



**Politechnika
Śląska**

**POLITECHNIKA ŚLĄSKA
WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI
KIERUNEK AUTOMATYKA I ROBOTYKA**

Projekt inżynierski

Wykorzystanie sieci neuronowych do detekcji ziaren w obrazach
termowizyjnych

Autor: Maciej Ziaja

Kierujący pracą: dr inż. Sebastian Budzan

Gliwice, styczeń 2020

Streszczenie

Tematem pracy jest klasyfikacja (rozpoznawanie typu) ziaren rud miedzi na podstawie nagrań dokonanych metodą termowizji aktywnej. Zaproponowano sposób wykonania pomiarów oraz zbierania danych z procesu stygnięcia badanych próbek miedzi. W celu budowy zbioru danych przeprowadzono eksperymenty z użyciem kamery termowizyjnej wyposażonej w makroobiektyw. Zgromadzone materiały przeanalizowano pod kątem użyteczności w zadaniu klasyfikacji. Zaproponowano możliwe sposoby ekstrakcji cech charakterystycznych rud miedzi oraz wybrano najlepszy, polegający na detekcji detali obrazu o niskiej emisjności cieplnej. Przedstawiono metody zliczania i śledzenia elementów charakterystycznych rud miedzi w cyklu zdjęć. Rozpatrywane sposoby ekstrakcji danych z nagrań zaimplementowano w języku programowania Python. Zgromadzone dane i opracowane metody eksploracji danych wykorzystano do konstrukcji prototypu sieci neuronowej klasyfikującej ziarna rud miedzi. Przeprowadzono ocenę działania klasyfikatora oraz wytypowano najlepszy uzyskany sposób na detekcję typu ziaren. Walidacja uzyskanego modelu wskazała na jego użyteczność w klasyfikacji nagrań procesu stygnięcia rud miedzi. Zaproponowano pomysły na rozbudowę stanowiska pomiarowego, automatyzację procesu zbierania danych, perspektywy rozwoju projektu oraz rozpatrzone możliwe alternatywne rozwiązania. Stworzone oprogramowanie może stanowić podstawę do dalszych badań ziaren rud miedzi metodą termowizyjną. Efekty pracy mogą stanowić punkt wyjścia dla kolejnych projektów oraz prób wdrożenia opracowanej technologii w przemyśle.

Spis treści

1 Wstęp	1
1.1 Motywacja projektu	1
1.2 Cel pracy	1
2 Założenia projektowe i wykorzystane narzędzia	3
2.1 Analizowane ziarna rud miedzi	3
2.2 Metody pomiarów termowizyjnych	3
2.3 Kamera termowizyjna FLIR A320	5
2.3.1 Opis sprzętowy wykorzystywanej kamery	5
2.3.2 Obiektyw kamery	7
2.3.3 Oprogramowanie do obsługi kamery	8
2.4 Narzędzia programistyczne	8
2.4.1 Język programowania Python	8
2.4.2 Biblioteka przetwarzania obrazu Scikit-image	9
2.4.3 Interfejs sieci neuronowych Keras	9
3 Analiza i ekstrakcja danych termowizyjnych	11
3.1 Proces pomiarowy i budowa zbioru danych	11
3.2 Analiza obrazów termowizyjnych	12
3.2.1 Prezentacja przykładowej serii pomiarowej	13
3.2.2 Przetwarzanie danych wizyjnych	13
3.2.3 Odczyt zakresu pomiarowego temperatur z obrazu	13
3.3 Eksploracja danych	15
3.3.1 Wybór cech obrazu użytecznych w klasyfikacji	17
3.3.2 Wybór algorytmu detekcji ziaren	19
3.3.3 Algorytm śledzenia ziaren w serii zdjęć	21
3.3.4 Wizualizacja zebranych cech i ocena ich użyteczności	27
4 Prototypowanie sieci neuronowej klasyfikującej ziarna miedzi	30
4.1 Konstrukcja prototypu sieci neuronowej	30
4.1.1 Dobór struktury sieci	30

Spis treści

4.1.2	Trening sieci neuronowej	33
4.2	Walidacja i ocena działania sieci	36
5	Podsumowanie	41
5.1	Wyniki	41
5.2	Wnioski	41
Dodatki		43
A	Porównanie konfiguracji i parametrów modelu sieci	44
B	Organizacja projektu	45
B.1	Struktura plików projektu	45
B.2	Dokumentacja projektu	46
B.3	Narzędzia pomocnicze	46
C	Spis wykorzystanych funkcji bibliotecznych	47

Rozdział 1

Wstęp

1.1 Motywacja projektu

Mielenie i granularyzacja materiałów są istotnymi elementami wielu procesów przemysłowych [2]. Badanie stopnia rozdrobnienia, składu oraz kształtów ziaren są kluczowe dla zapewnienia odpowiedniej jakości przetwarzania surowców. W ramach działalności Politechniki Śląskiej w Gliwicach powstało wiele prac naukowych oraz projektów związanych z procesem mielenia materiałów. Podczas badań opracowano, między innymi systemy: sterowania procesem mielenia, wizji komputerowej oraz klasyfikacji i obróbki danych związanych z ziarnami rud miedzi. W czasie trwania projektów zebrano wiele próbek materiałów pochodzących z przemysłowych młynów surowców [1, 2].

Politechnika Śląska jest także w posiadaniu laboratorium termowizji, które wyposażono w kamery pracujące w spektrum podczerwieni. W ramach projektu inżynierskiego zdecydowano się na realizację zadania detekcji i klasyfikacji ziaren rud miedzi z użyciem termowizji oraz sieci neuronowych. Ze względu na nieinwazyjny charakter pomiarów, techniki termowizyjne są popularnym narzędziem w realizacjach przemysłowych. Sieci neuronowe stanowią obecnie jedną z najszybciej rozwijających się i obiecujących technik uczenia maszynowego. Połączenie pomiarów termowizyjnych oraz sztucznych sieci neuronowych jest interesującym tematem pracy. Realizacja zadania klasyfikacji ziaren rud miedzi, z użyciem przedstawionych technik, stanowi próbę alternatywnego rozwiązania problemów rozpatrywanych we wcześniejszych prowadzonych w Politechnice Śląskiej.

1.2 Cel pracy

Celem pracy jest stworzenie systemu detekcji ziaren oraz klasyfikacji ich typu. Punktem wyjścia projektu są próbki rud miedzi, które posiadają odgórnie przypis-

sany typ (klasę). Budowa klasyfikatora polega na stworzeniu programu, który dla rudy miedzi o nieznanej klasie, będzie w stanie określić jej typ. W rozpatrywanym zbiorze znajdują się cztery rodzaje rud miedzi, czyli należy zbudować klasyfikator *wieloklasowy*.

Zdecydowano się klasyfikować rudy miedzi na podstawie nagrani ich procesu stygnięcia, do rejestrowania którego służy kamera termowizyjna. Na podstawie nagrani z kamery należy utworzyć zbiór danych, który będzie podstawą do treningu oraz oceny sieci neuronowej. Ponieważ próbki przeznaczone do treningu sieci mają znane klasy, to zadanie należy do grupy problemów *nadzorowanego uczenia maszynowego* [3, str. 108]. Oznacza to, że sieć będzie uczona poprzez podanie danych wejściowych oraz klasy jaka jest oczekiwana na wyjściu. Trening sieci będzie polegał na dopasowaniu jej do etykiet (klas) zbioru uczącego.

Zadania uczenia maszynowego wykonuje się w kilku etapach, które odpowiadają rozdziałom pracy. Kolejne kroki rozwiązania problemu klasyfikacji są następujące:

1. Analiza problemu oraz posiadanych materiałów.
2. Planowanie oraz przeprowadzenie eksperymentu uzyskania danych potrzebnych do nauki algorytmu klasyfikacji.
3. Analiza i wizualizacja zebranych danych.
4. Przygotowanie danych i ekstrakcja cech kluczowych dla algorytmów uczenia maszynowego.
5. Konstrukcja modelu oraz jego nauka.
6. Dostrojenie modelu na podstawie systemu walidacji.
7. Prezentacja rozwiązania.

Przedstawione kroki zadania klasyfikacji wymagają rozwiązania szeregu problemów związanych z rozpatrywanym przypadkiem. Prace należy zacząć od zapoznania się z dostępnym sprzętem termowizyjnym oraz oceną jego użyteczności w pomiarach ziaren rud miedzi. Następnie należy zaplanować i przeprowadzić eksperiment pomiarowy, którego wynikiem będą nagrania ziaren. Materiały wideo oraz zdjęcia zawierają wiele danych, należy je przeanalizować oraz poddać obróbce. Wynikiem eksploracji danych powinien być wektor liczb, które reprezentują numerycznie cechy charakterystyczne rozpatrywanych klas rud miedzi. Kolejnym krokiem jest implementacja sztucznej sieci neuronowej w postaci programu komputerowego. Sieć należy zaprojektować uwzględniając cechy zbudowanego zbioru danych. Ocena sieci powinna przebiegać na podstawie wyników systemu walidacji. Realizacja kolejnych etapów pracy prowadzi do celu jakim jest konstrukcja działającego klasyfikatora rud miedzi.

Rozdział 2

Założenia projektowe i wykorzystane narzędzia

2.1 Analizowane ziarna rud miedzi

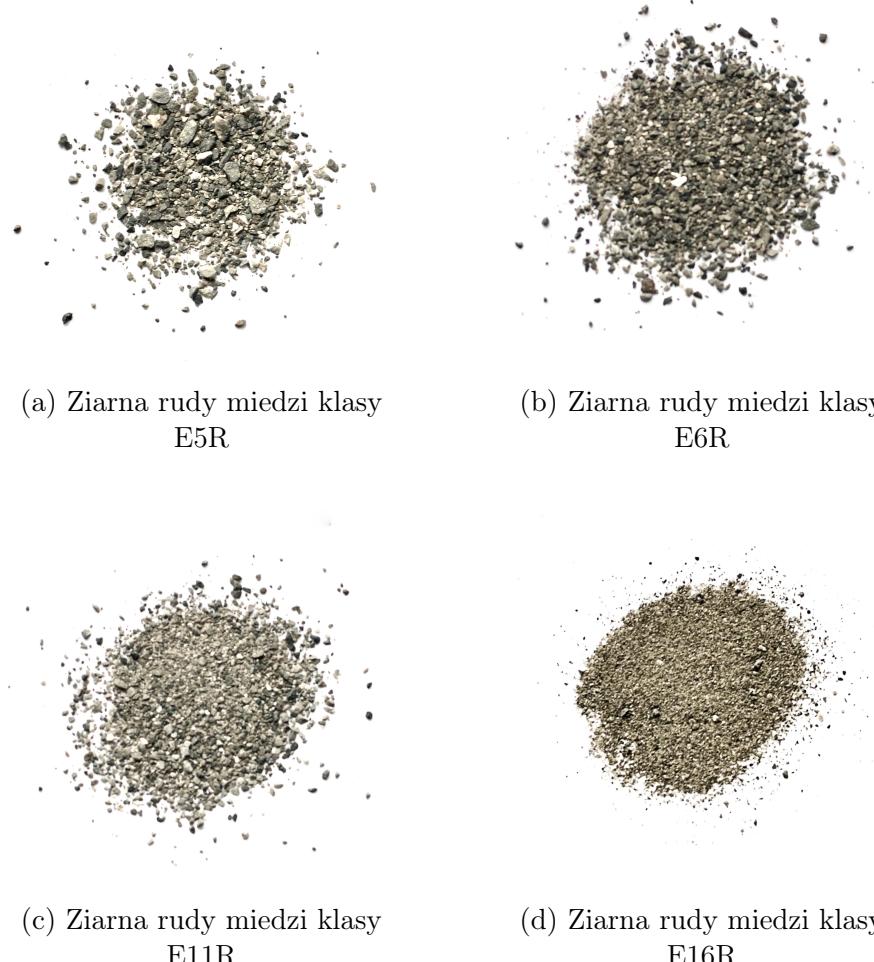
Zestaw analizowanych ziaren pochodzi z projektu SYSMEL (*System sterowania procesem mielenia w układzie z młynem elektromagnetycznym*)¹, którego współtwórcami są naukowcy z Politechniki Śląskiej. Uzyskane próbki powstały w procesie mielenia rud miedzi w młynie elektromagnetycznym. Poszczególne klasy ziaren różnią się granulacją i parametrami procesu ich mielenia. Spośród próbek, pochodzących z projektu SYSMEL, wybrano cztery typy ziaren, które będą przedmiotem zadania klasyfikacji. Zdjęcia rozpatrywanych materiałów przedstawiono na rysunku 2.1.

2.2 Metody pomiarów termowizyjnych

Pomiary termowizyjne (nazywane także *termografią*) polegają na rejestracji promieniowania cieplnego obiektów w celu ustalenia ich temperatury. Badania z wykorzystaniem termowizji można podzielić na dwie grupy: *termowizję pasywną* oraz *termowizję aktywną*.

Technika pasywna polega na rejestracji obrazów obiektów bez ingerencji w ich temperaturę w czasie trwania pomiarów. Można w ten sposób obserwować przepływ ciepła w urządzeniach technicznych, procesach przemysłowych oraz biologicznych. W rozważanym w pracy przypadku technika ta nie ma jednak zastosowania.

¹Pełen tytuł projektu naukowego: *Układ mielenia surowców mineralnych w młynie elektromagnetycznym wraz z systemem sterowania jego pracą, zapewniający wysoką efektywność technologiczną i niską energochłonność w zastosowaniach mikro i makro-przemysłowych*.



Rysunek 2.1: Próbki rozpatrywanych ziaren rud miedzi

W warunkach pokojowych badany materiał ma w całej swojej objętości podobną temperaturę, co skutkuje obrazem równomiernego szumu w rejestrowanym obrazie.

Bardziej zaawansowaną techniką są pomiary z wykorzystaniem termowizji aktywnej. W tym trybie pomiarów badane obiekty ogrzewają się w powtarzalnych warunkach, a następnie rejestruje proces ich stygnięcia. Podczas dostarczania ciepła do obiektu oraz jego oddawania na obrazach termowizyjnych można zaobserwować strukturę badanego obiektu. Przedmioty o niejednorodnej i złożonej budowie nagrzewają się oraz stygną nierównomiernie, co jest rejestrowane przez kamery termowizyjne. W ten sposób można dostrzec cechy obiektu niewidoczne gołym okiem, takie jak zmiany jego gęstości, składu oraz uszkodzenia struktury. Termowizję aktywną stosuje się szeroko w badaniach naukowych oraz przemyśle. Przykładowe aplikacje tej techniki to detekcja defektów w produktach przemysłowych

oraz wykrywanie części konstrukcji podatnych na zużycie. Termowizja aktywna ma charakter niedestruktywny oraz bezkontaktowy, co stanowi jej zalety w analizie materiałowej [5]. Technika ta wymaga jednak etapowego procesu pomiarowego, potrzebne jest przygotowanie instalacji grzewczej, a nagrzewanie i stygnięcie materiału może być czasochłonne. Przebieg zmian temperatury najlepiej rejestrować na materiałach wideo, aby zmaksymalizować ilość danych zgromadzonych w trakcie eksperymentu.

