UNIWERSYTET WARMIŃSKO MAZURSKI
W OLSZTYNIE
WYDZIAŁ MATEMATYKI I INFORMATYKI

Damian Murawski
Kierunek: Informatyka

Segmentacja obrazów medycznych

Praca magisterska wykonana w
Katedrze Metod Matematycznych
Informatyki
pod kierunkiem
dr hab. Aleksandra Denisiuka

Olsztyn 2024

UNIVERSITY OF WARMIA AND MAZURY
IN OLSZTYN FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Damian Murawski
Computer Science

Segmentation of medical images

Master's written in Department of Mathematical Methods of Computer Science under supervision of
dr hab. Aleksandra Denisiuka

Olsztyn 2024

Spis treści

Streszczenie

Celem pracy jest opracowanie zaawansowanej aplikacji, która ma umożliwić segmentację oraz klasyfikację obrazów medycznych w celu precyzyjnego rozpoznawania ich zawartości. Badania skupią się na analizie wpływu różnych parametrów i technik przetwarzania obrazów na efektywność algorytmów segmentacji i klasyfikacji. Przeprowadzone eksperymenty obejmą zróżnicowane rodzaje filtrów, metody segmentacji oraz architektury sieci neuronowych. Głównym celem jest zidentyfikowanie optymalnych ustawień parametrów, które zapewnią najwyższą precyzję i wydajność algorytmów. Ostatecznie, opracowana aplikacja ma umożliwić dokładną analizę obrazów medycznych, co potencjalnie może przyczynić się do podniesienia standardów opieki zdrowotnej.

Abstract

The aim of the study is to develop an advanced computer application to enable segmentation and classification of medical images for accurate recognition of their content. The research will focus on analyzing the impact of various image processing parameters and techniques on the effectiveness of segmentation and classification algorithms. The experiments to be carried out will include several types of filters, segmentation methods and neural network architectures. The main goal is to find the optimal parameter settings that will ensure the highest precision and efficiency of the algorithms. The developed application is expected to enable accurate analysis of medical images, with the potential to improve healthcare standards.

1 Wstęp

1.1 Kontekst badawczy

Kontekst badawczy obejmuje szerokie spektrum badań i rozwoju technik analizy obrazów medycznych w celu ich klasyfikacji oraz segmentacji.

Segmentacja obrazów medycznych jest procesem podziału obrazu na różne regiony lub segmenty w celu identyfikacji i izolacji obszarów zainteresowania. W przypadku obrazów płuc, segmentacja może pomóc w wyodrębnieniu struktur anatomicznych, takich jak płuca, oskrzela czy guzy, co jest kluczowe dla diagnostyki chorób płuc. Obszary badawczych, które można wskazać to:

Segmentacja płuc: Badania nad metodami segmentacji płuc w obrazach radiologicznych, takich jak tomografia komputerowa (CT) czy rezonans magnetyczny (MRI), mają na celu dokładne wyodrębnienie struktur płucnych, co jest istotne dla diagnozowania chorób płuc.

Detekcja guzów i zmian patologicznych: Opracowywanie algorytmów do wykrywania guzów, zmian patologicznych czy zmian strukturalnych na obrazach płucnych ma kluczowe znaczenie dla wczesnego wykrywania nowotworów i chorób płucnych.

Klasyfikacja chorób: CNN są wykorzystywane do klasyfikowania obrazów płuc na podstawie różnych chorób, takich jak zapalenie płuc, gruźlica czy rak płuc. Badania nad efektywnymi technikami uczenia maszynowego w tej dziedzinie mają na celu zwiększenie dokładności diagnozowania chorób płuc.

Segmentacja i analiza zmian dynamicznych: Opracowywanie metod segmentacji i analizy obrazów płuc w czasie rzeczywistym, na przykład w monitorowaniu progresji chorób lub ocenie skuteczności leczenia.

Interpretowalność modeli: Badania nad interpretowalnością modeli CNN w diagnostyce chorób płuc, aby lekarze mogli zrozumieć, dlaczego model podjął określone decyzje diagnostyczne.

Podsumowując, badania nad segmentacją obrazów medycznych oraz rozpoznawaniem chorób płuc za pomocą CNN mają na celu rozwój skutecznych i precyzyjnych narzędzi wspierających lekarzy w diagnozowaniu i leczeniu chorób płuc, co przyczynia się do poprawy opieki zdrowotnej i wyników klinicznych

1.2 Cel pracy

Celem niniejszej pracy jest opracowanie zaawansowanej aplikacji informatycznej, która będzie umożliwiała segmentację oraz klasyfikację obrazów medycznych w celu precyzyjnego rozpoznawania ich zawartości. Obrazy medyczne stanowią kluczowe źródło informacji w dziedzinie diagnostyki oraz leczenia różnych schorzeń i chorób.

Segmentacja obrazu medycznego polega na identyfikacji i wyodrębnieniu istotnych struktur, takich jak narządy, tkanki czy zmiany patologiczne, co jest kluczowym krokiem w analizie obrazów medycznych. Klasyfikacja natomiast pozwala na automatyczne identyfikowanie i rozpoznawanie rodzaju lub charakteru obiektów obecnych na obrazie, co ma istotne znaczenie w procesie diagnostycznym.

W ramach pracy przeprowadzimy badania mające na celu zrozumienie, jak różne parametry oraz techniki przetwarzania obrazów wpływają na skuteczność segmentacji i klasyfikacji obrazów medycznych. Testy będą obejmować eksperymenty z różnymi rodzajami filtrów, metodami segmentacji oraz architekturami sieci neuronowych wykorzystywanych do klasyfikacji obrazów. Poprzez systematyczne analizy wyników będziemy dążyć do identyfikacji optymalnych ustawień parametrów, które zapewnią najwyższą precyzję oraz wydajność algorytmów segmentacji i klasyfikacji.

Ostatecznie, celem naszej pracy będzie stworzenie aplikacji, która nie tylko umożliwi dokładną segmentację oraz klasyfikację obrazów medycznych, ale także będzie elastyczna i dostosowana do różnych rodzajów obrazów oraz zmiennych warunków diagnostycznych. Dzięki temu aplikacja będzie mogła znaleźć zastosowanie w praktyce medycznej, wspierając specjalistów w szybkiej i skutecznej analizie obrazów medycznych, co przyczyni się do poprawy jakości opieki zdrowotnej.

2 Technologie użyte w projekcie

Narzędzia programistyczne:

- **Visual Studio Code**: Środowisko programistyczne zintegrowane (IDE), które zapewnia edycję kodu, debugowanie, zarządzanie projektami i inne narzędzia pomocne w procesie programowania.

- **Google collab**: to bezpłatna usługa oferowana przez Google, umożliwiająca uruchamianie i udostępnianie notebooków Jupyter w chmurze. Umożliwia to wygodne tworzenie, edycję, uruchamianie i dzielenie się kodem w środowisku przeglądarki internetowej.

Języki programowania:

- **Python**: język programowania, który jest szeroko stosowany w dziedzinie analizy danych, uczenia maszynowego i przetwarzania obrazów medycznych. Język ten jest znany z czytelności i prostoty składni, co ułatwia pracę nad projektami związanymi z analizą danych.

Frameworki i biblioteki:

- **PyTorch:** to popularna biblioteka do uczenia maszynowego i głębokiego uczenia, stworzona przez grupę badawczą Facebooka. Dzięki elastyczności i prostocie użycia, umożliwia interaktywne eksperymentowanie z modelami i debugowanie kodu. Biblioteka ta oferuje również wsparcie dla obliczeń na GPU, co przyspiesza szkolenie modeli.

