, a następnie są one klasyfikowane przez architekturę przedstawioną w tym rozdziale. W związku z tym mogą wystąpić błędy, jak brak wykrycia danej struktury bądź źle sklasyfikowanie struktury. Jak widzimy na przedstawionych rysunkach (rys. 5.13 – 5.17), sieć klasyfikująca kształty wraz z całą sekwencją rozpoznawania pojedynczych struktur przynosi całkiem przyzwoite efekty. Wizualnie nie można tutaj niczego skrytykować. Natomiast często zdarza się tak, jak już wspomniano wcześniej, że struktury te są bardzo podobne do siebie i przygotowanie ich przez specjalistę w dziedzinie metalurgii mogłoby przynieść znaczny wzrost skuteczności modelu.



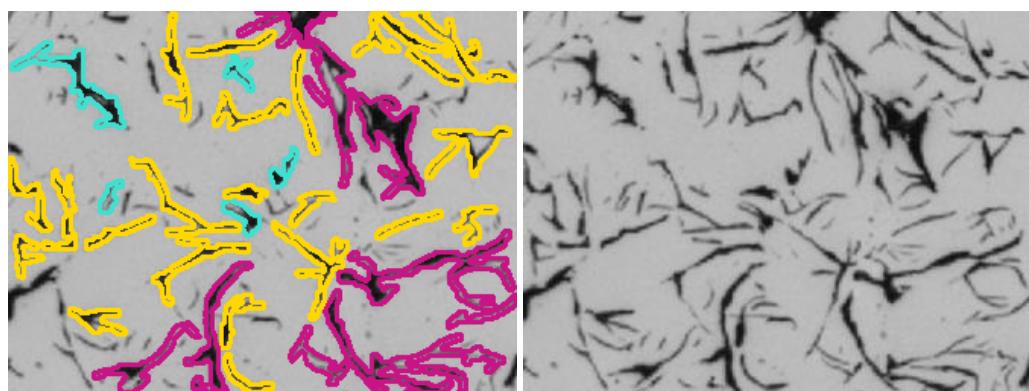
Rys. 5.12. Legenda użytych kolorów w celu oznaczenia struktur na obrazach, które należą do różnych klas. Źródło: opracowanie własne (z wykorzystaniem biblioteki *matplotlib*)

Ostatnią próbą w celu poprawienia skuteczności modelu było wyeliminowanie najmniejszych struktur z klasyfikacji. Są one najbardziej problematyczne, gdyż wizualnie, praktycznie się nie różnią. Dodatkowo sam autor miał problem z przypisaniem tych struktur do konkretnej klasy. Po usunięciu tych struktur otrzymaliśmy skuteczność na poziomie 82%, natomiast wartość straty logistycznej wynosi 0.8, co jest bardzo dobrym wynikiem. Tablica pomyłek została przedstawiona na rys. 5.18. Porównując tę tablicę pomyłek do poprzedniej, która dotyczyła wyników badań przeprowadzonych na wszystkich danych możemy dojść do kilku ciekawych wniosków. Otóż jak pokazano w rozdz. 5.2.3, liczba struktur klasy „0” wynosi zaledwie 14 i została dodana przez autora pracy. W tym wypadku ta klasa jest najmniej interesująca. Największe straty zanotowały klasy II oraz V, bo 7 i 10 punktów procentowych (odpowiednio). Natomiast największe zyski zanotowały klasy IV oraz VI, bo aż 33 i 6 punktów procentowych.

5.2.5. Wnioski

Po badaniach nad zagadnieniem przedstawionym w tym rozdziale można wyciągnąć wiele interesujących wniosków:

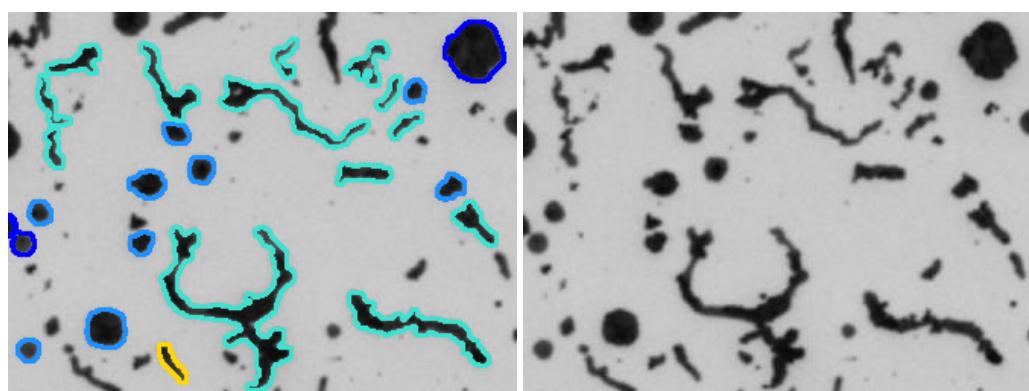
- w pierwotnym zbiorze wystąpiły dwa zdjęcia (przykładowo rys. 4.3, ich nazwy to *I_1E_t_500x* oraz *I_4E_t_500x*), które nie nadawały się do tej analizy, gdyż nie zawierały w sobie głównych form grafitowych (rys. 4.2)
- GHT zwracała dosyć precyzyjne wyniki (rys. 5.3), jednak miała wiele obostrzeń, przede wszystkim konieczność zastosowania identycznego obrazka referencyjnego do poszukiwanego, mimo to do innych zastosowań może być skutecznie wykorzystywana,



(a) Wykryte, sklasyfikowane i ubarwione struktury

(b) Zdjęcie mikrostruktury

Rys. 5.13. Zdjęcia mikrostruktury zawierające struktury klas I-III. Źródło: opracowanie własne



(a) Wykryte, sklasyfikowane i ubarwione struktury

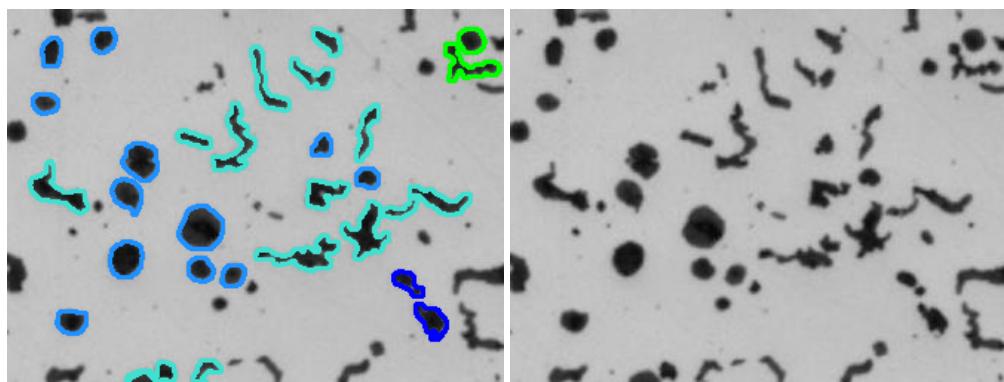
(b) Zdjęcie mikrostruktury

Rys. 5.14. Zdjęcia mikrostruktury wraz ze strukturami klasy III, V i VI. Można zauważyć również, że jedna, podłużna struktura została sklasyfikowana jako typ I. Źródło: opracowanie własne

- gdy wykorzystano rozmycie gaussowskie (ang. *Gaussian blur*), popękane struktury były wykrywane lepiej (gdyż pęknięcia nie były uznawane za struktury). Również klasy V-VI zyskiwały na skuteczności, natomiast traciły na tym pozostałe klasy (przede wszystkim I-II),
- przetestowano również progowanie obrazu (ang. *thresholding*), które polega na binaryzowaniu zdjęć, tj. przypisywaniu im czarnych bądź białych pikseli w zależności od poprzedniej wartości tych pikseli. Miało to zapobiegać traktowaniu pęknięć jako struktury, jednakże całkowita skuteczność również spadła.

5.3. Ocena jakości odlewów

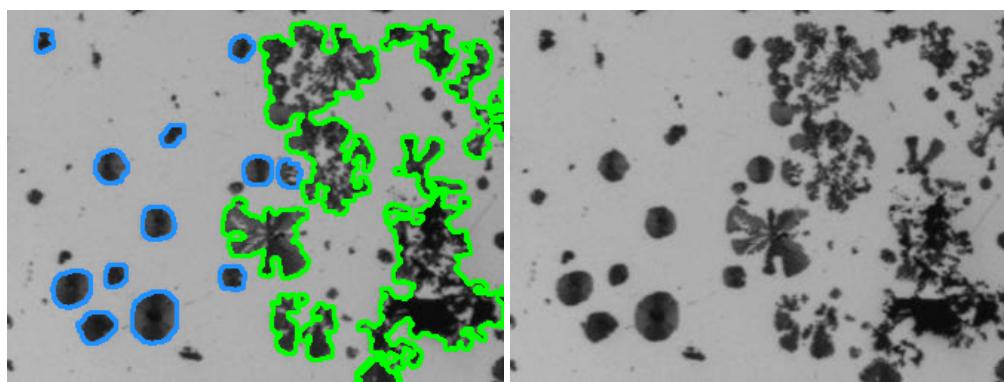
Jest to drugi najszerzy obszar badań (obok rozdz. 5.2), gdyż jest to główny punkt tych badań oraz temat niniejszej pracy. Zebrano i przedstawiono tutaj wszystkie testy związane z oceną jakości odlewów.



(a) Wykryte, sklasyfikowane i ubarwione struktury

(b) Zdjęcie mikrostruktury

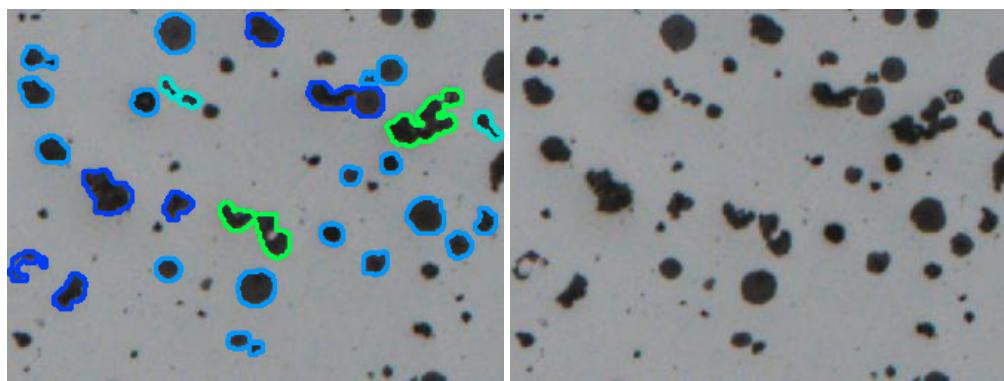
Rys. 5.15. Zdjęcia mikrostruktury z oznaczonymi formami grafitowymi, które należą do klas III-VI. Źródło: opracowanie własne



(a) Wykryte, sklasyfikowane i ubarwione struktury

(b) Zdjęcie mikrostruktury

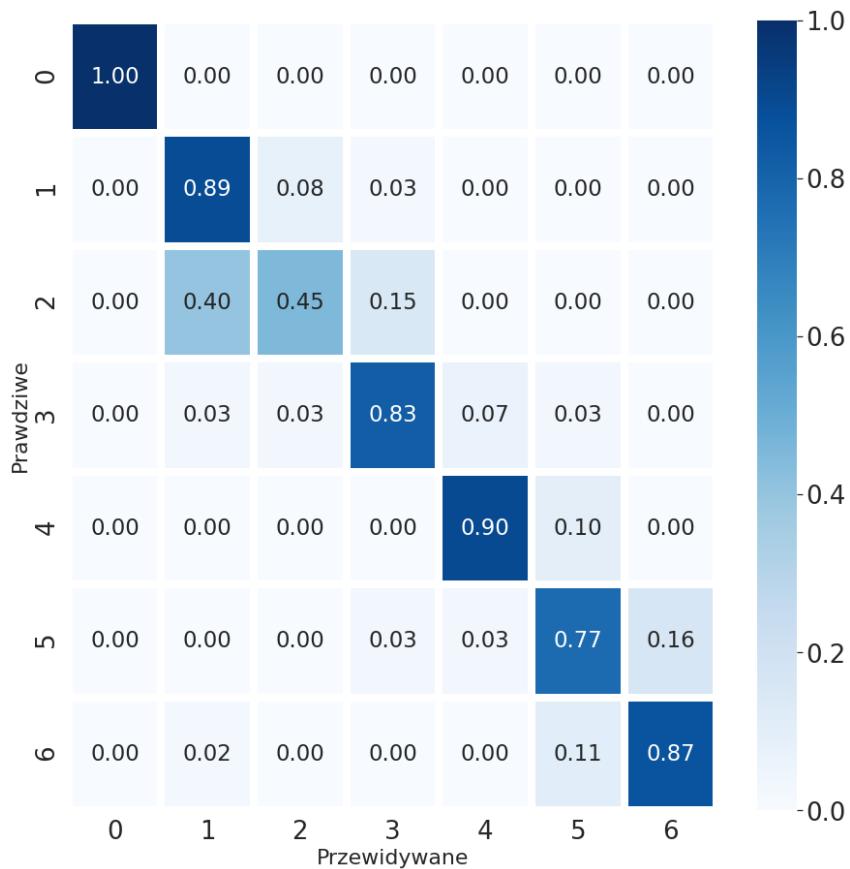
Rys. 5.16. Zdjęcia mikrostruktury zawierające kształty sklasyfikowane do klas IV oraz VI. Źródło: opracowanie własne