2.3 Kamera termowizyjna FLIR A320

Przedmiotem projektu jest analiza obrazów pochodzących z przemysłowej kamery termowizyjnej FLIR A320. Firma FLIR zajmuje się produkcją wysokiej jakości kamer i detektorów do celów profesjonalnych i przemysłowych. Używany aparat łączy wysoką jakość pomiaru z nowoczesnymi funkcjami integracji z oprogramowaniem komputerowym. Urządzenie komunikuje się z komputerem za pomocą sieci internetowej, pozwalając na kontrolę z poziomu aplikacji oraz bibliotek programistycznych. Kamera posiada także możliwość planowania automatycznych pomiarów i alarmów, oferuje również funkcje analityczne oraz wbudowany serwer internetowy [7].

2.3.1 Opis sprzętowy wykorzystywanej kamery

Kamera ma postać podłużnego korpusu, do którego swobodnie można podłączać obiektywy. Na rysunku 2.2 przedstawiono zdjęcie wykorzystywanego urządzenia. Egzemplarz kamery znajdujący się w laboratorium termowizji Politechniki



Rysunek 2.2: Kamera termowizyjna FLIR A320 [14]

Śląskiej został wyposażony w obiektyw, pozwalający oglądać ziarna rud miedzi w powiększeniu. Kamera cechuje się następującymi parametrami [14]:

- typ detektora: niechłodzony mikrobolometr,
- rozdzielcość: 320 na 240 pikseli,
- częstotliwość odświeżania: 9 Hz do 30 Hz,
- szerokość otworu: $f1,3$,
- autofocus z wbudowanym silnikiem,
- zakres pomiarowy temperatur:
 - ◊ od -15°C do 50°C ,
 - ◊ od 0°C do 350°C ,
- dokładność: $\pm 2^{\circ}\text{C}$ lub $\pm 2\%$ odczytu,
- zakres wykrywanego widma promieniowania: $7,5 \mu\text{m}$ do $13 \mu\text{m}$.

Kamera wykrywa temperaturę przez detektor zwany *bolometrem*, który mierzy energię niesioną przez fale elektromagnetyczne w spektrum podczerwieni. Kiedy fala pada na detektor aparatu, temperatura komórek matrycy rośnie i zwiększa się ich rezystancja elektryczna. Wartości rezystancji są mierzane, a na ich podstawie określana jest temperatura [17].

Zakres temperatur kamery jest odpowiedni do przeprowadzenia eksperymentów z pomiarami ziaren rud miedzi metodą termowizji aktywnej. Zaplanowano podgrzewanie próbek maksymalnie do temperatury około 80°C , wartość ta mieści się w zakresie pracy urządzenia. Dokładność kamery jest zadowalająca, próbne materiały nagraniowe wskazały, że na zdjęciach widocznych jest wiele szczegółów i detali badanego materiału. Przy klasyfikacji obrazów i wzorców stygnięcia próbek jest to bardziej istotne niż liczbowa dokładność pomiarowa przyrządu.

W porównaniu ze zwykłymi, współczesnymi aparatami, rozdzielcość kamery termowizyjnej może wydawać się mała. Należy sobie jednak uzmysłowić, że w standardowych aparatach piksele mają rozkład Bayera, a wartości składowych koloru są interpolowane. W kamerze termowizyjnej każda komórka matrycy dokonuje pełnego pomiaru wartości temperatury. Z tego powodu bezpośrednie porównanie rozdzielcości używanego przyrządu z popularnymi aparatami może być mylące. Oczywiście większa rozdzielcość kamery byłaby pożądana, jednak jej obecne możliwości pozwalają na szczegółowe pomiary i obserwację wielu detali ziaren rud miedzi.

2.3.2 Obiektyw kamery

Kamerę wyposażono w obiektyw zbliżeniowy FLIR T197415, pozwalający obserwować ziarna rud miedzi. Jest to sprzęt zaprojektowany przez producentów używanej kamery i przeznaczony do pracy z urządzeniami termowizyjnymi. Przyrząd przedstawiono na rysunku 2.3. Wybrany obiektyw jest przygotowany z myślą



Rysunek 2.3: Obiektyw FLIR T197415 [15]

o obserwacji drobnych detali powierzchni w dużym zbliżeniu. Używany model ma następujące parametry [15]:

- ogniskowa: 18,2 mm,
- powiększenie: 1×1 ,
- pole widzenia: 8 mm na 6 mm,
- odległość od płaszczyzny ostrzenia: 20 mm,
- głębia ostrości: 0,3 mm,
- przysłona: bez regulacji, równa otworowi systemu montażu kamery,
- budowa: trzy soczewki asferyczne.

Zgodnie ze specyfikacją producenta obiektyw ma powiększenie 1×1 , co może wydawać się niedużą wartością. Należy mieć jednak świadomość, że zwykłe obiektywy zmniejszają obraz padający na matrycę. Powszechnie jako granicę makrofotografii przyjmuje się powiększenie 1×1 . W obserwacji ziaren i detali powierzchni nie jest jednak istotne powiększenie, ale to że używany obiektyw jest *zbliżeniowy*. Oznacza to, że ma on bardzo małą odległość ostrzenia, czyli można go przysunąć blisko obserwowanej powierzchni. Obiektyw zbliżeniowy pozwala na obserwację z dystansu 20 mm, typowe obiektywy ostrzą z odległości parudziesięciu centymetrów do ponad metra. Dzięki temu obiektyw zbliżeniowy pozwala na obserwację bardzo drobnych detali powierzchni.

2.3.3 Oprogramowanie do obsługi kamery

Jedną z najważniejszych cech kamery jest łatwość integracji z oprogramowaniem komputerowym. Kamerę można obsługiwać za pomocą programu FLIR Tools². Pozwala on na podgląd obrazu oraz wykonywanie zdjęć termowizyjnych. Producent dostarcza również bibliotekę LabVIEW pozwalającą na zaawansowaną pracę z kamerą. Do obsługi stanowiska został napisany program używający tych bibliotek, który pozwala na nagrywanie materiałów wideo przy pomocy kamery. Nagrania mają własnościowy format firmy FLIR, można je odtwarzać w programie FLIR Tools. Program pozwala także na eksport stopklatek z nagrania w postaci plików JPEG. Przy obsłudze narzędzia ważne jest ustawianie zakresu temperatur na obrazie. Wybrany zakres określa w jaki sposób wartości temperatury są mapowane na kolory na zdjęciu, zawarte w tablicy LUT. Program oferuje tablice w skali szarości, takie zostały użyte w projekcie, możliwy jest także wybór tablic w postaci kolorowych gradientów. Wybór zakresu wpływa na wygląd wyświetlanego obrazu oraz eksportowanych klatek. Jego nieodpowiedni dobór może skutkować zbyt ciemnym, jasnym, lub mało kontrastowym obrazem. Aby zapewnić najlepsze wykorzystanie nagrani z kamery oraz powtarzalny charakter eksportu stopklatek korzystano z opcji automatycznego doboru zakresu temperatur, jaki jest wbudowany w program FLIR Tools.

2.4 Narzędzia programistyczne

2.4.1 Język programowania Python

Założenia projektu wymagają użycia języka programowania pozwalającego na zaawansowaną obróbkę obrazu oraz wydajne budowanie sieci neuronowych. W obu

²Strona produktu programu FLIR Tools: <https://www.flir.com/products/flir-tools/>

tych dziedzinach wiodącym językiem jest Python³, posiadający bogaty zestaw bibliotek. Język ten oferuje dużą wygodę programowania, co jest istotne przy pracy badawczej. Do wydajnych obliczeń numerycznych w Pythonie służy biblioteka NumPy. Oferuje ona klasy macierzy numerycznych oraz bogaty zestaw operacji matematycznych. Macierze biblioteki NumPy są mniej elastyczne od zwykłych list języka Python, jednak oferują dużo większą wydajność obliczeniową, między innymi dzięki wsparciu obliczeń wektorowych na odpowiednich procesorach. Wiele innych bibliotek języka wykorzystuje pakiet NumPy, na przykład przy przetwarzaniu obrazów, co pozwala na wysoką wydajność obliczeń. Połączenie wygody programowania z zadowalającą wydajnością bibliotek numerycznych, sprawia że Python jest dobrym wyborem do realizacji założeń projektu.

2.4.2 Biblioteka przetwarzania obrazu Scikit-image

Język Python oferuje bogaty zestaw bibliotek przetwarzania obrazu. Najbardziej popularną z nich jest biblioteka OpenCV napisana w języku C++. Zdecydowano się jednak na wybór mniej znanego pakietu Scikit-image⁴. Jest to biblioteka zorientowana na obliczenia naukowe i badawcze oraz napisana bezpośrednio z myślą o języku Python [16]. Scikit-image, w porównaniu z OpenCV, oferuje bardziej spójny interfejs programisty oraz lepiej wykorzystuje charakterystykę języka Python. Dodatkowo wybrana biblioteka jest częścią zestawu Scikit, w skład którego wchodzi pakiet Scikit-learn służący do uczenia maszynowego. Może on być przydatny przy tworzeniu sieci neuronowej, a spójność bibliotek z pakietem Scikit jest niewątpliwie zaletą. Wybrana biblioteka charakteryzuje się również dobrą dokumentacją [12] oraz zestawem przykładów.

2.4.3 Interfejs sieci neuronowych Keras

Wybór bibliotek głębokiego uczenia w języku Python jest bardzo szeroki. Do najpopularniejszych pakietów należą: Keras, TensorFlow oraz PyTorch. Zdecydowano się na wybór interfejsu biblioteki Keras⁴. Jest to pakiet nastawiony na elastyczność i możliwość eksperymentowania [4]. Moduł Keras oferuje także rozbudowaną dokumentację oraz zestaw poradników [9]. Keras nie jest jednak samodzielna biblioteką sieci neuronowych, a wyłącznie abstrakcyjnym interfejsem programistycznym. Używanie go wymaga wyboru wewnętrznego silnika biblioteki. Zdecydowano się na domyślną opcję użycia zaplecza pakietu TensorFlow. Użyto

³W projekcie użyto języka Python 3, który jest aktualnym standardem. Nie należy go mylić z niekompatybilną i przestarzałą wersją Python 2, która traci wsparcie w 2020 roku.

⁴Wersje używanych bibliotek są zawarte w pliku `requirements.txt` dołączonym do projektu. Na jego podstawie można utworzyć wirtualne środowisko języka Python z odpowiednim zestawem bibliotek. Więcej informacji o strukturze projektu zawarto w dodatku B.

wersji interfejsu Keras wchodzącej bezpośrednio w skład modułu TensorFlow. Taki sposób użycia bibliotek zapewnia ich najlepszą współpracę.

Rozdział 3

Analiza i ekstrakcja danych termowizyjnych

3.1 Proces pomiarowy i budowa zbioru danych

Zgodnie z opisem technik termowizyjnych przedstawionym w sekcji 2.2 zdecydowano się na przeprowadzenie pomiarów za pomocą termowizji aktywnej. Aby zrealizować pomiary przygotowano stanowisko laboratoryjne. Kamera termowizyjna została umieszczona na statywie, a do ogrzewania próbek zdecydowano się wykorzystać lampa halogenową.

W procesie termowizji aktywnej istotny jest charakter procesu nagrzewania materiału. Przy przygotowaniu pomiarów należy zdecydować pod jakim warunkiem zakończyć przekazywanie ciepła do próbki. Rozważono dwie możliwości:

- a) ogrzewanie próbek do osiągnięcia ustalonej temperatury,
- b) ogrzewanie próbek przez ustalony czas¹.

Zdecydowano się na metodę b), ze względu na wygodę jej realizacji. Doprowadzenie każdej próbki do tej samej temperatury wymaga pomiarów w czasie nagrzewania, co jest bardziej wymagające w realizacji. Wstępne obserwacje nagrani wskazują, że nagrzewanie materiału przez ustalony czas, pozwala na obserwacje unikalnych wzorców stygnięcia ziaren.

Następnie wybrano czas nagrywania materiałów wideo. Na podstawie próbnych nagrani zdecydowano się na ogrzewanie próbek przez jedną minutę oraz rejestrację ich stygnięcia przez cztery minuty. Taka konfiguracja daje, przy badanych próbkach rud miedzi, ostry i szczegółowy obraz w początkowej fazie nagrywania oraz widocznie rozmazany i mniej kontrastowy obraz pod koniec stygnięcia. Charakter

¹Przy założeniu stałej masy ogrzewanych próbek.

procesu przejścia między tymi stanami pozwoli na klasyfikację badanych ziaren rud miedzi.

Ostatnia decyzja kształtująca charakter pomiarów dotyczy chwili przechwytywania stopklatek z pozyskanych materiałów wideo. Na podstawie obserwacji zdecydowano się eksportować pięć klatek, jedną na początku każdej minuty nagrania.

Po ustaleniu planu eksperymentu pomiarowego przystąpiono do jego wykonania. Zgodnie z opisem badanych materiałów w sekcji 2.1, zgromadzono nagrania wideo dla czterech klas ziaren rud miedzi. Ze względu na czasochłonność pomiarów dla każdej klasy materiału nagrano trzy filmy. Przy nagrywaniu stygnięcia tej samej klasy materiału pomiędzy pomiarami próbki poddawano przemieszaniu, aby uniknąć powtarzania struktur ułożenia ziaren. Zgodnie z opisem w podsekcji 2.3.3, podczas zapisu zdjęć, używano opcji automatycznego doboru zakresu temperatur.

Proces pomiarowy w prototypowych warunkach był obarczony brakiem dużej precyzji i powtarzalności. Obserwacja uzyskanych nagrań pokazała, że obrazy tej samej klasy ziaren niekoniecznie osiągały równą temperaturę po ogrzewaniu przez ustalony czas. Problemem okazało się również dostosowanie ostrości obrazu z kamery. Jak wyjaśniono w podsekcji 2.3.2 wykorzystywana obiektyw cechuje się bardzo małą głębią ostrości, co powoduje że drobne ruchy kamery mogą spowodować utratę czytelności obrazu. Z kolei proces pomiarowy wymagał ciągłego przenoszenia próbki między stanowiskiem do podgrzewania materiału oraz nagrywania filmów. Należy mieć na uwadze że w czasie układania próbek pod kamerą oraz poprawiania ostrości postępowało stygnięcie ziaren, co sprzyjało brakowi powtarzalności pomiarów. Ze względu na przypadkowe utraty ostrości przy nagrywaniu oraz zbyt długi czas przenoszenia i przygotowania próbki do nagrywania, eksperiment wymagał czasem powtarzania. Ostatecznie jakość procesu pomiarowego oceniono na podstawie jego wyników w klasyfikacji ziaren, po skonstruowaniu sieci neuronowej.

