Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **Informatyka Techniczna (ITE)**
Specjalność: **Systemy Informatyki w Medycynie (IMT)**

PRACA DYPLOMOWA
INŻYNIERSKA

Projekt i implementacja algorytmu klasyfikującego raka piersi z zastosowaniem głębokich sieci konwolucyjnych

Paweł Krasicki

Opiekun pracy
Dr inż. Mariusz Topolski

Słowa kluczowe: rak piersi, uczenie maszynowe, sieci neuronowe

WROCŁAW 2023

STRESZCZENIE

W omawianej pracy zaprojektowano i zaimplementowano algorytm CNN klasyfikujący raka piersi na obrazach histopatologicznych. W celu korzystania z algorytmu przez użytkowników końcowych stworzono aplikację z graficznym interfejsem użytkownika umożliwiającą intuicyjną obsługę przetwarzania zdjęć. Rak piersi to najczęściej występujący nowotwór u kobiet, który często grozi śmiercią, ale przy odpowiednio wczesnym zdiagnozowaniu można go całkowicie wyleczyć. Jednym ze sposobów diagnozy jest badanie histopatologiczne, które dostarcza zdjęć pochodzących z mikroskopu. Dzięki rozwojowi sztucznej inteligencji możliwe jest wykrywanie obiektów na obrazach, w tym komórek raka piersi na zdjęciach histopatologicznych. Specjalnie w tym celu powstały sieci konwolucyjne, które potrafią wydajnie analizować i selekcjonować cechy graficzne. Dokument zawiera opis przebiegu prac dotyczących uczenia głębokiego sieci konwolucyjnej *resnet18*, implementacji algorytmu w celu przetwarzania przez sieć zdjęć użytkownika, testowanie parametrów uczenia i stworzonego GUI oraz instrukcję korzystania z graficznego interfejsu użytkownika. Na koniec przedstawione zostały opcje rozwoju aplikacji i wnioski z przeprowadzonych prac.

ABSTRACT

In this paper, a CNN algorithm was designed and implemented to classify breast cancer on histopathological images. In order to use the algorithm by end users, an application with a graphical user interface was created for intuitive operation of photo processing. Breast cancer is the most common cancer in women and is often fatal, but if diagnosed early enough, it can be completely cured. One way of diagnosis is histopathology, which provides pictures taken from a microscope. Thanks to the development of artificial intelligence, it is possible to detect objects in images, including breast cancer cells in histopathological images. Convolutional networks have been created especially for this purpose, which can efficiently analyze and select graphical features. The document includes a description of the workflow for deep learning of the convolutional network *resnet18*, implementation of the algorithm in order for the network to process user images, testing the learning parameters and the created GUI, and instructions for using the graphical user interface. Finally, options for the development of the application and conclusions of the work carried out are presented.

SPIS TREŚCI

WPROWADZNIE

Ludzki mózg od dawna jest obiektem fascynacji i wielu badań. Jedną z jego największych zalet jest inteligencja. Żyjemy w czasach, w których dokonuje się próby upodobnienia komputerów do ludzi, a efekty nazywa się sztuczną inteligencją. W wielu dziedzinach doskonale zastępuje ona człowieka, a w niektórych znacznie go wyprzedza. Z tego powodu jest to jeden z najbardziej popularnych i rozwijanych tematów w dziedzinie nauk komputerowych. Na sztuczną inteligencję składa się uczenie maszynowe, w skład którego wchodzi uczenie głębokie. Dotyczy ono wielowarstwowych sieci neuronowych, które między innymi realizują nieliniowe transformacje, generują cechy diagnostyczne i finalnie klasyfikują dane.

Głęboka sieć konwolucyjna to jedna ze struktur, którą wykorzystuje się na szeroką skalę do rozpoznawania obrazów. Analiza obrazów to czynność często powtarzająca się w medycynie. Na podstawie zdjęć rentgenowskich, obrazów USG, czy obrazów histopatologicznych, lekarze są w stanie rozpoznać różne stany organizmu, w tym choroby. Sieci konwolucujne, zwane inaczej splotowymi, to idealne narzędzie do zautomatyzowania tej pracy i zapobiegania błędom. Pozwalają one również skrócić czas analizy i zredukować nakłady ludzkie. Dzięki tym zaletom sieci splotowe zyskały uznanie i są powszechnie wykorzystywane m.in. do rozpoznawania raka piersi.

Niniejszy dokument stanowi opis implementacji algorytmu i tworzenia aplikacji do wykrywania raka piersi z wykorzystaniem głębokich sieci konwolucyjnych, należących do algorytmów głębokiego uczenia. Za pomocą aplikacji można wybrać zdjęcia do analizy, a następnie uzyskać wynik mówiący, czy znajduje się na nich rak piersi, czy są one wolne od komórek nowotworowych. Całość akcji wykonywana jest z poziomu graficznego interfejsu użytkownika. Algorytm został wytrenowany na podstawie publicznie dostępnego zbioru obrazów histopatologicznych. Aplikacja zyskała nazwę *Breast Cancer Detection*. Do jej poprawnego działania wymagane jest posiadanie dostarczanego wraz z nią pliku *.pth* zawierającego specjalnie wytrenowany model.

Należy pamiętać, że aplikacja jedynie wspomaga pracę lekarza, a nie go całkowicie zastępuje. Wzorcowym środowiskiem przedstawianej aplikacji jest klinika lub szpital, w którym zgromadzone zdjęcia są poddawane wstępnej analizie z użyciem aplikacji, a następnie wyniki przekazywane są odpowiedniemu specjaliście z zakresu patomorfologii, który może przyjrzeć się niepewnym wynikom i zdjęciom, które zaklasyfikowano jako zawierające raka piersi.

CEL PRACY

Celem pracy jest zrealizowanie projektu aplikacji, w której zostanie zaimplementowany algorytm klasyfikujący raka piersi na podstawie dostarczonych danych, wykorzystując do tego sieci konwolucyjne. Aplikacja ta powinna umożliwiać przekazanie własnego obrazu lub zestawu danych do analizy oraz zwracać wynik w postaci decyzji o wystąpieniu raka piersi.

ZAKRES PRACY

Sekcje niniejszego dokumentu rozpoczyna wstęp w postaci teorii z zakresu raka piersi, głębokiego uczenia i sieci konwolucyjnych oraz ich działania. Następnie przedstawiane jest środowisko eksperymentalne, czyli użyte technologie i wykorzystywany zbiór danych, wraz z jego wstępnym przygotowaniem, a następnie plan aplikacji w postaci wymagań funkcjonalnych i niefunkcjonalnych, diagramu przypadków użycia, planowanych widoków, planowanego uczenia głębokiego oraz schematu logicznego działania aplikacji. Kolejnym etapem pracy jest implementacja algorytmu realizowana w języku *Python*, w tym sposób trenowania algorytmu i późniejszej klasyfikacji zdjęć. W tej sekcji znajduje się również graficzny interfejs użytkownika w postaci zrzutów ekranu i wyjaśnień co się na nich znajduje. Kolejna sekcja dotyczy testowania aplikacji, wliczając w to eksperymentalny dobór parametrów trenowania algorytmu oraz końcowe testowanie działania wszystkich funkcjonalności aplikacji i wyników występowania raka piersi na podstawie danych testowych, niewidzianych nigdy wcześniej przez algorytm. Ostatnia sekcja przedstawia instrukcję użytkowania gotowego produktu. Dodatkowo przedstawiony został również sposób uruchomienia aplikacji i wskazówki korzystania. Po wszystkim wyciągnięto wnioski i podsumowano dokonane działania.

1. WSTĘP TEORETYCZNY

Rak piersi to najczęściej występująca postać nowotworu złośliwego u kobiet [27]. Pojawia się on w wyniku niekontrolowanego namnażania się wadliwych komórek nabłonkowych gruczołu piersiowego, które podczas wczesnych faz rozwoju tworzą guzek, który może być wyczuwalny przez pacjentkę. Zazwyczaj powstaje on w górnym, bocznym (zewnętrznym) fragmencie piersi, czyli w ćwiartce najbliżej dołu pachowego. Przy zlekceważeniu problemu może dojść do powstawania przerzutów, czyli przeniesienia się komórek rakowych na inne narządy i tworzenie tam kolejnych ognisk nowotworu [16]. Zazwyczaj wtedy jest już za późno na całkowite wyleczenie choroby. Nowotwór ten występuje najczęściej u kobiet po 50 roku życia [9]. Odpowiednio wczesne zdiagnozowanie i podjęcie leczenia pozwala całkowicie go wyeliminować. Dlatego tak ważna jest profilaktyka. Ryzyko wystąpienia tej choroby można również zmniejszyć poprzez redukcję używek, dbanie o właściwą wagę ciała i aktywność fizyczną, unikanie stresu i promieniowania jonizującego, jak również posiadanie dzieci przed 35 rokiem życia. Czynnikami wpływającymi na powstawanie raka piersi są [14, 27]:

— geny,
— dieta,
— nadwaga,
— brak aktywności fizycznej,
— niekarmienie piersią,
— bezdzietność,
— ekspozycja na działanie promieniowania jonizującego,
— czynniki hormonalne,
— stres.

1.1. WYKRYWANIE RAKA PIERSI

Najważniejszym czynnikiem warunkującym wyleczenie jest odpowiednio wczesne wykrycie problemu. Pomaga w tym profilaktyka w postaci samokontroli przeprowadzanej przez kobietę oraz częste badania, mające na celu postawienie diagnozy występowania komórek nowotworowych. Główny cel badania to stwierdzenie, czy w ogóle występuje choroba nowotworowa, a następnie przystępuje się do dokładniejszej analizy w postaci ustalenia typu i zaawansowania schorzenia. Badania te powinno wykonywać się okresowo, niezależnie od stanu zdrowia, szczególnie dotyczy to kobiet po 30 roku życia. Polskie

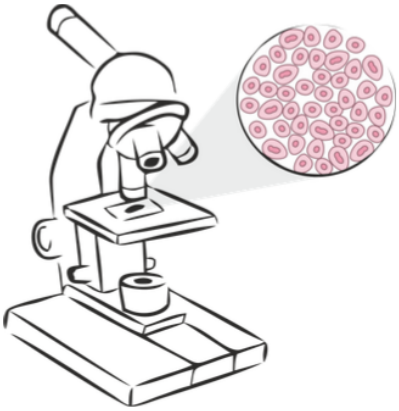
Towarzystwo Ultrasonograficzne zaleca wykonywanie badań kontrolnych raz do roku. Przeprowadzając samokontrolę w postaci oceny wizualnej i dotykowej, można wykryć następujące objawy:

— wyczuwalny guz w obrębie gruczołu piersiowego,
— wyczuwalny guz w okolicach pach,,
— zgrubienie lub obrzęk części piersi,
— widoczne zaczerwienienie w okolicy brodawki,
— łuszcząca się skóra w okolicy brodawki,
— wydzielina z sutków inna niż mleko,
— wciągnięcie brodawki lub ból w jej okolicy,
— poszerzenie żył skóry piersi,
— zmiana rozmiaru lub kształtu piersi,
— bolesność piersi.

Najpopularniejszą metodą profilaktyki w Polsce, niebędącej samokontrolą, jest prześwietlenie piersi promieniowaniem rentgenowskim. Jest to szybkie, proste i niebolesne badanie, które nie wymaga również szczególnego przygotowania. Zyskało ono nawet nazwę własną pod tytułem mammografia [15]. Badanie to ma największy sens u pacjentek powyżej 50 roku życia. Dla młodszych kobiet badanie to nie jest aż tak skuteczne ze względu na inną budowę piersi i zaleca się wykonanie USG, czyli ultrasonografii. Obydwa wspomniane rodzaje diagnozowania są skuteczne, ale nie są nieomylne.

Do najbardziej skutecznych metod o największej pewności zalicza się obecnie biopsję gruboigłową, która pozwala również na przeanalizowanie dodatkowych parametrów, m.in. złośliwości i szybkości namnażania. Jest to niestety metdota inwazyjna, wymagająca wkłucia się w ciało. Jej skuteczność wynika z fizycznego pobrania materiału biologicznego z tkanki piersiowej kobiety i jego analizy pod mikroskopem. Wkłucie igły następuje w podejrzany obszar tkanki piersiowej, która następnie jest zasysana do specjalnego rowka w igle, po czym element zewnętrzny igły ścina fragment tkanki, stający się materiałem badawczym [24, 8]. Generowany przez mikroskop obraz nazywany jest histopatologicznym, a za jego analizę odpowiada patomorfolog. Z uwagi na brak stuprocentowej pewności wyników badań USG i rentgenowskich, badanie histopatologiczne jest kluczowe i bez jego wyników nie można rozpocząć leczenia onkologicznego. Pamiętając jednak o wymaganiach, jakie za sobą niesie, nie jest ono wykorzystywane do regularnej profilaktyki. Naturalną koleją badań jest badanie palpacyjne, ultrasonografia lub mammografia, a dopiero później biopsja, kiedy wcześniejsze metody wykryją jakiekolwiek podejrzenia. Obrazy histopatologiczne mogą być zapisywane i magazynowane w formie cyfrowej. Pozwala to na powrót do analizy w innym terminie, przekazanie próbek innemu lekarzowi, jak również cyfrową obróbkę zdjęć.

Dzięki postępowi rozwoju technologicznego diagnostyka onkologiczna zyskuje nowe narzędzia do zautomatyzowania wykrywania raka oraz przyspieszenia całego procesu. Komputery są wykorzystywane nie tylko do przeprowadzenia procesu badań, ale coraz częściej przetwarzają one powstające obrazy i na podstawie ich analizy są w stanie podejmować odpowiednie decyzje [28]. Zdjęcia histopatologiczne mogą być użyte do analizy komputerowej za pomocą algorytmów głębokiego uczenia, w tym sieci konwolucyjnych, które odpowiednio wytrenowane są w stanie ocenić zawartość obrazu pod kątem występowania raka piersi i zwrócić wynik, czy rak występuje oraz jaką pewność ma algorytm co do tej decyzji. Stworzona i opisywana w tej pracy aplikacja bazuje na obrazach histopatologicznych, które poddawane są analizie z użyciem sieci CNN [5]. Można więc wysnuć wniosek, że aplikacja wykonuje i poniekąd zastępuje pracę patomorfologa. W dodatku automatyzacja z użyciem CNN wpływa korzystnie na szybkość klasyfikacji, zmniejsza ilość wymaganych lekarzy w procesie i zapewnia bardziej obiektywne decyzje, stąd przyszłość tej technologii wydaje się obiecująca.