- **Torchmetrics:** to biblioteka do oceny i ewaluacji modeli w PyTorch. Zapewnia gotowe metryki oceny, takie jak dokładność czy precyzja oraz umożliwia definiowanie własnych metryk i ich integrację z procesem trenowania i walidacji modelu.

- **Torchvision:** to biblioteka do przetwarzania obrazów i komputerowego widzenia w PyTorch. Zapewnia narzędzia do łatwego wczytywania, przetwarzania i analizowania obrazów, w tym gotowe modele pre-trenowane do klasyfikacji obrazów czy detekcji obiektów.

- **Matplotlib**: Biblioteka do tworzenia wykresów i wizualizacji danych w języku Python. Matplotlib jest często wykorzystywany do wizualizacji wyników analizy danych oraz prezentacji rezultatów uczenia maszynowego, takich jak krzywe uczenia, wykresy dokładności czy straty.

- **NumPy**: Biblioteka do obliczeń numerycznych w języku Python. NumPy umożliwia manipulację danymi numerycznymi oraz wykonywanie operacji algebraicznych na macierzach, co jest niezbędne w pracy z danymi w uczeniu maszynowym.

- **Scikit-image (skimage):** Biblioteka do przetwarzania obrazów w języku Python. Scikit-image zawiera zestaw narzędzi do segmentacji, filtrowania, transformacji i analizy obrazów, co jest przydatne w procesie analizy obrazów medycznych.

- **CNN (Convolutional Neural Network)**: Specjalny rodzaj sieci neuronowych, które są często wykorzystywane w zadaniach analizy obrazów, w tym segmentacji obrazów medycznych i rozpoznawaniu chorób płuc.

- **CUDA (Compute Unified Device Architecture):** to platforma programistyczna firmy NVIDIA, umożliwiająca wykorzystanie mocy obliczeniowej kart graficznych (GPU) do przyspieszenia obliczeń równoległych w aplikacjach wymagających dużych zasobów obliczeniowych.

- **Pandas:** to biblioteka programistyczna w języku Python, dedykowana do manipulacji i analizy danych. Dzięki Pandas, programiści mogą łatwo

importować, przetwarzać i analizować dane w formie tabelarycznej, podobnej do arkusza kalkulacyjnego, co czyni ją niezwykle przydatną w pracy z danymi w analizie danych i uczeniu maszynowym.

- **U-NET:** to głęboka architektura sieci neuronowej zaprojektowana do semantycznej segmentacji obrazu, czyli przypisywania etykiety (klasy) każdemu pikselowi obrazu. Jest stosowany głównie w medycynie do analizy obrazów medycznych, ale również znajduje zastosowanie w innych dziedzinach, takich jak przetwarzanie obrazów.

3 Powiązane prace

4 Algorytm

4.1 Algorytmy segmentacji

Segmentacja obrazu jest procesem podziału obrazu na różne obszary lub segmenty w celu identyfikacji i izolacji konkretnych struktur, obiektów lub cech na obrazie. Celem segmentacji jest wyodrębnienie istotnych informacji z obrazu poprzez podział go na bardziej spójne i homogeniczne regiony. Istnieje wiele technik segmentacji obrazu, z których każda może być stosowana w zależności od konkretnego problemu i cech obrazu.

Popularne algorytmów segmentacji obrazu:

1. **Progowanie globalne:**

- Opis: Podstawowa metoda segmentacji, która dzieli obraz na obszary na podstawie progu jasności.
- Działanie: Piksele o wartościach jasności powyżej lub poniżej ustalonego progu są przypisywane do różnych segmentów.
- Zastosowanie: Idealny dla obrazów o dobrze zdefiniowanych kontrastach między obszarami.

2. **Progowanie lokalne:**

   - Opis: Rozszerzenie progowania globalnego, które uwzględnia lokalne cechy obrazu.
   - Działanie: Dynamicznie dostosowuje próg na podstawie lokalnych wartości pikseli w sąsiedztwie.
   - Zastosowanie: Skuteczne w przypadku obrazów z niejednolitym oświetleniem lub tłem.

3. **Segmentacja przez rozrost obszarów:**

   - Opis: Metoda segmentacji, która rośnie obszary na podstawie podobieństwa pikseli.
   - Działanie: Rozpoczyna od punktów startowych i łączy piksele o podobnych właściwościach (np. jasności) w jednolite regiony.
   - Zastosowanie: Wykorzystywana w przypadkach, gdzie obiekty na obrazie mają jednolite cechy.

4. **Segmentacja oparta na krawędziach (np. Algorytm Canny'ego):**

   - Opis: Wykrywa krawędzie obiektów na obrazie, które są często granicami między różnymi regionami.
   - Działanie: Wykorzystuje detekcję zmian gradientu jasności w obrazie do lokalizacji krawędzi.
   - Zastosowanie: Skuteczna w przypadkach, gdzie obiekty mają wyraźne krawędzie lub kontury.

5. **Segmentacja oparta na rozmyciu (np. Algorytm watershed):**

   - Opis: Metoda segmentacji, która traktuje jasność pikseli jako wysokość terenu i wykorzystuje poziomy wody do podziału obrazu na baseny.
   - Działanie: Rozpoczyna od punktów startowych i "zalewa" obszary pikseli, aby wyznaczyć granice między segmentami.

- Zastosowanie: Skuteczna w segmentacji obiektów o zróżnicowanych kształtach i rozmiarach.

6. **Segmentacja oparta na uczeniu maszynowym (np. U-Net):**

- Opis: Wykorzystuje konwolucyjne sieci neuronowe (CNN) do automatycznego segmentowania obrazów.
- Działanie: Model CNN jest trenowany na danych wejściowych i ich etykietach, aby nauczyć się segmentacji obrazu.
- Zastosowanie: Skuteczna w wielu zastosowaniach medycznych, w tym segmentacji struktur anatomicznych na obrazach medycznych.

4.2 Wybór algorytmu segmentacji

Algorytm segmentacji za pomocą U-Net został wybrany ze względu na swoje liczne zalety, które czynią go atrakcyjnym rozwiązaniem w analizie obrazów. Oferuje między innymi:

1. **Złożona struktura sieci:** U-Net posiada symetryczną architekturę, która składa się z enkodera (skupiającego się na ekstrakcji cech) i dekodera (odpowiadającego za rekonstrukcję obrazu). Ta struktura pozwala na dokładne wyodrębnianie szczegółów z obrazów medycznych, co jest kluczowe dla diagnostyki.

2. **Połączenia skokowe:** U-Net wykorzystuje połączenia skokowe (skip connections), które przekazują informacje z enkodera do dekodera. Dzięki nim sieć może skutecznie zachować lokalne szczegóły i struktury, co jest niezwykle istotne w przypadku obrazów medycznych, gdzie detale mogą mieć kluczowe znaczenie dla diagnostyki.

3. **Odporność na niedobór danych:** W przypadku obrazów medycznych, gdzie dostępność dużych zbiorów danych treningowych może być ograniczona ze względu na prywatność pacjentów, U-Net jest szczególnie przydatny. Połączenia skokowe pozwalają na efektywne wykorzystanie nawet ograniczonej ilości danych

treningowych, co pozwala na uzyskanie skutecznych wyników przy mniejszym nakładzie danych.

4. **Skuteczność w wyodrębnianiu struktur:** Obrazy medyczne często zawierają subtelne struktury i detale, które są kluczowe dla diagnozy. U-Net, dzięki swojej złożonej strukturze i połączeniom skokowym, jest w stanie dokładnie wyodrębniać te struktury, co czyni go idealnym wyborem dla segmentacji obrazów medycznych.