(a) Wykryte, sklasyfikowane i ubarwione struktury

(b) Zdjęcie mikrostruktury

Rys. 5.17. Zdjęcia mikrostruktur przedstawiające kształty sklasyfikowane do klas III-VI. Źródło: opracowanie własne



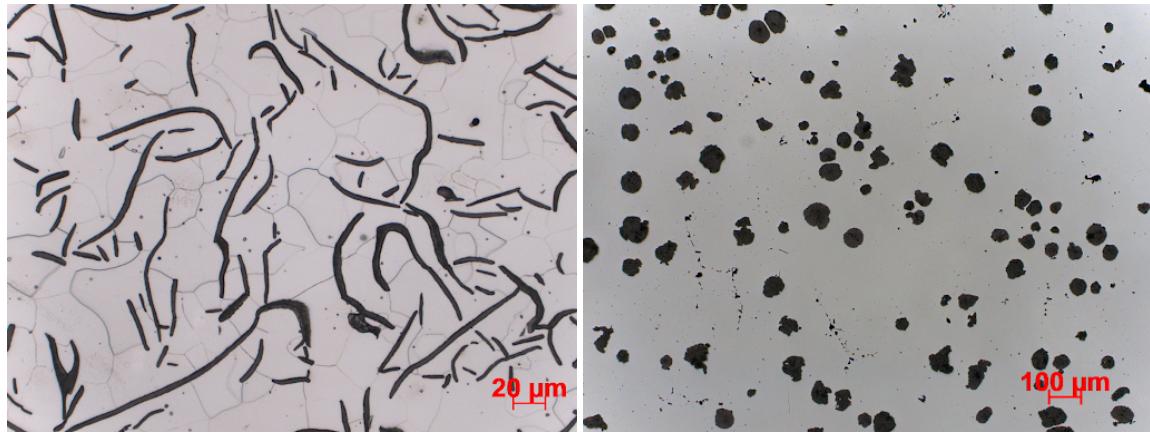
Rys. 5.18. Tablica pomyłek dla klasyfikacji struktur znajdujących się na zdjęciach mikrostruktur dla danych testowych. Wykorzystano większe struktury, które zostały wybrane na podstawie zajmowanej przez nie powierzchni, jak również ich obwodu.
 Źródło: opracowanie własne (z użyciem biblioteki *seaborn*)

Eksperymenty przedstawiono w kolejności, w jakiej zostały wykonywane, tj. od najprostszych metod do najbardziej skomplikowanych, które jednocześnie zwracały najlepsze wyniki. Badanie rozpoczęto od sprawdzenia skuteczności klasycznych klasyfikatorów na danych w postaci momentów Hu oraz tekstu Haralicka (rozdz. 5.3.1). Następnie przetestowano klasyczne klasyfikatory, które jako dane wejściowe dostawały poprzednio przygotowane dane w postaci liczności struktur poszczególnych klas. Jako ostatnie zbadano skuteczność sieci neuronowych oraz dodatkowo podejście hybrydowe, w którym wynik klasyfikacji sieci VGG19 dołączono do wejścia klasycznych klasyfikatorów (a więc otrzymywały one liczbę struktur oraz wynik klasyfikacji sieci VGG19).

5.3.1. Momenty Hu oraz tekstury Haralicka

Pierwszym podejściem było wykorzystanie tekstu Haralicka i momentów Hu do zidentyfikowania struktur na zdjęciu. Implementacja metody momentów Hu została zaczerpnięta z pakietu *opencv-python*, natomiast implementacja tekstu Haralicka została zaczerpnięta z modułu *mahotas*. Może się wydawać,

że użycie tych algorytmów jest poprawne i przyniesie pożądane rezultaty, ponieważ zostały zaprojektowane specjalnie do tego celu. Rysunek 5.19 przedstawia dwa zdjęcia różnych mikrostruktur, które potencjalnie można zidentyfikować przy użyciu technik opisanych powyżej. W pierwszym eksperymencie wy-



(a) Zdjęcie mikrostruktury z obiektem klasy I (rys. 4.2) (b) Zdjęcie mikrostruktury z obiektem klasy V (rys. 4.2)

Rys. 5.19. Dwa przykładowe zdjęcia mikrostruktur. Ich tekstury i znajdujące się tam kształty diametralnie się różnią. Źródło: [58]

korzystano teksturowi Haralicka, momenty Hu, a także modele maszyny wektorów nośnych (SVM) oraz lasów losowych (RF, RFC). Testy zostały przeprowadzone na zbiorze danych klasyfikowanych ze względu na wytrzymałość na rozciąganie. Niestety taka konfiguracja modeli i metod nie przyniosła oczekiwanych rezultatów. Testy przeprowadzono z wykorzystaniem sprawdzianu krzyżowego (ang. *cross-validation*), a dokładnie sprawdzian *k*-krotny (rozdz. 3.6.9.2). Następnie te *k* rezultatów jest uśrednianych. Wyko-

Tabela 5.2. Wyniki klasyfikacji binarnej z użyciem klasycznych klasyfikatorów, z wykorzystaniem momentów Hu oraz teksturowi Haralicka. Źródło: opracowanie własne

Model	Typ wejścia	Wagi klas	Dokładność
SVM	Hu ^a	—	71.5%
SVM	Haralick ^b	—	71.5%
SVM	Hu + Haralick ^c	—	71.5%
RFC	Hu	zrównoważone	58%
RFC	Haralick	zrównoważone	70%
RFC	Hu + Haralick	zrównoważone	70.1%

^a Momenty Hu wyliczone ze zdjęć (7 elementów).

^b Tekstury Haralicka wyznaczone ze zdjęć (13 elementów).

^c Konkatenacja wyników dwóch powyższych metod (20 elementów).

rzystano domyślne wartości parametrów bez wykorzystania strojenia parametrów, gdyż wyniki te są o wiele gorsze niż w przypadku innych metod. Wyniki zaprezentowano w tabeli 5.2. Jak widzimy, wyniki dla większości tych metod wynoszą około 71%. Nieco gorszy wynik w przypadku lasu losowego

Tabela 5.3. Wyniki klasyfikacji binarnej z użyciem klasycznych klasyfikatorów, z wykorzystaniem momentów Hu oraz tekstur Haralicka. Dodatkowo rozszerzono dwukrotnie liczbę zdjęć o niskiej odporności (augmentacja). Źródło: opracowanie własne

Model	Typ wejścia	Wagi klas	Dokładność
SVM	Hu	—	55.7%
SVM	Haralick	—	54.6%
SVM	Hu + Haralick	—	54.6%
RFC	Hu	zrównoważone	56.6%
RFC	Haralick	zrównoważone	86%
RFC	Hu + Haralick	zrównoważone	86%

może wynikać z zastosowania zrównoważenia wag klas. Jednakże zbiór danych składa się z 3358 zdjęć o wysokiej wytrzymałości na rozciąganie oraz 1337 zdjęć o niskiej wytrzymałości na rozciąganie (patrz 4.3.1). A więc liczba zdjęć o wysokiej wytrzymałości stanowi dokładnie 71% wszystkich danych, stąd klasyfikatory zamiast rzeczywiście rozpoznawać dane tak na prawdę mogą dopasowywać się do częstości występowania zdjęć określonej klasy. Stąd przeprowadzono dodatkowe testy z wykorzystaniem augmentacji (patrz 3.4, 4.4). Konkretnie zastosowano metodę odbicia na mniej licznej klasie (a więc na zdjęciach o niskiej wytrzymałości), dzięki czemu stosunek liczby zdjęć o wysokiej wytrzymałości do liczby wszystkich zdjęć spadł z 71% do 55.7%. Wyniki po przeprowadzeniu augmentacji znajdują się w tab. 5.3. Dodatkowo, aby nie zwiększać sztucznie dokładności ze względu na podobieństwo danych

Tabela 5.4. Wyniki klasyfikacji binarnej z użyciem klasycznych klasyfikatorów, z wykorzystaniem momentów Hu oraz tekstur Haralicka. Dane dodatkowo zostały zaszumione, aby zapobiec nadmierнемu dopasowaniu się modelu do danych. Źródło: opracowanie własne

Model	Typ wejścia	Wagi klas	Dokładność
SVM	Hu	—	55.7%
SVM	Haralick	—	55.5%
SVM	Hu + Haralick	—	55.5%
RFC	Hu	zrównoważone	51.6%
RFC	Haralick	zrównoważone	72.2%
RFC	Hu + Haralick	zrównoważone	72.6%

wygenerowanych sztucznie i oryginalnych, postanowiono nałożyć szum na dane wygenerowane syntetycznie. Zdecydowano się na szum biały. Wykorzystano bibliotekę *scikit-image* a precyzyjniej funkcję *scikit-image.util.random_noise* z domyślnymi wartościami parametrów. Wyniki zostały przedstawione w tabeli 5.4.

Porównując wyniki otrzymane w tych trzech eksperymentach można dojść z pewnością do kilku wniosków. Po pierwsze, widać wyraźnie, że wyniki pomiędzy modelami drzew oraz maszyny wektorów

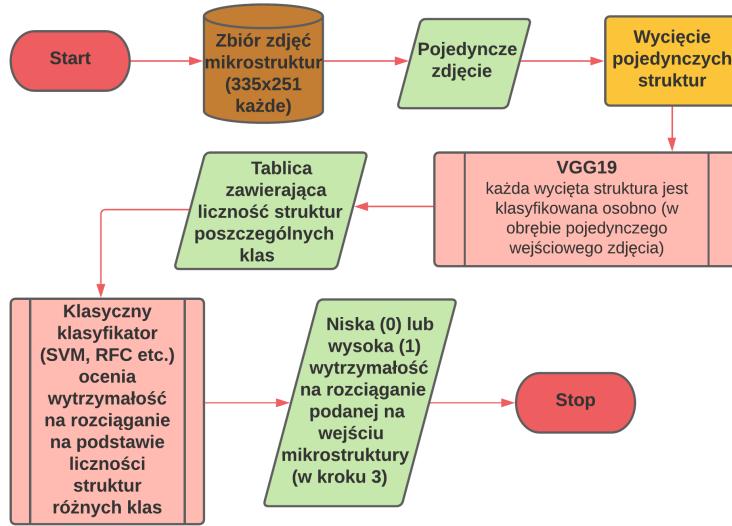
znacząco się różnią. W pierwszym eksperymencie wyniki dla SVM są lepsze od 1 do nawet 12 punktów procentowych. W pozostałych dwóch eksperymetach wyniki dla SVM są znacznie gorsze od tych dla lasu losowego, w ekstremalnym przypadku aż o 30 punktów procentowych. Po drugie, można zauważyć, że w większości przypadków wyniki dla momentów Hu są najgorsze, co można wytlumaczyć tym, iż jest to najprostsza metoda. Natomiast widzimy również wpływ augmentacji danych, jak również zaszumienia danych. Można przyjąć, iż najbardziej wiarygodne wyniki otrzymaliśmy w trzecim eksperymencie, gdyż, przede wszystkim, skuteczność algorytmu nie pokrywa się ze stosunkiem liczebności klas, oraz wyniki nie różnią się między sobą w niewytłumaczalny sposób. Mimo wszystko, ciężko do końca ocenić co tak naprawdę wpłynęło na te wyniki. Dodatkowo to podejście ma bardzo niską interpretowalność, tzn. ciężko ocenić dlaczego model wybrał jedną decyzję, zamiast innej. Natomiast atutem tego podejścia jest czas wykonywania (czas uczenia modeli). Mimo ustawienia parametrów SVM na wiele iteracji, a także ustawienie stosunkowo dużej liczby drzew dla algorytmu lasu losowego, czas uczenia tych modeli jest rzędu kilkunastu sekund. Jak widzimy, już nawet najprostsze metody dają pewne pozytywne rezultaty, dlatego zdecydowano się na nieco bardziej złożoną strategię, która być może podniesie skuteczność.