Rys. 1.1. Nowotwór pod mikroskopem

Rys. 1.2. Badanie histopatologiczne

1.2. UCZENIE MASZYNOWE

Uczenie maszynowe [7, 6] jest obszarem sztucznej inteligencji dedykowanym algorytmom, które poprawiają się wraz z dostarczaniem im nowych danych. Algorytmy te bazują na budowaniu modelu matematycznego z dostarczonych danych uczących, będących zbiorem treningowym o zamierzonej charakterystyce. Następnie algorytmy te są w stanie dokonywać predykcji etykiet nowo dostarczonych, nieznanych wcześniej danych. Etykiety te są zazwyczaj poszukiwanymi nazwami klas, których instancje są badane na podstawie różnych cech. Inteligencja polega na rozpoznaniu nowego, innego obiektu niż znany dotychczas, na podstawie zdobytej wiedzy, bez bycia w pełni zaprogramowanym przez człowieka pod kątem wybierania odpowiedniej klasy danego obiektu. Nierozłącznym elementem

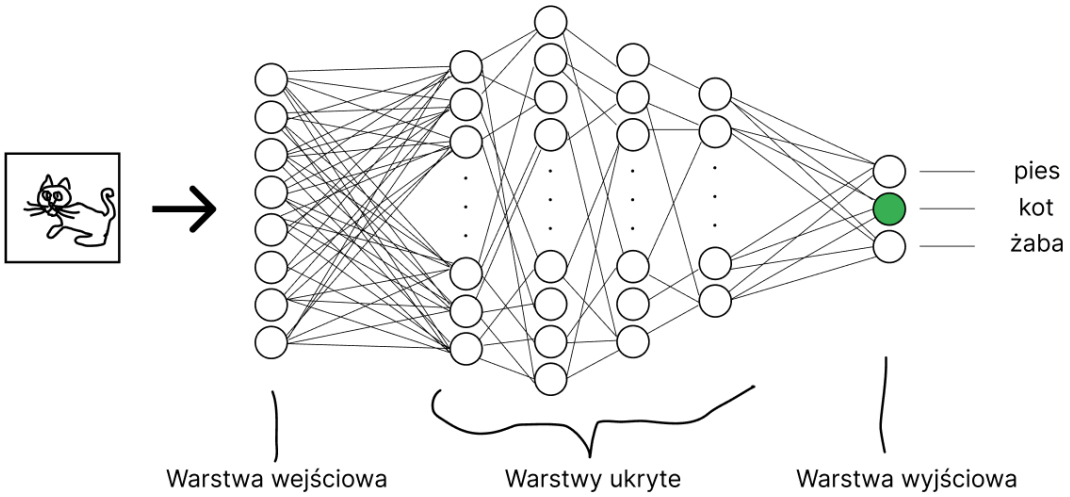
uczenia maszynowego jest klasyfikacja [2], która polega na przypisaniu badanych obiektów do odpowiednich klas poprzez analizę cechy tych obiektów i wnioskując na podstawie wcześniej dostarczonych danych treningowych. Klasyfikacja binarna to podział obiektów na dwie klasy, z których jedna zawiera daną cechę, a druga jej nie posiada. Przykładem jest podział zdjęć na zawierające komórki raka piersi i niezawierające tych komórek. Istnieje również klasyfikacja wieloklasowa, która polega na przypisaniu danemu obiektowi jednej z zadeklarowanych etykiet. Przykładem jest przydzielenie ptaka do danego gatunku lub nadanie wskazanemu owocowi konkretnej nazwy.

W celu przeprowadzenia klasyfikacji potrzebne jest poznanie badanego zbioru danych i odpowiednie nauczenie modelu wykrywania zadanych etykiet klas. Następnie wytrenowany model może być używany do określenia klasy nowego obiektu. Aby nauczyć model rozpoznawania klas, potrzebne jest posiadanie odpowiedniego zbioru danych, który będzie można podzielić na dane treningowe i dane testowe. Dane testowe powinny stanowić znaczną mniejszość, zazwyczaj od 15% do 30%. Z ich wykorzystaniem będzie można sprawdzić poprawność nauki algorytmu. Dane uczące powinny być odpowiednio zróżnicowane, ale również zbalansowane. Często występują problemy w tych aspektach, dlatego stosuje się na przykład obroty zdjęć w celu zróżnicowania instancji treningowych oraz różnego rodzaju metody radzenia sobie z niezbalansowanymi danymi, a czasem też z brakującymi wartościami cech. Dane treningowe muszą zawierać identyfikatory w postaci etykiet klas do jakich należą, aby algorytm mógł na tej podstawie nauczyć się wnioskowania podczas uczenia nadzorowanego.

1.3. UCZENIE GŁĘBOKIE

Uczenie głębokie (ang. deep learning) [25, 18, 10] to gałąź uczenia maszynowego zajmująca się sieciami neuronowymi. Proces uczenia nazywany jest głębokim ze względu na liczne warstwy sztucznych sieci neuronowych, dzielące się na wejściowe, ukryte i wyjściowe. Im więcej warstw posiada sieć, tym jest głębsza. Każda z warstw przekształca dane wejściowe w informacje, które przekazywane są odpowiednio kolejnej warstwie i służą jej jako dane wejściowe. Informacje te wykorzystywane są do wykonania predykcji [23, 4]. Schemat wykorzystania głębokich sieci neuronowych zobrazowano na poglądowym rysunku 1.3. Wśród najpopularniejszych aktualnie rodzajów sieci, możemy wyróżnić:

— DNN - głębokie sieci neuronowe,
— CNN - głębokie sieci konwolucyjne,
— RNN - głębokie sieci rekurencyjne,
— LSTM - długotrwała pamięć krótkotrwała,
— DBM - głęboka sieć przekonań.



Rys. 1.3. Logika działania głębokiej sieci neurnowej

W obrębie uczenia głębokiego możemy wyróżnić uczenie nadzorowane i nienadzorowane. Uczenie nadzorowane polega na trenowaniu algorytmu zestawem danych zawierającym właściwe etykiety. Dane wejściowe są wprowadzane wraz ze spodziewanym wyjściem. Za przykład posłuży wprowadzanie zdjęcia wraz z informacją, czy znajduje się na nim rak piersi, czy też nie. W przypadku uczenia nienadzorowanego mamy do czynienia z wykorzystaniem wyłącznie danych wejściowych, bez określonych wyjść. Takich danych bez etykiet jest znacznie więcej we wszechświecie, dlatego algorytmy te również zyskują na popularności. W przedstawianej pracy zastosowano uczenie nadzorowane z wykorzystaniem sieci CNN.

Rozszerzeniem uczenia maszynowego jest samodzielne tworzenie i dobieranie cech obiektów, które są kluczowe do ich odróżniania. Dzięki temu model może zostać lepiej dopasowany do danych, na których ma pracować. Naturalnym wyróżnikiem uczenia głębokiego jest też ilość warstw sieci, wśród których muszą znaleźć się przynajmniej dwie warstwy ukryte.

Uczenie głębokie jest powszechnie wykorzystywane do przetwarzania przede wszystkim danych, których analiza jest pracochłonna lub prawie niemożliwa konwencjonalnymi metodami. Do danych tego typu należą dźwięki, obrazy i filmy. Z uwagi na specyfikę tego typu danych, proces nauki sztucznej sieci neuronowej jest niezwykle czasochłonny. Mowa o czasie, który może być mierzony w dniach, a nawet tygodniach. Wielokrotne mnożenie pokaźnej liczby macierzy wymaga na tyle dużej mocy obliczeniowej, że proces ten jest najczęściej wspierany przez karty graficzne mające znacznie więcej rdzeni niż główny procesor komputera. Wykonanie akceleracji obliczeniowej umożliwiają m.in. sterowniki CUDA, które dostarcza producent kart graficznych Nvidia.

1.3.1. Uczenie transferowe

W przypadku głębokich sieci neuronowych niejednokrotnie stosuje się uczenie transferowe (ang. transfer learning) [29], czyli wykorzystanie wiedzy zdobytej przez jedną sieć w celu utworzenia i nauczenia drugiej. Sprowadza się to do wykorzystania wag wstępnie wytrenowanego modelu przez inną sieć. Podejście to sprawdza się dobrze w przypadku analizy obrazów, ponieważ obrazy zawierają wiele cech wspólnych takich jak krawędzie, czy jasność. Model sieci przygotowany uczeniem transferowym następnie doszkala się z użyciem obrazów zawierających konkretne obiekty, które algorytm ma rozpoznawać. Uczenie transferowe nie tylko pozwala skrócić proces uczenia, ale wpływa też korzystnie na otrzymywane wyniki.

1.3.2. Propagacja wsteczna

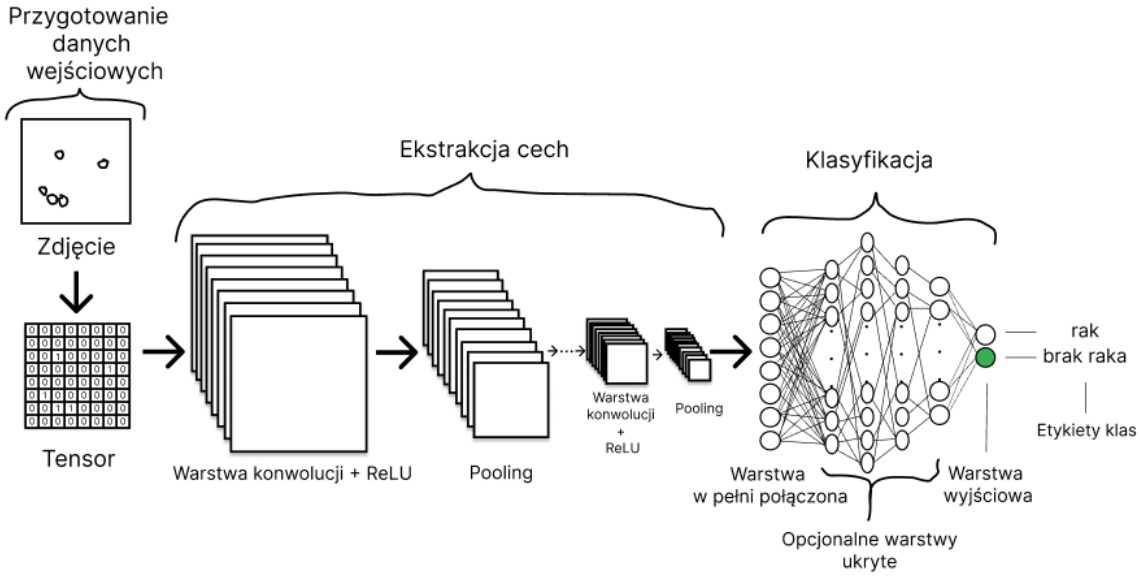
W procesie uczenia głębokiej sieci neuronowej niezwykle istotne jest wykorzystanie propagacji wstecznej (ang. backpropagation) [30], czyli metody aktualizowania wartości wag połączeń węzłów warstw sieci, mającej za zadanie korektę otrzymywanych rezultatów przy wprowadzaniu konkretnych danych wejściowych. To właśnie propagacja wsteczna odpowiada za faktyczne uczenie się danego modelu, ponieważ wpływa na poprawę predykcji dokonywanych przez model.

1.4. SIECI CNN

Sieci konwolucyjne [21] zwane również sieciami splotowymi powstały w celu analizy złożonych i trudnych do przetworzenia danych m.in. obrazów pod kątem rozpoznawania na nich obiektów [19, 3, 32]. Zawierają one specjalne warstwy realizujące operację konwolucji, które stanowią potężne narzędzie do wydobywania cech z badanych zdjęć. Warstwy te zawierają filtry nazywane również kernelami, które generują mapy cech. Po operacji splotu tensor wyjściowy poddawany jest działaniu nieliniowej funkcji aktywacji najczęściej *ReLu* 1.1. Oprócz tego sieci CNN zawierają warstwy pooling, które redukują rozmiar tensorów, a co za tym idzie wymagane moce obliczeniowe oraz uogólniają cechy reprezentowane przez sąsiednie grupy pikseli. Tensory w pewnym momencie są zamieniane na postać wektorową, która stanowi początek układu w pełni połączonego składającego się z warstw neuronów pozwalających na dokonanie klasyfikacji. Ilość warstw konwolucji i łączenia przed przejściem do warstw w pełni połączonych wpływa na zdolności predykcyjne sieci i stanowi główny wyróżnik między modelami sieci CNN. Na rysunku 1.4 przedstawiono schemat budowy sieci konwolucyjnej. Do powszechnie znanych modeli CNN możemy zaliczyć AlexNet, LeNet-5, GoogLeNet, ResNet i VGGNet. Zmienne charakterystyczne dla sieci CNN to:

— liczba warstw konwolucji,

— liczba warstw łączenia,
— nieliniowa funkcja aktywacji,
— rozmiar filtra,
— liczba filtrów,
— wielkość kroku (ang. stride) przesuwania filtra,
— wielkość kroku (ang. stride) warstwy łączenia,
— rozmiar warstwy łączenia (ang. pooling),
— sposób działania warstwy łączenia.



Rys. 1.4. Schemat budowy i działania sieci CNN

Dużym krokiem rozwojowym sieci CNN było powstanie w 2010 roku konkursu ImageNet large scale visual recognition challenge (ILSVRC) polegającego na wyłonieniu najlepszej sztucznej sieci neuronowej pod kątem klasyfikacji zbioru danych ImageNet. W 2012 roku zespół Alex Krizhevsky zdeklasował rywali, tworząc sieć CNN o nazwie AlexNet, która potrafiła rozpoznawać zawartość obrazów znacznie szybciej i dokładniej niż sieci innych drużyn, które nie były sieciami CNN. Kolejne lata konkursu przynosiły nowych zwycięzców i były to już tylko sieci CNN, kolejno m.in. ZFNet, VGGNET, GoogLeNet i ResNet. W omawianej pracy wykorzystano sieć ResNet, a jej dokładny opis znajduje się w dalszej części tego rozdziału.