5. **Popularność i dostępność:** U-Net jest dobrze zbadaną i powszechnie stosowaną architekturą w dziedzinie segmentacji obrazów medycznych. Istnieje wiele gotowych implementacji oraz badań potwierdzających jej skuteczność w różnych zastosowaniach medycznych, co sprawia, że jest to solidny i sprawdzony wybór dla projektów w tej dziedzinie.

4.3 Opis działania algorytmu

1. **Enkoder (skrócenie ścieżki):**
   - Pierwsza część U-Net to tzw. enkoder, który działa podobnie jak klasyczne konwolucyjne sieci neuronowe. Enkoder składa się z serii warstw konwolucyjnych i warstw poolingowych, które służą do ekstrakcji cech z obrazu.
   - Podczas przetwarzania enkoder redukuje rozmiar obrazu wejściowego, a jednocześnie zwiększa liczbę cech.

2. **Połączenia skokowe (skip connections):**
   - Kluczową cechą U-Net są połączenia skokowe, które łączą wyjścia warstw enkodera z odpowiadającymi im warstwami dekodera. Te połączenia pozwalają na przekazywanie szczegółowych informacji o lokalnych strukturach z enkodera do dekodera, co pomaga w rekonstrukcji obrazu w procesie dekodowania.

3. **Dekoder (rozszerzenie ścieżki):**
   - Druga część U-Net to tzw. dekoder, który rozszerza przetworzoną przez enkoder reprezentację na pierwotny rozmiar obrazu. Dekoder składa się z serii warstw

dekonwolucyjnych (transponowanych konwolucji), które zwiększają rozmiar przestrzenny wyjściowych predykcji.

- Podczas przetwarzania dekoder odtwarza lokalne detale, wykorzystując informacje przekazane przez połączenia skokowe.

4. **Ostatnia warstwa:**

   Na końcu U-Net zwykle znajduje się warstwa konwolucyjna z funkcją aktywacji, która przetwarza wyniki dekodera i generuje ostateczne mapy segmentacji dla każdego piksela obrazu.

4.4 Sieci konwolucyjne (Convolutional Neural Networks, CNN)

4.4.1 Co to są sieci konwolucyjne?

W uczeniu maszynowym klasyfikator przypisuje etykietę klasy do danych. Na przykład, klasyfikator obrazu tworzy etykietę klasy (np. ptak, samolot) dla obiektów na obrazie. Konwolucyjna sieć neuronowa (CNN) jest rodzajem klasyfikatora, który skutecznie rozwiązuje ten problem.

CNN to sieć neuronowa używana do rozpoznawania wzorców w danych. Składa się z neuronów zorganizowanych w warstwy, z własnymi wagami i odchyleniami, dostosowywanymi w trakcie uczenia. Sieci CNN operują na tensorach, zwykle 3-wymiarowych w przypadku warstw, z wyjątkiem warstwy wyjściowej.

Neurony przetwarzają wiele danych wejściowych na jedno wyjście, reprezentowane jako mapy aktywacji. Warstwy składają się z neuronów wykorzystujących te same operacje i hiperparametry.

Wagi jądra i odchylenia są dostosowywane w trakcie uczenia, umożliwiając sieci adaptację do problemu i danych. Ich wartości są wizualizowane poprzez skalę kolorów. CNN wykorzystują specjalne warstwy splotowe, doskonale nadające się do przetwarzania obrazów. Mogą być używane w różnych zadaniach związanych z wizją komputerową, takich jak klasyfikacja, segmentacja i wykrywanie obiektów.

4.4.2 Architektury sieci CNN

Najbardziej popularne architektury sieci konwolucyjnych neuronów (CNN) obejmują wiele wariantów, które ewoluowały wraz z postępem badań w dziedzinie sztucznej inteligencji:

1. **LeNet-5:** Pierwsza skuteczna architektura CNN opracowana przez Yanna LeCuna w latach 90. XX wieku, wykorzystywana do rozpoznawania ręcznie pisanych cyfr.

2. **AlexNet:** Opracowana przez Alexa Krizhevsky'ego, Ilya Sutskevera i Geoffreya Hintona w 2012 roku, AlexNet była przełomem w dziedzinie rozpoznawania obrazów, wygrywając konkurs ImageNet Large Scale Visual Recognition Challenge (ILSVRC) z przewagą ponad 10% nad konkurentami. Składa się z pięciu warstw konwolucyjnych, połączonych z operacjami ReLU oraz trzech warstw w pełni połączonych.

3. **VGG:** Zaproponowana przez grupę Visual Geometry Group na Uniwersytecie Oksfordzkim, architektura ta charakteryzuje się bardzo głęboką siecią o 16 lub 19 warstwach konwolucyjnych. Posiada małe filtry konwolucyjne (3x3) stosowane z dużą liczbą map cech, co prowadzi do bardzo dużego liczby parametrów.

4. **GoogLeNet (Inception):** Opracowana przez naukowców z Google w 2014 roku, ta architektura wykorzystuje moduły Inception, które łączą filtry o różnych rozmiarach w jednym punkcie. Ta struktura pomaga w efektywnym wykorzystaniu zasobów obliczeniowych.

5. **ResNet:** Zaprojektowana przez Kaiminga He i jego zespół w 2015 roku, architektura ta wprowadziła innowacyjny blok resztkowy (Residual Block), który umożliwia przekazywanie informacji przez warstwy pomijając połączenia (skip connections). To rozwiązanie pomaga w rozwiązywaniu problemu zanikającego gradientu i umożliwia budowanie bardzo głębokich sieci.

6. **MobileNet:** Opracowana przez Google, architektura ta została zoptymalizowana pod kątem wydajności na urządzeniach mobilnych. Wykorzystuje technikę

separowalnych konwolucji, aby zmniejszyć liczbę parametrów i obliczeń, co jest korzystne dla ograniczonych zasobów mobilnych.

4.4.3 Warstwy sieci

W konwolucyjnych sieciach neuronowych (CNN) warstwy są zorganizowane hierarchicznie, każda wykonując określone operacje przetwarzania danych. Oto opis kilku kluczowych warstw w CNN:

1. **Warstwa wejściowa:** Ta warstwa przyjmuje dane wejściowe, najczęściej obrazy i przekazuje je do sieci. Dla obrazów RGB, warstwa ta ma trzy kanały (czerwony, zielony, niebieski), reprezentujące kolory.

2. **Warstwa konwolucyjna:** To jedna z kluczowych warstw w CNN. W tej warstwie stosowane są filtry konwolucyjne, które przesuwają się po danych wejściowych, wydobywając lokalne wzorce. Każdy filtr generuje mapę cech, która reprezentuje aktywacje wykrytych cech.

3. **Warstwa aktywacji:** Po operacji konwolucji, na wynikach stosowana jest funkcja aktywacji, najczęściej ReLU (Rectified Linear Unit), która wprowadza nieliniowość do modelu, pomagając w uczeniu się złożonych zależności.

4. **Warstwa poolingowa:** Warstwa ta wykonuje operację poolingu, na przykład max-poolingu, która redukuje rozmiar map cech, zachowując najważniejsze informacje. Pomaga to zmniejszyć liczbę parametrów i obliczeń w sieci, co przyspiesza uczenie się i zmniejsza overfitting np.: Flatten ().

5. **Warstwa w pełni połączona**: Po przetworzeniu danych przez warstwy konwolucyjne i poolingowe, dane są przekształcane w wektor i przekazywane do warstwy w pełni połączonej (czyli gęsto połączonej). W tej warstwie każdy neuron jest połączony z każdym neuronem w poprzedniej warstwie, co umożliwia uczenie się złożonych zależności między cechami.