5.3.2. Klasyfikacja za pomocą liczby struktur

W kolejnym kroku przetestowano skuteczność klasycznych klasyfikatorów w ocenie jakości odlewów (tj. predykcji wytrzymałości na rozciąganie odlewów). Jako dane wejściowe zastosowano liczbę struktur poszczególnych klas (rys. 4.2), które są pozyskiwane w procesie przedstawionym w rozdz. 5.2. Stąd na wejście klasycznych klasyfikatorów jest podawana tablica siedmiu liczb, z czego każda z tych liczb odpowiada liczebności odpowiedniej klasy (tj. liczba „zerowa” odpowiada liczebności obiektów klasy „0” itd.) i na podstawie tych danych starają się przewidzieć wytrzymałość na rozciąganie odlewów przedstawianych na zdjęciach, których te dane dotyczą. Przetestowano również podejścia, w których zamiast bezwzględnych liczb podawano na wejście procentowy udział struktur poszczególnych klas, a także strategie, w której na wejście podawano wynik funkcji softmax (wzór 3.7) na bezwzględnych licznościach struktur poszczególnych klas. Na rys. 5.20 przedstawiono schemat działania całego procesu. W tym celu wykorzystano najpopularniejsze klasyfikatory, jak maszyna wektorów nonych (SVM), drzewo decyzyjne (DT), las losowy (RFC), regresja logistyczna (logit), perceptron wielowarstwowy (MLP) oraz AdaBoost. W kolejnych podrozdziałach są przedstawione analizy oraz wyniki uzyskane z użyciem wyżej wymienionych klasyfikatorów.

5.3.2.1. Klasyfikator SVM

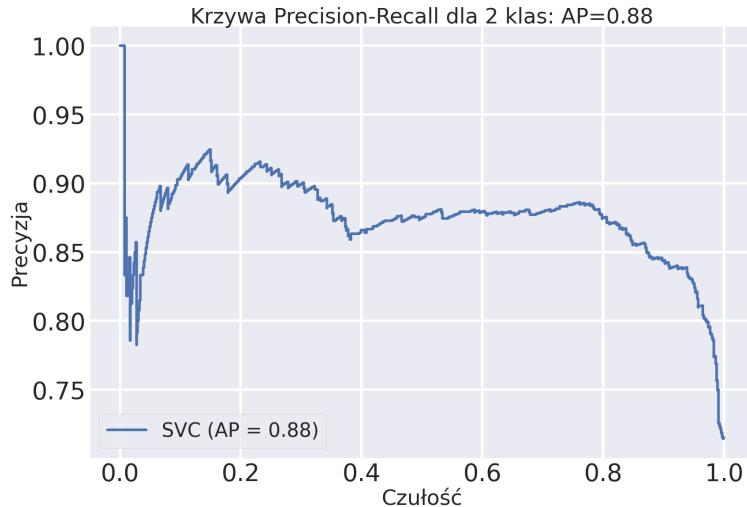
Jako pierwsze zostanie omówione podejście, w którym jako klasycznego klasyfikatora użyto SVM, czyli maszynę wektorów nośnych (rozdz. 3.6.1). Schemat ogólny został przedstawiony na rys. 5.20. W pierwszej kolejności przebadano strategię, w której na wejście SVM podawano bezwzględną liczbę struktur poszczególnych klas. Badania przeprowadzono wykorzystując walidację krzyżową (rozdz. 3.6.9.2) oraz przeszukiwanie siatki (rozdz. 3.6.9.1). Użyto już gotowej implementacji z biblioteki *scikit-learn* pod nazwą *GridSearchCV*, który obsługuje te dwie rzeczy naraz. W rezultacie otrzymano model,



Rys. 5.20. Schemat blokowy przedstawiający działanie klasyfikacji wytrzymałości na rozciąganie na podstawie liczności struktur poszczególnych klas. Źródło: opracowanie własne (z wykorzystaniem lucidchart)

w którym wartości najważniejszych parametrów były następujące: stropień (ang. *degree*) wyniósł 3 oraz jądro (ang. *kernel*) ustawiono na *rbf*. Model ten osiągnął 83.3% dokładności dla danych testowych oraz około cztery punkty procentowe więcej dla danych treningowych (86.7%). Wartość straty logistycznej wyniosła 0.39. To oznacza, że nie doszło do przetrenowania, ani niedotrenowania modelu do danych, czemu też zapobiega sprawdzian krzyżowy. W każdym razie wyniki wydają się być wysokie, tym bardziej, że opierają się na modelu, który miał mniejszą skuteczność (o pół punktu procentowego). Poniżej przedstawiono wykres krzywej Precision-Recall, który wskazuje jaką jest precyzyja modelu (ang. *precision*) przy zadanej czułości (ang. *recall*). Natomiast na rys. 5.22 przedstawiono tablicę pomyłek dla tej klasyfikacji.

Kolejnym przetestowanym pomysłem była strategia, w której zamiast bezwzględnych liczb określających liczbę struktur odpowiednich klas na wejście podawano procentowy udział struktur poszczególnych klas. Tym razem skuteczność wyniosła 78.7%, a więc dokładnie o cztery punkty procentowe mniej, niż w przypadku liczności struktur. Średnia wartość precyzyji wyniosła 0.8, a więc również gorzej niż w poprzednim wypadku. Również wartość straty logistycznej jest większa i wynosi 0.47. Stąd wykres nie jest nawet wymagany, ponieważ po samych wynikach liczbowych możemy z całą pewnością stwierdzić, iż tutaj lepiej zadziałało poprzednie podejście. Próbując wyjaśnić takie rozbieżności tak naprawdę ciężko dojść do sensownego wyjaśnienia tej sytuacji. Wydawać mogłoby się, iż wyniki powinny być znacznie bardziej zbliżone, a to ze względu na charakter danych wejściowych. Procentowy udział struktur poszczególnych to nic innego jak znormalizowane wyniki biorące pod uwagę liczbę struktur danej klasy względem wszystkich pozostałych. A jednak wyniki są dużo gorsze. Natomiast wystąpiła jedna drobna różnicę, a mianowicie było dziewięć zdjęć, dla których nie wykryto ani jednej struktury,



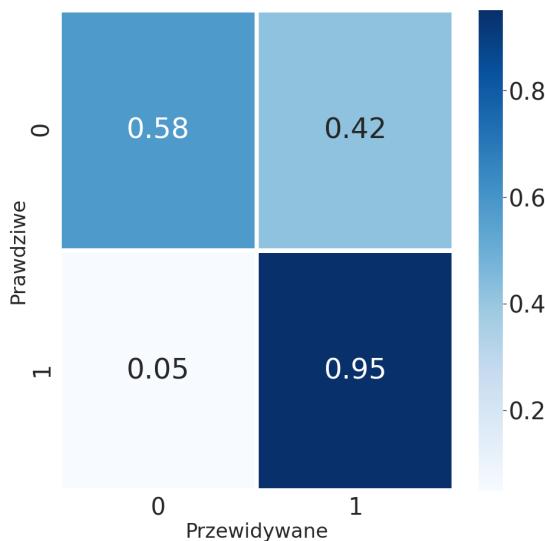
Rys. 5.21. Wykres krzywej Precision-Recall. Średnia wartość precyzji wynosi 0.88.

Źródło: opracowanie własne (z wykorzystaniem biblioteki *scikit-learn*)

stąd niemożliwe było wyliczenie względnego udziału struktur poszczególnych klas, wskutek czego konieczne było ich wyeliminowanie ze zbioru. Tak czy inaczej, jest to tylko dziewięć przykładów spośród 4695 wszystkich, stąd można ten fakt pominać.

Jako ostatnia została przetestowana strategia, w której na wejście klasycznych klasyfikatorów jest podawany wynik funkcji softmax na pierwotnych danych (tj. liczności struktur poszczególnych klas). W tym przypadku otrzymaliśmy dokładność na poziomie 79%, średnią wartość precyzji równą 0.72 oraz wartość straty logistycznej równą 0.51. Jak widzimy, z wszystkich trzech zasosowanych tutaj strategii najlepiej sprawdza się bezwzględna liczność struktur poszczególnych klas, dlatego w dalszych badaniach będziemy się głównie skupiać na tego typu danych.

Jednakże dane są mocno niebalansowane, co zostało pokazane w rozdz. 4.3.1. W ramach przypomnienia, zdjęć reprezentujących odlewy z wysoką wytrzymałością na rozcięganie jest 3358, natomiast tych z niską jest 1337. To mogłoby wyjaśniać przesunięcie w stronę klasy odpowiadającej wysokiej wytrzymałości, co zostało przedstawione na rys. 5.22 (tzw. błąd drugiego rodzaju). Co prawda zgodnie z terminologią jest to zaledwie „delikatnie niebalansowany zbiór”, aczkolwiek warto sprawdzić czy to niezrównoważenie ma wpływ na wyniki. W tym celu wykorzystano najprostszą technikę równoważenia zbiorów, a mianowicie wyeliminowano z danych nadmiarowe przykłady nadreprezentowanej klasy. W ten sposób mamy teraz 1337 przykładów zarówno mikrostruktur wysokiej, jak i niskiej wytrzymałości. Jednakże po zbalansowaniu klas problem nadal występuje (rys. 5.23). Przeprowadzono te same testy i wartości wszystkich metryk się pogorszyły, mimo że tablica pomyłek po tym zabiegu wygląda nieco lepiej. Stąd wiadomo, że przesunięcie w stronę klasy wysokich wytrzymałości nie jest kwestią niebalansowania klas. Co jest przyczyną tego stanu rzeczy postaramy się dogłębniej wyjaśnić przy okazji testów kolejnych klasyfikatorów klasycznych, w tym bardziej interpretowalnych (m.in. drzewa decyzyjne) w kolejnych podrozdziałach.



Rys. 5.22. Tablica pomyłek dla klasyfikacji wytrzymałości na rozciąganie odlewów.

Użyto niezbalansowanych danych (oryginalny zbiór). Źródło: opracowanie własne (z wykorzystaniem biblioteki *seaborn*)

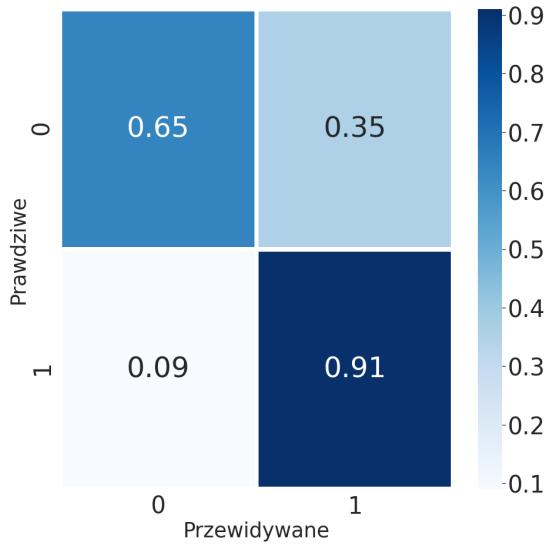
Podsumowując wyniki badań dla SVM przygotowano szkic, w którym są przedstawione zbiorczo zebrane wyniki (tab. 5.5). Widzimy, że najlepsze wyniki otrzymujemy dla oryginalnych danych wej-

Tabela 5.5. Podsumowanie wyników klasyfikacji binarnej z użyciem klasyfikatora SVM w celu predykcji wytrzymałości na rozciąganie odlewów. Źródło: opracowanie własne

Typ wejścia	Dokładność	AP ^a	Strata logistyczna
Liczba struktur	82.7%	0.88	0.39
Procentowy udział struktur	78.7%	0.8	0.47
Wartość funkcji softmax dla liczby struktur	79%	0.72	0.51
Liczba struktur (zbalansowane dane)	78.4%	0.79	0.49

^a AP – średnia wartość precyzji.