3.2 Analiza obrazów termowizyjnych

Zgodnie z planem pomiarów, przedstawionym w sekcji 3.1, z każdego nagrania wybrano pięć stopklatek. Podczas eksperymentu uzyskano łącznie dwanaście pomiarów, zawierających sumarycznie 60 zdjęć. W ramach uczenia maszynowego jest to bardzo mały zbiór danych, biorąc jednak pod uwagę wstępno-badawczy charakter pracy oraz czasochłonność procesu pomiarowego zdecydowano, że jest to rozmiar zadawalający do pierwszych prób klasyfikacji. Zebrane materiały mają format JPEG. Do oznaczania zdjęć przyjęto schemat nazw jak w przykładzie: 115_E11R_1, gdzie człony nazwy oznaczają kolejno: automatyczny numer nagrania w programie FLIR Tools, klasę próbki oraz minutę nagrania.

3.2.1 Prezentacja przykładowej serii pomiarowej

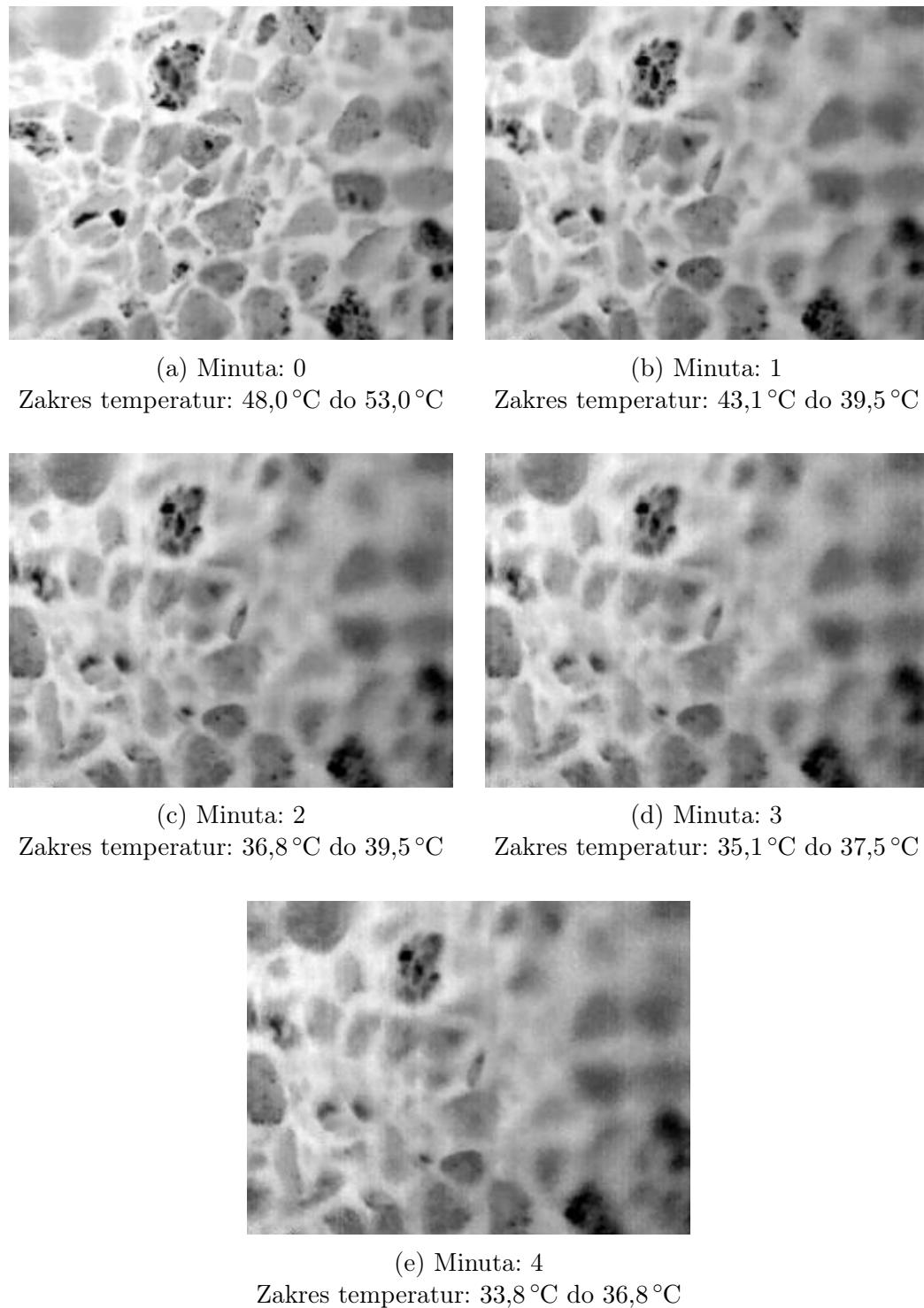
Jak opisano w rozdziale 3.1 jedna próbka w zbiorze danych składa się z serii pięciu zdjęć o malejącym kontraście i szczegółowości. Na rysunku 3.1 przedstawiono proces stygnięcia przykładowej próbki 104 klasy E5R. Wartości zakresu temperatur na zdjęciach maleją. Wskazuje to na stopniowy spadek sumarycznej temperatury obserwowanego materiału. Wraz z upływem czasu obraz staje się także coraz mniej wyraźny i kontrastowy. Nagrzana próbka emitemie ciepło ze swoich zróżnicowanych struktur w niejednorodny sposób. Wraz z ochłodzeniem ziaren temperatura na obrazie wyrównuje się i kamera termowizyjna rejestruje coraz mniej szczegółów. Detale i elementy charakterystyczne obrazu zlewają się na kolejnych zdjęciach. Wraz z opadaniem temperatury na zdjęciu pojawia się także coraz więcej szumów.

3.2.2 Przetwarzanie danych wizyjnych

Wykorzystanie wizji komputerowej w rozmaitych zadaniach często wymaga odpowiedniego przygotowania używanych obrazów. Do standardowych technik należą: wyrównywanie histogramu oraz redukcja szumów i wyostrzanie zdjęć. Po analizie zebranych obrazów i wypróbowaniu wymienionych technik zdecydowano się jednak nie modyfikować zgromadzonych nagrań. Użyty podczas pomiarów program FLIR Tools posiada wbudowaną funkcję doboru zakresu temperatur, co przekłada się na ekspozycję i kontrast eksportowanych zdjęć. Próba wyrównywania histogramu oraz korekcji kontrastu sprawiłaaby, że obraz rozkładu temperatur na zdjęciu stałby się przekłamany. Ze względu na niską, jak na standardy typowego przetwarzania zdjęć cyfrowych, rozdzielcość obrazów zrezygnowano także z redukcji szumów. Wiele detali ziaren ma wielkość zaledwie paru pikseli, proces usuwania szumu mógłby spowodować ich zanik. Nie oznacza to jednak, że zdjęcia charakteryzują się przypadkową ekspozycją. Algorytmy automatycznego doboru zakresu temperatur oprogramowania kamery powodują, że jasność elementów na zdjęciu dobrze reprezentuje właściwości cieplne obiektu.

3.2.3 Odczyt zakresu pomiarowego temperatur z obrazu

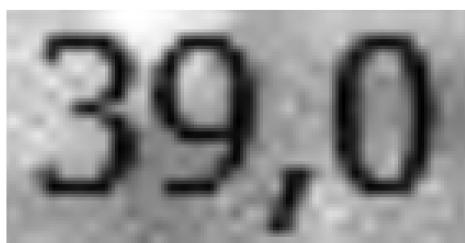
Jak wspomniano w sekcji 2.3.3 jednym z kluczowych czynników decydujących o wyglądzie obrazów pochodzących z kamery jest zakres temperatur mapowany na kolory w obrazie. Niestety aplikacja FLIR Tools nie pozwala na eksport zakresu temperatur wraz ze zdjęciami w formacie liczbowym. W czasie zapisu zdjęć oprogramowanie dodaje do nich interfejs z aktywną skalą pomiarową, jednak jest on graficznie naniesiony na obraz. Aby ułatwić w przyszłości pracę z materiałami z kamery opracowano mechanizm ekstrakcji zakresu temperatur ze zdjęć pochodzących



Rysunek 3.1: Zdjęcia procesu stygnięcia przykładowej próbki 104 klasy E5R

z kamery.

W celu odczytu wartości liczbowych z obrazu posłużyono się gotową siecią neuronową zaprojektowaną do detekcji tekstu na zdjęciach. Zdecydowano się na użycie popularnej biblioteki *Pytesseract*². Analizowane zdjęcia są kadrowane, by wyizolować odczytywane liczby. Wyjęte fragmenty mają bardzo małą rozdzielcość, aby ułatwić sieci rozpoznawanie liczb, zdecydowano się przeskalać obrazy w górę. W czasie skalowania włączono mechanizm anty aliasingu aby wyrównać krawędzie cyfr. Ponieważ używana sieć uznaje za tło kolor biały oraz poszukuje liczb w kolorze czarnym, barwy na zdjęciu odwrócono. Następnie obraz poddano binaryzacji metodą *otsu*. Jest to popularna i wydajna metoda binaryzacji, jej efektywność jest maksymalna kiedy liczba pikseli tła oraz pierwszego planu jest zbliżona [13]. Z tego powodu poprawne kadrowanie liczb sprzyja jakości ich binaryzacji. Na rysunku 3.2 przedstawiono kolejne etapy przygotowania obrazu do rozpoznania liczb. Implementację opisanego mechanizmu odczytywania zakresu temperatur ze zdjęć przedstawiono na listingu 1.



(a) Przeskalowany kadr z liczbą



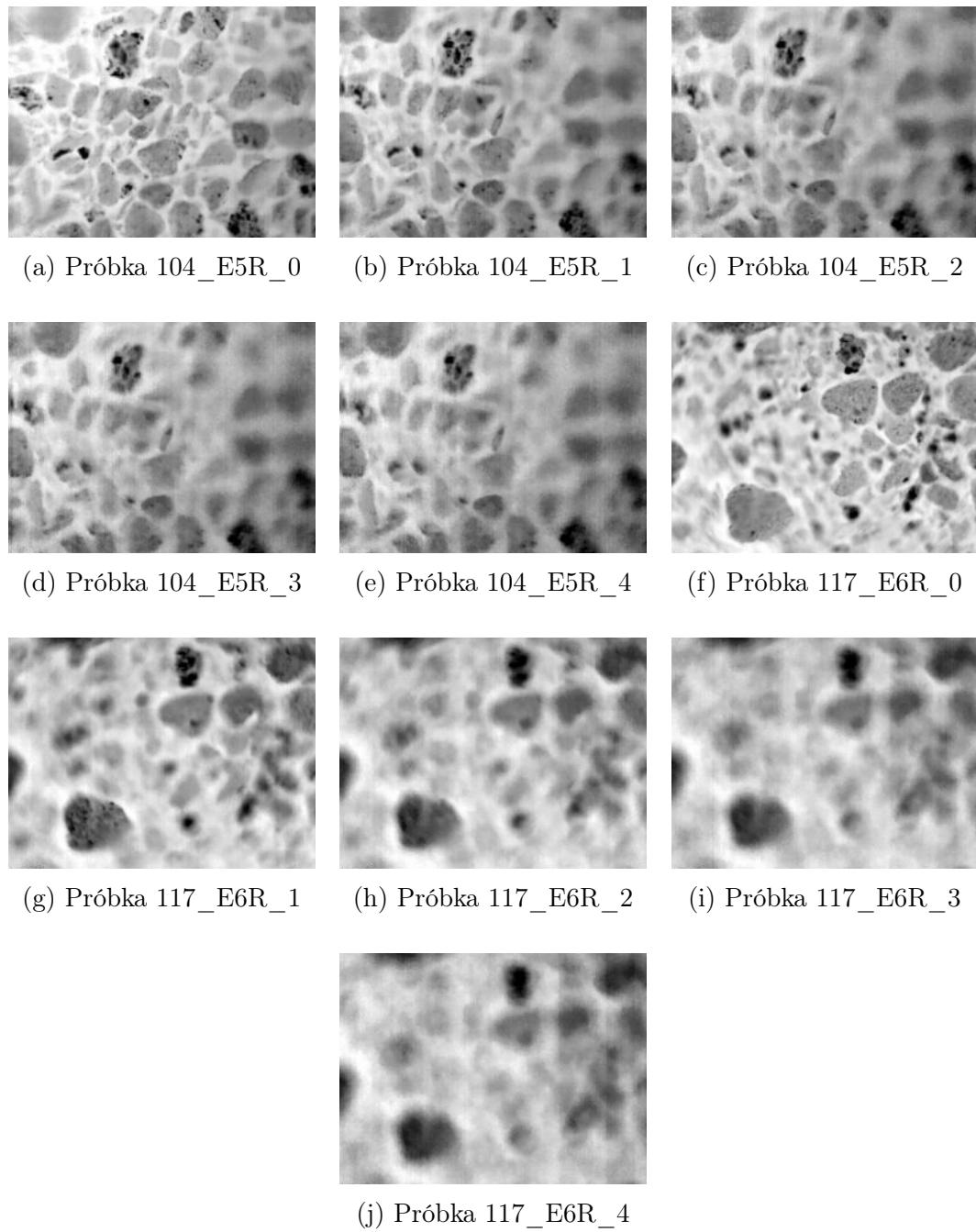
(b) Kadr z liczbą po binaryzacji

Rysunek 3.2: Przygotowanie zakresu temperatur do odczytu przez sieć neuronową

3.3 Eksploracja danych

Aby móc klasyfikować dane należy zastanowić się nad cechami, którymi się różnią. Na rysunku 3.3 przedstawiono porównanie stygnięcia dwóch rodzajów próbek: E5R oraz E6R. W klasyfikacji użytkczne będą dane, które są unikalne dla danej klasy. Posiadane dane stanowią serie obrazów postępującego procesu studzenia materiału. Aby dobrze wykorzystać zebrane zdjęcia należy szukać cech charakterystycznych w całym procesie stygnięcia.

²Repozytorium projektu w serwisie GitHub: <https://github.com/madmaze/pytesseract>



Rysunek 3.3: Porównanie procesu stygnięcia próbek klasy E5R oraz E6R

```
def get_temperature_bounds(img, bounds=((6, 24), (283, 318)),
                           ((219, 236), (283, 318))):  
    '''Extract temperature values from FLIR UI on image.'''
    img = invert(img)
    temp_txt = []
    for bound in bounds:  
        bound_img = img[slice(*bound[0]), slice(*bound[1])]  
        bound_img = rescale(bound_img, 4, anti_aliasing=True)  
        thr = threshold_otsu(bound_img)  
        img_txt = bound_img > thr  
        img_txt = Image.fromarray(img_txt)
        temp = pytesseract.image_to_string(img_txt, config='digits')
        if temp != '':
            temp = float(temp) / 10
        else:
            temp = 0
        temp_txt.append(temp)
    return temp_txt
```

Listing 1: Funkcja języka Python odczytująca zakres temperatur ze zdjęć

3.3.1 Wybór cech obrazu użytecznych w klasyfikacji

Po przyjrzeniu się rysunkowi 3.3 widoczne jest, że w próbce E6R temperatura ziaren zaczęła wyrównywać się szybciej. Zdjęcia tej próbki wcześniej stają się szare i tracą kontrast. Na podstawie analizy nagrani rozpatrzone zostały możliwości ekstrakcji ich cech charakterystycznych:

- a) klasyfikacja całych zdjęć z użyciem zaawansowanych sieci konwolucyjnych,
- b) analiza częstotliwościowa obrazów,
- c) użycie macierzy GLCM,
- d) detekcja krawędzi ziaren i wyznaczanie reprezentacji liczbowej ich kształtów,
- e) śledzenie zlewania się i zanikania małych detali na obrazie.