Zwycięskie modele sieci noszą tytuł *State of the Art*. Dzięki uczeniu transferowemu możliwe jest wykorzystanie wytrenowanych modeli tych sieci w indywidualnych projektach. Modele *State of the Art* można również douczyć, dostosowując je do posiadanego zbioru danych i konkretnego celu, dzięki czemu końcowy model może być spersonalizowany, ale bazować na światowym wzorze. Dostrajanie (ang. fine-tuning) modelu CNN powstałego w wyniku uczenia transferowego jest powszechnie stosowane i ma na celu redukcję przetrenowania. Ucząc sieć CNN od podstaw, kiedy posiadany zbiór danych nie jest wystarczająco

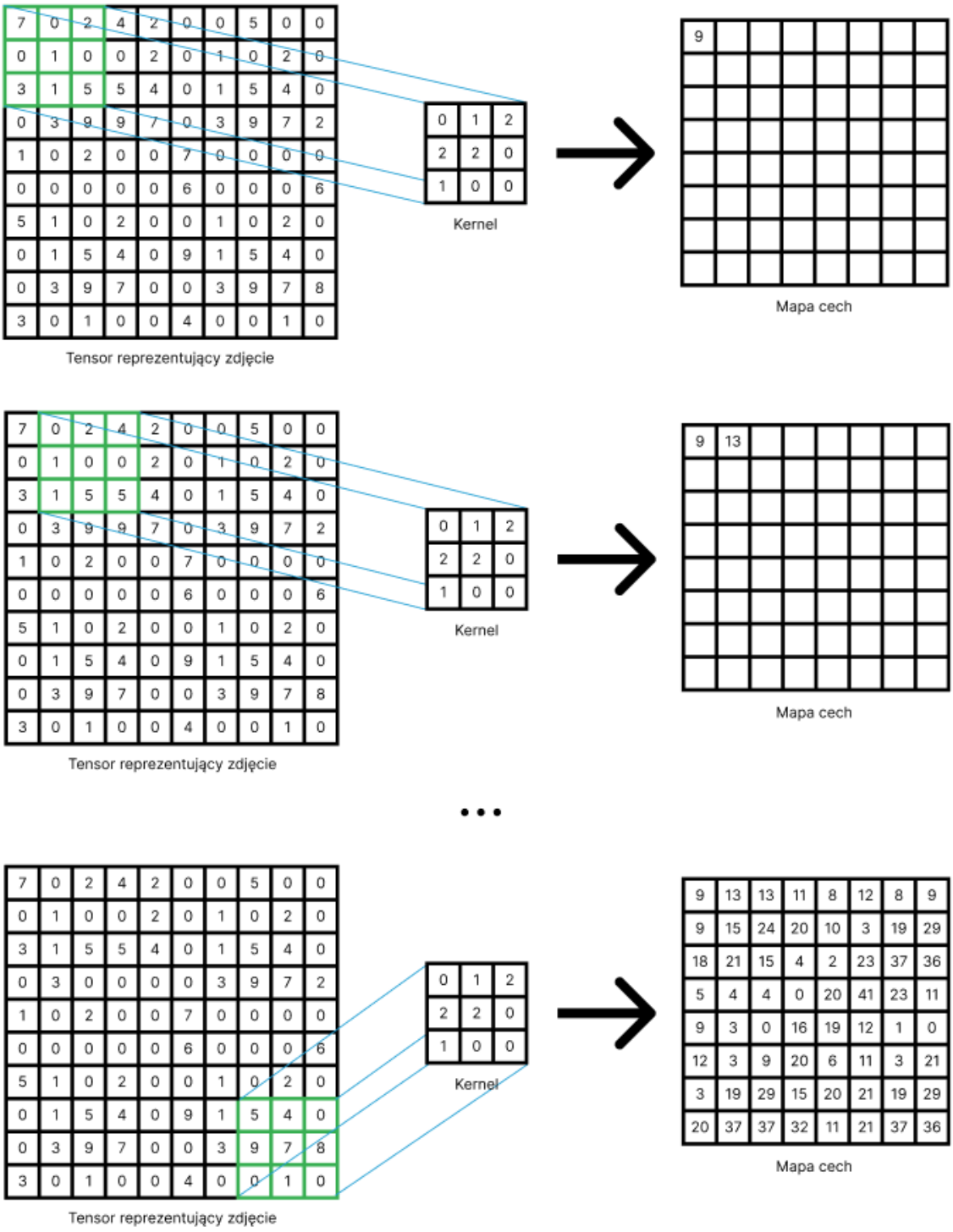
duży i różnorodny łatwo o zbytnie dopasowanie modelu do danych treningowych, a dostrajanie modelu trenowanego na ogromnym zbiorze zdjęć ImageNet niweluje ten problem. W dodatku już na starcie wagi połączeń są dobrane lepiej niż losowo, więc proces uczenia może być krótszy przy zachowaniu lepszych wyników. Możliwe jest również wykorzystanie dostępnych modeli sieci splotowych tylko do ekstrakcji cech (ang. feature extraction) bez wykonywania propagacji wstecznej na warstwach modelu bazowego. Ma ono zastosowanie w przypadku małych i podobnych zbiorów danych do tych, które były wykorzystywane do uczenia modelu bazowego.

1.4.1. Kernel i warstwa konwolucji

Podstawową warstwą sieci CNN jest warstwa konwolucji, której głównym narzędziem są kernele, czyli filtry odpowiedzialne za szukanie i wyodrębnianie odpowiednich cech z obrazów. Kernele to dwuwymiarowe macierze, które poddawane są matematycznej operacji splotu z kolejnymi fragmentami tensora wejściowego reprezentującego zdjęcie. Fragmenty te są wycinkami obrazu będącymi dwuwymiarowymi macierzami liczb o rozmiarze odpowiadającym rozmiarowi filtra. Filtry są następnie przesuwane o pewną liczbę jednostek w celu przeliczania ich z kolejnymi fragmentami zdjęcia, realizując operację konwolucji. Liczbę jednostek przesunięcia określa parametr kroku (ang. stride). Kernele zawierają wartości liczbowe, które są dobierane i dostrajane podczas trenowania sieci. Wagi połączeń przypisane są do wartości filtra, który obsługuje całe zdjęcie. Wag jest więc tyle ile liczb w filtrze, czyli znacznie mniej niż pikseli na zdjęciu, co jest ogromną zaletą sieci CNN, znacznie zwiększającą wydajność przetwarzania danych. Kernele wykrywają początkowo krawędzie, kolory i inne podstawowe parametry, a z biegiem kolejnych warstw coraz bardziej skomplikowane wzory generując przy tym mapy cech. Rysunek 1.5 jest wizualizacją działania filtra podczas operacji konwolucji. Warstwa konwolucji może zawierać wiele filtrów. Po każdym wykonaniu splotu danym kernelem obraz wyjściowy w postaci mapy cech jest pomniejszany względem obrazu wejściowego. Dodatkowym kłopotem jest nierównomierna częstość rozpatrywania poszczególnych liczb w tensorze. Liczby na krawędziach są rozpatrywane tylko raz, a te leżące bliżej środka macierzy wielokrotnie poddawane są działaniu filtra. Aby temu zapobiec, stosuje się padding polegający na stworzeniu ramki wokół tensora wyjściowego składającej się z zer. Dodatkowo pozwala to zachować oryginalny wymiar tensora. Rysunek 1.6 przedstawia zasadę działania padding.

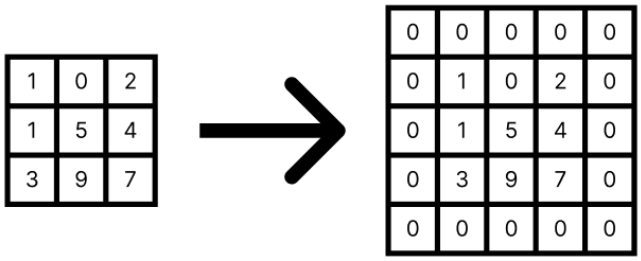
1.4.2. ReLU

Po wykonaniu operacji konwolucji następuje przekazanie map cech do nieliniowej funkcji aktywacji, zazwyczaj *ReLU*, która wprowadza do sieci nieliniowość, dzięki czemu zapobiega się występowaniu przetrenowania. W przypadku *ReLU* dokonuje się tego poprzez zamianę wszystkich negatywnych liczb na 0 i pozostawienie pozostałych bez zmian. Wzór 1.1 prezentuje działanie funkcji *ReLU*.



Rys. 1.5. Operacja konwolucji z wykorzystaniem kernela o rozmiarze 3x3 i stride 1

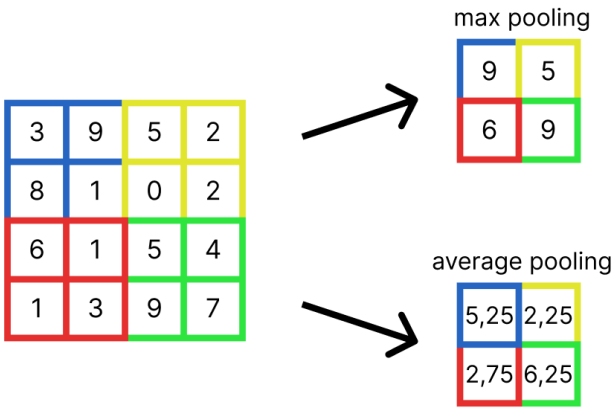
$$y(x) = \begin{cases} x & \text{dla } x \geq 0 \\ 0 & \text{dla } x < 0 \end{cases} \tag{1.1}$$



Rys. 1.6. Prezentacja działania padding

1.4.3. Pooling

Drugą charakterystyczną warstwą CNN jest warstwa pooling, która wykonuje operację redukcji rozmiaru tensora poprzez łączenie sąsiadujących wartości, z zachowaniem kluczowych cech wykrytych w warstwach konwolucji. Operacja łączenia dokonywana jest poprzez wybranie największej lub średniej wartości z liczb znajdujących się w dwuwymiarowej macierzy o zadanym rozmiarze, która stanowi wycinek tensora wejściowego. Przykładowo dla obszaru 2x2 wybierana jest jedna z czterech liczb, która jest największa i zapisywana jest ona na macierzy wyjściowej. Z każdym krokiem jest brany pod uwagę inny obszar tensora wejściowego, aż do przeanalizowania go w całości. Przemieszczanie się realizowane jest identycznie jak w warstwie konwolucji i określane jest przez wielkość kroku (ang. stride). Na rysunku 1.7 zaprezentowano strategię wybierania wartości maksymalnych i średnich. Najczęściej stosuje się *max pooling*.

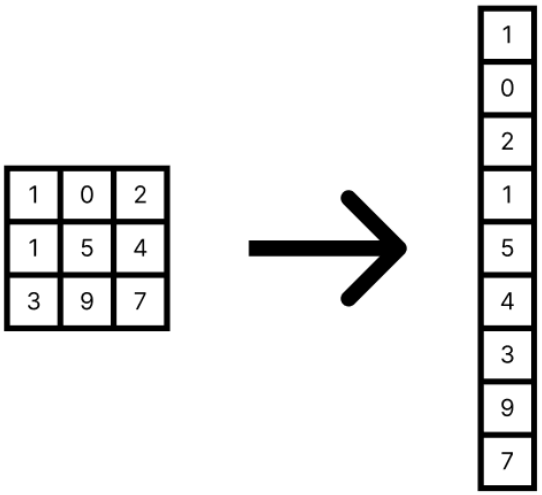


Rys. 1.7. Pooling o rozmiarze 2x2 i stride 2

1.4.4. Flattening

Po przeprowadzeniu danych wejściowych przez kilka warstw konwolucji i pooling dane muszą trafić na liniową warstwę neuronów, która umożliwi wykonanie procesu klasyfikacji. Spłaszczanie (ang. flattening) tensora do jednego wymiaru zostało zaprezentowane na

rysunku 1.8. Zazwyczaj po liniowej warstwie powstałej w wyniku operacji spłaszczania występuje jeszcze w pełni połączona warstwa (ang. fully connected), w której wszystkie neurony połączone są ze wszystkimi neuronami poprzedniej warstwy i warstwy wyjściowej, zawierającej tyle neuronów ile etykiet klas ma być rozpoznanych. Do klasyfikacji najczęściej używana jest funkcja *softmax* przetwarzająca aktywacje neuronów ostatniej warstwy w wyniki w postaci predykcji.

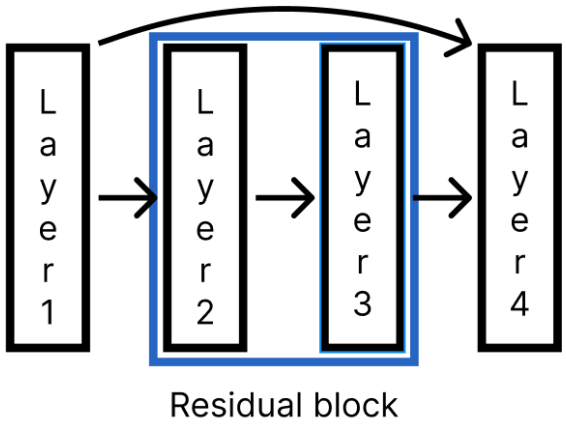


Rys. 1.8. Flattening

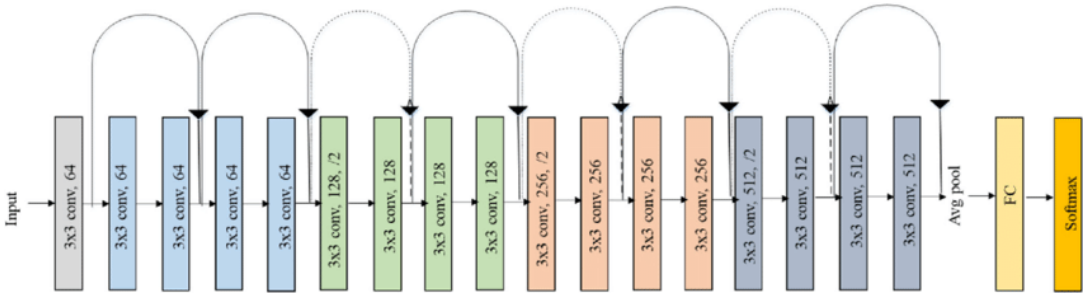
1.4.5. ResNet

ResNet [31, 26] to sieć CNN zaprezentowana po raz pierwszy w 2015 roku przez pracowników firmy Microsoft w artykule Deep Residual Learning for Image Recognition [12], która wygrała konkurs przetwarzania zbioru obrazów ImageNet. Różni się ona od podstawowej konstrukcji CNN omawianej dotychczas specjalnym dodatkiem w postaci bloków resztkowych (ang. residual blocks), które zaprezentowano na poglądowym rysunku 1.9. W sieci tej występuje dodatkowe połączenie między warstwami niewystępującymi bezpośrednio po sobie. Warstwy pomiędzy dodatkowym połączeniem nazywane są blokiem resztkowym i są to dwie lub trzy warstwy konwolucji o rozmiarze filtra 3x3. Sieć składa się z wielu bloków resztkowych i warstw pooling między nimi wraz z nieliniowością ReLU. Podejście to zastosowano z uwagi na brak poprawy wyników poprzez dokładanie kolejnych warstw w dotychczasowych sieciach. Bloki resztkowe rozwiązują ten problem i pozwalają sieci składać się nawet z 152 warstw, które przekładają się na faktyczną poprawę zdolności predykcyjnych. ResNet to pierwsza sieć, która przekroczyła głębokość 100 warstw. Występuje wiele odmian sieci ResNet z różną liczbą warstw podawaną w nazwie modelu, przykładowo *resnet18*. Architekturę sieci *resnet18* przedstawiono na rysunku 1.10, gdzie widać poszczególne warstwy i ich budowę, w tym rozmiar filtra i ich ilość. W zależności

od biblioteki występują drobne różnice w implementacji. Detale poszczególnych odmian sieci ResNet przedstawiono w artykule Deep Residual Learning for Image Recognition.