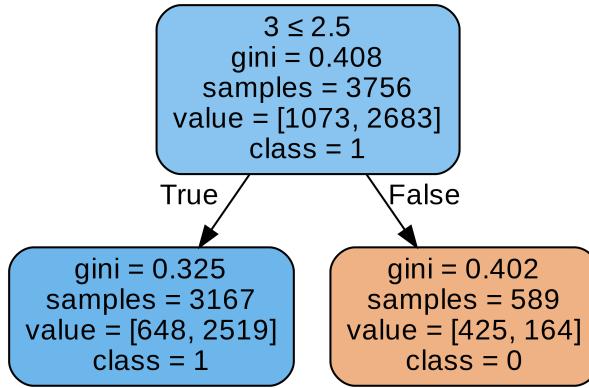
ściowych, tj. w postaci liczności struktur poszczególnych klas, stąd w kolejnych badaniach z użyciem pozostałych klasyfikatorów skupimy się głównie na tego typu danych. Dodatkowo, w ramach próby użyskania najlepszego możliwego wyniku skorzystano z biblioteki *Optuna*, która służy do optymalizacji hiperparametrów. Polega ona na nieco innym wyszukiwaniu, niż w przypadku poprzedniego modułu. Po zastosowaniu kroswalidacji ta strategia przyniosła najlepsze wyniki. Dla danych w postaci liczby struktur otrzymujemy 84% dokładności, 0.88 średniej wartości precyzji oraz 5.66 wartości straty logistycznej.



Rys. 5.23. Tablica pomyłek dla klasyfikacji wytrzymałości na rozciaganie odlewów. Dane zostały zbalansowane. Źródło: opracowanie własne (z wykorzystaniem biblioteki *seaborn*)

5.3.2.2. Drzewo decyzyjne

W tym podrozdziale zajmiemy się wynikami osiąganyimi za pomocą drzew decyzyjnych (3.6.2). Mimo, iż jest to bardzo prosty model, często zwraca przyzwoite wyniki, zaś jego interpretowalność jest jego największym atutem (obok jego prostoty). Jak wiemy istnieje wiele algorytmów generowania drzew decyzyjnych. Jako że autor czerpie implementacje klasycznych klasifikatorów z biblioteki *scikit-learn*, toteż pierwszym przebadanym algorytmem będzie ten wykorzystywany przez wspomnianą bibliotekę. Jest to zoptymalizowana wersja algorytmu CART. Ponownie rozpoczniemy od zaprezentowania wyników, gdy jako dane wejściowe wykorzystano bezwzględną licznosć struktur poszczególnych klas. Testy podzielono względem głębokości testowanych drzew. Ponieważ wymiar danych wejściowych wynosi siedem, nie ma sensu bliżej przyglądać się drzewom o głębokości większej niż trzy. Natomiast w ramach porównania również one zostaną uwzględnione w podsumowaniu. W pierwszej kolejności przeprowadzono badania dla drzew o głębokości równej jeden. Dla takich drzew średnia dokładność modelu wynosi 78%, co już jest bardzo dobrym wynikiem. Wizualizacja takiego drzewa została przedstawiona na rys. 5.24. Jak możemy zauważyć, najważniejsze są tutaj struktury typu III (rys. 4.2). Biorąc pod uwagę tylko ten jeden typ struktur otrzymujemy dokładność na wspomnianym poziomie 78%. Zobaczmy teraz jak prezentują się wyniki dla drzew o głębokości równej dwa. Otóż dokładność wyniosła 81.2%, natomiast wizualizacja została przedstawiona na rys. 5.25. Jak możemy zauważyć, klasa III jest ponownie najważniejszą klasą, natomiast dochodzą nam również klasy I oraz VI. Logicznym wytłumaczeniem tego stanu rzeczy może być fakt, iż właśnie struktury tych trzech klas najbardziej różnią się od pozostałych wizualnie. Dla drzewa o głębokości równej trzy otrzymujemy dokładność na poziomie 81.3%, a więc zaledwie jedną dziesiątą punktu procentowego więcej, niż dla drzewa o głębokości dwa. To może świadczyć o nadmiernym dopasowaniu się modelu do danych, czy też słabej generalizacji. Dla większych



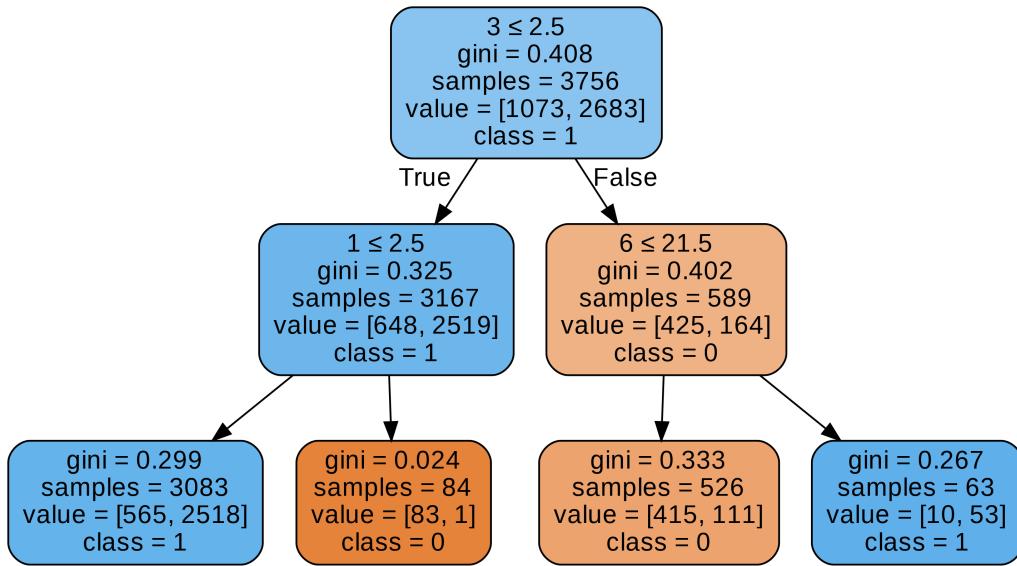
Rys. 5.24. Wizualizacja drzewa decyzyjnego. Warunek jest przedstawiony w pierwszym wierszu pierwszego klocka decyzyjnego. Sprawdzamy, czy struktur III typu jest mniej lub równo 2.5. Na tej podstawie jest wyznaczana wytrzymałość odlewu. Źródło: opracowanie własne (z wykorzystaniem biblioteki *graphviz*)

głębokości drzewa nie odnotowano wzrostu dokładności. Ciekawie również prezentują się wyniki dla innych typów wejść, toteż zostaną one przedstawione na samym końcu tej sekcji w formie tabeli.

Tymczasem przejdziemy do kolejnego algorytmu generowania drzewa decyzyjnego. Obok algorytmu CART przetestowano również algorytm ID3, którego implementacja została zaczerpnięta z biblioteki *decision-tree-id3*. Skonfrontowano ze sobą takie cechy, jak czas uczenia, czas zwracania wyników czy też dokładność. W pierwszej kolejności przyjrzymy się czasom uczenia tych modeli w zależności od wykorzystanego algorytmu oraz głębokości drzew. Na rys. 5.26 przedstawiono wyniki tego porównania. Jak możemy zauważyć, czas uczenia jest wyraźnie dłuższy dla algorytmu ID3 i rośnie wraz ze wzrostem głębokości drzewa. Jest to znana własność tych algorytmów, gdyż na ogół algorytm CART działa szybciej, szczególnie gdy korzystamy z zoptymalizowanej wersji algorytmu CART.

Następną metryką jest czas uzyskania wyniku. Wyniki tego porównania przedstawiono na rys. 5.27. Jak możemy zauważyć na wykresie, również w tym zestawieniu lepiej prezentuje się algorytm CART. Jego wyjście jest zwracane niemal natychmiast, natomiast na wyniki drzewa zbudowanego za pomocą algorytmu ID3 trzeba czekać dłużej i czas ten wydłuża się jeszcze bardziej, gdy obsługujemy gębsze drzewa.

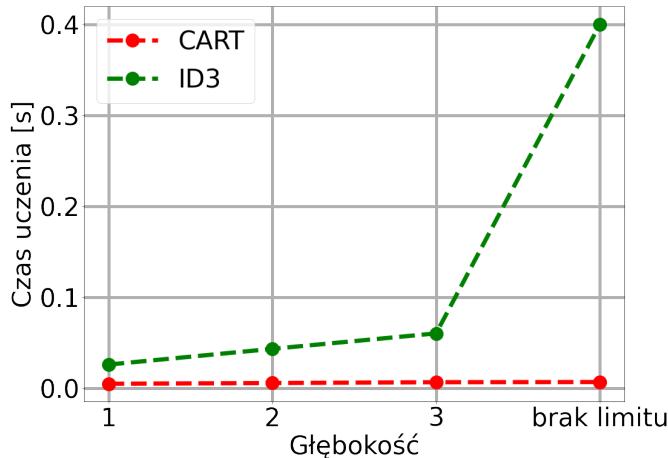
Ostatnią metryką, która posłuży nam do porównania tych dwóch algorytmów będzie dokładność. Otóż postaramy się zbadać który algorytm generuje bardziej dokładne drzewa pod względem klasyfikacji. Jak wiadomo, różne algorytmy generowania drzewa stosują różne kryteria podziału drzewa. I tak algorytm ID3 wykorzystuje entropię (ang. *entropy*) lub też przyrost informacji (ang. *information gain*). Z drugiej strony algorytm CART może korzystać przykładowo z miary nieczystości Giniego (ang. *Gini impurity*). Stąd mogą wynikać różnice w dokładności tych drzew. Na rys. 5.28 przedstawiono porównanie tych wyników. Widzimy, że orientacyjnie wyniki są bardzo zbliżone. Nie można jednoznacznie stwierdzić które z tych podejść jest dokładniejsze, gdyż w zależności od głębokości raz jeden, raz drugi model osiąga nieznacznie lepsze wyniki. Jednakże biorąc pod uwagę poprzednie czynniki wydaje się,



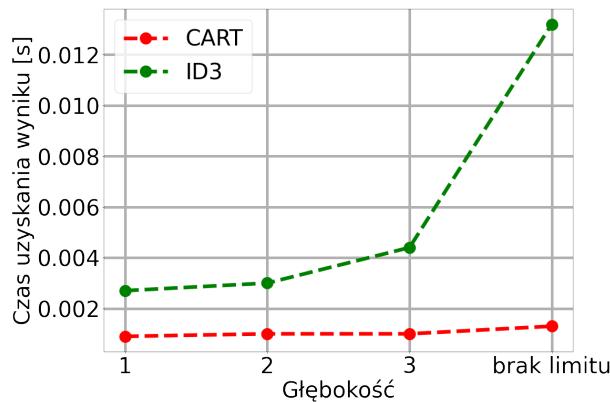
Rys. 5.25. Wizualizacja drzewa decyzyjnego o głębokości równej dwa. Źródło: opracowanie własne (z wykorzystaniem biblioteki *graphviz*)

że algorytm CART może być faktycznie lepszym wyborem, głównie, jeśli trenujemy głębokie drzewo i zależy nam na czasie, uwzględniając w tym czas uzyskania wyników.

Na koniec, zanim przedstawimy tabelę podsumowującą wszystkie testy związane z drzewem decyzyjnym, przyjrzymy się jeszcze wynikom osiąganym za pomocą drzewa decyzyjnego skonstruowanego przy pomocy algorytmu CART, gdy uprzednio zbalansujemy dane. Rys. 5.29 przedstawia wizualizację drzewa skonstruowanego za pomocą algorytmu CART używając optymalizacji hiperparametrów. Dokładność tego modelu wynosi 73.5%, a więc zanotowaliśmy spadek o 7.7 punktu procentowego. Można zauważyć, iż liczność struktury typu III jest tu decydująca. Gdy jest tych struktur pewna ilość (więcej niż dwie), to wtedy z dużą pewnością możemy stwierdzić, iż struktura ma małą wytrzymałość na rozciąganie. W przeciwnym przypadku (a więc gdy struktur jest mniej niż trzy) klasyfikacja nie jest już taka dokładna. Stąd można wyciągnąć wnioski, iż obecność struktur typu III powoduje, że struktura ma niską wytrzymałość na rozciąganie. Tablica pomyłek została przedstawiona na rys. 5.30. Jak można zauważyć, klasyfikacja jest nieco przesunięta w stronę wysokiej wytrzymałości (klasa 1). Można to wytlumaczyć przy pomocy wizualizacji drzewa (rys. 5.29). Gdy struktur typu III jest mniej niż trzy, dokładność naszej klasyfikacji wynosi około 69% (lewe poddrzewo). A ponieważ większość obecnych tam struktur ma wysoką wytrzymałość, stąd przesunięcie w kierunku tej klasy. Natomiast widzimy również spadki skuteczności w wykrywaniu wysokiej wytrzymałości, co nastąpiło wskutek ograniczenia liczby danych o tej własności. O ile wyniki są teraz bardziej wiarygodne, o tyle widzimy, że za pomocą samych liczności struktur nie jesteśmy w stanie stwierdzić z bardzo wysoką skutecznością jaka jest wytrzymałość tej struktury. Stąd teza, iż sieci neuronowe oceniące tę cechę odlewów na podstawie pikseli mogą osiągać wyższe skuteczności wydaje się jeszcze bardziej prawdopodobna.