Wszystkie przedstawione opcje mają uzasadnienie i mogą się dobrze sprawdzić jako podstawa klasyfikacji. Metody b), c) oraz d) były już w różnym stopniu badane w ramach działalności Politechniki Śląskiej [2]. Należy rozpatrzyć, którą technikę wykorzystać do budowy prototypu klasyfikatora.

Metoda bazująca na sieciach konwolucyjnych może wykorzystywać najnowsze rozwiązania w dziedzinie uczenia maszynowego. Utrudnieniem w użyciu tej metody jest bardzo mały rozmiar posiadanej zbioru danych. Na zgromadzonych obrazach znajdują się zarówno użyteczne informacje, jak i szумy pomiarowe, a złożoność jednego zdjęcia jest na tą chwilę nieproporcjonalna do wielkości zestawu danych. Pomysł ten można wypróbować po rozszerzeniu zbioru nagrani.

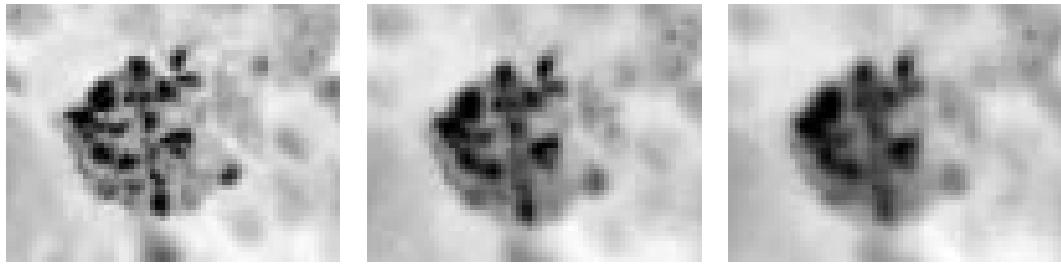
Technika polegająca na analizie częstotliwościowej obrazów wymaga zastosowania transformacji Fouriera. Takie podejście charakteryzuje się złożonością obliczeniową. Dodatkowo efekt analizy widmowej może być nieoczywisty w interpretacji. Wynikiem zastosowania transformacji Fouriera na obrazach są dwuwymiarowe macierze. Użycie takich danych wejściowych może wymagać konstrukcji złożonych sieci neuronowych. Analizę widmową ziaren przeprowadzono w pracach naukowców Politechniki Śląskiej, w rozpatrywanym przypadku zdecydowano się na poszukiwanie innego rozwiązania.

Macierz GLCM (ang. *Gray-Level Co-Occurrence Matrix*), na której może bazować opcja c), to tablica zawierająca informacje o relacjach wszystkich par pikseli w obrazie. Pozwala ona na analizę takich wartości jak: kontrast, korelacja, energia oraz homogeniczność. Jest to opcja dająca możliwość analizy dużej ilości informacji, z pewnością warta rozpatrzenia, jednak dosyć skomplikowana. Zdecydowano się na wybór mniej złożonej metody.

W obrazie można także wykrywać kształty ziaren za pomocą filtrów detekcji krawędzi. Opcję tę testowano przy pomocy filtra *Canny*. Uzyskane krawędzie nie pozwalają jednak na łatwą detekcję kształtów. Tradycyjna segmentacja ziaren jest trudna ze względu na małą rodzielcość oraz duże upakowanie ziaren. Operacje morfologiczne domykania kształtów powodowały bardzo duże zmiany w obrazie i zlewały ziarna. Rozwój takiego podejścia przy analizowanych obrazach wymaga zaawansowanej i ostrożnej obróbki zdjęć.

Ostatnia opcja e) wynika z obserwacji detali na obrazach. Na przedstawionych zdjęciach próbek można zauważać drobne ciemne punkty, które są obszarami o wolniejszej wymianie ciepła z otoczeniem niż reszta powierzchni ziaren. Jest to spowodowane ich niską emisjnością cieplną. Na rysunku 3.4 przedstawiono zbliżenie na grupę takich detali, w czterech częściach procesu stygnięcia.

Analizując próbki można zauważyc, że wraz ze stygnięciem, ciemne punkty w grupach zlewają się, a następnie zanikają. Dodatkowo ich liczba dla poszczególnych klas materiałów jest różna. Zdecydowano się na wybór metody polegającej na śledzeniu liczby tych punktów i ich zaniku. Taka analiza wiąże się z przetwarzaniem obrazów i utworzeniem algorytmów śledzenia detali. Opcja ta jest obiecująca, ponieważ nawet podczas wstępnej obserwacji próbek można dopatrywać się zależności między klasami, a obecnością omawianych detali. Można również zaobserwować, że w czasie stygnięcia na kolejnych zdjęciach pojawiają się sporadycznie także nowe



(a) Grupa detali w próbce 119_E5R_0 (b) Grupa detali w próbce 119_E5R_1 (c) Grupa detali w próbce 119_E5R_2

Rysunek 3.4: Zbliżenie na charakterystyczne grupy detali materiału

detale. Przy zdecydowanej tendencji do zanika i rozmywania plam zjawisko ich powstawania jest jednak pomijalne. Fakt pojawiania się nowych detali wzięto pod uwagę przy późniejszym procesie projektowania algorytmów ich śledzenia.

3.3.2 Wybór algorytmu detekcji ziaren

Zgodnie z rozważaniami przedstawionymi w podsekcji 3.3.1 w celu klasyfikacji ziaren zdecydowano się na obserwację ilości punktów o małej emisyjności. Należy więc wybrać metodę detekcji charakterystycznych elementów. W wykrywaniu omawianych detali użyteczne są algorytmy wykrywania plam na podstawie analizy pochodnych wartości w obrazie. Biblioteka Scikit-image udostępnia trzy algorytmy tego typu wykorzystujące:

- a) laplasjan funkcji Gaussa,
- b) różnicę funkcji Gaussa,
- c) wyznacznik Hesjanu.

Przedstawione algorytmy pozwalają na wykrycie na obrazie plam o kształcie zbliżonym do kolistego oraz oszacowanie ich promieni. Badane detale, przedstawione na rysunku 3.4, mają kształt zbliżony do kolistego, więc próba użycia rozpatrywanych algorytmów jest uzasadniona. Należy porównać dostępne warianty detekcji i wybrać algorytm, który działa najskuteczniej na zgromadzonych próbkach.

Metoda bazująca na laplasjanie funkcji Gaussa jest najdokładniejsza, ale także najwolniejsza. Funkcja Gaussa, jest dana wzorem 3.1.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \quad (3.1)$$

Symbole we wzorze mają następujące znaczenia:

σ to odchylenie standardowe,

μ to wartość średnia.

Laplasjan to operator różniczkowy drugiego rzędu. Omawiany algorytm oblicza wartości funkcji Gaussa dla coraz większego odchylenia standardowego i układa je w sześcianie. Poszukiwane plamy to lokalne maksima w tym sześcianie. Wadą tego rozwiązania jest bardzo wolne wykrywanie dużych plam z powodu złożoności obliczeniowej.

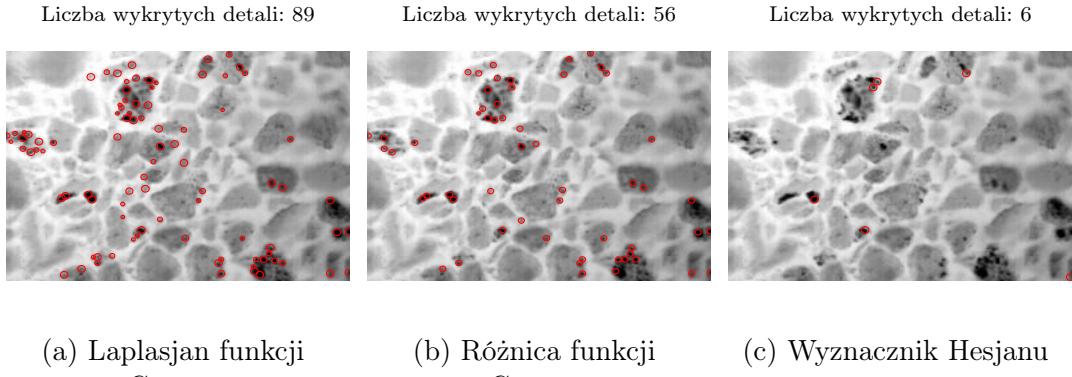
Różnica funkcji Gaussa jest metodą podobną do poprzedniej. Ponownie rozmywa ona obrazy z narastającymi odchyleniami standardowymi z użyciem funkcji Gaussa. Następnie różnice rozmytych obrazów są układane w sześcianie, gdzie maksima to plamy. Metoda ta jest szybsza i mniej dokładna od algorytmu bazującego na laplasjanie funkcji Gaussa, ale podobnie jak ona jest wolna w wykrywaniu dużych elementów.

Ostatnia metoda jest najszybsza, ale najmniej dokładna. Polega na wyszukiwaniu maksimów w macierzy Hesjanu. Jest to macierz drugich pochodnych cząstkowych. Postać takiej macierzy w n-wymiarowej przestrzeni zmiennych x przedstawia wzór 3.2. Prędkość tej metody nie zależy od wielkości wykrywanych plam, ale małe elementy mogą nie zostać przez nią wykryte [12].

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (3.2)$$

Porównanie działania wymienionych metod przedstawiono na grafice 3.5. Zgodnie z teoretycznym opisem funkcji najlepsza okazała się metoda Laplasjanu funkcji Gaussa. Funkcję korzystającą z wyznacznika Hesjanu należy odrzucić, ponieważ w rozważanym przypadku istotne jest wykrywanie małych plam. Z tego samego powodu korzystne jest użycie najdokładniejszej funkcji. Ponieważ program nie powinien wykrywać dużych elementów, nie ma ryzyka zbyt powolnych obliczeń na obszernych plamach.

Aby wybrana funkcja laplasjanu funkcji Gaussa wykrywała tylko małe plamy należy podać jej odpowiednie parametry. Uczyniono to już na etapie porównania metod detekcji. Wywołanie omawianej funkcji ma postać przedstawioną na listingu



Rysunek 3.5: Porównanie bibliotecznych algorytmów wykrywania plam na obrazie

2. Funkcji podano dwa dodatkowe argumenty, które są istotne dla pożądanego działania. Argument `max_sigma` ogranicza odchylenie standardowe obliczanych funkcji Gaussa, przez co wykrywane są tylko małe elementy. Drugi argument `threshold` decyduje o poziomie, powyżej którego punkt jest uznany za maksimum w sześciu laplasjanów. Domyślna wartość tego argumentu `threshold` okazała się zbyt duża, zmniejszono ją aby wykrywać bardziej subtelne detale. Na podstawie opisanego wywołania metody laplasjanu funkcji Gaussa opracowano funkcję zwracającą położenie i promień wykrytych plam.

```
def find_blobs(img):
    """
    Find blobs in given image and get list of their positions
    and radii.
    ...
    # Detect blobs with Difference of Gaussian
    blobs = blob_dog(img, max_sigma=2, threshold=0.1)
    # Get blobs radii from each kernel sigma
    blobs[:, 2] = blobs[:, 2] * sqrt(2)
    return blobs
```

Listing 2: Funkcja języka Python wykrywająca detale na obrazie

3.3.3 Algorytm śledzenia ziaren w serii zdjęć

Użycie funkcji bibliotecznych opisanych w podsekcji 3.3.2 pozwala na detekcję detali na pojedynczym zdjęciu. Aby wykorzystać detekcję plam w serii pięciu

```
def find_blob_series(imgs, only_remaining=True):
    """
    Return list of list of blobs found in each of given images.
    """
    stages = []
    remaining = None
    for img in imgs:
        new_blobs = find_blobs(img)
        if remaining is not None and only_remaining:
            remaining = find_remaining_blobs(new_blobs, remaining)
        else:
            remaining = new_blobs
        stages.append(remaining)
    return stages
```

Listing 3: Funkcja języka Python śledząca detale w serii zdjęć

zdjęć, które reprezentują stygnięcie jednej próbki, należy opracować zestaw funkcji śledzący zmiany na obrazie. Na tym etapie prac nie jest jednoznaczne jaki sposób obserwacji i zliczania detali będzie najlepszy do późniejszego prototypowania sieci neuronowej klasyfikującej ziarna rud miedzi. Dlatego zdecydowano się na napisanie funkcji zliczania plam, tak by umożliwić trzy warianty pracy algorytmu:

- a) zliczanie wszystkich detali na każdym etapie stygnięcia,
- b) zliczanie detali śledzonych od początku stygnięcia,
- c) zliczanie stosunku śledzonych detali do ich ilości na początku stygnięcia.

Przez śledzenie detali rozumiane jest zliczanie jedynie tych plam, które były wykryte na początku stygnięcia. Oznacza to, że metody b) oraz c) ignorują detale pojawiające się w późniejszych etapach studzenia ziaren.

W celu realizacji planu śledzenia detali utworzono funkcję języka Python. Jej kod przedstawiono na listingu 3. Aby umożliwić wariantowość algorytmu funkcja posiada opcjonalny parametr, który pozwala aktywować śledzenie wyłącznie tych ziaren, które były obecne od początku stygnięcia. Omawiany kod wykorzystuje funkcję wykrywania detali na zdjęciu przedstawioną na listingu 2. Wybór opcji detekcji ziaren obecnych jedynie od początku stygnięcia, powoduje wywołanie funkcji porównania wykrytych detali, z punktami na poprzednim zdjęciu. Po analizie każdego zdjęcia nowy zestaw ziaren staje się zbiorem poprzednim w kolejnej iteracji. Dzieje się tak, by uwzględnić fakt, że podczas zlewania się ziaren środki ciężkości

```
def find_remaining_blobs(new_blobs, old_blobs):
    """
    Return list of blobs present in both lists, where blob
    is considered same if is in proximity of 2 times it's radius.
    """
    remaining = []
    for new_blob in new_blobs:
        yn, xn, _ = new_blob
        for old_blob in old_blobs:
            yo, xo, ro = old_blob
            if inside_circle(xn, yn, xo, yo, 2 * ro):
                remaining.append(new_blob)
```

Listing 4: Funkcja języka Python wykrywającej te same detale w kolejnych obrazach

```
def inside_circle(x, y, a, b, r):
    """
    Return True if point (x, y) lies inside circle
    with center of (a, b) and radius r.
    """
    return (x - a) * (x - a) + (y - b) * (y - b) < r * r
```

Listing 5: Funkcja języka Python sprawdzająca czy dany punkt leży wewnątrz podanego okręgu

i promienie detali ulegają zmianie. Dlatego kolejne grupy plam nie są zawsze porównywane z grupą z początku serii, a z zestawem z poprzedniego analizowanego zdjęcia.