Rys. 1.9. Wizualizacja poglądowa bloku resztkowego



Rys. 1.10. Architektura modelu *resnet18*

1.4.6. Narzędzia do pracy z sieciami CNN

Powstało wiele bibliotek przeznaczonych do uczenia głębokiego i pracy z sieciami konwolucyjnymi. Biblioteki te implementują wymyślone przez wszystkie lata nauki i badań modele sieci CNN oraz techniki przeprowadzenia treningu, klasyfikacji i przetwarzania zbiorów danych. Najwięcej z nich dotyczy języka Python. Wśród dostępnych bibliotek możemy wyróżnić:

— TensorFlow,
— PyTorch,
— DeepLearning4J,
— Microsoft Cognitive Toolkit,
— Keras,
— ONNX,
— MXNET,
— Caffe.

2. ŚRODOWISKO EKSPERYMENTALNE

W porozumieniu z promotorem pracy dyplomowej została wybrana biblioteka PyTorch [22, 13, 17] przeznaczona do pracy z językiem Python. Promotor pomógł również z wyborem zbioru danych potrzebnego do wytrenowania sieci, aby była zdolna do wykrywania raka piersi. Wybrano zbiór Breast Histopathology Images [1] z popularnej platformy Kaggle.

2.1. WYKORZYSTYWANE TECHNOLOGIE

Projekt został wykonany przy użyciu następujących technologii, które zostały wybierane i dokładane w trakcie wykonywanych prac. Działania rozpoczęto od przygotowania modelu sieci CNN i trenowania go w celu klasyfikacji raka piersi w środowisku Python z wykorzystaniem biblioteki PyTorch. Kolejne biblioteki były wynikiem potrzeby spełnienia zdefiniowanych założeń i udoskonaleń aplikacji.

- Python 3.10.6
- PyTorch 1.13.0
- PyTorch-cuda 11.7
- PyCharm 2021.3.2
- CUDA 11.8
- tkinter 8.6
- Pillow 9.2.0
- matplotlib 3.5.3
- scikit-learn 1.1.3
- numpy 1.23.3
- imbalanced-learn 0.10.0
- torchvision 0.14.0

Rys. 2.1. Technologie wykorzystane w celu powstania aplikacji do wykrywania raka piersi

2.2. PYTORCH

PyTorch to potężne narzędzie do pracy z sieciami neuronowymi w środowisku języka Python, który jest przodującym językiem w dziedzinie uczenia maszynowego. Biblioteka ta zawiera otwarty kod źródłowy i jest stale rozwijana, głównie przez grupę badawczą platformy Facebook w dziedzinie sztucznej inteligencji. Dzięki bibliotece możliwe jest opracowywanie i szkolenie modeli głębokich sieci neuronowych oraz wykorzystywanie modeli

do przetwarzania danych. Biblioteka zapewnia również wsparcie akceleracji obliczeniowej z wykorzystaniem kart graficznych. PyTorch zawiera szereg przydatnych klas i funkcji, które mają intuicyjny interfejs i zapewniają zoptymalizowaną pracę. Wśród najważniejszych elementów można wyróżnić klasę *Dataset*, *DataLoader* i funkcję *transforms*, które odpowiadają za przygotowanie zbioru zdjęć i zamianę ich na tensory, moduł *models*, który zawiera wszystkie modele *State of the Art*, w tym *resnet18*, funkcje takie jak *softmax*, *max* potrzebne do uzyskania wyników klasyfikacji, moduł *optim* zawierający klasy optymalizatorów, w tym *SGD* i *Adam*, moduł *nn* zawierający wszystkie najważniejsze elementy sieci neuronowych, w tym funkcję kosztu *CrossEntropyLoss*, zapisywanie i wczytywanie modelu poprzez funkcje *save* i *load* oraz funckje *train* i *eval* danego modelu służące odpowiednio do treningu modelu i wykorzystywania go w celu uzyskiwania predykcji po wyuczeniu. Istotne są jeszcze funkcje *backward* klasy funkcji kosztu i *step* klasy optymalizatora obsługujące propagację wsteczną.

Sterowniki CUDA pozwalające na wykonywanie obliczeń na kartach graficznych Nvidia są wspierane przez bibliotekę PyTorch i zostały wykorzystane również na potrzeby tworzenia opisywanej aplikacji. Szczególnie było to potrzebne podczas procesu trenowania. Dodatkowo uwzględniono obecność kart graficznych Nvidia na komputerach użytkowników końcowych, optymalizując pod tym kątem obliczenia wykonywane do rozpoznania raka piersi, przez co użytkownicy posiadający tę technologię po jej wykryciu będą mogli cieszyć się szybszą pracą aplikacji.

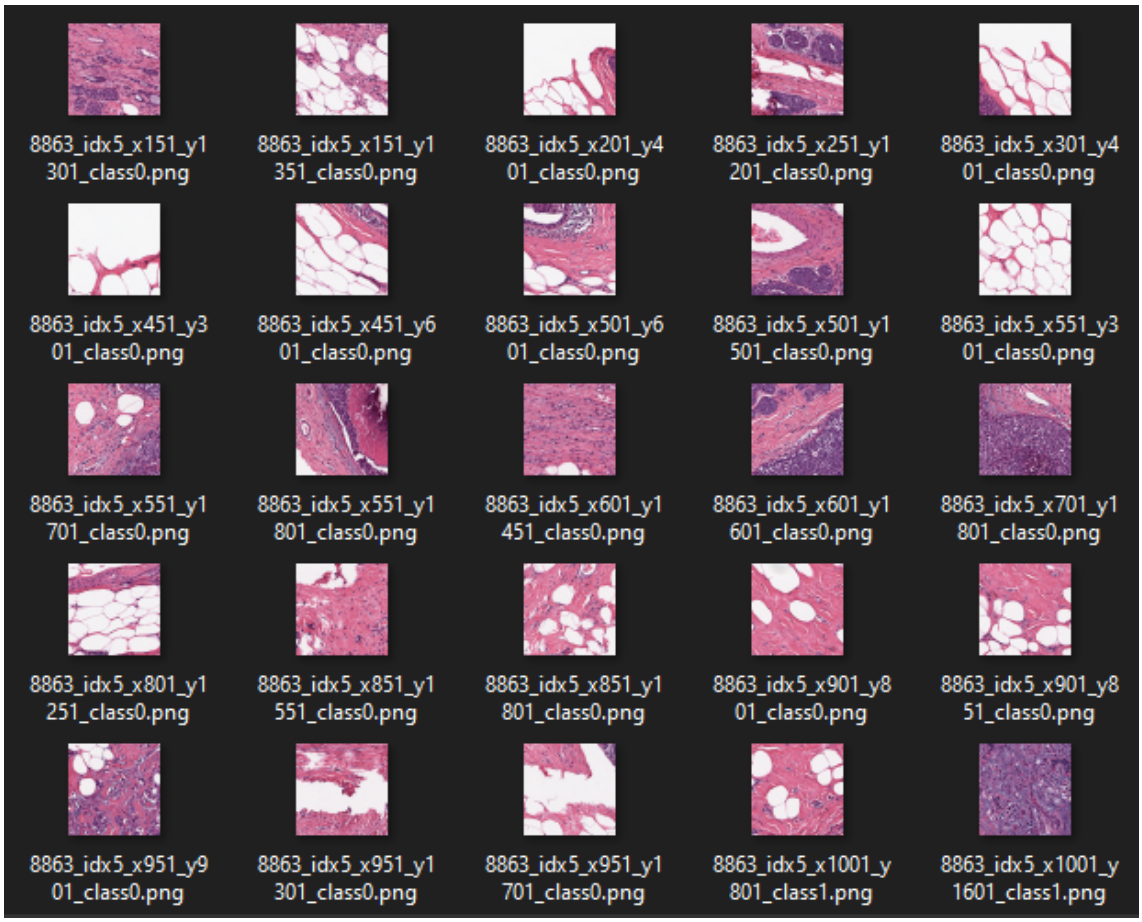
2.2.1. Model sieci

Na potrzeby implementacji algorytmu klasyfikującego raka piersi na obrazach zdecydowano się na model sieci ResNet omawianej we wstępie teoretycznym w najmniejszym jej wariancie, z uwagi na klasyfikację binarną.

W projekcie użyto model *resnet18* dostępny w bibliotece PyTorch. Zawiera on pierwszą warstwę konwolucji z kernelem 7x7, stride 2 i padding o wartości 3, następnie *ReLu*, warstwę pooling o rozmiarze 3x3 i stride 2, po czym występuje 8 bloków resztkowych. W blokach 1, 2, 4, 6, 8 występują dwie warstwy konwolucji z filtrem 3x3 i stride 1, natomiast trzeci, piąty i siódmy blok składa się z trzech warstw konwolucyjnych, przy czym pierwsza ma kernel 3x3 i stride 2, druga kernel 3x3 i stride 1, a trzecia kernel 1x1 i stride 2. Na koniec stosowany jest pooling z wybieraniem wartości średnich i warstwa w pełni połączona zawierająca 512 wejść oraz 2 wyjścia. Oryginalnie model ten zawiera 1000 wyjść z uwagi na dostosowanie do zbioru danych ImageNet, ale w przypadku wykrywania raka piersi potrzebne są dwa wyjścia, więc dokonano zamiany ostatniej warstwy w pełni połączonej podczas inicjalizacji modelu. Zostało to zaprezentowane w rozdziale dotyczącym implementacji.

2.3. ZBIÓR DANYCH

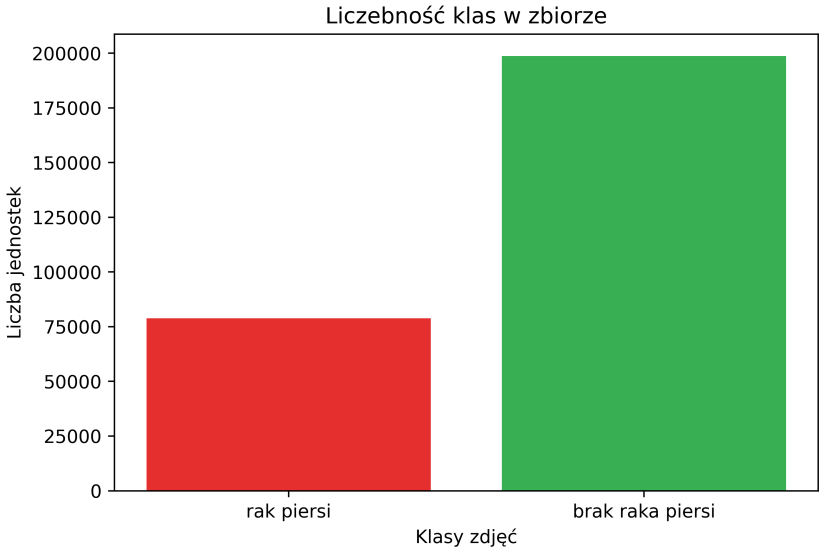
Wykorzystywany w projekcie zbiór danych o nazwie Breast Histopathology Images [1] pochodzi z platformy Kaggle i zawiera 277 524 zdjęć histopatologicznych pochodzących ze skanowania próbek mikroskopem z 40-krotnym przybliżeniem. Każde zdjęcie ma rozmiar 50 pikseli wysokości i szerokości, z czego 198 738 obrazy są wolne od raka piersi, a 78 786 zawierają komórki rakowe. Zdjęcia są zapisane pod nazwą, która sugeruje występowanie raka piersi bądź nie i nazwa ta stanowi etykietę klasy. Przykładowo *10253idx5x1351y1101class0.png* nie zawiera komórek nowotworowych, ponieważ w nazwie występuje *class0*. Zdjęcia zawierające w nazwie pliku *class1* zawierają komórki raka piersi. Na rysunku 2.2 przedstawiono kilka przykładowych zdjęć ze zbioru. Zdjęcia te są trudne do analizy przez człowieka, a osoba nieposiadająca wiedzy na temat komórek nowotworowych nie doszuka się na nich zależności. Ilość zdjęć również byłaby niemożliwa do analizy przez lekarzy w rozsądnym czasie. Duża ilość próbek działa za to bardzo korzystnie na uczenie algorytmu CNN, a ich małe wymiary bazowe pozwalają na sprawne wykonywanie przeliczeń tensorów.



Rys. 2.2. Przykładowe próbki obrazowe z badanego zbioru danych

Tabela 2.1. Rozkład liczbowy zdjęć w wybranym zbiorze danych z podziałem na klasy

	Rak piersi	Brak raka
Liczba zdjęć	78 786	198 738



Rys. 2.3. Wykres liczby instancji klas w wybranym zbiorze

2.4. PRZYGOTOWANIE ZBIORU DANYCH

Biblioteka PyTorch zawiera klasę *Dataset*, która zapewnia obsługę zbioru danych w efektywny sposób. Klasa przechowuje ścieżki do plików, a w momencie konieczności pobrania danej puli zdjęć są one pojedynczo otwierane i zwracane za pośrednictwem funkcji *__getitem__*. W celu załadowania posiadanego zbioru danych należy stworzyć własną klasę, która będzie dziedziczyła klasę *Dataset*. Na potrzeby aplikacji została stworzona klasa *BreastDataset* przechowująca ścieżki do zdjęć histopatologicznych raka piersi. Klasa ta jest odpowiedzialna za dane treningowe. Do celów analizowania zdjęć przez użytkowników końcowych stworzono też klasę *BreastValDataset*, która nie przechowuje etykiet klas danych. Do poprawnego działania sieci potrzebny jest jeszcze obiekt *DataLoader*, pozwalający iterować po obiekcie wcześniej stworzonej klasy *BreastDataset*. *DataLoader* dzieli również zbiór na odpowiednie porcje, których rozmiar wskazywany jest przez argument *batch_size*.

Przygotowanie danych rozpoczęto od podziału całego zbioru na dane przeznaczone do uczenia głębokiego i dane wykorzystywane później do weryfikacji poprawności działania aplikacji. Na listingu 2.1 przedstawiono fragment kodu realizującego podział zbioru danych na dane treningowe i symulacyjne. Dane służące do uczenia sieci stanowią 90% całego zbioru danych. Skrypt wykorzystuje funkcję *test_train _split* biblioteki *scikit-learn* i tworzy dwa nowe foldery pod nazwami train_test i validate. W katalogu train_test znajdują się dane wykorzystywane na potrzeby nauki modelu, a w folderze validate są dane przeznaczone do późniejszego testowania gotowej aplikacji.