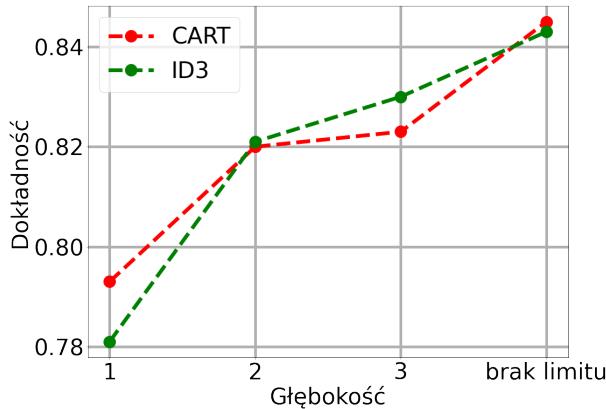


Rys. 5.26. Wykres przedstawiający czas uczenia modeli skonstruowanych przy pomocy różnych algorytmów generowania drzewa, w zależności od jego głębokości.
Źródło: opracowanie własne (z wykorzystaniem biblioteki *matplotlib*)



Rys. 5.27. Wykres przedstawiający czas uzyskania wyniku przez modele skonstruowane przy pomocy różnych algorytmów generowania drzewa, w zależności od jego głębokości.
Źródło: opracowanie własne (z wykorzystaniem biblioteki *matplotlib*)

Jako podsumowanie prac z drzewami decyzyjnymi zostaje przedstawiony szkic (tab. 5.6), który zawiera wyniki większości wartych uwagi testów, również tych, które nie zostały tutaj bliżej opisane. Oczywiście najlepsze wyniki otrzymujemy dla głębszych drzew. Jednakże warto zwrócić uwagę przede wszystkim na dwie rzeczy. Po pierwsze, po zbalansowaniu danych zanotowaliśmy znaczny spadek skuteczności, co już zostało omówione wyżej w tym podrozdziale. Po drugie, najwyższą dokładność uzyskujemy dla danych w postaci bezwzględnej liczności struktur poszczególnych klas, nieco gorsze wyniki, gdy na wejściu podajemy procentowy udział struktur poszczególnych klas, a najgorsze wyniki, gdy użyjemy funkcji softmax. Poniekąd jest zrozumiałe, że dla funkcji softmax otrzymujemy nieco gorsze wyniki, gdyż usuwa ona z danych pewną informację. Natomiast spadek dla danych w postaci procentowego udziału jest nietypowy, aczkolwiek podobne wyniki uzyskano dla modelu SVM. Proste wyjaśnienie

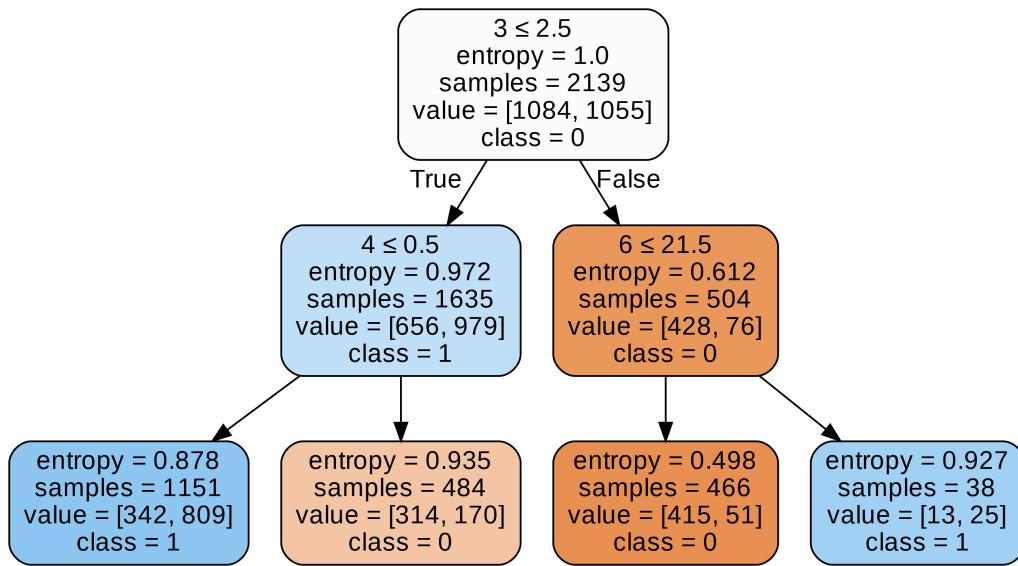


Rys. 5.28. Wykres przedstawiający dokładność uzyskaną przez modele skonstruowane przy pomocy różnych algorytmów generowania drzewa, w zależności od jego głębokości. Źródło: opracowanie własne (z wykorzystaniem biblioteki *matplotlib*)

może opierać się na tym, iż tak na prawdę większy wpływ na wytrzymałość struktur ma to, czy struktury konkretnej klasy w ogóle wystąpiły niż stosunek ich liczności względem pozostałych klas.

5.3.2.3. Las losowy

Kolejny klasyfikator klasyczny, którego możliwości w ocenie wytrzymałości na rozciąganie odlewów zostały przebadane, to las losowy (rozdz. 3.6.3). Tutaj szczegółowo zostaną zaprezentowane wyniki uzyskane z wykorzystaniem optymalizacji hiperparametrów oraz kroswalidacji, na danych oryginalnych oraz zbalansowanych. Natomiast w tabeli na końcu podrozdziału zostaną przedstawione wszystkie znaczące wyniki, których testy były analogiczne do przedstawionych. W pierwszej kolejności przebadamy model dla danych oryginalnych (rozdz. 4.3.1). Otóż model ten uzyskał skuteczność na poziomie 83%, średnią wartość precyzji na poziomie 0.91 oraz wartość funkcji straty logistycznej na poziomie 0.4. Wartość metryki F1 wyniosła 0.89. Ogólnie można stwierdzić, że wyniki są jak najbardziej poprawne, jednakże patrząc na wykres tablicy pomyłek (rys. 5.31) ponownie widać to, co występowało w poprzednich klasyfikatorach. Mianowicie jest to przesunięcie w stronę klasy z wysoką wytrzymałością. Dlatego powtórzymy te testy dla zbalansowanych danych i porównamy je z poprzednimi wynikami. Zbalansowane dane są tworzone na tej samej zasadzie, co w rozdz. 5.3.2.1, tzn. wszystkie przykłady o niskiej wytrzymałości są uwzględniane w zbiorze (ponieważ jest ich zdecydowanie mniej), natomiast z przykładów o wysokiej wytrzymałości wybieramy losowo tyle przykładów, aby było ich tyle samo, co w poprzedniej klasie. Wyniki, jakie udało się osiągnąć dla tego zestawienia, to dokładność na poziomie 80.9%, średnia wartość precyzji na poziomie 0.85, wartość funkcji straty logistycznej na poziomie 0.47 oraz wartość metryki F1 na poziomie 0.81. Jak więc widzimy, zanotowano spadek tych metryk, jednakże nie jest on tak duży, jak w przypadku drzew decyzyjnych (rozdz. 5.3.2.2). Na rys. 5.7 przedstawiono tablicę pomyłek. Jak widzimy, zbalansowanie danych przyniosło oczekiwany efekt. Dodatkowo pokazało, że o ile faktycznie zanotowano spadki w skuteczności, o tyle były one bardzo niewielkie, co dowodzi skuteczności tego podejścia. Tabela dodatkowo podsumowuje większość przeprowadzonych badań,



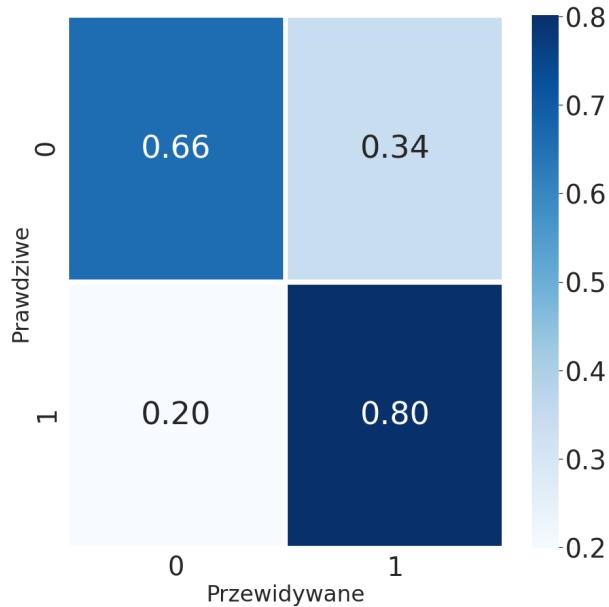
Rys. 5.29. Wizualizacja drzewa decyzyjnego o głębokości równej dwa. Dodatkowo zbalansowano dane oraz użyto optymalizacji hiperparametrów. Źródło: opracowanie własne (z wykorzystaniem biblioteki *graphviz*)

które osiągnęły istotne wyniki. Jak możemy zauważyć, tendencja skuteczności względem różnych typów wejścia jest zachowana. Ponownie najlepsze wyniki otrzymujemy na danych w postaci liczności struktur, aczkolwiek wyniki pomiędzy różnymi typami wejścia nie różnią się znacząco. Ponadto wyniki dla danych zbalansowanych są zaledwie 2.1 punktu procentowego gorsze od najlepszych wyników osiągniętych przy pomocy lasu losowego.

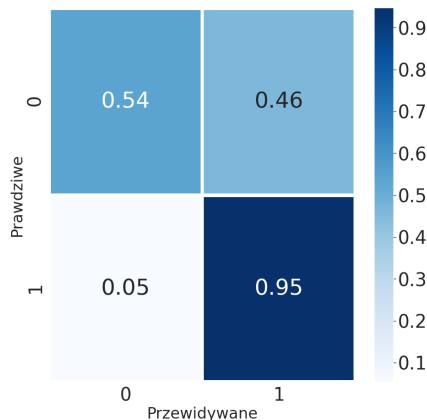
5.3.2.4. Regresja logistyczna

Następnym modelem, który został przetestowany jest regresja logistyczna (rozdz. 3.6.5). Trzymając się schematu z poprzednich podrozdziałów, ponownie przedstawimy szczegółowe wyniki dla danych w postaci liczności klas dla danych niezbalansowanych i zbalansowanych, po czym na końcu rozdziału zostanie przedstawiona tabela podsumowująca większość prac z użyciem tego modelu. Jak się okazało, nieco nieoczekiwanie, wyniki dla regresji logistycznej były całkiem przyzwoite. Otóż zaczynając od danych niezbalansowanych, otrzymaliśmy następujące wyniki: 83.1% dokładności, 0.91 średniej wartości prezentacji, 0.41 wartości funkcji straty logistycznej oraz 0.88 wartości metryki F1. Na rys. 5.33 przedstawiono tablicę pomyłek dla tej klasyfikacji i jak możemy zauważyć, problem z poprzednich klasyfikatorów wciąż jest aktualny. Natomiast dla danych zbalansowanych otrzymaliśmy następujące wyniki:

- dokładność – 82.6%,
- średnia wartość precyzji – 0.85,
- wartość funkcji straty logistycznej – 0.52,
- wartość metryki F1 – 0.8.



Rys. 5.30. Tablica pomyłek dla klasyfikacji wytrzymałości na rozciąganie odlewów wykorzystując drzewo decyzyjne o głębokości równej dwa. Testy przeprowadzono na zbalansowanych danych. Źródło: opracowanie własne (z wykorzystaniem biblioteki *seaborn*)



Rys. 5.31. Tablica pomyłek dla klasyfikacji wytrzymałości na rozciąganie odlewów wykorzystując las losowy. Testy przeprowadzono na danych niezbalansowanych. Źródło: opracowanie własne (z wykorzystaniem biblioteki *seaborn*)

Tablica pomyłek zaprezentowana została na rys. 5.34. Ponownie widzimy, iż zbalansowanie danych powoduje, że wartości metryk nieco spadają, ale tym razem spadki te są mniejsze niż w przypadku poprzednich klasyfikatorów. W tab. 5.8 przedstawiono najważniejsze wyniki osiągnięte z wykorzystaniem regresji logistycznej. Porównując te wyniki do tych osiągniętych przez las losowy (tab. 5.7) możemy zauważyc, że zarówno dla danych oryginalnych najlepszy wynik został osiągnięty przez regresję logistyczną (o 0.1 punktu procentowego), jak i dla danych zbalansowanych – tutaj przewyższając las losowy aż o 1.7 punktu procentowego.