Na listingu 4 przedstawiono funkcję testującą, które z detali znajdują się na kolejnych etapach stygnięcia. Przyjmuje ona listy plam wykrytych w dwóch kolejnych etapach iteracji. Potem następuje porównanie każdego ziarna z nowej i starej próbki. Jeżeli nowy punkt znajduje się w odległości do dwóch promieni od środka starej plamy, to zostaje on uznany za powtarzający się. Powtarzające się punkty są dołączane do zwracanej listy. Aby zrealizować ten zamysł zaimplementowano funkcję przedstawioną na listingu 5. Kod tej funkcji wynika wprost z równania matematycznego okręgu. Ponieważ wiele punktów może znajdować się blisko siebie istnieje ryzyko, że zostaną one zliczone wiele razy. Z tego powodu przed zwróceniem listy należy usunąć z niej duplikaty. W tym celu utworzono funkcje pomocniczą widoczną na listingu 6. Najprostszym sposobem eliminacji powtarzających

```
def unique(multilist):
    '''Get list without repeating values.'''
    return list(set(tuple(i) for i in multilist))
```

Listing 6: Funkcja języka Python usuwająca duplikaty z listy

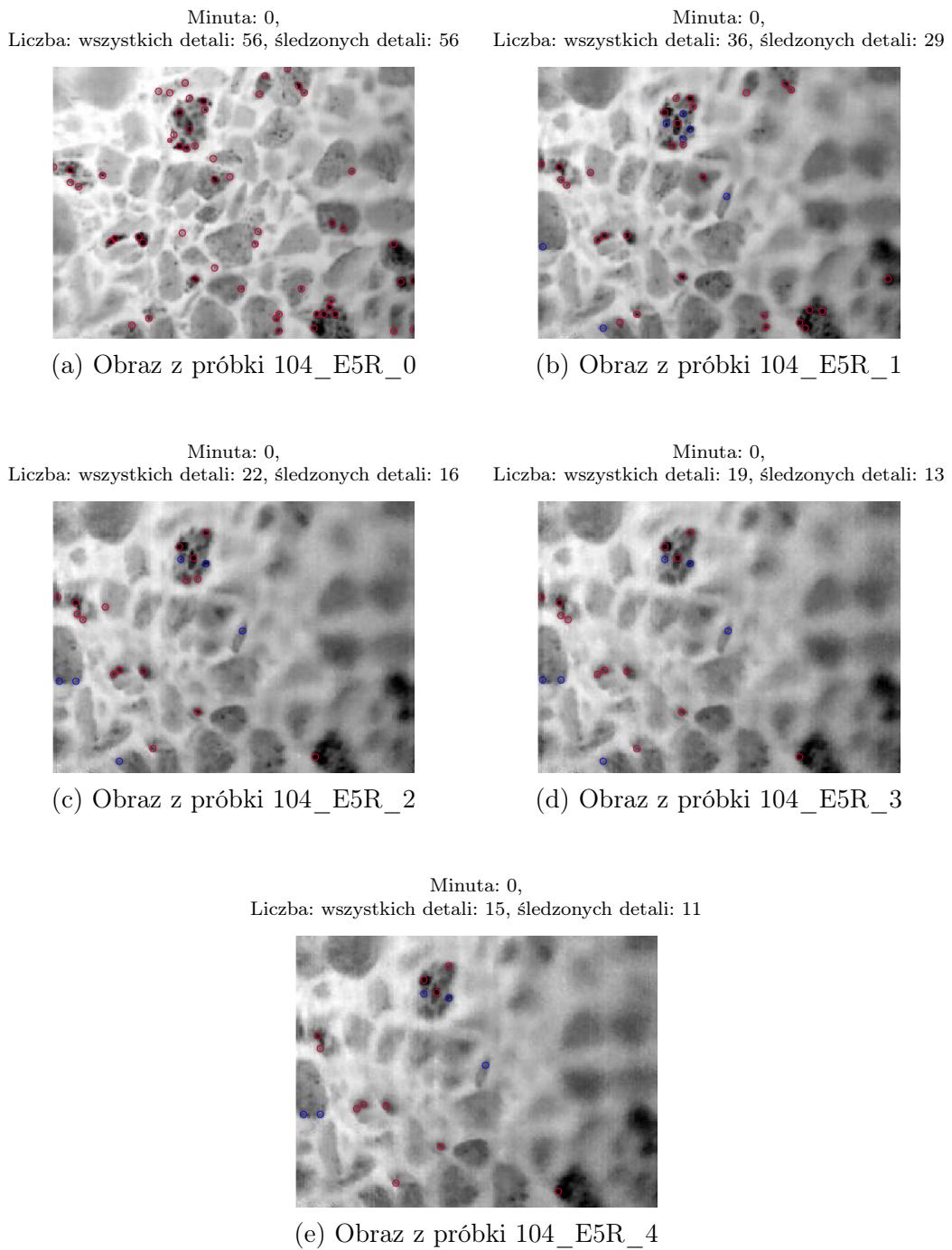
się wartości w języku Python jest konwersja listy na zbiór, który jest agregatem unikalnych nieuszeregowanych wartości. Aby zwrócić z funkcji listę należy z powrotem skonwertować zbiór na typ listowy. W naszym przypadku argument funkcji jest listą, której elementy są listami zawierającymi współrzędne oraz promień każdej plamy. Lista list nie podlega konwersji do zbioru, ponieważ typ jej elementów nie posiada funkcji hashowania. Dlatego każdy punkt na liście zamieniono uprzednio w typ krotki. W ten sposób wszystkie konieczne przekształcenia są możliwe. Aby zrealizować pomysł liczenia stosunku plam pozostałych w kolejnych etapach stygnięcia zaimplementowano funkcję przedstawioną na listingu 7.

```
def ratio_of_remaining_blobs_in_stages(stages):
    """
    In each stage calculate ratio of remaining blobs
    to their initial number.
    """
    num_of_blobs = [len(stage) for stage in stages]
    return [remaining / num_of_blobs[0] for remaining in num_of_blobs]
```

Listing 7: Funkcja języka Python obliczająca jaka część ziaren z początku stygnięcia pozostała w jego kolejnych etapach

Przyglądając się zaimplementowanym funkcjom można docenić wybór języka Python do realizacji projektu. Elastyczność języka Python, w szczególności jego list oraz wygoda dynamicznego typowania pozwoliły na szybkie i eleganckie zaimplementowanie potrzebnych funkcji. Wykorzystano bardzo zwięzły i ekspresyjny element języka nazywany *wyrażeniami listowymi*. Pozwala on na efektywne stosowanie jedno-linijkowych pętli do przeglądania i modyfikowania obiektów iterowalnych. Użyteczność tego mechanizmu można zaobserwować na przedstawionych listingach.

Przedstawiony zestaw funkcji daje możliwość ekstrakcji liczby plam, w serii zdjęć, na trzy sposoby. Efekty działania utworzonego kodu przetestowano na obrazach pomiarowych. Działanie sposobów zliczania ziaren a) oraz b) przedstawiono na rysunku 3.6. Zaznaczono na nim wykryte detale oraz wyróżniono wśród nich, te które są śledzone od początku stygnięcia.

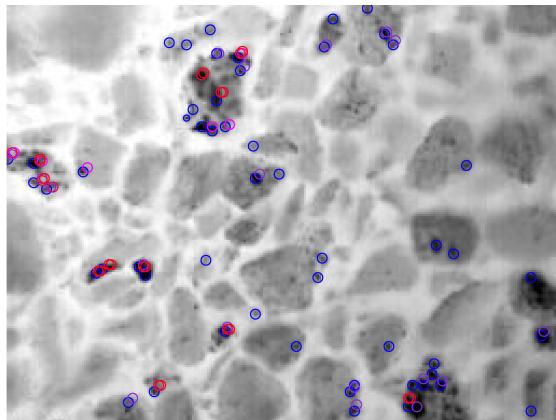


Kolor czerwony detale śledzone od początku stygnięcia

Kolor niebieski nowe wykryte detale

Rysunek 3.6: Zliczanie śledzonych detali w próbce dwoma sposobami

Detekcja metodą laplasjanu funkcji Gaussa



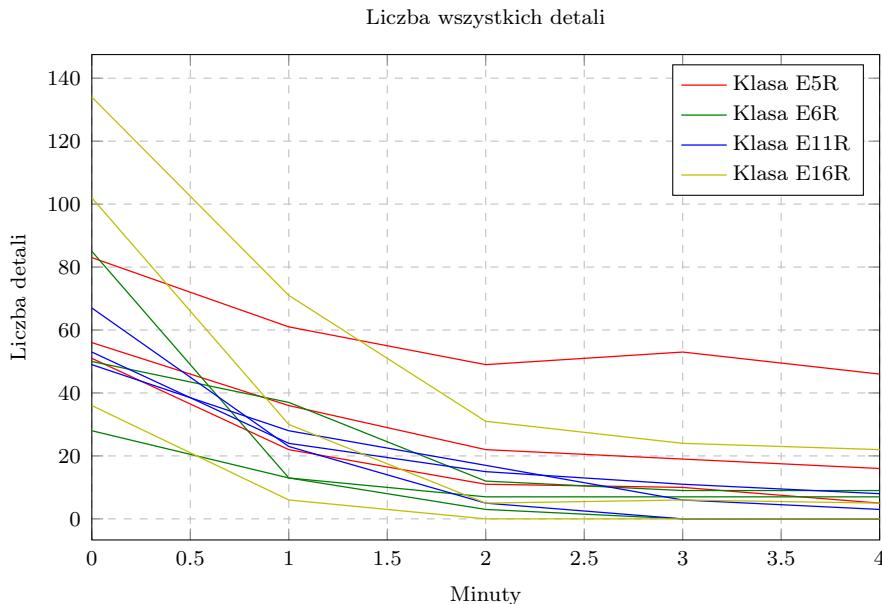
Kolorem niebieskim zaznaczono detale wykryte na początku stygnięcia. Wraz z upływem czasu kolor oznaczenia detali przechodzi w czerwień

Rysunek 3.7: Wykryte detale, śledzone od początku stygnięcia, zaznaczone na pierwszym obrazie w serii pomiarowej

Próbka	Minuta 0	Minuta 1	Minuta 2	Minuta 3	Minuta 4
104_E5R	1.0	0.52	0.29	0.23	0.20
107_E6R	1.0	0.20	0.12	0.02	0.02
106_E11R	1.0	0.12	0.02	0.00	0.00
111_E16R	1.0	0.05	0.01	0.01	0.01

Tablica 3.1: Stosunek liczby detali śledzonych w kolejnych etapach stygnięcia do ich liczby na początku pomiaru

Metodę śledzenia plam i ich zanikania przedstawiono na rysunku 3.7. Pokazuje on położenie plam wykrytych w kolejnych etapach stygnięcia. Wykryte punkty zaznaczono na obrazie z początku tego procesu. Zmieniające się kolory obrazują detale wykryte w kolejnych chwilach. Pozwala to zaobserwować przemieszczanie się środków ciężkości i promieni plam podczas stygnięcia i zlewania się. Działanie metody c), zliczającej stosunek śledzonych detali do ich pierwotnej liczby zobrazowano w tabeli 3.1. Działanie zliczania przedstawiono dla każdej rozpatrywanej klasy ziaren.

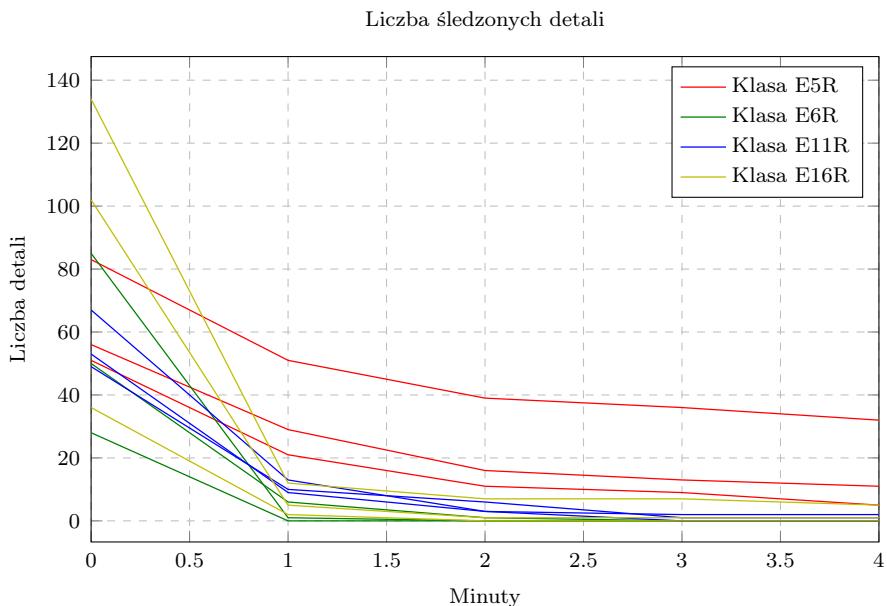


Rysunek 3.8: Liczba wykrytych detali na każdym etapie stygnięcia, w wariantie zliczającym wszystkie plamy

3.3.4 Wizualizacja zebranych cech i ocena ich użyteczności

W podsekcji 3.3.3 przedstawiono opracowane algorytmy śledzenia ilości detali w rozpatrywanych obrazach stygnięcia ziaren rud miedzi. Opracowane metody mają sens przy klasyfikacji, jeśli pozyskane cechy są charakterystyczne dla rozpatrywanych klas. Aby oszacować użyteczność pozyskanych cech należy przeprowadzić ich wizualizację. W tym celu stworzono wykresy ilości wykrytych plam dla poszczególnych metod ich zliczania.