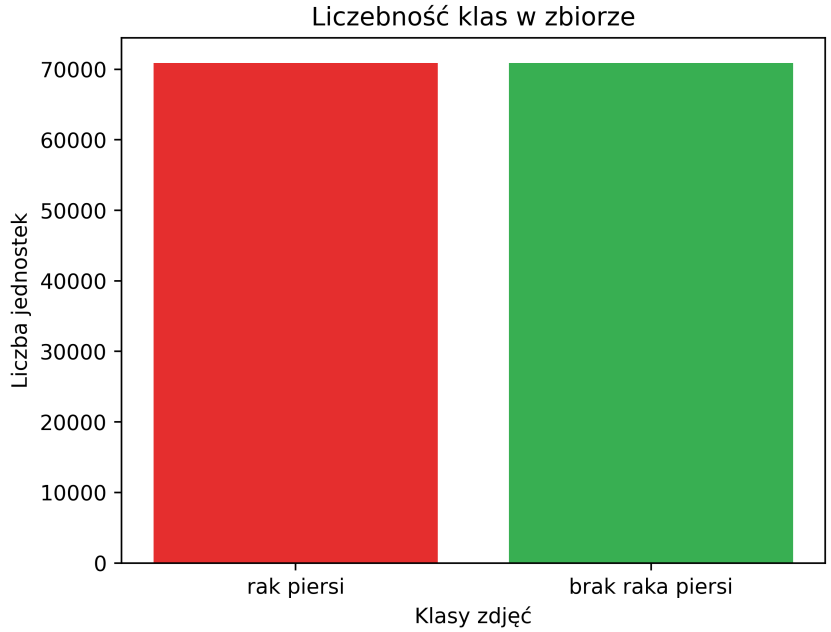
```
dir_path = r"C:\IDC_regular_ps50_idx5"
train_test_dir = "../train_test"
train_test_dir_balanced = "../train_test_balanced"
validate_dir = "../validate"

def initial_split(class_label=1):
    paths = glob.glob(f'{dir_path}/**/{class_label}/*.png')
    labels = [int(path[-5]) for path in paths]

    X_train, X_val, y_train, y_val = train_test_split(paths, labels,
    ↪  test_size=.1, random_state=42)

    for x in X_val:
        shutil.copy2(x, validate_dir)
    print("validation files copied!")

    for x in X_train:
        shutil.copy2(x, train_test_dir)
    print("train_test files copied!")


def prepare_dirs():
    if not os.path.exists(train_test_dir):
        os.makedirs(train_test_dir)
    if not os.path.exists(validate_dir):
        os.makedirs(validate_dir)
    if not os.path.exists(train_test_dir_balanced):
        os.makedirs(train_test_dir_balanced)


def split_dataset():
    prepare_dirs()
    initial_split(0)
    initial_split(1)
```

Listing 2.1: Podział zbioru danych dane treningowo-testowe i do finalnego korzystania z aplikacji

Tabela 2.2. Rozkład liczbowy zdjęć w zbiorze treningowym po zbalansowaniu

	Rak piersi	Brak raka
Liczba zdjęć	70 907	70 907

Z uwagi na lekkie niezbalansowanie danych [11], widoczne w tabeli 2.1 i na rysunku 2.3 zdecydowano się na wyrównanie liczby próbek. Dzięki dużemu rozmiarowi zbioru danych możliwe było zastosowanie metody *undersampling* [20], czyli wyrównania liczebności klas poprzez odrzucenie odpowiedniej liczby instancji klasy większościowej. Zazwyczaj próbki klasy większościowej usuwa się losowo i tak również postąpiono ze zdjęciami wolnymi od raka piersi. Losowe odrzucanie jest szybkie, co ma duże znaczenie w przypadku tak dużej liczby zdjęć, wymagających znacznie większych mocy obliczeniowych niż dane tekstowe. Liczebność klas po zbalansowaniu prezentuje rysunek 2.4, a fragment kodu realizujący tę operację znajduje się na listingu 2.2. Jak widać też w tabeli 2.2, dane treningowe znajdujące się w folderze train_test_balanced zawierają identyczną liczbę instancji obu klas i posłużą do wykonania uczenia głębokiego modelu *resnet18*.



Rys. 2.4. Wykres danych po zbalansowaniu

```python
def random_under_sampling():
    X = glob.glob(f'{train_test_dir}/*.png')
    y = [int(path[-5]) for path in X]
    X = np.array(X).reshape(-1, 1)
    print('Original dataset shape %s' % Counter(y))

    rus = RandomUnderSampler(random_state=0)
    X_resampled, y_resampled = rus.fit_resample(X, y)
    print('Resampled dataset shape %s' % Counter(y_resampled))
    X_resampled = X_resampled.flatten().tolist()
    for x in X_resampled:
        shutil.copy2(x, train_test_dir_balanced)
    print("balanced files copied!")


def balance_dataset():
    prepare_dirs()
    random_under_sampling()


def prepare_data():
    split_dataset()
    balance_dataset()


prepare_data()
```

Listing 2.2: Zbalansowanie zbioru danych metodą *undersampling*

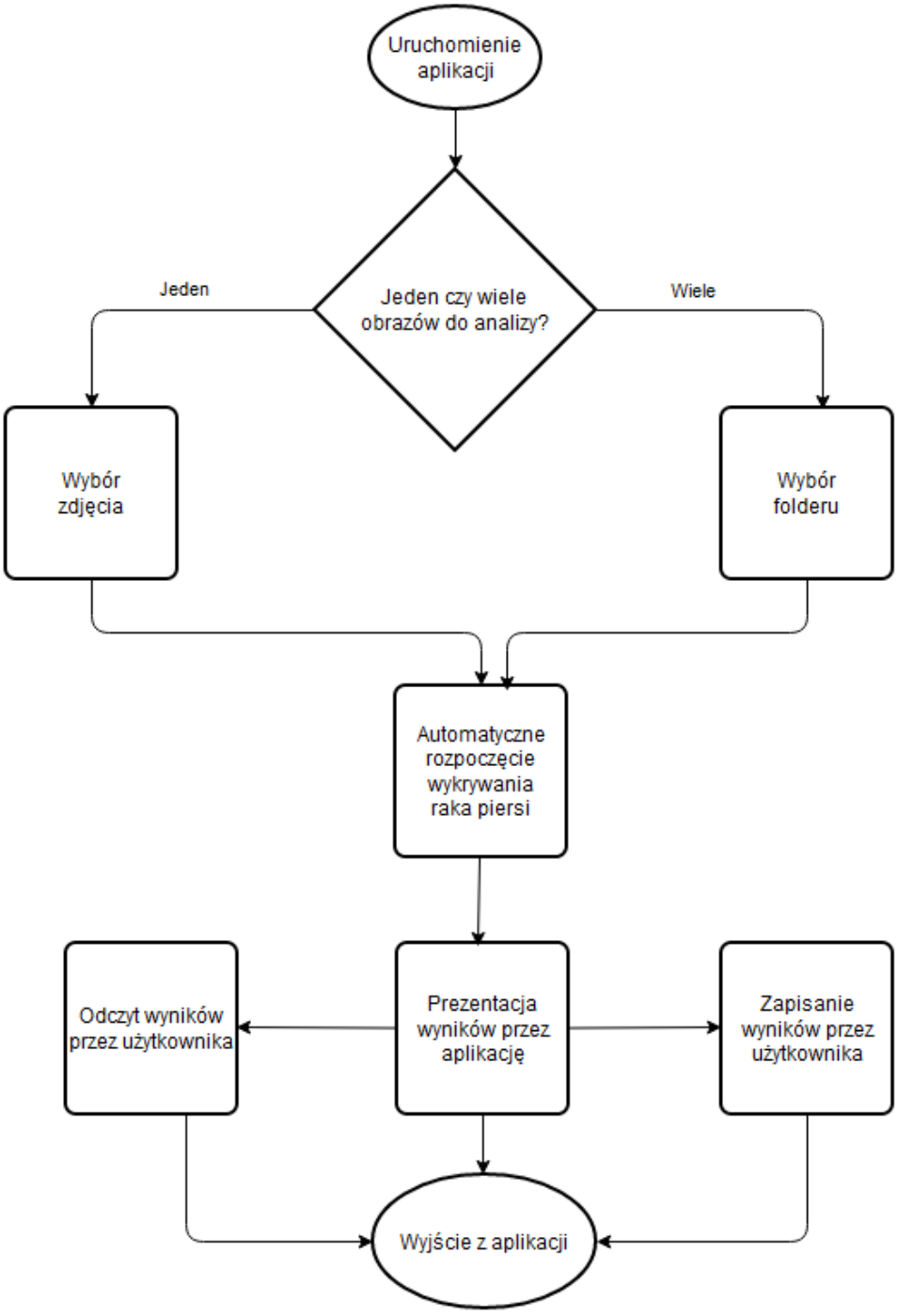
3. PLAN APLIKACJI

Aplikacja umożliwia przeanalizowanie obrazu pochodzącego z badania onkologicznego piersi pod kątem występowania na nim komórek nowotworowych. Badane obrazy to zdjęcia histopatologiczne wygenerowane przez mikroskop. Przeanalizowany może zostać pojedynczy obraz lub wiele obrazów naraz, znajdujących się w wybranym folderze. Po uruchomieniu aplikacji użytkownik wybiera, czy chce przeanalizować jeden obraz, czy cały folder. Po wskazaniu obrazów proces wykrywania raka piersi rozpocznie się automatycznie. Proces analizy zakończy się, kiedy zostaną wypisane wyniki, zawierające decyzję, czy na obrazie wykryto raka piersi, czy nie. W celu utrwalenia wyników, a następnie przekazania ich pacjentce, aplikacja umożliwia zapis przeprowadzonych badań. W tym celu należy nacisnąć przycisk zapisu wyników, po czym wskazać folder, do którego ma być wykonany zapis i wpisać nazwę pliku. Wyświetlona zostanie informacja zwrotna o pomyślnym zapisie. W tym momencie można zamknąć aplikację lub przeprowadzić kolejną analizę, naciskając przyciski przeznaczone do analizy jednego lub wielu obrazów. Aby odczytać zapisane wyniki, należy znaleźć na dysku wskazany folder, a w nim plik tekstowy o wybranej nazwie, po czym go otworzyć. Wyniki zapisywane są w sposób prosty i czytelny do analizy.

Poprawne działanie aplikacji w obrębie wykrywania raka piersi zależy od modelu sieci CNN, który będzie przetwarzał obrazy. Model ten musi być wytrenowany pod kątem rozpoznawania komórek nowotworowych odpowiednim zbiorem danych. Prace nad aplikacją muszą uwzględnić wytrenowanie modelu, a następnie używanie go do analizy zdjęć. Planowane jest użycie modelu sieci ResNet, konkretnie *resnet18*, który będzie wstępnie wytrenowany zbiorem ImageNet pod kątem rozpoznawania podstawowych cech obrazów. Do zrealizowania tego założenia wykorzystane zostanie uczenie transferowe, które w przypadku biblioteki PyTorch realizowane jest poprzez argument *pretrained=True*. Zastosowane zostanie dostrajanie modelu bazowego przez *fine-tuning*. Uczenie głębokie planuje się wykonywać przez 20 epok, a parametry i poszczególne zmienne należy dobrać eksperymentalnie w celu uzyskania jak najlepszych wyników. Do liczenia funkcji kosztu wykorzystana zostanie klasa *CrossEntropyLoss*, a za propagację wsteczną odpowiadać będzie algorytm *SGD* lub *Adam*, w zależności który z nich okaże się lepszy podczas testów. Testom będzie podlegał również parametr *learning_rate* wskazujący na stopień uczenia, czyli wielkość zmian wprowadzanych podczas propagacji wstecznej. Parametrem badanym podczas testów będzie również *batch_size*, czyli liczba zdjęć przetwarzanych równolegle przez model, tzw. porcja danych. Wytrenowany model powinien być automatycznie wczytywany przez aplikację.

3.1. SCHEMAT LOGICZNY APLIKACJI

Wykonano schemat 3.1, który pomaga zrozumieć działanie aplikacji i korzystanie z niej przez użytkownika końcowego. Po uruchomieniu wybierana jest jedna z opcji analizy zdjęć, pojedyncza lub masowa, a następnie aplikacja przeprowadza detekcję raka piersi i wypisuje wyniki. Użytkownik może opcjonalnie zapisać wyniki i je odczytać, a ostatecznie zamknąć aplikację.



Rys. 3.1. Schemat logiczny działania aplikacji

3.2. WYMAGANIA FUNKCJONALNE I NIEFUNKCJONALNE

Wymagania funkcjonalne:

— wybór pojedynczego obrazu do analizy
— wybór folderu zawierającego wiele obrazów do analizy
— rozpoczęcie procesu analizy obrazów
— wyświetlanie wyniku decyzji o wystąpieniu nowotworu
— zapis wyników do pliku
— wyświetlanie postępów pracy aplikacji

Wymagania niefunkcjonalne:

— graficzny interfejs użytkownika
— wyniki w postaci decyzji o występowaniu raka piersi w prostej i jednoznacznej formie
— w przypadku analizy pojedynczego zdjęcia wyniki zawierające procent ich pewności
— interfejs użytkownika w języku angielskim
— stabilne działanie na komputerach z systemem Windows 10 i 11
— możliwa analiza folderu zawierającego maksymalnie 1000 zdjęć
— zabezpieczenia przed próbą zmiany kolejności wykonywania działań
— automatyczne wczytanie domyślnego wytrenowanego modelu CNN *resnet18*
— automatyczne rozpoczęcie analizy obrazów po ich wskazaniu
— automatyczne wyświetlenie wyników klasyfikacji

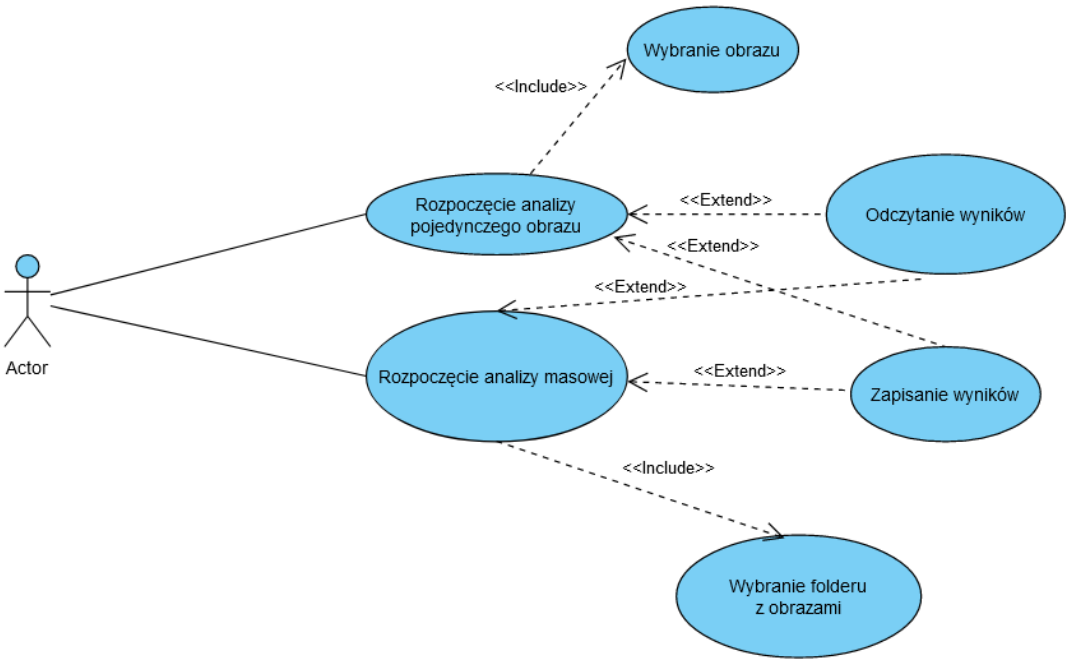
3.3. DIAGRAM PRZYPADKÓW UŻYCIA

Diagram 3.2 prezentuje przypadki użycia z perspektywy użytkownika aplikacji. Początkowo może on podjąć jedną z dwóch opcji. Może być to rozpoczęcie analizy pojedynczego obrazu lub rozpoczęcie analizy katalogu obrazów. W zależności od wyboru musi on następnie wybrać zdjęcie lub folder. Następnie rozpocznie się wykrywanie raka piersi. Po jego zakończeniu użytkownik może opcjonalnie odczytać wyniki i je zapisać.