Tabela 5.6. Podsumowanie wyników klasyfikacji binarnej z użyciem drzew decyzyjnych w celu predykcji wytrzymałości na rozciąganie odlewów. Źródło: opracowanie własne

Typ wejścia	Głębokość	Dane zbalansowane	Dokładność
Liczność ^a	1	Nie	78%
Liczność	2	Nie	81.2%
Liczność	2	Tak	73.5%
Liczność	3	Nie	81.4%
Liczność	Brak limitu	Nie	84.3%
Procent ^b	1	Nie	75%
Procent	2	Nie	79%
Procent	3	Nie	80%
Softmax ^c	1	Nie	74.5%
Softmax	2	Nie	75%
Softmax	3	Nie	78%

^a Liczność struktur poszczególnych klas.

^b Procentowy udział struktur poszczególnych klas.

^c Wartość funkcji softmax dla liczności struktur.

Tabela 5.7. Podsumowanie wyników klasyfikacji binarnej z użyciem lasu losowego w celu predykcji wytrzymałości na rozciąganie odlewów. Źródło: opracowanie własne

Typ wejścia	Dane zbalansowane	Dokładność
Liczność ^a	Nie	83%
Liczność	Tak	80.9%
Procent ^b	Nie	81.8%
Softmax ^c	Nie	81.9%

^a Liczność struktur poszczególnych klas.

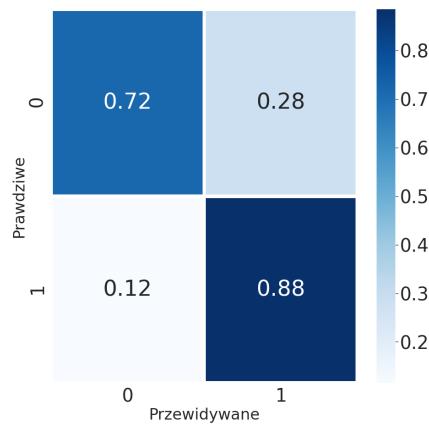
^b Procentowy udział struktur poszczególnych klas.

^c Wartość funkcji softmax dla liczności struktur.

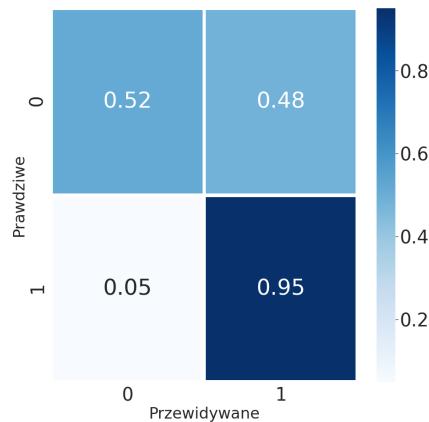
5.3.2.5. Perceptron wielowarstwowy

Kolejnym przetestowanym modelem był perceptron wielowarstwowy (rozdz. 3.6.8). Jest to nieco inny model od wszystkich przedstawionych w tym rozdziale, gdyż jest to najpopularniejszy typ sztucznych sieci neuronowych. Wyniki dla danych niezbalansowanych prezentują się następująco:

- dokładność – 82.6%,
- średnia wartość precyzji – 0.85,
- wartość funkcji straty logistycznej – 0.41,



Rys. 5.32. Tablica pomyłek dla klasyfikacji wytrzymałości na rozciąganie odlewów dla algorytmu lasu losowego. Wykorzystano dane zbalansowane. Źródło: opracowanie własne (z wykorzystaniem biblioteki *seaborn*)



Rys. 5.33. Tablica pomyłek dla klasyfikacji wytrzymałości na rozciąganie odlewów dla regresji logistycznej. Wykorzystano dane niezbalansowane. Źródło: opracowanie własne (z wykorzystaniem biblioteki *seaborn*)

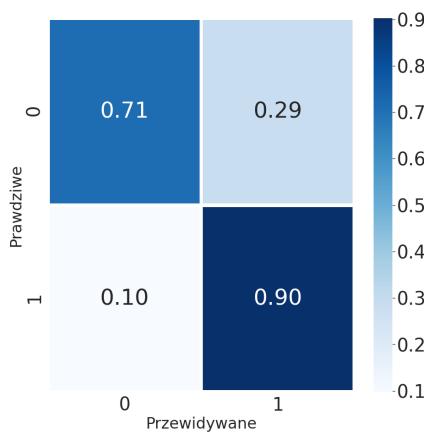
Tabela 5.8. Podsumowanie wyników klasyfikacji binarnej z użyciem regresji logistycznej w celu predykcji wytrzymałości na rozciąganie odlewów. Źródło: opracowanie własne

Typ wejścia	Dane zbalansowane	Dokładność
Liczność ^a	Nie	83.1%
Liczność	Tak	82.6%
Procent ^b	Nie	81.8%
Softmax ^c	Nie	77.7%

^a Liczność struktur poszczególnych klas.

^b Procentowy udział struktur poszczególnych klas.

^c Wartość funkcji softmax dla liczności struktur.



Rys. 5.34. Tablica pomyłek dla klasyfikacji wytrzymałości na rozciąganie odlewów dla regresji logistycznej. Wykorzystano dane zbalansowane. Źródło: opracowanie własne (z wykorzystaniem biblioteki *seaborn*)

- wartość metryki F1 – 0.88.

Oprócz dokładności, wartości wszystkich pozostałych metryk są najlepsze w porównaniu do innych modeli. W poniższej tabeli (tab. 5.9) przedstawiono wszystkie istotne wyniki osiągnięte z wykorzystaniem tego modelu. Jak widzimy, wyniki są dosyć imponujące na tle poprzednich modeli, szczególnie jeśli

Tabela 5.9. Podsumowanie wyników klasyfikacji binarnej z użyciem perceptronu wielowarstwowego w celu predykcji wytrzymałości na rozciąganie odlewów. Źródło: opracowanie własne

Typ wejścia	Dane zbalansowane	Dokładność
Liczność	Nie	82.7%
Liczność	Tak	80.6%
Procent	Nie	81.9%
Softmax	Nie	79.6%

chodzi o dane zbalansowane. Dodatkowo wyniki są bardziej równomierne dla różnych typów wejścia w porównaniu do poprzednich modeli.

5.3.2.6. AdaBoost

Ostatnim modelem przetestowanym w tym zestawieniu jest algorytm AdaBoost, który został przedstawiony w rozdz. 3.6.7.1. Składa się on z wielu tzw. słabych klasyfikatorów, a w każdej iteracji waga źle zaklasyfikowanych obserwacji jest zwiększana. W bibliotece, z której korzystano (*scikit-learn*) domyślnym słabym klasyfikatorem jest drzewo decyzyjne i jest ich 50. Można by oczekiwać wysokich skuteczności tego modelu, niewiele gorszych od lasu losowego. Wyniki dla danych zbalansowanych przedstawiają się następująco:

- dokładność – 83.4%,

- średnia wartość precyzyji – 0.91,
- wartość funkcji straty logistycznej – 0.41,
- wartość metryki F1 – 0.88.

Jest to najwyższa dokładność jaką udało się uzyskać dla tego typu danych. W poniższej tabeli (tab. 5.10)

Tabela 5.10. Podsumowanie wyników klasyfikacji binarnej z użyciem algorytmu AdaBoost w celu predykcji wytrzymałości na rozciąganie odlewów. Źródło: opracowanie własne

Typ wejścia	Dane zbalansowane	Dokładność
Liczność	Nie	83.4%
Liczność	Tak	72%
Procent	Nie	81.3%
Softmax	Nie	80.6%

przedstawiono wszystkie istotne wyniki osiągnięte z wykorzystaniem tego modelu. Warto tutaj zwrócić uwagę na dane zbalansowane. Jak można zauważyć, nastąpił znaczny spadek skuteczności. Możliwe wyjaśnienie opiera się na nadmiernym dopasowaniu się modelu do danych treningowych. Jak wiadomo, modele oparte na wzmacnianiu mają taką tendencję, tym bardziej, iż dane zbalansowane zawierają 40% wszystkich dostępnych danych.

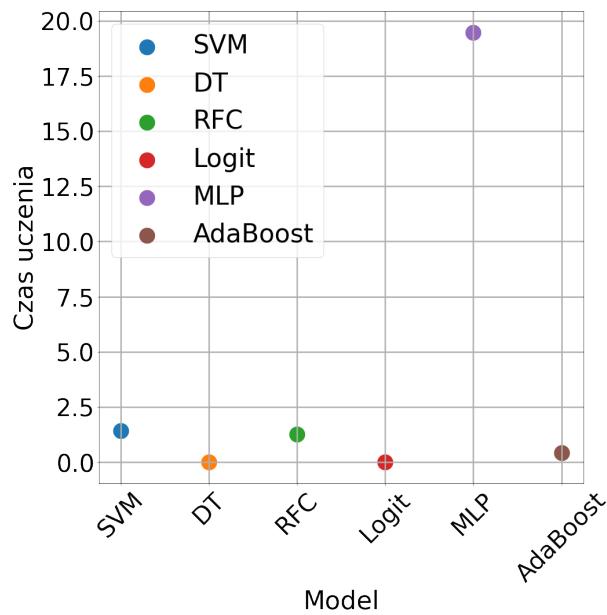
5.3.2.7. Podsumowanie

Podsumowując wszystkie najważniejsze wyniki osiągnięte w tym podrozdziale przygotowano tabelę (tab. 5.11). Zawarto w niej najlepsze osiągnięte wyniki przez wszystkie przetestowane modele na danych zarówno zbalansowanych, jak i niezbalansowanych, wykorzystując optymalizację hiperparametrów oraz kroswalidację. Uwzględnione zostały również modele, których przebieg badań nie został szczegółowo opisany w powyższych podrozdziałach. Jak można zauważyć, dla niezbalansowanych danych dobrze prezentują się modele maszyny wektorów nośnych, lasu losowego oraz AdaBoost, co jest oczekiwany efektem, gdyż są to bardzo popularne algorytmy, ogólnie uważane za solidne. Natomiast dla danych zbalansowanych bezsprzecznie najlepszym modelem jest perceptron wielowarstwowy, który osiągnął najlepsze wartości dla wszystkich metryk.

Oprócz tego szerokiego porównania modeli na podstawie metryk wydajności przygotowano również wykresy przedstawiająca uśrednione czasy uczenia oraz czasy zwracania wyników dla poszczególnych modeli. Jak możemy zauważyć, zdecydowanie najdłużej trwa wytrenowanie modelu perceptronu wielowarstwowego, co jest zrozumiałe ze względu na to, że jest to tak naprawdę sieć neuronowa z wieloma neuronami na warstwę oraz setkami iteracji. Następnie mamy modele maszyny wektorów nośnych oraz lasu losowego. SVM opiera się na funkcji jądra (ang. *kernel function*) i dla dużych danych może działać powoli. Natomiast las losowy składa się z setek pojedynczych modeli (tzw. słabych klasyfikatorów), w tym przypadku są to tzw. kikuty decyzyjne, a więc drzewa decyzyjne o głębokości równej jeden. Uczenie modelu RFC trwa ponad 400 razy dłużej niż pojedyncze drzewo decyzyjne, co by się zgadzało z tym,

Tabela 5.11. Podsumowanie wyników klasyfikacji binarnej dla wszystkich wykorzystanych klasyfikatorów. W zestawieniu przedstawiono najlepsze wyniki, które udało się osiągnąć dla poszczególnych modeli dla danych niezbalansowanych oraz zbalansowanych. Źródło: opracowanie własne