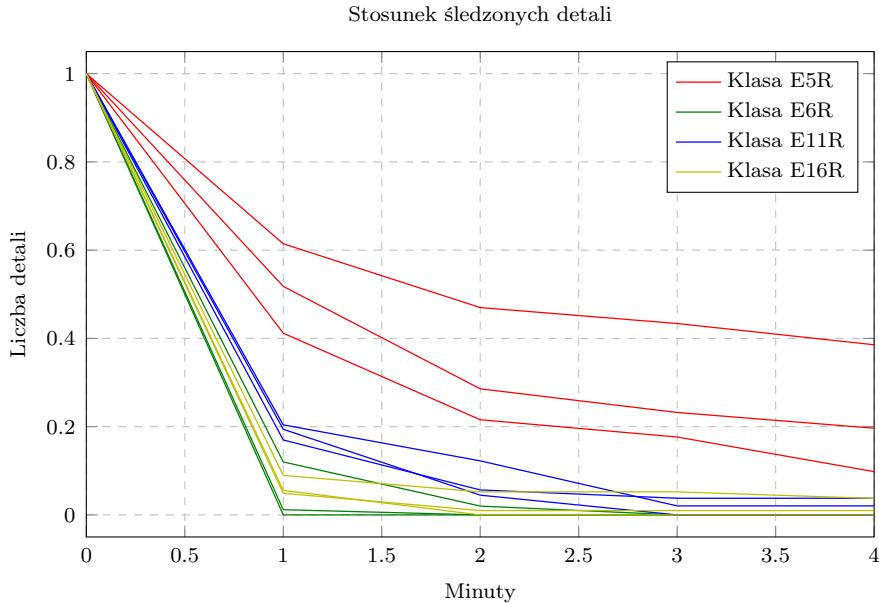
Rysunek 3.8 przedstawia wykres procesu zanikania plam dla wariantu z wykrywaniem wszystkich detali na każdym etapie stygnięcia. Wykreślone krzywe dla różnych klas materiałów przecinają się, na wykresach nie widać uporządkowania, ani wyraźnych zależności. Liczba charakterystycznych detali może być cechą ziaren rud miedzi, ze względu na ich różną budowę, jednak w analizowanym przypadku nie jest to cecha dająca nadzieję na dobre wyniki klasyfikacji. Powodem takiej sytuacji jest duża losowość tego typu danych. Niekwestionowanie od własności materiału, jego ułożenie podczas pomiaru jest przypadkowe, co ma wpływ na bezwzględną liczbę zliczonych detali. Na liczbę widocznych plam wpływa również sposób ustalenia ostrości kamery. Po przyjrzeniu się wykresom można jednak domniemywać, że pewną cechą charakterystyczną jest nachylenie krzywych. Wskazuje to, że bardziej widoczne mogą być względne cechy czasowe, a nie bezwzględna liczba zliczonych detali.



Rysunek 3.9: Liczba detali wykrytych na zdjęciach, w wariantie zliczającym plamy śledzone od początku procesu stygnięcia

Efekty drugiego sposobu zliczania punktów przedstawiono na rysunku 3.9. W tym przypadku zliczano jedynie detale obecne na obrazie od początku stygnięcia. Ten wariant śledzenia ilości plam daje dużo lepsze perspektywy na klasyfikację ziaren. Dla różnych typów próbek widoczne są charakterystyczne przebiegi krzywych. Obserwacja dotycząca nachylenia krzywych staje się bardziej uzasadniona, pochodne poszczególnych klas wydają się zbliżone. Wyeliminowanie pojawiących się detali pomogło w obserwacji procesu stygnięcia i zmniejszyło czynniki losowe. Na podstawie analizy nagrani można domniemywać, że nowe ziarna pojawiały się w wyniku zaobserwowanych delikatnych ruchów materiału. Te mogły wynikać z drgań stanowiska pomiarowego, istotnym czynnikiem może być także mała głębia ostrości obiektywu. W późniejszych etapach stygnięcia na obrazach pojawia się także coraz więcej szumów, które mogą zostać wykryte przez algorytm. Śledzenie detali, które znajdują się na obrazie od początku eliminuje ten problem.

Wyniki ostatniej metody, polegającej na ustaleniu stosunku śledzonych detali do ich pierwotnej liczby, został przedstawiona na rysunku 3.10. Widoczne jest odseparowanie różnych klas rud miedzi, różne typy próbek charakteryzują się odmiennymi postępami rozmycia detali w czasie. Śledzenie ilości detali w stosunku względnym dało najlepsze rezultaty. Nastąpiła także pewna normalizacja danych, która była trudna do realizacji przy bezwzględnym zliczaniu wszystkich plam na obrazie. Należy także zauważyć, że przedstawione krzywe przedstawione na rysun-



Rysunek 3.10: Stosunek śledzonych detali do ich liczby na początku procesu stygnięcia

kach 3.9 oraz 3.10 mają kształt zbliżony do krzywych eksponentjalnych. Jest to obserwacja wskazująca, że pozyskane cechy dobrze oddają naturę procesu stygnięcia. Proces opadania temperatury ciał opisuje *prawo stygnięcia Newtona*, które ma postać równania różniczkowego, przedstawionego we wzorze 3.3 [10].

$$\frac{dT(t)}{dt} = -k(T(t) - T_R) \quad (3.3)$$

Oznaczenia we wzorze mają następujące znaczenia:

$T(t)$ to funkcja temperatury ciała w czasie,

T_R to temperatura otoczenia,

k to współczynnik liczbowy, charakterystyczny dla danego ciała.

Rozwiązań równania przedstawia wzór 3.4, jak widać ma ono charakter eksponentjalny, do którego zbliżone są wykresy na rysunkach 3.9 oraz 3.10.

$$T(t) - T_R = \Delta T(t) = \Delta T(0) e^{-kt} \quad (3.4)$$

Analiza pozyskanych cech wskazuje, że istnieje możliwość ich wykorzystania w klasyfikacji rud miedzi. Dalsza ocena wyników pracy będzie zależała od jakości działania sieci neuronowej stworzonej do rozpoznawania przygotowanych danych.

Rozdział 4

Prototypowanie sieci neuronowej klasyfikującej ziarna miedzi

4.1 Konstrukcja prototypu sieci neuronowej

Po zakończeniu procesu eksploracji danych można przystąpić do konstrukcji sieci neuronowej odpowiedzialnej za rozpoznawanie ziaren rud miedzi. Przedstawiony problem wymaga klasyfikacji wieloklasowej wielowymiarowych danych. Problemy tego typu można rozwiązywać zarówno za pomocą klasycznych algorytmów uczenia maszynowego jak i uczenia głębokiego. Zdecydowano się na wybór sieci neuronowych do klasyfikacji ziaren. Jest to rozwiązanie najbardziej nowoczesne i elastyczne. Zgodnie z opisem narzędzi programistycznych w podsekcji 2.4.3, proces budowania sieci oparto na interfejsie Keras, z zapleczem TensorFlow.

4.1.1 Dobór struktury sieci

Kluczowym czynnikiem decydującym o architekturze sieci jest posiadany zbiór danych. Wejściowy zestaw cech klas jest pięciowymiarowy, do klasyfikacji takich danych wystarczająca powinna być standardowa sieć neuronowa. Nie ma potrzeby używania bardziej złożonych sieci rekurencyjnych, czy konwolucyjnych.

Projektowanie sieci neuronowych wymaga podjęcia szeregu wyborów, wśród nich można wyróżnić następujące decyzje dotyczące:

- ilości warstw sieci,
- typu warstw w sieci,
- ilości neuronów w poszczególnych warstwach sieci,
- rodzajów funkcji aktywacji w neuronach poszczególnych warstw sieci.

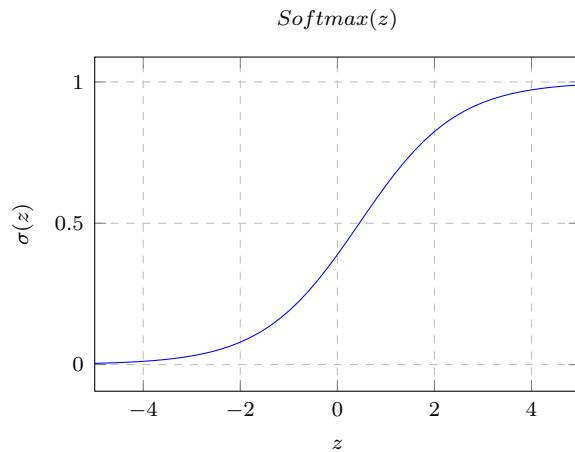
Definiując model należy rozpatrzyć przedstawione cechy budowanej sieci. Nie istnieje jednoznaczny sposób bezpośredniego określenia najlepszego klasyfikatora dla danego problemu. Przy projektowaniu sieci należy kierować się przesłankami teoretycznymi, doświadczeniem oraz testując i porównując różne rozwiązania. Przedstawiona konstrukcja oraz uzasadnienie jej wyboru bazuje na wynikach walidacji, której mechanizm przedstawiono w sekcji 4.2. Szczegółowe dane, dotyczące porównań różnych konstrukcji i parametrów sieci przedstawiono w dodatku A.

Konstrukcję klasyfikatora rozpoczęto od wyboru ilości warstw. W standardowej konstrukcji każda sieć posiada warstwę wejściową oraz wyjściową, których parametry są związane odpowiednio z wektorem cech oraz reprezentacją klas na wyjściu sieci. Aby zwiększać efektywność sieci pomiędzy warstwą wejściową oraz wyjściową umieszcza się zestaw warstw nazywanych *ukrytymi*. Zazwyczaj posiadają one najwięcej neuronów spośród warstw w sieci. Matematycznie dowiedzono, że jedna warstwa ukryta¹ wystarcza do aproksymacji dowolnej funkcji nieliniowej [6]. Nie oznacza to jednak, że w praktyce tak mała sieć wystarczy, aby rozwiązać dowolny problem uczenia maszynowego. Dodanie większej ilości warstw ukrytych jest jednym z najlepszych sposobów na poprawę działania sieci w realnych warunkach [8, str. 273]. Biorąc pod uwagę złożoność rozpatrywanego problemu zdecydowano się na strukturę czterowarstwową (warstwa wejściowa, wyjściowa i dwie warstwy ukryte). Jest to rozmiar często spotykany w sieciach tego typu, późniejsze testy pokazały, że wybrana liczba warstw daje dobre rezultaty. Sprawdzono, że zastosowanie jednej warstwy ukrytej powodowało pogorszenie działania sieci. Dodanie do sieci większej ilości warstw nie powodowało znacznej poprawy działania, a komplikowało jej budowę i wydłużało czas treningu.

Kolejnym krokiem projektowania sieci jest decyzja co do ilości neuronów w poszczególnych warstwach. W przypadku pierwszej i ostatniej warstwy jest to wartość prosta do określenia. Zaleca się, aby wejście sieci miało liczbę neuronów równą liczbę elementów wektora cech, który w rozpatrywanym przypadku ma długość równą pięć [8, str. 272]. Ostatnia warstwa powinna mieć tyle neuronów ile wynosi liczba rozpoznawanych klas, tak by każde wyjście sieci oznaczało prawdopodobieństwo przynależności próbki do odpowiedniej klasy. W analizowanym zbiorze ziaren znajdują się cztery typy rud miedzi i tyle neuronów powinno znajdować się w warstwie wyjściowej sieci. W warstwach ukrytych umieszczono większą liczbę neuronów, eksperymentalnie stwierdzono, że sieć osiąga dobre wyniki dla 256 neuronów w pierwszej warstwie ukrytej i 128 w drugiej. Zmniejszenie liczby neuronów w kolejnej warstwie ukrytej jest często spotykanym zabiegiem.

Następnie rozpatrzone zostały dostępne funkcje aktywacji poszczególnych warstw. Dla warstwy wyjściowej należy wybrać funkcję o wartościach od zera do jeden. Jest to uzasadnione faktem, że wyjścia powinny reprezentować prawdopodobieństwo

¹Przy pewnych ograniczeniach co do doboru funkcji aktywacji.



Rysunek 4.1: Funkcja aktywacji softmax

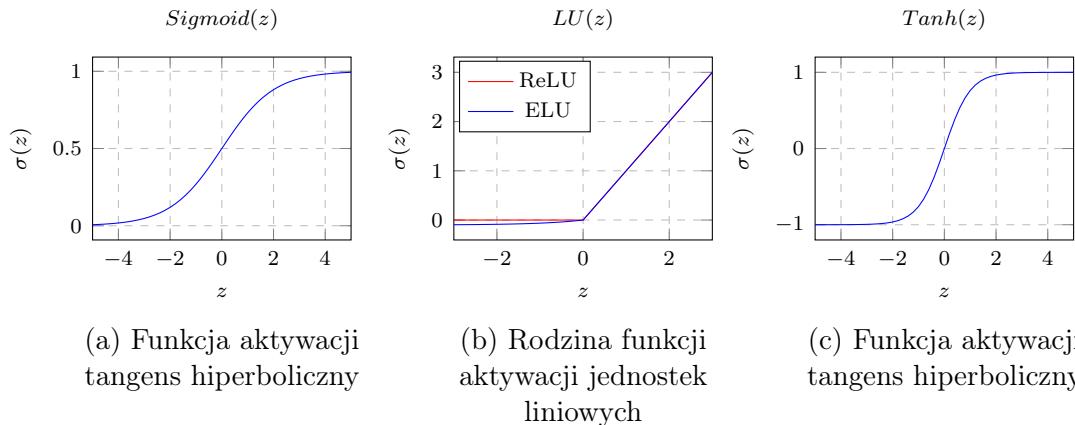
zaliczenia próbki do danej klasy. Przy klasyfikacji zazwyczaj stosuje się funkcję softmax, jej wykres przedstawiono na rysunku 4.1. W przypadku pozostałych warstw wybór funkcji aktywacji jest mniej oczywisty. Rozpatrzone trzy rodziny funkcji:

- funkcję sigmoid,
- funkcje z grupy jednostek liniowych (ang. *linear unit*):
 - ReLU (ang. *Rectified Linear Unit*),
 - ELU (ang. *Exponential Linear Unit*),
- funkcję tangens hiperboliczny².

Rozpatrywane funkcje aktywacji przedstawiono na rysunku 4.2. Funkcja sigmoid jest często stosowaną, klasyczną funkcją aktywacji. Jej wadą jest jednak zanikanie wartości pochodnej. Obecnie częściej stosowane są funkcje z rodziny jednostek liniowych, szczególnie z wyciekiem, czyli małymi wartościami dla ujemnych argumentów. Przetestowano również funkcję tangens hiperboliczny, która ma kształt podobny do funkcji sigmoid. Mimo, że obecnie funkcje jednostek liniowych są najbardziej popularne w rozpatrywanym przypadku sieć osiągała najlepsze wyniki dla funkcji tangens.

Opisany model sieci neuronowej należy zdefiniować za pomocą interfejsu Keras. Do inicjalizacji sieci o standardowej liniowej strukturze służy funkcja `Sequential()`, która może przyjąć listę warstw w sieci. Funkcja `Dense()` tworzy warstwę łączącą każdy neuron z każdym wyjściem poprzedniej warstwy. Jej parametry definiują liczbę neuronów w warstwie oraz ich funkcję aktywacji. Funkcję języka Python

²Wzory matematyczne rozpatrywanych funkcji można znaleźć w książce [8, str. 273].