6. **Warstwa wyjściowa:** Warstwa końcowa, która przewiduje etykietę lub klasyfikację dla danych wejściowych. Dla problemów klasyfikacyjnych, może to być warstwa softmax, która zwraca prawdopodobieństwa przynależności do poszczególnych klas.

To są podstawowe warstwy konwolucyjnych sieci neuronowych, ale w zależności od architektury sieci mogą być stosowane również inne warstwy lub modyfikacje tych podstawowych.

4.4.4 Hyperparametry

Hyperparametry w konwolucyjnych sieciach neuronowych (CNN) to parametry, które nie są uczone podczas procesu treningu, ale muszą być ustawione ręcznie przed rozpoczęciem uczenia. Oto kilka kluczowych hyperparametrów w CNN:

1. **Rozmiar filtru konwolucyjnego:** Określa wielkość filtra stosowanego w warstwach konwolucyjnych. Typowy rozmiar to 3x3 lub 5x5, ale może być dostosowany do specyfiki problemu.

2. **Kroki (stride):** Określa, o ile pikseli przesuwa się filtr podczas konwolucji. Zwiększanie kroku zmniejsza rozmiar wyjściowy mapy cech.

3. **Wypełnienie (padding):** Decyduje, jak dane wejściowe są uzupełniane wokół krawędzi przed zastosowaniem filtra konwolucyjnego. Wypełnienie może być "same" (zero-padding) lub "valid" (bez wypełnienia).

4. **Liczba filtrów konwolucyjnych:** Określa, ile filtrów konwolucyjnych ma być stosowanych w każdej warstwie konwolucyjnej. Każdy filtr wykrywa inny zestaw cech.

5. **Rozmiar poolingu:** Określa wielkość obszaru, na którym stosowana jest operacja poolingowa (np. max-pooling). Pomaga to zmniejszyć rozmiar map cech.

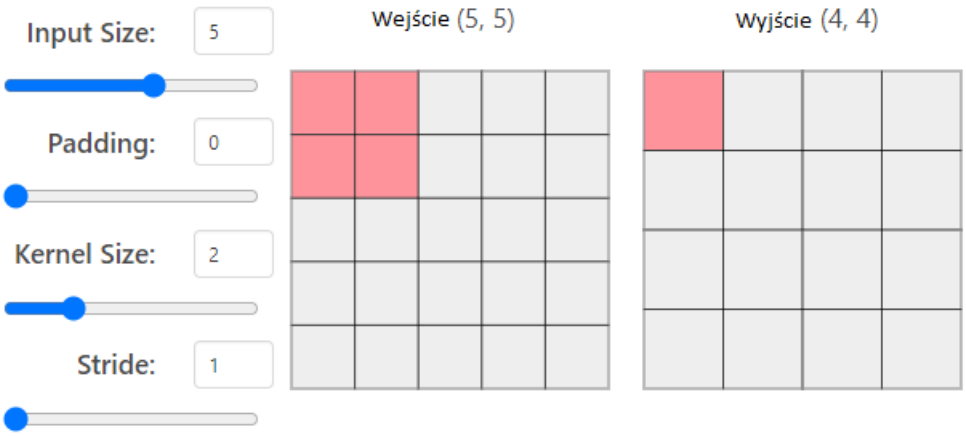
6. **Rodzaj pooling:** Określa rodzaj operacji poolingowej, np. max-pooling, average-pooling.

7. **Liczba neuronów w warstwach gęsto połączonych:** Określa liczbę neuronów w warstwach gęsto połączonych, które występują po warstwach konwolucyjnych i poolingowych.

8. **Współczynnik uczenia:** Określa, jak szybko model się uczy podczas procesu uczenia. Zbyt wysoki współczynnik uczenia może prowadzić do niestabilności, podczas gdy zbyt niski może wydłużyć czas uczenia.

9. **Rozmiar wsadu (batch size):** Określa liczbę przykładów treningowych przekazywanych do modelu w jednej iteracji podczas treningu.

Dobór odpowiednich hyperparametrów jest kluczowy dla skuteczności i wydajności CNN. Może wymagać eksperymentowania i dostosowywania na podstawie specyfiki problemu oraz dostępnych zasobów obliczeniowych.



5 Implementacja

5.1 Klasyfikacja

5.1.1 Klasyfikacja wieloklasowa

Klasyfikacja wieloklasowa jest jednym z rodzajów problemów klasyfikacji w uczeniu maszynowym, w którym zadaniem jest przypisanie jednego z wielu możliwych etykiet (klas) do danego obiektu na podstawie jego cech. W odróżnieniu od klasyfikacji binarnej, gdzie istnieją tylko dwie klasy, klasyfikacja wieloklasowa dotyczy sytuacji, w której obiekty mogą być przyporządkowane do jednej z wielu kategorii.

5.1.2 Klasyfikacja binarna

Klasyfikacja binarna to jeden z najprostszych rodzajów problemów klasyfikacji w uczeniu maszynowym. W tym rodzaju klasyfikacji zadaniem jest przypisanie jednej z dwóch możliwych klas do danego obiektu na podstawie jego cech. Typowym przykładem klasyfikacji binarnej jest rozpoznawanie czy e-mail jest spamem czy nie-spamem, czy pacjent jest chory na daną chorobę czy nie, czy transakcja finansowa jest podejrzana czy nie.

5.2 Tworzenie zbioru danych

5.2.1 Podział danych na zbiory treningowy i testowy

Zbiór treningowy to zestaw danych, które służą do nauki modelu w trakcie procesu uczenia. Model przystosowuje swoje parametry, starając się minimalizować błędy predykcji na tych konkretnych danych. Jakość oraz różnorodność danych treningowych bezpośrednio wpływają na jakość i skuteczność ostatecznie wytrenowanego modelu.

Natomiast zbiór testowy to zbiór danych, który nie był wykorzystywany w procesie uczenia. Po przeszkoleniu modelu na danych treningowych, jest on testowany na zbiorze testowym, aby ocenić jego zdolność do radzenia sobie z nowymi, nieznajomymi danymi. Testowanie na zbiorze testowym pozwala na ocenę ogólnej

zdolności modelu do generalizacji, czyli na jego zdolność do dokonywania trafnych predykcji na nowych danych, które nie były częścią zbioru treningowego.

5.2.2 Augmentacja (Rozszerzanie danych)

Augmentacja danych to technika stosowana w uczeniu maszynowym, szczególnie w zadaniach związanych z przetwarzaniem obrazów, w celu zwiększenia różnorodności danych treningowych poprzez generowanie syntetycznych przykładów na podstawie istniejących danych treningowych. Istnieje kilka sposobów, w jaki można przeprowadzić augmentację danych w przypadku obrazów:

- Rotacja: Obracanie obrazów o różne kąty, aby model mógł nauczyć się rozpoznawać obiekty pod różnymi kątami.
- Przesunięcie: Przesuwanie obrazów w poziomie i pionie, aby model mógł nauczyć się rozpoznawać obiekty w różnych położeniach na obrazie.
- Zmiana skali: Zmniejszanie lub zwiększanie obrazów, aby model mógł nauczyć się rozpoznawać obiekty różnych rozmiarów.
- Odbicie lustrzane: Odbicie obrazów w poziomie lub pionie, aby model mógł nauczyć się rozpoznawać obiekty w różnych orientacjach.
- Zmiana jasności i kontrastu: Modyfikowanie jasności, kontrastu i innych właściwości obrazu, aby model mógł nauczyć się radzić sobie z różnymi warunkami oświetleniowymi.
- Dodawanie szumów: Dodawanie szumów do obrazów, aby model mógł nauczyć się rozpoznawać obiekty w bardziej realistycznych warunkach.
- Przycinanie: Przycinanie obrazów w różnych miejscach, aby model mógł nauczyć się rozpoznawać obiekty w różnych kontekstach.