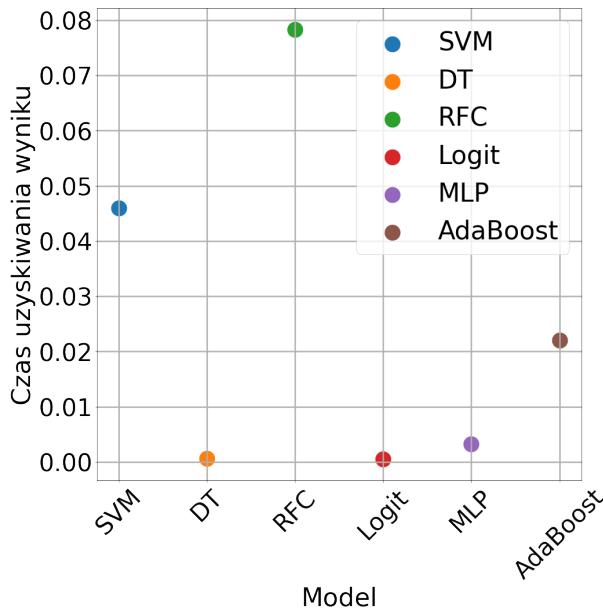
Model	Dane zbalansowane	Dokładność	Log loss	F1	AP
SVM	Nie	83.3%	0.39	0.9	0.9
SVM	Tak	76.9%	0.49	0.78	0.81
DT	Nie	81.6%	0.46	0.89	0.81
DT	Tak	69.5%	0.58	0.74	0.65
RFC	Nie	83.3%	0.4	0.9	0.92
RFC	Tak	78.4%	0.47	0.79	0.83
Logit	Nie	82.2%	0.41	0.89	0.91
Logit	Tak	77.1%	0.52	0.78	0.8
MLP	Nie	82.7%	0.41	0.88	0.91
MLP	Tak	80.6%	0.46	0.83	0.84
AdaBoost	Nie	83.4%	0.67	0.89	0.92
AdaBoost	Tak	72%	0.62	0.76	0.77



Rys. 5.35. Uśrednione czasy uczenia dla poszczególnych modeli. Czasy obliczono dla danych niezbalansowanych. Źródło: opracowanie własne (z wykorzystaniem biblioteki *matplotlib*)

że w testach wykorzystano model lasu losowego z 473 drzewami. Następnie mamy model AdaBoost. Ten model również składa się z wielu słabych klasyfikatorów i również są to kikuty decyzyjne. Ponieważ jest

ich 50, jego czas uczenia też jest odpowiednio dłuższy. Pozostałe dwa modele, tj. drzewo decyzyjne i regresja logistyczna mają zdecydowanie najkrótsze czasy uczenia (bliskie zera). Są to najprostsze modele niewymagające wielu obliczeń. W tym przypadku badania potwierdziły to, co znaliśmy z teorii. Poniżej znajduje się wykres dot. czasów uzyskania wyników dla tych samych modeli. W przypadku tego wy-



Rys. 5.36. Uśrednione czasy zwracania wyników dla poszczególnych modeli. Czasy obliczono dla danych niebalansowanych. Źródło: opracowanie własne (z wykorzystaniem biblioteki *matplotlib*)

kresu wyniki również pokrywają się z oczekiwaniami. Na pierwszy rzut oka może zdziwić tak niski czas zwracania wyniku dla sieci neuronowej, aczkolwiek sieci neuronowe są bardzo szybkie, jeżeli chcemy otrzymać predykcję dla pojedynczej instancji wejściowej. Jest to tak naprawdę kilka działań mnożenia i otrzymujemy wynik. Natomiast najdłużej należy czekać na wynik lasu losowego. Jest to zrozumiałe, gdyż składa się on z ponad 450 drzew decyzyjnych. Następnie mamy modele SVM oraz AdaBoost. Wytyłumaczenie dla AdaBoosta jest analogiczne do przypadku lasu losowego. Natomiast ponownie najkrócej należy oczekwać na wyniki pojedynczego drzewa decyzyjnego oraz regresji logistycznej. Jak wiadomo są to algorytmy bardzo proste matematycznie.

Podsumowując, w zależności od wymagań użytkownika powinien samodzielnie wybrać który algorytm będzie miał najlepsze zastosowanie. W naszym przypadku, gdzie danych nie było zbyt dużo oraz nie były bardzo skomplikowane, tak naprawdę wszystkie klasyfikatory spełniły swoje wymagania. Natomiast, jak pokazały badania, modele te różnią się między sobą i dla dużych danych te różnice czasowe mogą stać się bardzo duże i mieć ogromne znaczenie dla całego procesu (uczenia jak i ewaluacji).

5.3.3. Sieci neuronowe

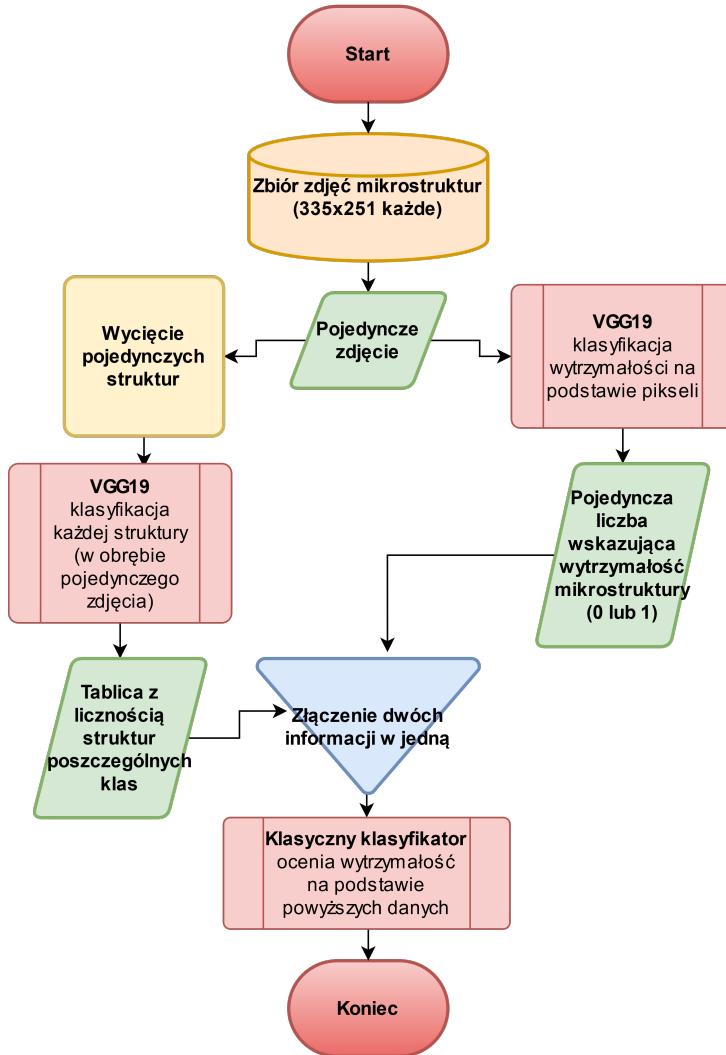
Gdy mamy do czynienia z klasyfikacją obrazów, zwyczajowym już podejściem jest zastosowanie sieci neuronowych. Od wielu lat jest to metoda dająca najwyższe skuteczności i wciąż jest rozwijana. Dlatego postaramy się ją również zastosować w naszych badaniach. Ze względu na to, iż nie posiadamy danych w liczbie dziesiątek tysięcy, wykorzystamy technikę zwaną uczeniem przez transfer (rozdz. 3.5). W skrócie polega ona na tym, iż na sieć neuronową trenowaną na milionach przykładów oraz wielu klasach nakładamy kilka dodatkowych warstw i staramy się ją doczycić do własnych celów. W ten sposób korzystamy z już posiadanej wiedzy przez tę sieć neuronową. W naszych badaniach użyjemy sieć VGG19. Oprócz tego nałożymy na sam koniec sieci trzy dodatkowe warstwy i będą to jedyne trenowalne warstwy w naszej architekturze. Ponadto zostanie zastosowany tzw. **dropout**, a więc losowane usuwanie pojedynczych neuronów w sieci w trakcie uczenia, co przeciwdziała nadmiernemu dopasowaniu. Wartości hiperparametrów zostały wybrane na drodze optymalizacji wykorzystując bibliotekę *optuna*. W ramach przypomnienia, najlepszą skuteczność jak do tej pory uzyskaliśmy za pomocą MLP, które również jest siecią neuronową, a wyniosła ona 80.6%. Dlatego oczekujemy wzrostu skuteczności, jak również wartości pozostałych metryk. Po przeprowadzeniu badań otrzymano następujące wyniki:

- dokładność – 83.2%,
- średnia wartość precyzji – 0.88,
- wartość funkcji straty logistycznej – 0.45,
- wartość metryki F1 – 0.81.

Jak widzimy są to najwyższe jak do tej pory wyniki dla danych zbalansowanych, co jest oczekiwany wynikiem. Aczkolwiek można byłoby spodziewać się większej skuteczności ze względu na to, iż jest to ogromna sieć, która wcześniejsze warstwy miała wytrenowane na milionach przykładów. Stąd w następnym, ostatnim już podrozdziale postaramy się połączyć podejścia, tzn. wykorzystamy sieć neuronową (do zliczania struktur) oraz klasyczny klasyfikator (do klasyfikacji wytrzymałości). W ten sposób model będzie miał więcej informacji na wejściu i być może poprawi to jego skuteczność.

5.3.4. Podejście hybrydowe

Ostatnim przebadanym podejściem jest połączenie sieci neuronowych z klasycznymi klasyfikatorami. Sieć neuronowa nie ma wiedzy o tym, jakie struktury znajdują się na obrazach, natomiast klasyczne klasyfikatory nie dostają na wejście pikseli, a więc również mają ograniczone dane. Stąd pojawił się pomysł, aby wynik klasyfikacji sieci neuronowej dołączyć do liczby struktur (która de facto również jest zliczana przez sieć neuronową), a następnie to wejście w postaci ośmiu liczb (w rozdz. 5.3.2 było ich siedem) klasyfikować za pomocą klasycznych modeli. W ten sposób w pełni wykorzystamy korzyści jakie dają nam zarówno sieci neuronowe, jak i klasyczne klasyfikatory. Schemat został przedstawiony na rys. 5.37. Można byłoby się spodziewać, iż to podejście będzie przynajmniej tak dobre, jak sama sieć



Rys. 5.37. Schemat blokowy przedstawiający działanie klasyfikacji wytrzymałości na rozcięganie na podstawie liczności struktur poszczególnych klas a także wyniku klasyfikacji sieci neuronowej. Źródło: opracowanie własne (z wykorzystaniem *diagrams.net*)

neuronowa klasyfikująca wytrzymałość, gdyż tutaj wykorzystujemy jej predykcję. Tak jak poprzednio, obliczono wartości najważniejszych metryk i prezentują się one następująco:

- dokładność – 83.2%,
- średnia wartość precyzji – 0.82,
- wartość funkcji straty logistycznej – 0.47,
- wartość metryki F1 – 0.83.

Jak więc widzimy, wyniki są bardzo zbliżone do tych otrzymanych przez sieć neuronową VGG19 (rozdz. 5.3.4). Mimo, iż zostały połączone dwa najlepsze podejścia nie udało się osiągnąć lepszych wyników,

aczkolwiek dokładność osiągnęła taki sam poziom jak najlepszy model dla danych zbalansowanych. Z racji ograniczonych zasobów czasowych autor nie mógł kontynuować dalszych badań, aczkolwiek podejście hybrydowe wydaje się mieć potencjał i warto byłoby go rozwinać w przyszłości.

6. Podsumowanie i wnioski

W trakcie realizacji tego projektu pojawiało się wiele problemów, które należało rozwiązać. Pomijając przegląd literatury, następnym krokiem było odpowiednie przygotowanie danych. Przede wszystkim konieczne było sprowadzenie zdjęć do tej samej skali oraz przybliżenia, a także usunięcie z nich niepotrzebnych elementów (jak rozmiar skali). Dodatkowo sztucznie wygenerowano kolejne dane za pomocą technik augmentacji, które mogłyby się przydać w przyszłości. Po obsłużeniu prac związanych z samymi danymi można już było przejść do poważniejszych zagadnień, które były tematem tej pracy. Idąc chronologicznie, pierwszym problemem była klasyfikacja struktur. Pierwszym przebadanym podejściem była uogólniona transformata Hougha, która to metoda służy do znajdowania na obrazie zapytania kształtów przedstawionych na obrazie referencyjnym. Co prawda, po kilku modyfikacjach procedury, wizualnie wyniki wyglądały świetnie (rozdz. 5.2.1), aczkolwiek było to podejście mało praktyczne, gdyż dla każdego zdjęcia mikrostruktury potrzebowalibyśmy co najmniej sześć obrazów referencyjnych, a potencjalnie nawet więcej, gdyż w rzeczywistości struktury na zdjęciach nieco się różnią od tych wskazanych w normie (rys. 4.2). Mimo iż ta metoda była już blisko rozwiązania, to tak na prawdę dopiero kolejne przebadane metody spełniły nasze oczekiwania, a mianowicie algorytmy z biblioteki *OpenCV*, takie jak wygładzanie, progowanie, a przede wszystkim metoda wykrywania krawędzi Canny'ego [63]. Dzięki tym metodom stało się możliwe wycinanie praktycznie wszystkich struktur, a następnie ręczne ich klasyfikowanie w celu stworzenia nowej bazy danych. Wyniki, jakie udało się osiągnąć (najlepszy wynik to 82.2% dokładności) zostały zaprezentowane w rozdz. 5.2.4. Wyniki wizualnie wyglądają znakomicie, aczkolwiek weryfikując je za pomocą tablicy pomyłek (rys. 5.10) okazuje się, że występują lekkie zakłamania. Przede wszystkim wyniki dla pięciu z siedmiu klas wyglądają przyzwoicie, natomiast dla pozostałych dwóch klas (II oraz IV) są nieco gorsze. Przeprowadzono wiele testów i wszystko niestety wskazuje na to, iż za pomocą tych danych nie można osiągnąć już wiele lepszych wyników.