Rysunek 4.2: Funkcje aktywacji rozpatrywane do użycia w warstwach ukrytych

implementują opisywaną sieć, za pomocą przedstawionych elementów biblioteki Keras, przedstawiono na listingu 8.

```
def default_grain_classifier_model():
    """
    Get default uncompiled model for grain classification,
    based on 5 step cooling process using number of blobs.
    """

    model = keras.Sequential([
        keras.layers.Dense(5, activation='tanh'),
        keras.layers.Dense(256, activation='tanh'),
        keras.layers.Dense(128, activation='tanh'),
        keras.layers.Dense(4, activation='softmax')
    ])
    return model
```

Listing 8: Funkcja języka Python definiująca model sieci neuronowej

4.1.2 Trening sieci neuronowej

Kolejnym krokiem w budowie klasyfikatora ziaren jest trening sieci neuronowej. Jest to proces, w którym wagi neuronów są dopasowywane do zbioru uczącego. Rozpatrywany problem jest typu uczenia nadzorowanego. Przygotowane dane mają znane etykiety klas, to znaczy są odgórnie przypisane do danego typu. Narzucone etykiety nazywane są *wiedzą ekspercką*, ich prawdziwość jest pewna. Etykiety

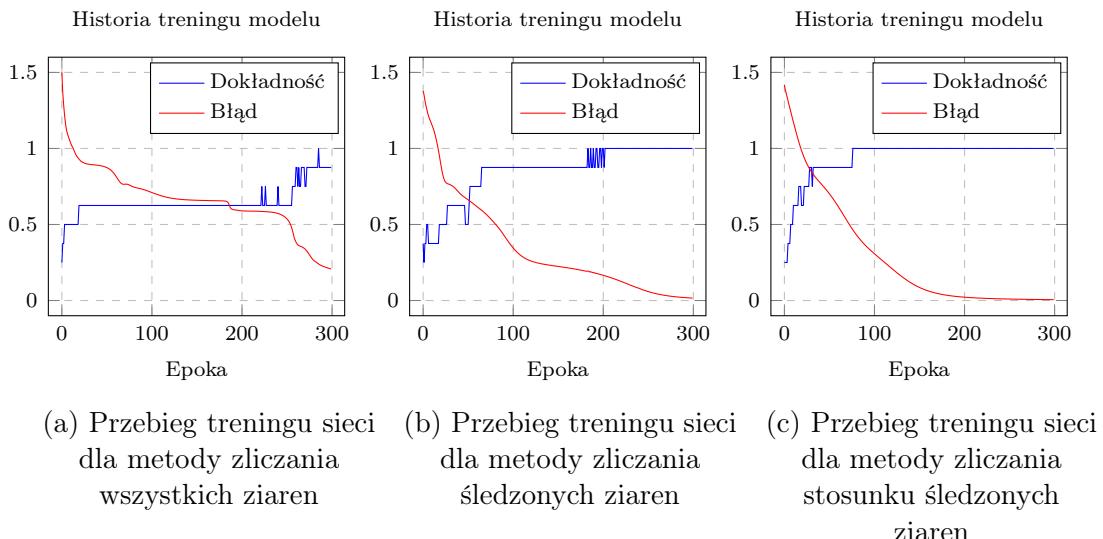
przypisane przez klasyfikator będą oceniane przez porównanie z etykietami eksperckimi. Trening polega na zbliżaniu wyników działania sieci do etykiet zbioru uczącego.

Przed rozpoczęciem uczenia sieci należy podzielić dostępne dane na zbiór treningowy oraz testowy. W skład zbioru testowego nie powinny wchodzić dane uczące. Ocena zdolności sieci do generalizacji problemu może bazować tylko na danych, które nie brały udziału w jej treningu. Aby ułatwić podział Biblioteka Scikit-learn udostępnia funkcję `train_test_split()`, która zwraca podane zbiory, podzielone na części treningowe i testowe. Odpowiednie wykorzystanie funkcji wymaga użycia jej parametrów opcjonalnych. Argument `stratify` sprawia, że podzielone zbiory zawierają takie same proporcje klas. Taka metoda podziału nazywana jest *losowaniem warstwowym*. Uaktywnienie tej opcji jest szczególnie ważne, ze względu na mały rozmiar posiadanej zbioru danych. Gdyby dane dzielić w pełni przypadkowo istniałoby ryzyko nadmiernej reprezentacji klasy w danej grupie oraz jej braku w innej. Kolejnym istotnym parametrem jest `test_size`, który decyduje o rozmiarze zbioru testowego. Ponieważ w zbiorze są trzy egzemplarze każdej klasy odpowiednie jest wydzielenie do testów jednej trzeciej próbki. Ostatni parametr `random_state`, to ziarno generatora losowego podziału zbiorów. Aby móc porównywać działanie sieci, pomiędzy wielokrotnymi uruchomieniami programu, należy wyeliminować z niego czynniki przypadkowe i podać funkcji stałe ziarno, co zapewni powtarzalny podział zbioru. Oczywiście wydzielenie w narzucony sposób zbioru testowego, szczególnie w przypadku tak małej ilości danych, nie daje obiektywnej oceny modelu. Jest on jednak wystarczający do budowy i testowania pierwszego prototypu sieci. W sekcji 4.2 przedstawiono konstrukcję i wyniki działania bardziej miarodajnego procesu walidacji.

Aby móc przystąpić do treningu należy określić parametry uczenia sieci. Konfiguracja tego procesu odbywa się przez wywołanie metody komplikacji. Metoda przyjmuje parametry algorytmu optymalizacji sieci, miary błędu oraz metryki. Argument `optimizer` przyjmuje nazwę stosowanego algorytmu uczenia sieci. Rozważono dwie metody treningu: `sgd` (ang. *stochastic gradient descent*) oraz `adam` (ang. *adaptive moment estimation*). Metoda `sgd` jest najbardziej popularnym i podstawowym sposobem uczenia, jednak algorytm `adam`, jest rozwiązaniem nowszym, i bardziej wydajnym. Jedną z zalet drugiego algorytmu jest adaptacyjne obliczanie współczynnika uczenia, przez co strojenie tego parametru nie jest konieczne. Wadą nowszego rozwiązania jest spadek jakości wyników dla szczególnych zbiorów danych [8, str. 299]. Porównanie algorytmów w rozpatrywanym przypadku wskazało, że algorytm `adam` daje lepsze rezultaty i to na jego użycie się zdecydowano. Parametr `loss` przyjmuje nazwę sposobu liczenia błędu w sieci. Przykładem miary błędu używanej w sieciach neuronowych jest popularny błąd średniokwadratowy. Dla problemów klasyfikacji lepszy jest jednak błąd obliczany metodą *rzadkiej ka-*

tegoryzacyjnej entropii krzyżowej (ang. *sparse categorical crossentropy*). Jest to złożona metoda wykorzystująca funkcję logarytmiczną. Metoda kategoryzacyjnej entropii okazała się najlepszą funkcją błędu dla analizowanego przypadku. Ostatni parametr metryki to wartość obliczana w celu oszacowania poprawności działania sieci. Wybrano standardową metrykę dokładności sieci, czyli stosunku poprawnie zakwalifikowanych próbek do wszystkich analizowanych.

Ostatnim etapem uczenia jest realizacja treningu sieci, dokonywana za pomocą metody `model.fit()`. Metoda przyjmuje argumenty zbioru treningowego wraz z zestawem etykiet oraz liczbę epok treningu, zwraca obiekt zawierający przebieg uczenia sieci. Na rysunku 4.3 przedstawiono dokładność i błąd w kolejnych epokach treningu. Na ich podstawie określono, że odpowiednia liczba epok wynosi 300. Opisany plan treningu sieci realizuje kod przedstawiony na listingu 9.



Rysunek 4.3: Przebieg treningu sieci dla różnych metod zliczania ziaren

Działanie sieci można sprawdzić na zbiorze testowym. Należy mieć na uwadze, że przy małej ilości danych i przedstawionym podziale na zbiory treningowy oraz testowy nie będzie to test miarodajny. Jest on jednak akceptowalny przy pierwszych testach sieci podczas jej konstrukcji. Wyniki działania sieci przedstawia tabela 4.1. Przedstawione wartości błędu oraz dokładności potwierdzają przypuszczenia na temat użyteczności różnych metod zliczania detali, które przedstawiono w podsekcji 3.3.4.

```

def classification_demo(X, y):
    """
    Demo grain classification on given data.
    Train and test default model.
    """
    X = np.array(X)
    y = np.array(y)

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, stratify=y, test_size=0.33, random_state=1)

    model = default_grain_classifier_model()
    model.compile(
        optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
    history = model.fit(X_train, y_train, epochs=300, verbose=0)

```

Listing 9: Kod treningu sieci neuronowej klasyfikującej ziarna miedzi

Metoda zliczania detali	Wskaźnik	
	Błąd	Dokładność
Wszystkie detale	1,45	0,25
Śledzone detale	7,50	0,5
Stosunek śledzonych detali	0,63	0,75

Tablica 4.1: Wskaźniki oceny działania sieci na zbiorze testowym

4.2 Walidacja i ocena działania sieci

Po stworzeniu prototypu klasyfikatora przedstawionego w rozdziale 4.1.1 należy przeprowadzić walidację zbudowanej sieci neuronowej. Najpopularniejszą metodą miarodajnej walidacji sieci jest *k-krotny sprawdzian krzyżowy* (ang. *k-fold cross-validation*). Metoda ta polega na podziale dostępnych danych na k części, i kolejnym wydzielaniu jednej z części danych jako zbioru testowego. Walidacje powtarza się k-krotnie, tak by każda część była wykorzystana jako dane testowe. Na końcu testu wyniki są uśrednianie. Dzięki temu sprawdzian krzyżowy daje dobry pogląd na ogólną zdolność sieci do generalizacji rozwiązywanego problemu.

Często po procesie walidacji dokonuje się testu sieci przy użyciu osobnego ze-

```
def network_cross_validation(model, X, y, n_splits):
    '''Compute cross validation fold scores for given keras model.'''
    eval_scores = []

    folds = StratifiedKFold(n_splits=n_splits).split(X, y)
    for train_index, test_index in folds:
        x_train, x_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        model.fit(x_train, y_train, epochs=300, verbose=0)
        eval_scores.append(model.evaluate(x_test, y_test, verbose=0))
    return eval_scores
```

Listing 10: Funkcja języka Python definiująca model sieci neuronowej

stawu danych. Niestety posiadany zbiór jest zbyt mały, aby wydzielić z niego osobny zestaw testowy. Z tego powodu zdecydowano się oceniać sieć jedynie na podstawie metody sprawdzianu krzyżowego.

Biblioteka Keras nie posiada wbudowanych funkcji zaawansowanych technik walidacji, dlatego należy przygotować je samodzielnie. W tym celu użyto biblioteki Scikit-learn, która oferuje funkcje podziału zbiorów danych. Jedną z nich jest funkcja zwracająca iterowalny zestaw k-krotnych podziałów zbioru danych. Jak przedstawiono w sekcji 4.1.2, ze względu na mały rozmiar zbioru danych, przy podziale należy zastosować losowanie warstwowe. Na listingu 10 przedstawiono funkcję pozwalającą na walidację modelu klasyfikatora biblioteki Keras z użyciem sprawdzianu krzyżowego. Funkcja przyjmuje argumenty w postaci skompilowanego modelu sieci oraz zbioru danych i etykiet. Zwracana jest macierz, w której rzędy oznaczają wyniki testów dla kolejnych podziałów zbioru danych. Sposób wykorzystania zbudowanej funkcji walidacji modelu przedstawia listing 11. Na końcu procesu oceny sieci wyniki kolejnych kroków sprawdzianu krzyżowego należy uśrednić. Efekty walidacji dla trzech sposobów zliczania detali na obrazach przedstawia tabela 4.2.

Zgodnie z wcześniejszymi przewidywaniami, metoda zliczania stosunku śledzonych detali do ich pierwotnej liczby okazała się dawać najlepsze rezultaty. Dokładność na poziomie 91% jest wystarczająca do uznania sieci za dobry klasyfikator. Pozostałe metody zliczania okazały się mniej precyzyjne, co wskazuje że należy je odrzucić.

Przedstawiony mechanizm walidacji wykorzystano do dostrojenia parametrów sieci opisywanych w podsekcji 4.1.1. Uzasadnienie doboru przedstawionej w toku pracy struktury sieci wynika z prób maksymalizacji dokładności obliczonej przy po-

mocy sprawdzianu krzyżowego. W czasie prototypowania sieci dokonano licznych porównań, aby znaleźć najlepszą strukturę. Szczegółowe zestawienie rozpatrywanych konfiguracji i wartości parametrów załączono w dodatku A.

```
def cross_val_demo(X, y):
    """
    Demo cross validation of default grain classifier
    on given data.
    """
    X = np.array(X)
    y = np.array(y)

    model = default_grain_classifier_model()
    model.compile(
        optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])

    scores = network_cross_validation(model, X, y, 3)

    print('Folds scores: (loss, acc)\n', scores)
    scores = np.array(scores)
    print('Cross validation mean score (loss, acc):\n',
          scores.mean(axis=0), '\n')
```

Listing 11: Wykorzystanie funkcji sprawdzianu krzyżowego do oceny działania sieci

Dodatkowo opracowano kod generujący *macierz pomyłek* (ang. *confusion matrix*). Jest to tablica porównująca wiedzę eksperta z wynikami klasyfikacji na zbiorze testowym. Wiersze macierzy oznaczają poprawną klasyfikację kolejnych próbek, zgodnie z etykietami eksperckimi. Kolumny tabeli to typy klasyfikatora. Przekątna tabeli zawiera informacje o elementach poprawnie sklasyfikowanych, etykiety klasyfikatora są zgodne z danymi eksperckimi. Komórki spoza przekątnej oznaczają błędную klasyfikację. Obserwacja macierzy pozwala zaobserwować, które typy ziaren są ze sobą mylone w trakcie klasyfikacji.

Ze względu na małą ilość posiadanych danych macierz pomyłek opracowano na podstawie mechanizmu sprawdzianu krzyżowego. Na każdym jego etapie budowano macierz pomyłek, a ostatecznie tablicę uśredniono. Ułamki dziesiętne w tabeli reprezentują jaka część próbek danej klasy została sklasyfikowana w dany sposób. Funkcję generującą uśrednioną macierz pomyłek przedstawia listing 12.

Metoda zliczania detali	Wskaźnik	
	Błąd	Dokładność
Wszystkie detale	6,59	0,41
Śledzone detale	3,99	0,50
Stosunek śledzonych detali	1,37	0,91

Tablica 4.2: Wskaźniki oceny działania sieci uzyskane metodą sprawdzianu krzyżowego

Zdecydowano się wygenerować macierz pomyłek dla najlepszego sposobu klasyfikacji, bazującego na określaniu stosunku pozostałych detali do ich liczby na początku procesu stygnięcia. Wyniki zestawiono w tabeli 4.3. Zgodnie z wynikami sprawdzianu krzyżowego, większość elementów tablicy znajduje się na jej przekątnej. To znaczy, że wszystkie próbki klas E5R, E11R oraz E16R są poprawnie klasyfikowane. Widoczna jest pomyłka sieci polegająca na niewłaściwej klasyfikacji próbki klasy E6R jako E16R.