3.4. SCENARIUSZE WYBRANYCH PRZYPADKÓW UŻYCIA

1. Użytkownik analizuje pojedynczy obraz

   Aktor: użytkownik
   Zdarzenie inicjujące: wciśnięcie przycisku analizy pojedynczego obrazu
   Warunki początkowe:
   — wczytanie wytrenowanego modelu CNN
   — wskazanie ścieżki do pliku podlegającego analizie



Rys. 3.2. Diagram przypadków użycia

Warunki końcowe: została podjęta decyzja o występowaniu raka piersi

Scenariusz:

a) Uruchomienie aplikacji
b) Wybranie i wciśnięcie na panelu aplikacji przycisku do analizy pojedynczego obrazu
c) Wskazanie obrazu przez okno wyboru plików i naciśnięcie przycisku akceptacji
d) Odczekanie, aż proces analizy dobiegnie końca
e) Odczytanie wyniku występowania raka piersi
f) Zamknięcie aplikacji

2. Użytkownik zapisuje wyniki analizy wielu obrazów z wybranego folderu

Aktor: użytkownik
Zdarzenie inicjujące: wciśnięcie przycisku zapisu wyników
Warunki początkowe:
— proces analizy dobiegł końca i podjęto decyzję o występowaniu raka piersi
Warunki końcowe: wynik analizy raka piersi został pomyślnie zapisany

Scenariusz:

a) Odczytanie wyników występowania raka piersi
b) Wybranie i wciśnięcie na panelu aplikacji przycisku do zapisu wyników

c) Wskazanie folderu, do którego ma zostać zapisany wynik analizy
d) Podanie nazwy zapisywanego pliku i naciśnięcie przycisku akceptacji
e) Odczytanie informacji o pomyślnym zapisie wyniku
f) Zamknięcie aplikacji

3.5. WIDOKI

1. Widok główny aplikacji
   Aplikacja zawiera jeden panel główny, na którym znajdują się wszystkie istotne funkcjonalności oraz informacje zwrotne. Główny panel zawiera panel sterujący i panel wynikowy do odczytu wyników przeprowadzonej detekcji raka piersi. Panel sterujący zawiera dwa przyciski rozpoczynające analizę, jeden przycisk służący do przeprowadzenia dokładnej analizy pojedynczego obrazu i drugi do analizy całego folderu obrazów. Dodatkowo zawiera on przycisk zapisu rezultatów. Panel wynikowy zawiera pole tekstowe, na którym wyświetlane są aktualne stany aplikacji i wyniki obecności raka piersi. W przypadku zakończenia przetwarzania pojedynczego zdjęcia na panelu tym powinno się ono wyświetlić z podjętą decyzją.

2. Widok wyboru pojedynczego obrazu
   Po wciśnięciu przycisku do analizy pojedynczego obrazu zostanie otworzone okno systemowe, w którym będzie możliwe wskazanie obrazu. Należy wskazać obraz z rozszerzeniem *.png*, *.jpeg*, *.jpg* lub *.gif* i nacisnąć przycisk akceptacji.

3. Widok wyboru wielu obrazów
   Po wciśnięciu przycisku do analizy wielu obrazów zostanie otworzone okno systemowe, w którym będzie możliwe wskazanie folderu. Z wybranego katalogu zostaną pobrane wszystkie obrazy z rozszerzeniem *.png*, *.jpeg*, *.jpg* i *.gif*. Po wskazaniu folderu należy nacisnąć przycisk akceptacji.

4. Widok zapisu wyników
   Po wciśnięciu przycisku do zapisu wyników zostanie otworzone okno systemowe, w którym będzie możliwe wskazanie folderu i podanie nazwy pliku. Domyślne rozszerzenie pliku będzie ustawione jako *.txt*. Po wskazaniu folderu i podaniu nazwy pliku należy nacisnąć przycisk akceptacji.

4. WYKONANIE PROJEKTU I IMPLEMENTACJA APLIKACJI

Prace nad aplikacją zaczęły się od stworzenia kodu odpowiedzialnego za powstanie modelu sieci CNN *resnet18* i próby wytrenowania go z użyciem danych train_test_balanced. Na potrzeby tworzenia kodu i sprawdzania jego poprawności działania przygotowano zbiór train_test_small, który stanowi 5% docelowego zbioru danych i umożliwia przeprowadzenie uczenia w rozsądnym czasie kilkunastu minut. Zbiór ten wykorzystano również do początkowego dobrania parametrów i odrzucenia ich skrajnych kombinacji. Dokładne testy parametrów opisano w kolejnym rozdziale dotyczącym testowania aplikacji. Ostatecznie do treningu sieci użyto optymalizator *SGD* odpowiedzialny za propagację wsteczną, *learning_rate* o wartości 0,01 i *batch_size* o wartości 64. Kod odpowiedzialny za wytrenowanie sieci nie jest potrzebny w aplikacji dla użytkowników końcowych, dlatego nie stworzono do niego interfejsu graficznego. Raz przeprowadzone uczenie głębokie dostarcza wyszkolony model, który zapisano i wykorzystuje się później w aplikacji. Po wytrenowaniu modelu zaczęto prace nad interfejsem graficznym, aby spełnić wszystkie założenia i przypadki użycia. Interfejs graficzny nie jest głównym celem omawianej pracy, więc wybrano bibliotekę tkinter, która pozwala na tworzenie prostych i podstawowych elementów takich jak przyciski i etykiety tekstowe w łatwy sposób. Na tym etapie należało również stworzyć logikę automatyzacji zaplanowanych wymagań dotyczących wczytywania modelu i rozpoczynania pracy modelu oraz połączenie interfejsu graficznego z kodem odpowiedzialnym za wykorzystywanie wytrenowanego modelu *resnet18* do przetwarzania zdjęć. Konieczne było również zaimplementowanie obsługi okien systemowych do wyboru plików i załadowanie ich do pamięci aplikacji. Po przygotowaniu interfejsu zajęto się kodem przetwarzającym zdjęcia pod kątem wykrywania raka piersi i wyświetlaniem wyników powstałych z tego procesu. Kod ten poddano testom na zbiorze walidacyjnym, aby zweryfikować poprawność jego działania. Ostatecznie zaimplementowano funkcjonalność zapisywania wyników do pliku tekstowego i zajęto się dopracowywaniem GUI.

4.1. OMÓWIENIE KODU INICJALIZUJĄCEGO MODEL RESNET

W celu przeprowadzenia uczenia głębokiego na wybranym zbiorze zdjęć histopatologicznych konieczne było utworzenie modelu sieci CNN, który będzie można wytrenować do wykrywania raka piersi. Za pomocą zaprezentowanego na listingu 4.1 kodu wybrany

model *resnet18* został przygotowany i odpowiednio przystosowany do problemu klasyfikacji binarnej raka piersi. Domyślnie model ten jest tworzony z parametrem *pretrained* i *fine_tuning* ustawionymi na wartość **True**. Oznacza to, że za pomocą uczenia transferowego wykorzystany zostanie model *State of the Art* wyszkolony zbiorem ImageNet z możliwością douczania. Aby można było wykorzystać go do wykrywania raka piersi, należy zmienić warstwę wyjściową tak, żeby zawierała dwa neurony wyjściowe. Warstwę tą reprezentuje pole *model.fc*. Za wywoływanie tej funkcji odpowiedzialny jest obiekt klasy *TrainCNN* stworzonej w celu obsługi procesu uczenia głębokiego. Listing 4.2 prezentuje wybrane fragmenty klasy *TrainCNN* odpowiedzialne za stworzenie, wytrenowanie i zapisanie modelu. Cała klasa jest zbyt długa, żeby można było ją zaprezentować na listingu, więc zaprezentowano fragmenty. Klasa ta znajduje się w module *backend_deep_learning.py* i używana jest w tym module w funkcji *main*. Model zapisywany jest do pliku z rozszerzeniem *.pth* w automatycznie tworzonym folderze *models*. Najlepszy uzyskany model został zapisany pod nazwą **cnn_breast_cancer_full_64_sgd_01_20e.pth**, mówiącą przy okazji jakie były użyte parametry podczas głębokiego uczenia. Model ten jest później automatycznie wczytywany przez aplikację w celu wykonywania dalszych zadań.

```
import torch.nn as nn
import torchvision.models as models


def set_parameter_requires_grad(model, fine_tuning):
    if not fine_tuning:
        for param in model.parameters():
            param.requires_grad = False


def create_resnet_model(num_classes, pretrained=True, fine_tuning=True):
    model = models.resnet18(pretrained=pretrained)
    set_parameter_requires_grad(model, fine_tuning)
    model.fc = nn.Linear(model.fc.in_features, num_classes)
    return model
```

Listing 4.1: Moduł odpowiedzialny za przygotowanie modelu *resnet18* sieci CNN

4.2. KOD TRENINGU SIECI

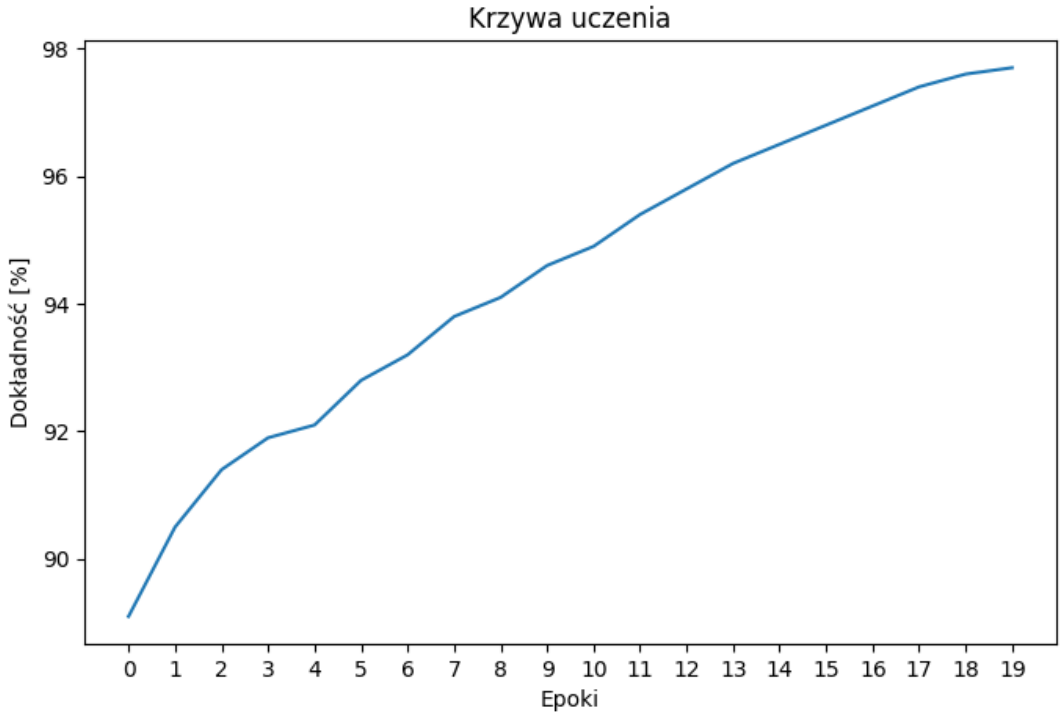
Po dostrojeniu parametrów tak, aby dokładność sieci była jak najlepsza, przeprowadzone zostało 20 epok treningu na pełnym zbiorze danych zbalansowanych. Proces ten zajął ponad 3 dni ciągłego działania algorytmu. Funkcja treningowa wykonująca główną pracę głębokiego uczenia sieci *resnet18* przedstawiona została na listingu 4.3. Z uwagi na długość kodu,

```
class TrainCNN(DeepLearning):
...

    def init_model(self, num_classes=2, pretrained=True, fine_tuning=True,
    ↪  input_required_size=224):
        self.model = create_resnet_model(num_classes, pretrained,
        ↪  fine_tuning)
        self.input_required_size = input_required_size
        self.transformations = train_test_trans(input_required_size)
        self.optimizer = torch.optim.SGD(self.model.parameters(), lr=0.01,
        ↪  momentum=0.9)
...

    def load_train_data(self):
        self.dls = load_train_test_dataset(
            self.train_test_dir,
            ts=self.test_size,
            batch_size=self.batch_size,
            input_required_size=self.input_required_size
        )

    def train(self):
        if self.model is None or self.dls is None or self.loss_func is None
        ↪  or self.optimizer is None:
            return False

        self.model, self.acc_hist, self.best_acc = universal_train(
            self.device,
            self.model,
            self.dls,
            self.loss_func,
            self.optimizer,
            self.num_epochs
        )
        return self.model, self.acc_hist, self.best_acc

    def save_state_dict(self, save_model_path):
        torch.save(self.model.state_dict(), save_model_path)

    def save_model(self, save_model_path):
        torch.save(self.model, save_model_path)
...
```

Listing 4.2: Fragment klasy *TrainCNN* służącej do przeprowadzenia uczenia głębokiego

zostały przedstawione tylko fragmenty funkcji. Funkcja ta przyjmuje kolejno argumenty: *device*, odpowiedzialny za rodzaj jednostki obliczeniowej (CPU lub GPU), *model*, czyli używany model sieci *resnet18* podlegający treningowi, *data_loaders_phases*, czyli słownik języka Python zawierającym obiekty *DataLoader* dla danych treningowych i testowych, *loss_func*, czyli funkcję kosztu, *optimizer*, czyli optymalizator realizujący propagację wsteczną i *num_epochs*, czyli liczbę epok uczenia głębokiego. Podczas przebiegu funkcji realizowane są pętle, które dla danego obiektu *DataLoader* iterują po porcjach danych, tworząc tensory zdjęć i odpowiadające im etykiety klas, a następnie przekazują te dane do modelu, który przetwarza je od pierwszej do ostatniej warstwy sieci i zwraca wyniki, a następnie obliczana jest funkcja kosztu i wykonywana jest propagacja wsteczna. Po zakończeniu cyklu pętla wraca na swój początek i wykonuje się do momentu przetworzenia wszystkich danych treningowych.