5.3 Metryki ewaluacji modelu klasyfikacji

Metryki ewaluacji modelu klasyfikacji to narzędzia używane do oceny skuteczności modelu w zadaniu klasyfikacji. Służą one do pomiaru jakości predykcji modelu na podstawie rzeczywistych i przewidywanych wartości. Najczęściej stosowane metryki to:

1. **Dokładność (Accuracy**): Proporcja poprawnych predykcji do wszystkich próbek. Wartość bliższa 1 oznacza lepszą dokładność.

2. **Precyzja (Precision):** Proporcja poprawnych pozytywnych predykcji do wszystkich pozytywnych predykcji. Ocena, jak wiele z pozytywnych predykcji było poprawnych.

3. **Czułość (Recall**): Proporcja poprawnych pozytywnych predykcji do wszystkich prawdziwych pozytywnych próbek. Ocena, jak wiele z prawdziwych pozytywnych próbek zostało poprawnie wykrytych.

4. **Współczynnik F1 (F1-score):** Średnia harmoniczna precyzji i czułości. Uwzględnia zarówno precyzję, jak i czułość.

5. **Macierz pomyłek (Confusion matrix):** Tabela przedstawiająca liczbę prawdziwie pozytywnych, fałszywie pozytywnych, prawdziwie negatywnych i fałszywie negatywnych predykcji.

6. **Raport klasyfikacji (Classification report):** Zawiera informacje o dokładności, precyzji, czułości, współczynniku F1 oraz innych metrykach dla każdej klasy w problemie wieloklasowej klasyfikacji.

5.4 Funkcje strat

Funkcje strat (ang. loss functions) są kluczowym elementem procesu uczenia maszynowego, ponieważ pozwalają mierzyć, jak bardzo predykcje modelu różnią się od rzeczywistych wartości (etykiet) w danych treningowych. Przykłady podstawowych funkcji strat:

1. Binary Crossentropy (Binary Cross-Entropy Loss):

- Jest to popularna funkcja straty stosowana w problemach binarnej klasyfikacji, gdzie mamy tylko dwie możliwe klasy (np. 0 i 1).

- Funkcja ta jest szczególnie użyteczna, gdy chcemy mierzyć różnicę między prawdopodobieństwem przewidywanym przez model a rzeczywistą etykietą.
- W przypadku PyTorch funkcję tę można znaleźć jako *torch.nn.BCELoss.*

2. Cross Entropy (Cross-Entropy Loss):

- Jest to powszechnie używana funkcja straty w problemach wieloklasowej klasyfikacji, gdzie mamy więcej niż dwie klasy.
- Funkcja ta mierzy różnicę między rozkładem prawdopodobieństwa przewidywanym przez model a rzeczywistymi etykietami, wykorzystując zwykle funkcję softmax.
- W PyTorch funkcję tę można znaleźć jako *torch.nn.CrossEntropyLoss*.

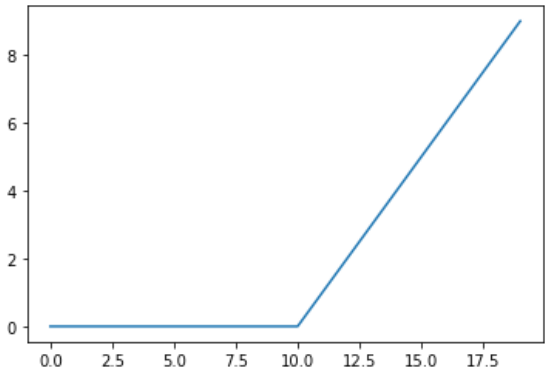
5.5 Funkcje aktywacji

Funkcje aktywacji są nieliniowymi funkcjami stosowanymi wewnątrz neuronów sztucznych sieci neuronowych. Służą one do wprowadzenia nieliniowości do modelu, co pozwala na uczenie się złożonych zależności między danymi wejściowymi a wyjściowymi.

5.5.1 Funkcje aktywacji w warstwach ukrytych (Hidden Layer Activation Functions):

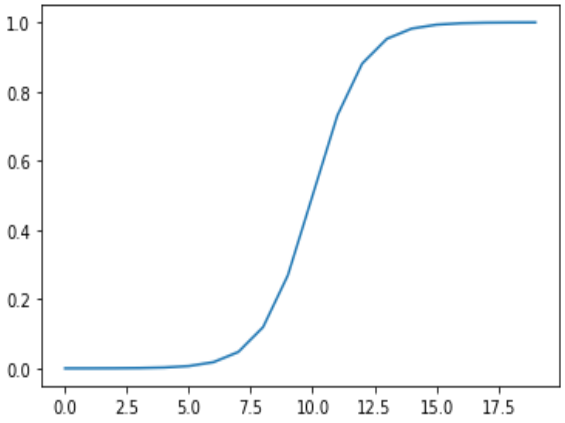
Funkcje aktywacji w warstwach ukrytych sieci neuronowych są nieliniowymi funkcjami stosowanymi do przekształcania sygnałów na wyjściu neuronów w tych warstwach. Oto krótki opis kilku popularnych funkcji aktywacji:

1. **ReLU (Rectified Linear Unit):**



- Najczęściej stosowana funkcja aktywacji w warstwach ukrytych. Jest prosta i efektywna w uczeniu głębokich sieci neuronowych.
- ReLU jest zdefiniowane jako f(x)=max (0, x), co oznacza, że dla wartości dodatnich zachowuje się liniowo, a dla wartości ujemnych jest równe zero.

2. **Sigmoid:**



- Funkcja sigmoidalna przekształca wartości na przedziale (0, 1), co czyni ją odpowiednią do zastosowań, gdzie wynik modelu ma interpretację jako prawdopodobieństwo.
- Jest zdefiniowane jako:

$$f(x) = \frac{1}{1+e^{-x}}$$

- Jest często stosowane w warstwach wyjściowych dla problemów binarnej klasyfikacji.

5.5.2 Funkcje aktywacji w warstwie wyjściowej (Output Activation Functions):

Funkcje aktywacji w warstwie wyjściowej sieci neuronowej są stosowane do przetwarzania sygnałów na wyjściu sieci, które zwykle reprezentują końcowe predykcje modelu. Oto krótki opis kilku popularnych funkcji aktywacji używanych w warstwie wyjściowej:

1. **Sigmoid:**

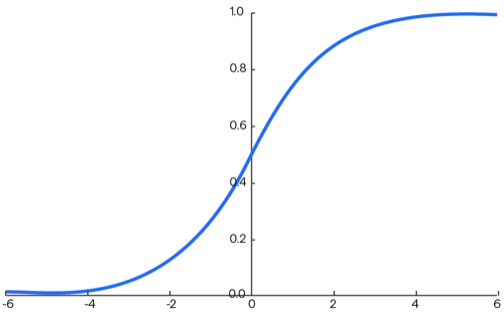
- Jak wspomniano wcześniej, jest często stosowane w warstwach wyjściowych dla problemów binarnej klasyfikacji, gdzie wynik modelu jest interpretowany jako prawdopodobieństwo przynależności do jednej z dwóch klas.