Następnie, w rozdz. 5.3 przedstawiono badania, które były głównym tematem niniejszej pracy. W celu klasyfikacji binarnej wytrzymałości na rozciąganie odlewów wykorzystano metody zaprezentowane w poprzednich rozdziałach. Aczkolwiek pierwszymi przebadanymi metodami były momenty Hu oraz tekstury Haralicka (rozdz. 5.3.1), które mogłyby się wydawać idealnymi narzędziami do tego celu, jeśli tylko struktury obecne na zdjęciach mają wpływ na wytrzymałość. Dodatkowo wykorzystano klasyczne modele klasyfikujące. Niestety, mimo iż wyniki były zróżnicowane, od niskich (55% dokładności) do wysokich (70% dokładności) to wciąż nie są to wyniki w pełni satysfakcjonujące. Następnie zostały przetestowane klasyczne klasyfikatory z wykorzystaniem wygenerowanych przez autora danych

oraz klasycznych klasyfikatorów (rozdz. 5.3). Jest to najbardziej obszerny punkt badań, jednakże wyniki zostały zbiorczo przedstawione w rozdz. 5.3.2.7. Jak się okazało, jest to bardzo skuteczna metoda, gdyż udało się osiągnąć skuteczność na poziomie aż 83.4% dokładności! W dodatku są to metody wysoce interpretowalne, przez co mogłyby służyć jako pomoc dla specjalistów, którzy za pomocą zdjęć chcieliby ocenić wytrzymałość danego metariału. Zaś w ostatnich testach (rozdz. 5.3.4) przebadano skuteczność bardzo popularnych obecnie sieci neuronowych (w szczególności *VGG19*), a także połączenie sieci neuronowych z klasycznymi klasyfikatorami (rozdz. 5.3.4). Jak się jednak okazało, nie udało się osiągnąć lepszych wyników, zaledwie dorównując najlepszym otrzymanym wynikom do tej pory. Aczkolwiek wiemy, że sieci neuronowe sprawdzają się znacznie lepiej dla dużej liczby danych, co tutaj było lekkim problemem (opisane w dalszej części).

Jako dalszą pracę nad tym tematem można uznać konieczność lepszego przygotowania danych uczących (sześć kształtów z rysunku 4.2), na przykład przy pomocy specjalistów w tej dziedzinie. Przede wszystkim najważniejsze jest, aby w danej klasie znajdowały się możliwie tylko poprawne przykłady. Natomiast problemem niższej rangi (aczkolwiek w celu otrzymania jak najlepszych wyników również ważnym) jest zbalansowanie klas. Dodatkowo autor wybrał zaledwie ułamek z wyciętych struktur (około 1500), gdyż po sprowadzeniu zdjęć do tej samej skali (tzn. zdjęcia z przybliżeniem 100x sprowadzono do przybliżenia 500x, tym samym uzyskując z takiego pojedynczego zdjęcia kolejnych 25) jest ich aż 4797, z czego z każdego takiego zdjęcia jest wycinanych średnio 20 struktur, a więc liczba możliwych struktur uzyskanych z tego zbioru może być nawet większa, niż 100 000 (słownie: sto tysięcy), a więc wykorzystano około 1.5% dostępnych danych. Stąd widać, że w kwestii danych jest jeszcze sporo miejsca na poprawę. Natomiast widzimy, że zarówno modele klasyczne jak i sieci neuronowe uzyskały przyzwoite skuteczności. W szczególności na ten moment uzasadnione wydaje się wykorzystanie modeli klasycznych (które de facto pośrednio również korzystają z efektów sieci *VGG19*), które mogłyby posłużyć ekspertom w tej dziedzinie jako pomoc w uzasadnieniu decyzji co do wytrzymałości danego materiału. Jeśli chodzi zaś o sieci neuronowe, tutaj konieczne wydaje się uzbieranie większej ilości danych i powtórzenie podobnych testów w celu wykazania, że mogą one osiągać jeszcze lepsze wyniki.

Bibliografia

- [1] W. Reczek. *Raport z pracowni problemowej: Analiza obrazów w celu identyfikacji parametrów materiałów odlewniczych*. 2021.
- [2] D. Olson. „Prediction of Austenitic Weld Metal Microstructure and Properties”. W: *Weld. J. (Miami); (United States)* (1985).
- [3] H. F. Reid i W. T. DeLong. „Making sense out of ferrite requirements in welding stainless steels”. W: *Metal Progress* (1973).
- [4] 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).
- [5] *Measure Ferrite Content of Austenitic and Duplex Steel*. source. Diverse.
- [6] R. Saluja. „Formation, Quantification and Significance of Delta Ferrite for 300 Series Stainless Steel Weldments”. W: *Academia.edu* (2015).
- [7] S. S. Babu et al. „New Model for Prediction of Ferrite Number of Stainless Steel Welds”. W: *Taylor & Francis* (2013).
- [8] D. Kotecki i T. Siewert. „WRC-1992 constitution diagram for stainless steel weld metals: a modification of the WRC-1988 diagram”. W: (1992).
- [9] J. Vitek et al. „Improved Ferrite Number Prediction Model That Accounts for Cooling Rate Effects: Part 2: Model Results”. W: *Semantic Scholar* (2003).
- [10] 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).
- [11] H. K. D. H. Bhadeshia. „Neural Networks in Materials Science”. W: *ISIJ International, The Iron and Steel Institute of Japan* (2007).
- [12] 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).
- [13] I. Santos et al. „Machine-learning-based Mechanical Properties Prediction in Foundry Production”. W: *IEEE Xplore* (2009).

- [14] J. Nieves et al. „Mechanical Properties Prediction in High-Precision Foundry Production”. W: *IEEE Xplore* (2009).
- [15] *Klasyfikacja Statystyczna*. https://pl.wikipedia.org/wiki/Klasyfikacja_statystyczna. 2019.
- [16] Y. Y. Yang et al. „Tensile Strength Prediction for Hot Rolled Steels by Bayesian Neural Network Model”. W: *IFAC Proceedings Volumes, Elsevier* (2016).
- [17] 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).
- [18] Y. K. Penya i in. „Advanced fault prediction in high-precision foundry production”. W: *IEEE Xplore* (2008).
- [19] S. L. Shrestha et al. „An Automated Method of Quantifying Ferrite Microstructures Using Electron Backscatter Diffraction (EBSD) Data”. W: *Ultramicroscopy* (2013).
- [20] A. Schneider D. Britz J. Webel i F. Mücklich. „Identifying And Quantifying Microstructures in Low-alloyed Steels: A Correlative Approach”. W: *Metallurgia Italiana* (2017).
- [21] S. M. Azimi et al. „Advanced Steel Microstructural Classification by Deep Learning Methods”. W: *Nature News, Nature Publishing Group* (2018).
- [22] D. Britz J. Pauly i F. Mücklich. „Advanced Microstructure Classification Using Data Mining Methods”. W: *Computational Materials Science* (2016).
- [23] A. R. Durmaz et al. „A Deep Learning Approach for Complex Microstructure Inference”. W: *Research Square* (2021).
- [24] A. Stoll i P. Benner. „Machine Learning for Material Characterization with an Application for Predicting Mechanical Properties”. W: *Wiley Online Library* (2021).
- [25] T. M. Mitchell. „Machine Learning”. W: *McGraw-Hill* (1997).
- [26] 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).
- [27] K. Sawka. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow*. Wydawnictwo Helion, 2018.
- [28] *Tablica pomyłek*. https://pl.wikipedia.org/wiki/Tablica_pomy%C5%82ek. 2020.
- [29] K. Sawka. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow. Wydanie II*. Wydawnictwo Helion, 2020.
- [30] A. Burns, B. Dobbing i T. Vardanega. *Machine Learning and AI via Brain simulations*. Spraw. tech. Stanford University & Google, 2014.
- [31] *Big Data: Week 3 Video 3 - Feature Engineering*. Spraw. tech. CCNMTL, 2014.
- [32] C. Shorten i T. M. Khoshgoftaar. „A Survey on Image Data Augmentation for Deep Learning”. W: *Journal of Big Data, Springer International Publishing* (2019).

- [33] A. Handhi. *Data Augmentation | How to use Deep Learning when you have Limited Data — Part 2*. Spraw. tech. Nanonets, 2021.
- [34] D. Sarkar. *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning*. Spraw. tech. Medium, Towards Data Science, 2018.
- [35] P. Chmiel. *Czym Jest Transfer Learning? Trenowanie Sieci Neuronowych w Deep Learning'u*. Spraw. tech. Greenlogic, 2020.
- [36] S. Bickel. „ECML-PKDD Discovery Challenge 2006 Overview”. W: *CiteSeerX* (2006).
- [37] 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).
- [38] A. B. Hur et al. „Support Vector Clustering”. W: *The Journal of Machine Learning Research* (2002).
- [39] Drzewo decyzyjne. https://pl.wikipedia.org/wiki/Drzewo_decyzyjne. 2020.
- [40] J. Grus. *Data science od podstaw. Analiza danych w Pythonie*. Wydawnictwo Helion, 2018.
- [41] J. R. Quinlan. „Improved Use of Continuous Attributes in C4.5”. W: *ArXiv.org* (1996).
- [42] Decision tree. https://en.wikipedia.org/wiki/Decision_tree. 2021.
- [43] T. K. Ho. „Random Decision Forests”. W: *IEEE Xplore* (1995).
- [44] P. Ramchandani. *Random Forests and the Bias-Variance Tradeoff*. Spraw. tech. Medium, Towards Data Science, 2021.
- [45] *What is the difference between Bagging and Boosting?* Spraw. tech. QuantDare, 2016.
- [46] 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).
- [47] 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).
- [48] *K-Nearest Neighbors Algorithm*. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm. 2021.
- [49] *Logistic regression*. https://en.wikipedia.org/wiki/Logistic_regression. 2021.
- [50] S. Menard. „Applied Logistic Regression Analysis”. W: *SAGE Publications* (2002).
- [51] *Naive Bayes Classifier*. https://en.wikipedia.org/wiki/Naive_Bayes_classifier. 2021.
- [52] V. Kurama. *A Guide To Understanding AdaBoost*. Spraw. tech. Paperspace Blog, 2021.
- [53] 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).

- [54] C. Li. *A Gentle Introduction to Gradient Boosting*. Spraw. tech. College of Computer and Information ScienceNortheastern University.
- [55] R. Tadeusiewicz i M. Szaleniec. *Leksykon Sieci Neuronowych*. Academia.edu, Projekt Nauka Fundacja na rzecz promocji nauki polskiej, 2015.
- [56] K. Simonyan i A. Zisserman. „Very Deep Convolutional Networks for Large-Scale Image Recognition”. W: *ArXiv.org* (2015).
- [57] C. C. Chang C. W. Hsu i C.J. Lin. *A Practical Guide to Support Vector Classification*. 2003.
- [58] Z. Pirowski. *Opracowanie innowacyjnych elementów roboczych maszyn sektora leśnego i przetwórstwa biomasy w oparciu o wysokoenergetyczne technologie powierzchniowej modyfikacji warstwy wierzchniej elementów odlewanych*. 2017.
- [59] Nowej generacji system posuwu wysokowydajnych kompleksów ścianowych. 2017.
- [60] H. J. Watson. „The CRISP-DM Model: The New Blueprint for Data Mining”. W: *International Journal of Data Warehousing and Mining* (2000).
- [61] Mikrostruktura żeliwa – Część 1: Klasyfikacja wydzielień grafitu na podstawie analizy wizualnej. Spraw. tech. KT 301, Odlewnictwa, Polski Komitet Normalizacyjny, 2018.
- [62] Vinnie Monaco. <https://github.com/vmonaco/general-hough>. General Hough implementation. 2013.
- [63] J. Canny. „A Computational Approach to Edge Detection”. W: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1986).