Dla każdej epoki sprawdzany i zapisywany jest bieżący koszt i dokładność modelu oraz jeśli są one lepsze niż dotychczas, zapisywany jest obecny stan wewnętrzny modelu uwzględniający wagi połączeń. Na koniec funkcja zwraca wyuczony model z najlepszymi wagami połączeń oraz historię uzyskanych dokładności i najlepszą zanotowaną dokładność. Rysunek 4.1 przedstawia krzywą uczenia modelu. Dokładność wyrażona w procentacch powinna stopniowo rosnąć wraz z kolejnymi epokami uczenia i właśnie tak się dzieje.



Rys. 4.1. Krzywa uczenia modelu *resnet18*

```
def train(device, model, data_loaders_phases, loss_func, optimizer,
↪  num_epochs=1):
...

    for epoch in range(num_epochs):
        ...

        for phase in data_loaders_phases:
            model.train() if phase == 'train' else model.eval()
            current_loss = 0.0
            current_corrects = 0

            for imgs, labels in data_loaders_phases[phase]:
                imgs = imgs.to(device)
                labels = labels.to(device)
                optimizer.zero_grad()
                with torch.set_grad_enabled(phase == "train"):
                    outputs = model(imgs)
                    loss = loss_func(outputs, labels)
                    _, preds = torch.max(outputs, 1)
                    if phase == "train":
                        loss.backward()
                        optimizer.step()

                current_loss += loss.item() * imgs.size(0)
                current_corrects += torch.sum(preds == labels.data)

            epoch_loss = current_loss /
            ↪  len(data_loaders_phases[phase].dataset)
            epoch_acc = current_corrects.double() /
            ↪  len(data_loaders_phases[phase].dataset)

            if phase == "test" and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict())
            if phase == "test":
                test_acc_history.append(epoch_acc)
    ...

    model.load_state_dict(best_model_wts)
    return model, test_acc_history, best_acc
```

Listing 4.3: Funkcja treningowa

4.3. DALSZE KROKI PO TRENINGU

Po wytrenowaniu modelu sieci *resnet18* danymi dotyczącymi raka piersi i bezpiecznym zapisaniu go do pliku zajęto się pracą nad aplikacją przeznaczoną dla użytkownika końcowego. W tym celu powstał graficzny interfejs użytkownika z dwoma przyciskami do analizy zdjęć i jednym do zapisu wyników. Główna część pracy dotyczyła stworzenia logiki, która kryje się pod tymi przyciskami. Stworzono odpowiednie funkcje do wyboru i wczytywania plików oraz przetwarzania przekazywanych obrazów przez model, a następnie wyświetlania wyników. Listing 4.4 przedstawia najważniejsze obiekty wykorzystywane przez moduł odpowiedzialny za GUI, w tym *settings* czyli ustawienia, w których zawarte są kolory, ścieżki plików składających się na aplikację i pamięć pomocnicza, *cnn* czyli obiekt odpowiedzialny za sterowanie logiką przetwarzania zdjęć za pomocą modelu sieci CNN oraz *root*, czyli główny obiekt okna GUI odpowiedzialny za graficzną prezentację interfejsu.

```
settings = Settings()
cnn = CNN(settings.default_model_path)
root = tk.Tk()
```

Listing 4.4: Najważniejsze obiekty modułu *frontend*

Następnie stworzono logikę odpowiedzialną za obsługę przycisków GUI i mechanizm wzajemnej komunikacji między interfejsem a funkcjami wykonującymi operacje na wytrenowanym modelu sieci neuronowej. Przyciski odpowiedzialne za analizę pojedynczego zdjęcia i masową analizę zdjęć wywołują kod z modułu *frontend*, który odwołuje się do kodu z modułu *backend* poprzez obiekt klasy *cnn*. Przycisk zapisu wyników zawiera całą logikę w module *frontend*. Stworzono na tym etapie również kod odpowiedzialny za widoki aplikacji, czyli jej wygląd oraz wyświetlanie informacji zwrotnych na panelu wynikowym.

4.4. KLASYFIKACJA RAKA PIERSI

Główna logika rozpoznawania raka piersi na pojedynczym zdjęciu znajduje się na listingu 4.5, z kolei logika dotycząca masowej analizy zdjęć została zawarta na listingu 4.6. Funkcje te znajdują się w klasie *CNN*, która została stworzona na potrzeby wygodnego dostępu do wszystkich potrzebnych algorytmów logiki z poziomu front-endu, czyli kodu odpowiedzialnego za widoki aplikacji. W obrębie klasy *CNN* znajduje się jeszcze wczytywanie modelu, które jest wykonywane automatycznie. Model jest wczytywany ze ścieżki do pliku podawanej podczas tworzenia obiektu klasy *CNN*, więc w razie powstania kolejnych wytrenowanych modeli można będzie je wykorzystywać w aplikacji. Model *resnet18* wytrenowany pod kątem raka piersi wczytywany jest domyślnie.

Proces analizy pojedynczego obrazu rozpoczyna się od wskazania przez użytkownika interesującego go zdjęcia. Przekazana jest ścieżka do pliku, która następnie pozwala na utworzenie tensora. Tensor jest przekazywany na urządzenie obliczające, czyli procesor lub kartę graficzną, po czym wykonywana jest funkcja *forward* modelu *resnet18*, która odpowiada za przeprowadzenie tensora przez wszystkie warstwy sieci od pierwszej do końcowej i otrzymanie aktywacji na ostatniej warstwie. Aktywacje te są następnie rozpatrywane przez funkcję *softmax*, która zwraca ostateczną decyzję klasyfikacji wraz z jej pewnością. Zwrócone wartości są następnie przetwarzane przez kod aplikacji odpowiedzialny za graficzny interfejs użytkownika i prezentowane na panelu wynikowym aplikacji. Przetwarzanie pojedynczego zdjęcia pozwala na osiągnięcie dokładniejszego opisu wyników, wraz z pewnością i zdjęciem na wykresie. Opcja ta jest więc dobra do analizy zdjęć, które wydają się podejrzane lub gdy jest ich niewiele.

```
def check_single(self, img_path: str) -> tuple:
    img_tensor = transform_singe_image(img_path)
    if img_tensor is None:
        return None, None

    img_tensor = img_tensor.to(self.device)
    result = self.model.forward(img_tensor)

    decision = torch.nn.functional.softmax(result, dim=1)

    confidence, is_cancer = torch.max(decision, 1)
    confidence = confidence.item()
    confidence *= 100
    is_cancer = is_cancer.item()
    return is_cancer, confidence
```

Listing 4.5: Funkcja wykrywania raka piersi na pojedynczym obrazie

Analiza wielu obrazów rozpoczyna się od wskazania przez użytkownika interesującego go folderu systemowego, w którym powinny znajdować się zdjęcia. Wskazany katalog jest następnie przeszukiwany pod kątem występowania próbek i tworzony jest obiekt klasy *BreastValDataset*, a następnie *DataLoader*, który będzie dzielił dane na porcje o domyślnym rozmiarze 4. Wszelkie domyślne ustawienia aplikacji znajdują się w module *settings.py* i mogą w łatwy sposób zostać zmienione. Wartość 4 została wybrana po wielu testach jako optymalna pod kątem zajętości pamięci i częstości informowania użytkownika o wynikach, aby dostarczyć mu jak najlepsze wrażenia użytkowania. Po przygotowaniu danych tworzona jest lista predykcji, która będzie przechowywała wyniki. Następnie pętla iteruje po utworzonym zbiorze próbek, wykonując klasyfikację. Wyniki klasyfikacji określonej porcji danych są na bieżąco wyświetlane na panelu wynikowym GUI. Po ukończeniu całego

procesu zwracana jest lista wyników, które później użytkownik może zapisać do pliku tekstowego. Przetwarzanie całego folderu zdjęć pozwala na szybkie wygenerowanie dużej liczby wyników. Opcja ta jest więc dobra do analizy dużej ilości próbek, kiedy nie ma czasu na przyglądanie się każdej z osobna.

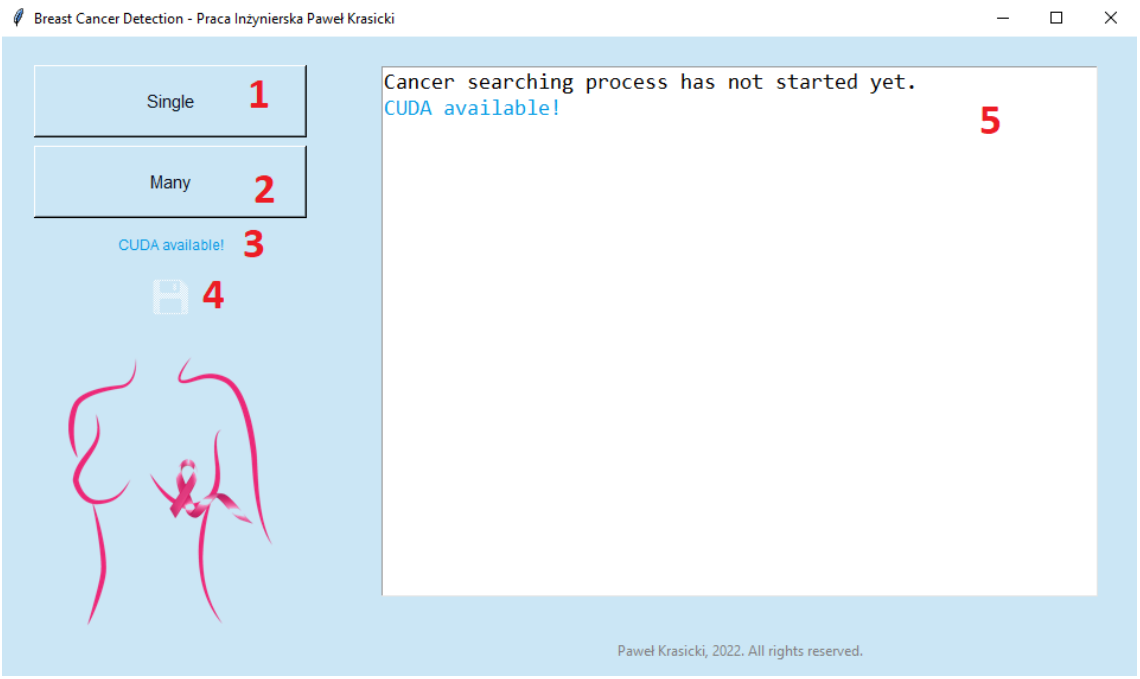
```
def check_many(self, dir_path, print_text, val_batch_size):
    val_dataloader = load_val_dir(dir_path, batch_size=val_batch_size)
    if val_dataloader is None:
        return None
    val_dataloader_len = len(val_dataloader.dataset)
    pred_list = []
    for idx, (images, paths) in enumerate(val_dataloader):
        images = images.to(self.device)

        outputs = self.model(images)
        _, predictions = torch.max(outputs, 1)

        predictions = list(predictions.cpu().detach().numpy())
        for idx_inner, (path, pred) in enumerate(zip(paths, predictions)):
            pred_list.append((path, pred))
            image_num = idx*val_batch_size + idx_inner + 1
            print_text(f"\n{image_num}/{val_dataloader_len}")
            print_text(f"{path}:")
            is_cancer_txt = "CANCER" if pred else "OK"
            print_text(is_cancer_txt, is_cancer_txt.lower())

    return pred_list
```

Listing 4.6: Funkcja wykrywania raka piersi w katalogu obrazów

4.5. GRAFICZNY INTERFEJS UŻYTKOWNIKA

Graficzny interfejs użytkownika skrótowo nazywany GUI odpowiedzialny jest za widoki aplikacji, czyli okna, które użytkownik widzi i może z nimi wejść w interakcję. Wybrana została biblioteka *tkinter* języka Python, która jest jego częścią. Kod obsługujący GUI znajduje się w module *frontend.py* i zawiera obiekty odpowiedzialne za wyświetlanie okien, ramek, przycisków i tekstów na panelu głównym aplikacji oraz funkcje wywoływane po naciśnięciu odpowiednich klawiszy na klawiaturze i przycisków GUI za pomocą myszy. Funkcje te odpowiedzialne są za wskazywanie ścieżek do plików i folderów, zapisywanie wyników, wyświetlanie aktualnych stanów aplikacji i procesu klasyfikacji oraz kontakt z modułem *backend.py*, czyli główną logiką obsługującą model sieci CNN.