2. **Softmax:**

- Funkcja softmax przekształca wyniki modelu na rozkład prawdopodobieństwa dla wielu klas.
- Jest zdefiniowane jako

$$f(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \text{ dla i = 1...K,}$$

gdzie K to liczba klas



5.6 Optymalizator (Optimizer)

Optymalizator to algorytm używany w uczeniu maszynowym do aktualizacji parametrów modelu w celu minimalizacji funkcji straty. Jego głównym celem jest dostosowanie parametrów modelu tak, aby lepiej dopasować się do danych treningowych i/lub lepiej generalizować na nowe dane. Przykłady najpopularniejszych optymalizatorów:

1. **SGD (Stochastic Gradient Descent):** jest podstawowym optymalizatorem, który aktualizuje parametry modelu proporcjonalnie do negatywnego gradientu funkcji straty, z dodanym czynnikiem uczenia. Jest to iteracyjny proces, w którym parametry są aktualizowane na podstawie pojedynczych lub małych grup próbek.

2. **Adam (Adaptive Moment Estimation):** to zaawansowany optymalizator, który łączy cechy SGD z adaptacyjnym podejściem do wyboru prędkości uczenia się dla każdego parametru. Adam wykorzystuje ruchome średnie momentów gradientów do aktualizacji parametrów, co pomaga w skutecznym dostosowaniu prędkości uczenia się dla różnych parametrów modelu.

5.7 Ulepszanie modelu

1. Dodaj więcej warstw - daj modelowi więcej szans na poznanie wzorców w danych.
2. Dodaj więcej jednostek ukrytych - przejdź z 5 do 10 jednostek ukrytych
3. Dłuższe dopasowanie
4. Zmiana funkcji aktywacji
5. Zmiana szybkości uczenia
6. Zmiana funkcji straty

5.7.1 Train model, Testing loop

1. Przechodzenie do przodu (obejmuje to dane przechodzące przez funkcje forward () naszego modelu) w celu przewidywania danych - zwane również propagacją do przodu.
2. Obliczenie straty (porównanie przewidywań w przód z etykietami prawdy)
3. Optymalizator zero grad
4. Loss backward - poruszanie się wstecz przez sieć w celu obliczenia gradientów każdego z parametrów naszego modelu w odniesieniu do straty (wsteczna propagacja).
5. Krok optymalizatora - użyj optymalizatora do dostosowania parametrów naszego modelu, aby spróbować poprawić stratę (zejście gradientowe).

5.8 Tensory

5.8.1 Operacje wykonywane na tensorach

Operacje wykonywane na tensorach to fundamentalne działania przetwarzania danych w bibliotece PyTorch, umożliwiające manipulowanie kształtem i wymiarami tensorów. Oto krótki opis wybranych operacji:

1. Reshaping (Przekształcanie kształtu):

Reshaping pozwala zmieniać kształt tensora bez zmiany jego danych. Możemy zmieniać tensor z jednym kształtem na inny, pod warunkiem, że liczba elementów

pozostaje taka sama. Na przykład, tensor o kształcie (2, 3) może zostać przekształcony na tensor o kształcie (3, 2) za pomocą operacji reshaping.

2. **Stacking (Łączenie):**

   Operacja stacking pozwala na łączenie wielu tensorów w jeden większy tensor. Możemy stosować stacking wzdłuż istniejącego wymiaru lub na nowym wymiarze, tworząc tensor o większym wymiarze. Na przykład, możemy zastosować stacking do połączenia dwóch tensorów o kształcie (3, 4) w jeden tensor o kształcie (2, 3, 4).

3. **Squeezing (Ściskanie):**

   Operacja squeezing usuwa wymiary o jednostkowej długości, czyli o długości równej 1, z tensora. Jest to przydatne, gdy chcemy zmniejszyć liczbę wymiarów tensora. Na przykład, możemy ścisnąć tensor o kształcie (1, 3, 1, 5) w tensor o kształcie (3, 5).

4. **Unsqueezing (Rozszerzanie):**

   Unsqueezing dodaje nowe wymiary o jednostkowej długości do tensora. Możemy wybrać, w którym miejscu chcemy dodać nowy wymiar. Na przykład, możemy rozszerzyć tensor o kształcie (3, 4) poprzez dodanie nowego wymiaru na początku, tworząc tensor o kształcie (1, 3, 4).

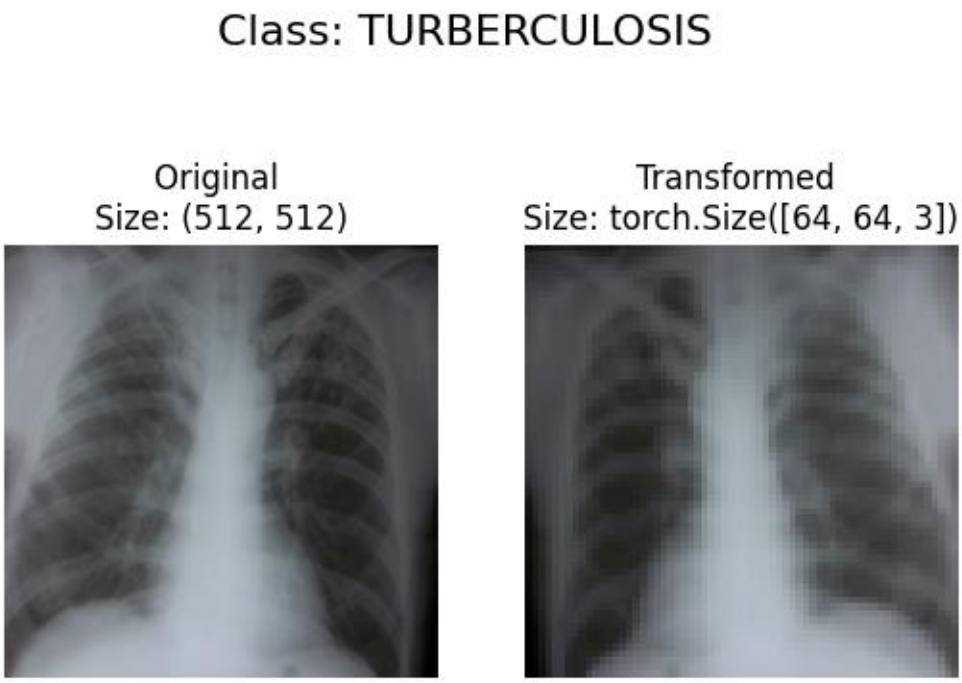
6 Eksperymenty numeryczne

6.1 Opis zbioru danych

Zbiór danych zawiera zróżnicowane obrazy radiologiczne (RTG) płuc, podzielone na zestawy treningowy i testowy, z każdym obrazem przypisanym do jednej z czterech klas:

1. **Płuca zdrowe:** Obrazy płuc osób zdrowych, które służą jako punkt odniesienia do porównań z obrazami pacjentów z podejrzeniem choroby.
2. **Zapalenie płuc:** Obrazy płuc pacjentów cierpiących na zapalenie płuc
3. **Gruźlica:** Obrazy płuc pacjentów zdiagnozowanych z gruźlicą
4. **COVID-19:** Obrazy płuc pacjentów zarażonych wirusem SARS-CoV2 charakteryzujące się różnymi objawami i stopniem nasilenia zmian w płucach.

Zbiór danych RTG zawiera obrazy w formacie plików graficznych, takich jak JPEG lub PNG, które przedstawiają różne cechy i objawy chorób płuc. Obrazy te są wykorzystywane do treningu i testowania modeli uczenia maszynowego w celu rozpoznawania i klasyfikacji chorób płuc na podstawie cech obrazowych. Ważne jest, aby zbiór danych był zrównoważony, co oznacza, że liczba obrazów w każdej klasie jest zbliżona, aby model mógł równomiernie nauczyć się rozpoznawać różne przypadki. Dodatkowo, jakość i dokładność etykietowania danych są kluczowe dla skuteczności modelu w klasyfikacji i diagnozowaniu chorób płuc.