		Wiedza klasyfikatora			
		klasa	E5R	E6R	E11R
Wiedza eksperta	E5R	1,00	0,00	0,00	0,00
	E6R	0,00	0,67	0,00	0,33
	E11R	0,00	0,00	1,00	0,00
	E16R	0,00	0,00	0,00	1,00

Tablica 4.3: Tabela pomyłek najlepszego klasyfikatora

```
def mean_confusion_matrix(model, X, y, n_splits):
    """
    Compute mean confusion matrix using cross validation
    with n splits.
    """
    conf_matrix = np.zeros((4, 4))

    folds = StratifiedKFold(n_splits=n_splits).split(X, y)
    for train_index, test_index in folds:
        x_train, x_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        model.fit(x_train, y_train, epochs=300, verbose=0)
        y_pred = model.predict_classes(x_test)

        for test, pred in zip(y_test, y_pred):
            conf_matrix[test][pred] = conf_matrix[test][pred] + 1

    return conf_matrix / n_splits
```

Listing 12: Funkcja języka Python generująca uśrednioną macierz pomyłek

Rozdział 5

Podsumowanie

5.1 Wyniki

Wykonany projekt zrealizował cel jakim była konstrukcja klasyfikatora ziaren rud miedzi, z użyciem termowizji aktywnej oraz sieci neuronowych. Zaproponowane sposoby zbierania danych, ekstrakcji cech oraz model sieci okazały się wystarczające do osiągnięcia ponad 90-procentowej dokładności. Jest to zadawalający wynik, świadczący o dobrej jakości klasyfikacji. W toku projektu wyciągnięto liczne wnioski dotyczące alternatywnych rozwiązań oraz możliwości rozwoju projektu.

Skonstruowany prototyp sieci dobrze pełni rolę klasyfikatora rud miedzi. Zbudowane mechanizmy walidacji modelu mogą być pomocne przy jego przyszłej rozbudowie oraz ocenie alternatywnych struktur sieci neuronowych. Ze względu na bezkontaktowy i niedestruktywny charakter pomiarów termowizją aktywną, przedstawiono sposób klasyfikacji może znaleźć liczne zastosowania. Opracowane algorytmy mogą być podstawą do wykorzystania sieci neuronowych do klasyfikacji ziaren rud miedzi w środowisku przemysłowym. Połączenie termowizji aktywnej oraz uczenia maszynowego może znaleźć zastosowanie w kontroli jakości półproduktów zawierających rudy miedzi. Skonstruowane funkcje przetwarzania obrazów z kamery, automatyczne kadrowanie oraz odczytywanie temperatur ze zdjęć mogą być pomocne w przyszłej pracy z wykorzystywanym sprzętem. Zgromadzone materiały, techniki pomiarowe i algorytmy mogą być pomocne w przyszłych projektach bazujących na wykorzystaniu sieci neuronowych w klasyfikacji ziaren różnorakich materiałów.

5.2 Wnioski

Jak wspomniano w sekcji 3.1 proces budowy bazy danych był czasochłonny oraz nieopozbawiony niedokładności. Zgromadzony zestaw nagrani okazał się wy-

starczający do spełnienia zadania klasyfikacji rud miedzi, jednak możliwe są liczne usprawnienia stanowiska pomiarowego. Zgodnie z opisem wykorzystywanej kamery przedstawionym w sekcji 2.3, istnieją liczne narzędzia pozwalające na automatyzację procesu nagrywania materiałów wideo. Używana w projekcie kamera współpracuje z bibliotekami LabVIEW. Istnieje zatem możliwość realizacji projektu rozbudowy stanowiska laboratoryjnego termowizji w celu automatyzacji procesu ogrzewania oraz nagrywania próbek. Pozwoliłyby to na znacznie szybsze utworzenie dużego zestawu danych oraz poprawiłyby jakość zbieranych materiałów. Budowa bardziej stabilnego i zautomatyzowanego stanowiska pomiarowego wyeliminowałaby ryzyko utraty ostrości obrazu pochodzącego z kamery. Dodatkowo rozbudowa stanowiska pozwoliłaby na realizację alternatywnej metody pomiarów, polegającej na ogrzewaniu próbek do osiągnięcia zadanej temperatury.

Zebrane materiały wideo zawierają wiele szczegółów oraz danych charakteryzujących badane materiały. W podsekcji 3.3.1 rozpatrzone zostały sposoby ekstrakcji cech ze zgromadzonych nagrań. Wybrana metoda śledzenia detali o małej emisyjności okazała się odpowiednia do wykorzystania jej w klasyfikacji ziaren. Przeprowadzona analiza świadczy, że uzyskany zestaw cech próbek dobrze oddaje charakter fizyczального procesu stygnięcia materiałów. Opracowana technika ekstrakcji danych jest efektywna i nie wymaga użycia skomplikowanych modeli sieci neuronowych. Rozpatrzone także alternatywne rozwiązania, które mogą również okazać się skuteczne i interesujące. Po rozbudowie stanowiska oraz zebraniu większej ilości danych warta rozpatrzenia może być się możliwość wykorzystania nowoczesnych konwolucyjnych sieci neuronowych. Zebrane w pracy materiały i wnioski mogą posłużyć do realizacji projektów, które rozwinią temat badania ziaren metodą termowizji.

Projekt spełnił swoje założenia, a jego efektem jest działający klasyfikator rud miedzi. Wymienione wnioski mogą prowadzić do jego rozwoju oraz testów alternatywnych rozwiązań. Projekt może być podstawą do dalszych badań naukowych, jak i prób wdrożenia w praktycznych zastosowaniach.

Repozytorium projektu w serwisie GitHub: <https://github.com/MaciejZj/Copper-grains-classification>

Repozytorium tego dokumentu w serwisie GitHub: <https://github.com/MaciejZj/Bachelors-thesis>

Dodatki

Dodatek A

Porównanie konfiguracji i parametrów modelu sieci

Parametr	Wartość	Błąd	Dokładność
Funkcja aktywacji	sigmoid	1,28	0,58
	relu	1,71	0,83
	elu	2,04	0,92
	tanh	1,44	0,92
Liczba warstw ukrytych	1	1,71	0,67
	2	1,22	0,92
Liczba neuronów w warstwach ukrytych	128 i 64	1,47	0,83
	256 i 128	1,42	0,92
	512 i 256	0,97	0,92
Algorytm uczenia	sgd	0,96	0,58
	adam	0,68	0,92

Dodatek B

Organizacja projektu

B.1 Struktura plików projektu

Podczas pisania funkcji języka Python dbano o ich zwięzłość, dobry podział realizowanych zadań oraz ograniczenie długości bloków kodu. Dzięki temu program zachowuje zasady dobrego programowania [11]. Każdą funkcję opatrzono w specjalny komentarz *docstring* będący standardem w świecie języka Python. Funkcje umieszczone w odpowiednich plikach, które zawierają także przykłady ich użycia i obrazują działanie oraz porównują efekty. Struktura plików w projekcie jest następująca:

img_processing.py funkcje przetwarzania obrazu i wczytywania danych,

blob_finder.py funkcja wykrywania detali na zdjęciach termowizyjnych,

blob_series_tracker.py funkcje śledzenia i zliczania detali w serii zdjęć,

neural_network.py funkcje dostępu do modelu sieci i metod walidacji,

pliki demo.py pliki demonstrujące działanie utworzonego oprogramowania,

thesis_generator.py funkcje generujące obrazy, wykresy oraz tabele,

docs dokumentacja projektu.

Projekt zaopatrzono w plik **requirements.txt** zawierający listę wymaganych modułów bibliotecznych. Stworzone programy najlepiej uruchamiać w środowisku wirtualnym języka Python *virtualenv*¹. Takie rozwiązanie pozwala na izolację projektu od innych modułów Pythona zainstalowanych na komputerze oraz daje

¹Dokumentacja narzędzia virtualenv: <https://virtualenv.pypa.io/en/latest/>

pewność zgodności wersji używanych bibliotek. Programując w języku Python starało się utrzymać zgodność ze standardem *PEP8*² zapewniającym wysoką jakość i czytelność kodu.

B.2 Dokumentacja projektu

Za pomocą modułu Pydoc wygenerowano dokumentację projektu w postaci plików html. Zawierają one listę utworzonych funkcji oraz ich opisy. Pliki dokumentacji zawarto w repozytorium projektu.

B.3 Narzędzia pomocnicze

W czasie programowania oraz pisania pracy korzystano z systemu wersjonowania Git. Używanie systemu kontroli wersji zapewnia porządek w rozwoju projektu oraz chroni przed przypadkową utratą postępów. Aby zapewnić odpowiednią jakość oprogramowania użyto programu Pylint³ (*lintera* języka Python), analizującego kod w celu wykrycia błędów. Dokument opisujący projekt inżynierski napisano z użyciem zestawu makr LaTex oraz systemu bibliografii BibTex.

²Strona standardu PEP8: <https://www.python.org/dev/peps/pep-0008/>

³Strona projektu Pylint: <https://www.pylint.org>

Dodatek C

Spis wykorzystanych funkcji bibliotecznych

Biblioteka standardowa języka Python moduły: `glob`, `math` oraz `natsort`.

Keras funkcje i klasy z przestrzeni nazw `keras`.

Matplotlib funkcje z przestrzeni nazw `mpatches` oraz `plt`.

Numpy funkcje z przestrzeni nazw `np`.

Pillow funkcje z przestrzeni nazw `Image`.

Pytesseract funkcje z przestrzeni nazw `pytesseract`.

Scikit-image funkcje: `blob_dog`, `blob_doh`, `blob_log`, `crop`, `histogram`, `img_as_ubyte`, `imread`, `imsave`, `invert`, `rescale`, `rgb2gray` oraz `threshold_otsu`.

Scikit-learn funkcja `train_test_split` oraz klasa `StratifiedKFold`.

TensorFlow funkcje i klasy z przestrzeni nazw `tf`.

Tikzplotlib funkcje z przestrzeni nazw `tikzplotlib`.

Spis rysunków

2.1	Próbki rozpatrywanych ziaren rud miedzi	4
2.2	Kamera termowizyjna FLIR A320 [14]	5
2.3	Obiektyw FLIR T197415 [15]	7
3.1	Zdjęcia procesu stygnięcia przykładowej próbki 104 klasy E5R	14
3.2	Przygotowanie zakresu temperatur do odczytu przez sieć neuronową	15
3.3	Porównanie procesu stygnięcia próbek klasy E5R oraz E6R	16
3.4	Zbliżenie na charakterystyczne grupy detali materiału	19
3.5	Porównanie bibliotecznych algorytmów wykrywania plam na obrazie	21
3.6	Zliczanie śledzonych detali w próbce dwoma sposobami	25
3.7	Wykryte detale, śledzone od początku stygnięcia, zaznaczone na pierwszym obrazie w serii pomiarowej	26
3.8	Liczba wykrytych detali na każdym etapie stygnięcia, w wariancie zliczającym wszystkie plamy	27
3.9	Liczba detali wykrytych na zdjęciach, w wariancie zliczającym plamy śledzone od początku procesu stygnięcia	28
3.10	Stosunek śledzonych detali do ich liczby na początku procesu stygnięcia	29
4.1	Funkcja aktywacji softmax	32
4.2	Funkcje aktywacji rozpatrywane do użycia w warstwach ukrytych .	33
4.3	Przebieg treningu sieci dla różnych metod zliczania ziaren	35

Spis tablic

3.1	Stosunek liczby detali śledzonych w kolejnych etapach stygnięcia do ich liczby na początku pomiaru	26
4.1	Wskaźniki oceny działania sieci na zbiorze testowym	36
4.2	Wskaźniki oceny działania sieci uzyskane metodą sprawdzianu krzy- żowego	39
4.3	Tabela pomyłek najlepszego klasyfikatora	39

Spis listingów

1	Funkcja języka Python odczytująca zakres temperatur ze zdjęć	17
2	Funkcja języka Python wykrywająca detale na obrazie	21
3	Funkcja języka Python śledząca detale w serii zdjęć	22
4	Funkcja języka Python wykrywające te same detale w kolejnych obrazach	23
5	Funkcja języka Python sprawdzająca czy dany punkt leży wewnątrz podanego okręgu	23
6	Funkcja języka Python usuwająca duplikaty z listy	24
7	Funkcja języka Python obliczająca jaka część ziaren z początku stygnięcia pozostała w jego kolejnych etapach	24
8	Funkcja języka Python definiująca model sieci neuronowej	33
9	Kod treningu sieci neuronowej klasyfikującej ziarna miedzi	36
10	Funkcja języka Python definiująca model sieci neuronowej	37
11	Wykorzystanie funkcji sprawdzianu krzyżowego do oceny działania sieci	38
12	Funkcja języka Python generująca uśrednioną macierz pomyłek	40

Bibliografia

- [1] D. Buchczik, J. Wegehaupt, O. Krauze. Indirect measurements of milling product quality in the classification system of electromagnetic mill. *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, strony 1039–1044, Aug 2017.
- [2] Sebastian Budzan, Dariusz Buchczik, Marek Pawełczyk, Jirí Tůma. Combining segmentation and edge detection for efficient ore detection in an electromagnetic mill classification system. *Sensors*, 19(8), 2019.
- [3] Fran,cois Chollet. *Deep Learning, Praca z językiem Python i biblioteką Keras*. Helion, 2019.
- [4] François Chollet, i in. Keras. <https://github.com/fchollet/keras>, 2015.
- [5] Francesco Ciampa, Pooya Mahmoodi, Fulvio Pinto, Michele Meo. Recent advances in active infrared thermography for non-destructive testing of aerospace components. *Sensors*, 18:25, 02 2018.
- [6] George Cybenko. Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.
- [7] FLIR Systems. *FLIR A320 FLIR A325 User's manual*, April 2008.
- [8] Aurélien Géron. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow*. Helion, 2018.
- [9] Keras contributors, i in. Keras documentation. <https://keras.io>, 2019.
- [10] J.H. i Lienhard V J.H. Lienhard, IV. *A Heat Transfer Textbook*, strona 19. Phlogiston Press, wydanie 5, 2019.
- [11] Robert C. Martin. *Czysty kod. Podręcznik dobrego programisty*. Helion, 2015.
- [12] Scikit-image contributors, i in. Scikit-image API reference. <https://scikit-image.org/docs/stable/api/api.html>, 2019. 0.16.1.

- [13] Mehmet Sezgin, Böлent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–165, 2004.
- [14] FLIR Systems. Specyfikacja kamer FLIR A320/A325. <https://www.flir.com/support/products/a325sc#Overview>, 2019.
- [15] FLIR Systems. Specyfikacja obiektywu FLIR T197415. <https://www.flircameras.com/t197415-close-up-ir-lens.htm>, 2019.
- [16] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, the scikit-image contributors. Scikit-image: image processing in Python. *PeerJ*, 2, 6 2014.
- [17] Chris Van Hoof, Piet De Moor. Chapter 12 - PolySiGe uncooled microbolometers for thermal IR detection. Mohamed Henini, Manijeh Razeghi, redaktorzy, *Handbook of Infra-red Detection Technologies*, strony 449–479. Elsevier Science, 2002.