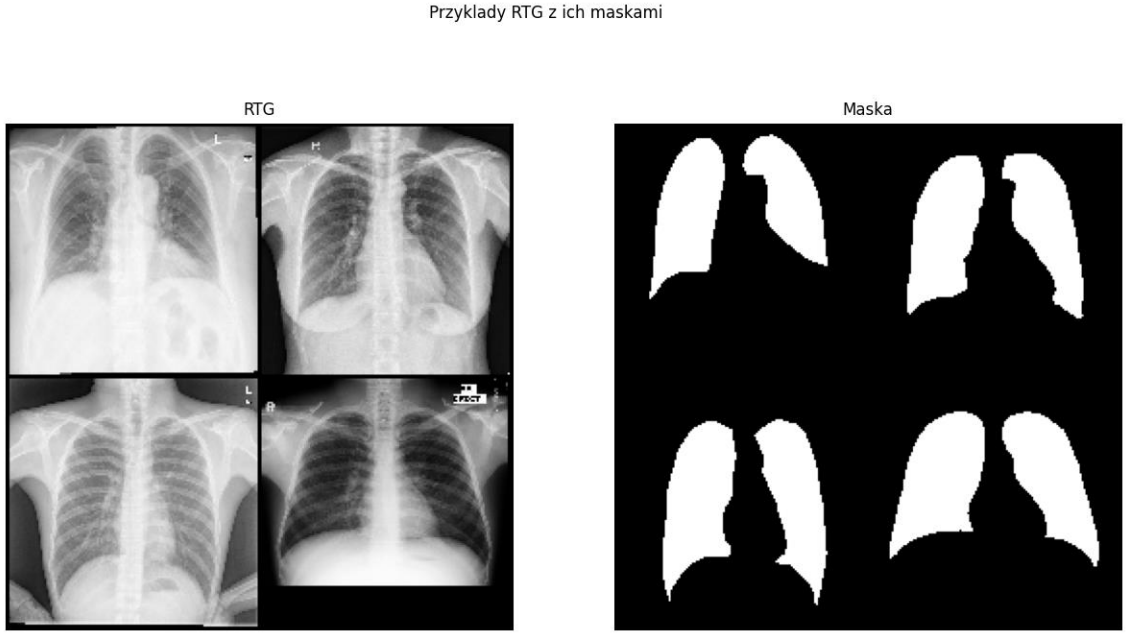
Zbiór danych zawiera zdjęcia rentgenowskie (RTG) płuc wraz z maskami segmentacji. Ten typ zbioru danych jest idealny do treningu i testowania modeli segmentacji obrazów, które mają na celu identyfikację różnych struktur anatomicznych na obrazach RTG płuc. Typowe elementy, które można segmentować na takich obrazach, to np. płuca, oskrzela, tchawica, naczynia krwionośne, a także obszary zmian patologicznych, takich jak guzy, blizny czy płyny.

Struktura takiego zbioru danych to:

1. **Zdjęcia RTG płuc**: Zawierają obrazy rentgenowskie płuc, zwykle w formacie 2D. Mogą być w różnych rozdzielczościach i formatach, ale najczęściej są to obrazy w skali szarości (grayscale), które przedstawiają anatomię klatki piersiowej.

2. **Maski segmentacji**: Są to obrazy o takich samych wymiarach co obrazy RTG, ale zazwyczaj zawierają binarną informację, gdzie obszary zainteresowania (np. płuca) są oznaczone jako białe, a tło jako czarne. Może być kilka klas segmentacji, w zależności od potrzeb (np. płuca, obszary zmian patologicznych). Każde zdjęcie RTG ma odpowiadającą mu maskę segmentacji, która określa, które piksele należą do danej klasy.

3. **Zbiór treningowy, walidacyjny i testowy**: Zbiór jest zazwyczaj podzielony na te trzy części. Zbiór treningowy służy do trenowania modelu, zbiór walidacyjny służy do strojenia parametrów modelu i zapobiegania nadmiernemu dopasowaniu (overfitting), a zbiór testowy służy do oceny wydajności modelu na nowych danych, które nie były używane w trakcie treningu.



Takie zestawy danych są niezwykle cenne dla społeczności medycznej i badawczej, ponieważ umożliwiają rozwój i ocenę skuteczności algorytmów segmentacji obrazów, które mogą wspierać lekarzy w diagnozowaniu i leczeniu chorób płuc.

6.2 Procedury testowe

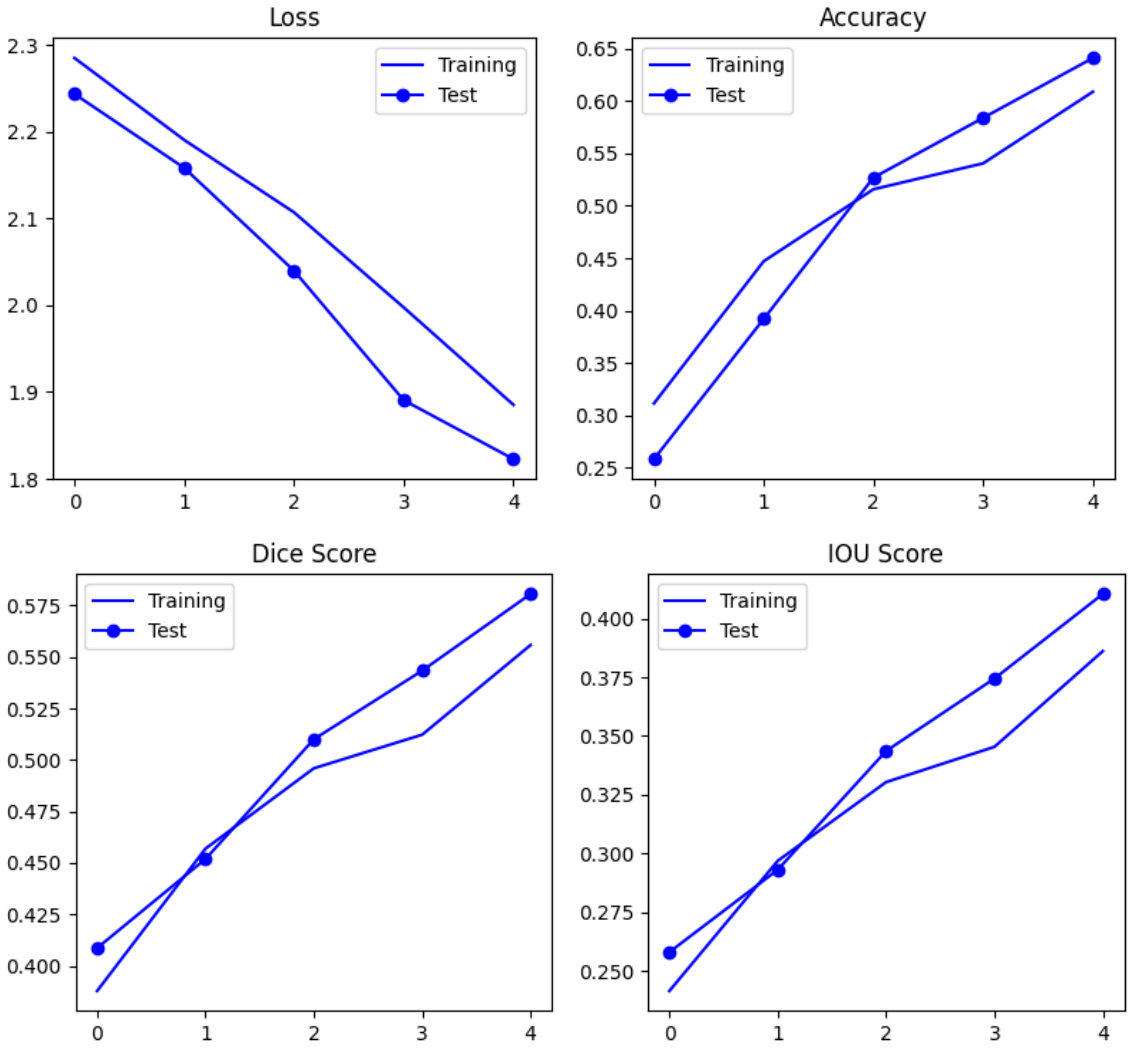
Procedury testowe, które zostały użyte w aplikacji, miały na celu ocenę różnych wskaźników, sprawdzających, jak dobrze model radzi sobie z segmentacją obrazów rentgenowskich (RTG) płuc. Głównymi metrykami oceny są:

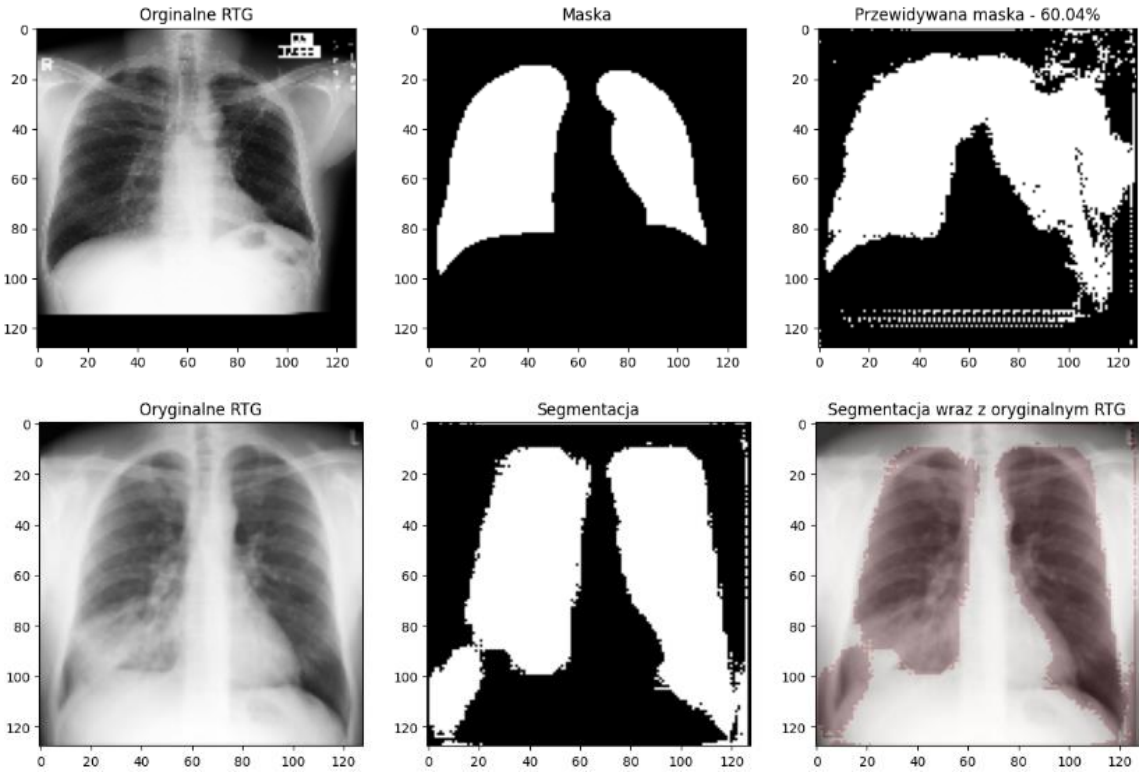
- strata (loss),
- dokładność (accuracy),
- wynik Dice,

- wskaźnik Intersection over Union (IOU).

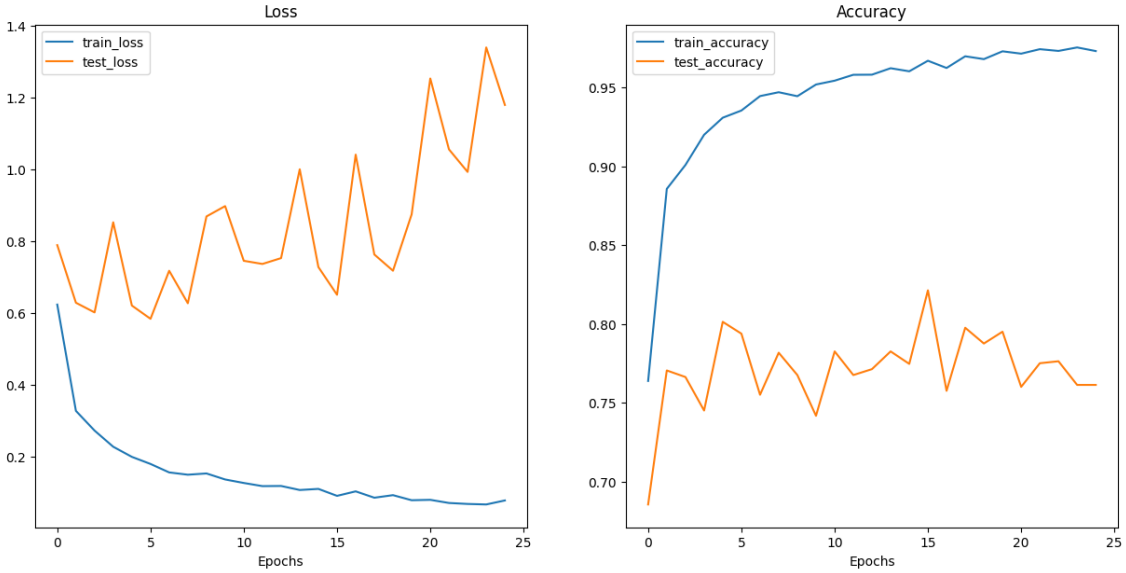
Dodatkowo, opracowaliśmy własną funkcję compare_masks, która umożliwia porównanie dwóch masek segmentacji i ocenę, jak bardzo są one do siebie podobne.

Aby ocenić skuteczność segmentacji w praktyce, został przeprowadzony eksperyment, w którym użyto już wytrenowanego modelu do segmentacji nowego obrazu RTG. Potem predykcja modelu została porównana z rzeczywistą maską segmentacji, patrząc na oba obrazy oraz oryginalny obraz RTG. Ocena skuteczności segmentacji jest dokonywana przez obserwację podobieństwa między predykcją a rzeczywistą maską segmentacji. Procedura ta pozwoliła nam zobaczyć, jak dobrze model radzi sobie z segmentacją w praktyce.





6.3 Zmiana hyperparametrów



6.4 Zmiana funkcji aktywacji, optimizera, funkcji straty

6.5 Wpływ wielkości datasetu na skuteczność

6.6 Problem przeuczenia

6.7 Wpływ architektury sieci na końcowy wynik

Zmniejszanie/ zwiększenie ilości warstw

Zmiana liczby cech „features”

6.8 Inna metoda augmentacji

6.9 Testy na nowych danych

6.10 Analiza wyników

7 Wnioski

8 Podsumowanie

9 Bibliografia

https://poloclub.github.io/cnn-explainer/

https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

https://tinyurl.com/tensorflowPG

10 Spis tabel

11 Spis diagramów