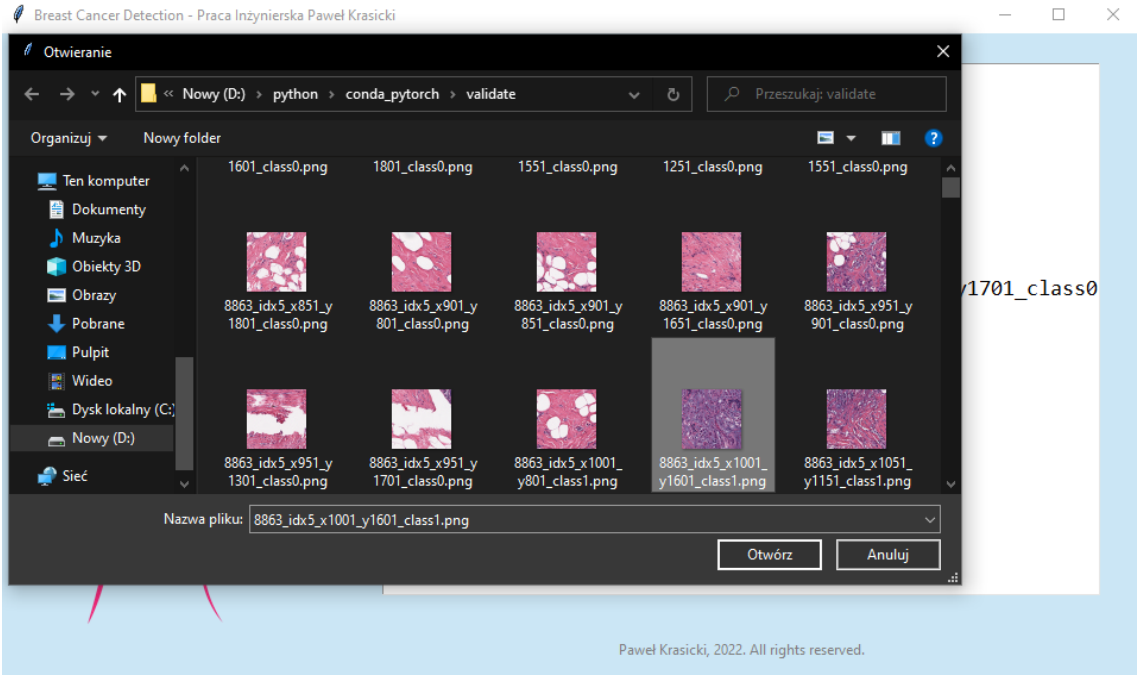
Zrealizowane zostały wszystkie zaplanowane widoki aplikacji. Panel główny zawiera panel sterujący z przyciskami i panel wynikowy, będący dziennikiem zdarzeń. Co więcej, w odpowiednich momentach otwierają się dodatkowe okna służące do wybierania plików bądź ich zapisywania. Panel wynikowy przeznaczony jest do informowania użytkownika o stanie działania aplikacji i wynikach wykrywania raka piersi. W ten sposób użytkownik ma możliwość śledzenia postępów pracy aplikacji. Zaprezentowano go na rysunku 4.2 pod numerem 5. Ważniejsze komunikaty prezentowane są w odpowiednich kolorach. Pozytywny wynik raka piersi wypisywany jest na zielono, a negatywny na czerwono. Błędy wynikające z nieumiejętnego korzystania z aplikacji wypisywane są na fioletowo. Aplikacja została zabezpieczona przed próbami wytrącenia jej z poprawnego działania, takimi jak wskazanie pliku, który nie jest obrazem lub niewskazanie go w ogóle. Zapis wyników jest wyłączony, dopóki wyniki nie powstaną. Kolejnym zabezpieczeniem jest brak możliwości uruchomienia analizy zdjęć, jeśli poprzednia analiza nie dobiegła końca. Proces analizy nie rozpocznie się po wskazaniu folderu zawierającego więcej niż 1000 zdjęć, a użytkownik zostanie o tym poinformowany. Interfejs jest skonstruowany tak, aby używanie aplikacji było intuicyjne i nie stwarzało niejasności lub miejsc, w których jest możliwość zrobienia czegoś inaczej, niż zaplanowano.



Rys. 4.2. Widok główny po uruchomieniu aplikacji

Na rysunku 4.2 widać uruchomiony program gotowy do rozpoczęcia detekcji raka piersi. Domyślnie uruchamiane jest okno wypełniające cały ekran. Na potrzeby prezentacji zrzutów ekranu okno zostało pomniejszone. Elementy graficznego interfejsu zostały na rysunku 4.2 oznaczone numerami, aby móc się lepiej do nich odnieść podczas opisu. Cyfrą 3 oznaczony został komunikat informujący użytkownika o trybie wydajnościowym, w

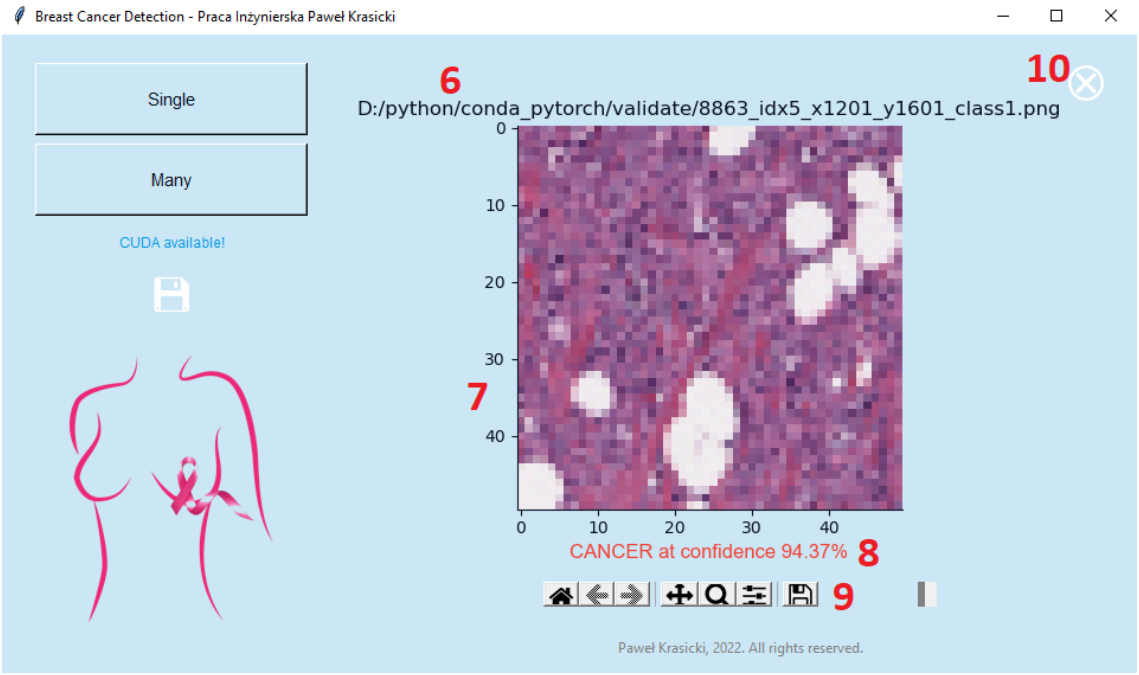
jakim program będzie pracował. Informuje on o wykorzystaniu karty graficznej Nvidia w przypadku napisu "CUDA available!" lub samego procesora w przypadku komunikatu "CPU only". Przycisk oznaczony cyfrą 1 służy do rozpoczęcia procesu detekcji raka piersi na jednym zdjęciu. Po jego wciśnięciu ukaże się okno systemowe do wybrania pliku obrazowego widoczne na rysunku 4.3, a po wyborze zdjęcia aplikacja zacznie analizować obraz pod kątem zawartości.



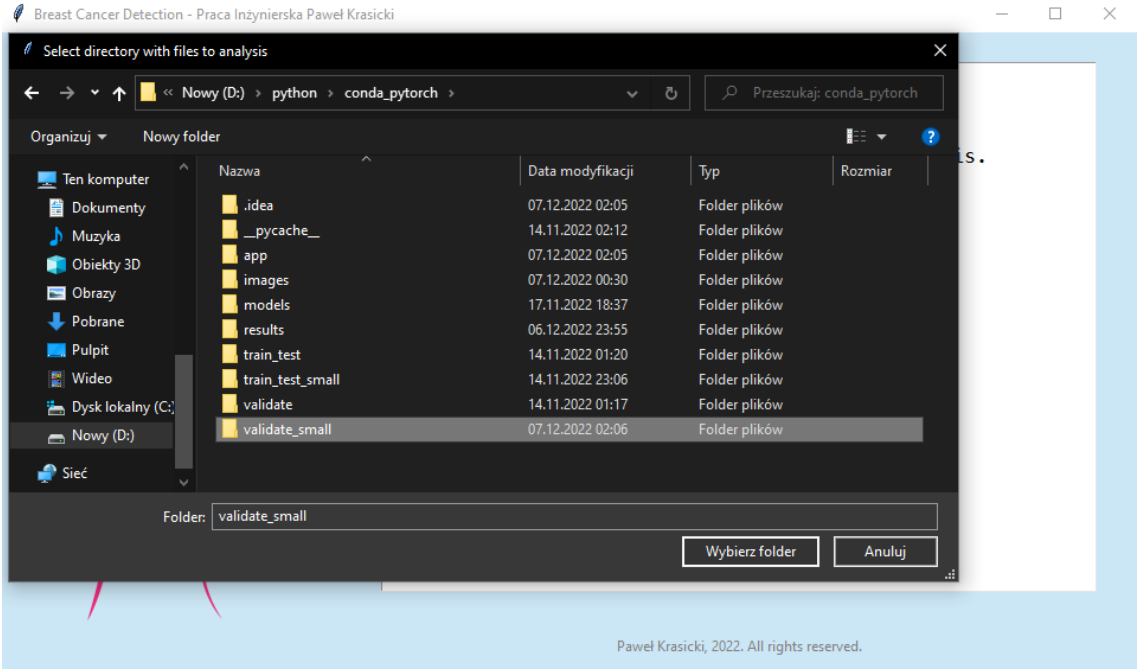
Rys. 4.3. Widok wyboru pojedynczego pliku do wykrywania raka piersi

Wynik zostanie wypisany na panelu wynikowym oznaczonym cyfrą 5. W tym samym momencie zostanie również otworzony i wyświetlony wykres oznaczony liczbą 7 na rysunku 4.4, przez co początkowo użytkownik nie zobaczy tekstowego panelu wynikowego. Cyfra 6 wskazuje ścieżkę pliku przetwarzanego zdjęcia, 8 to wynik klasyfikacji będący informacją czy na zdjęciu wykryto raka piersi, czy nie i z jaką pewnością, a 9 to pasek nawigacyjny, za pomocą którego można wykonywać na wykresie różnego rodzaju operacje takie jak przybliżanie, przesuwanie i zapisywanie zdjęcia. Przycisk pod numerem 10 służy do zamknięcia widoku wykresu i powrotu do tekstowego panelu wynikowego.

Na rysunku 4.2 przycisk oznaczony cyfrą 2 służy do uruchomienia procesu analizy wielu obrazów. Po jego wciśnięciu otworzy się okno systemowe widoczne na rysunku 4.5, za pomocą którego będzie można wybrać folder. Po jego wybraniu uruchomi się proces detekcji raka piersi na obrazach występujących w folderze. Jeśli folder będzie pusty lub liczba obrazów w nim zawartych przekroczy 1000, użytkownik zostanie o tym poinformowany i proces się nie wykona. Pomyślne wykonanie procesu zaprezentowano na rysunku 4.6.

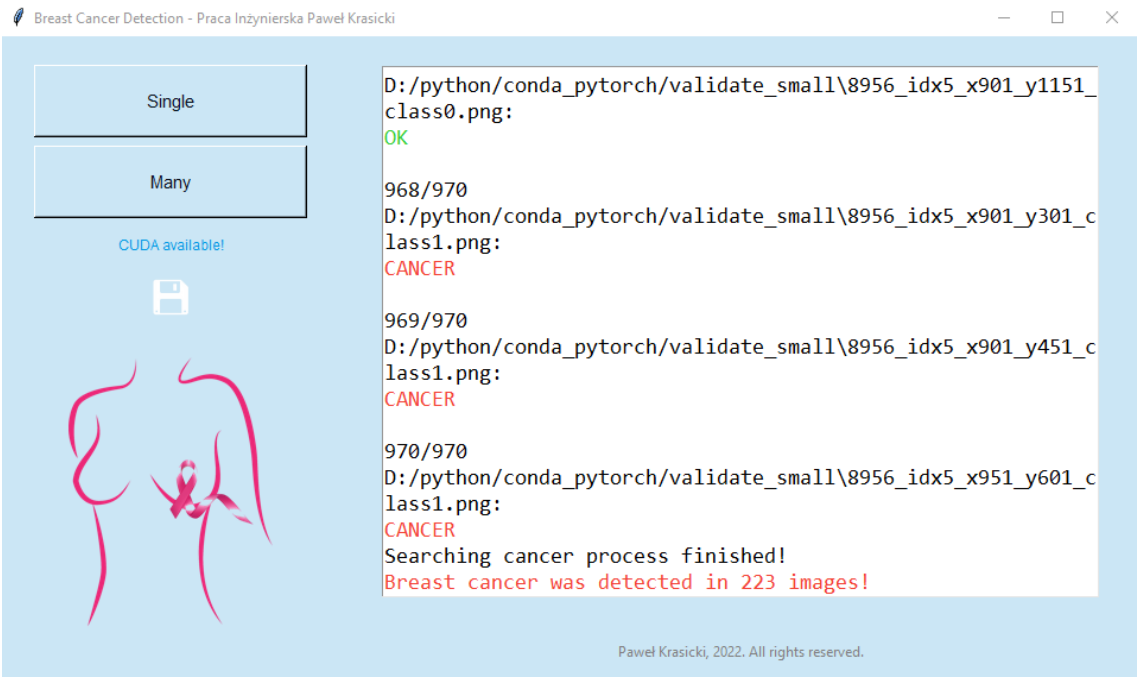


Rys. 4.4. Widok wyniku wykrywania raka piersi na jednym obrazie

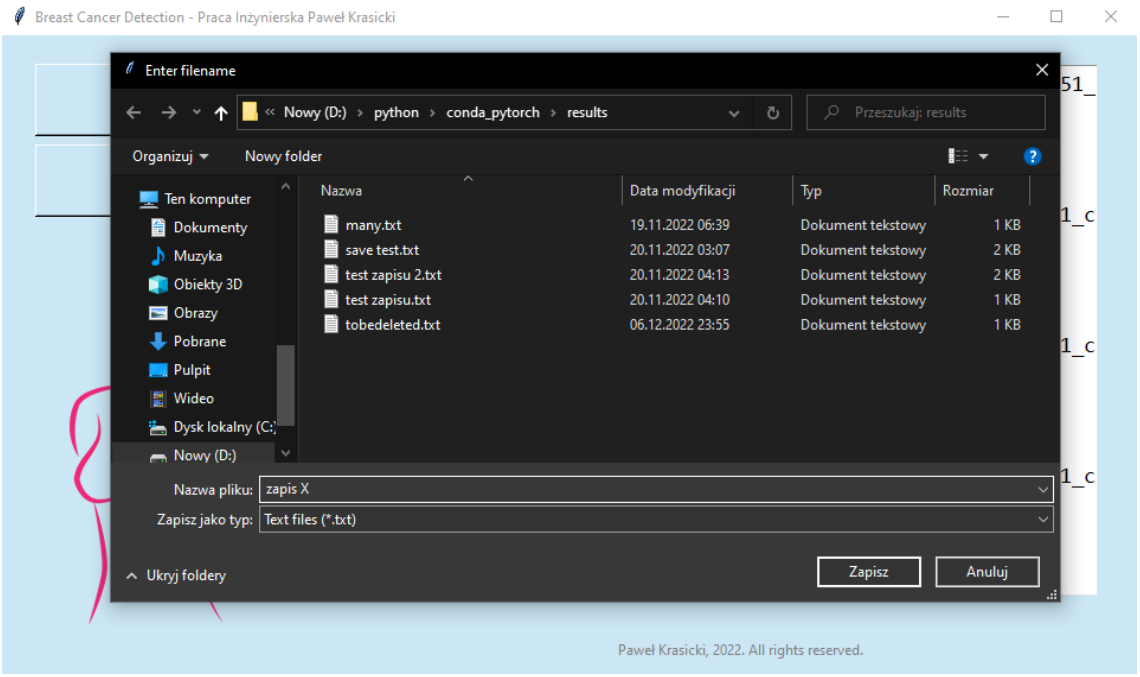


Rys. 4.5. Widok wyboru folderu

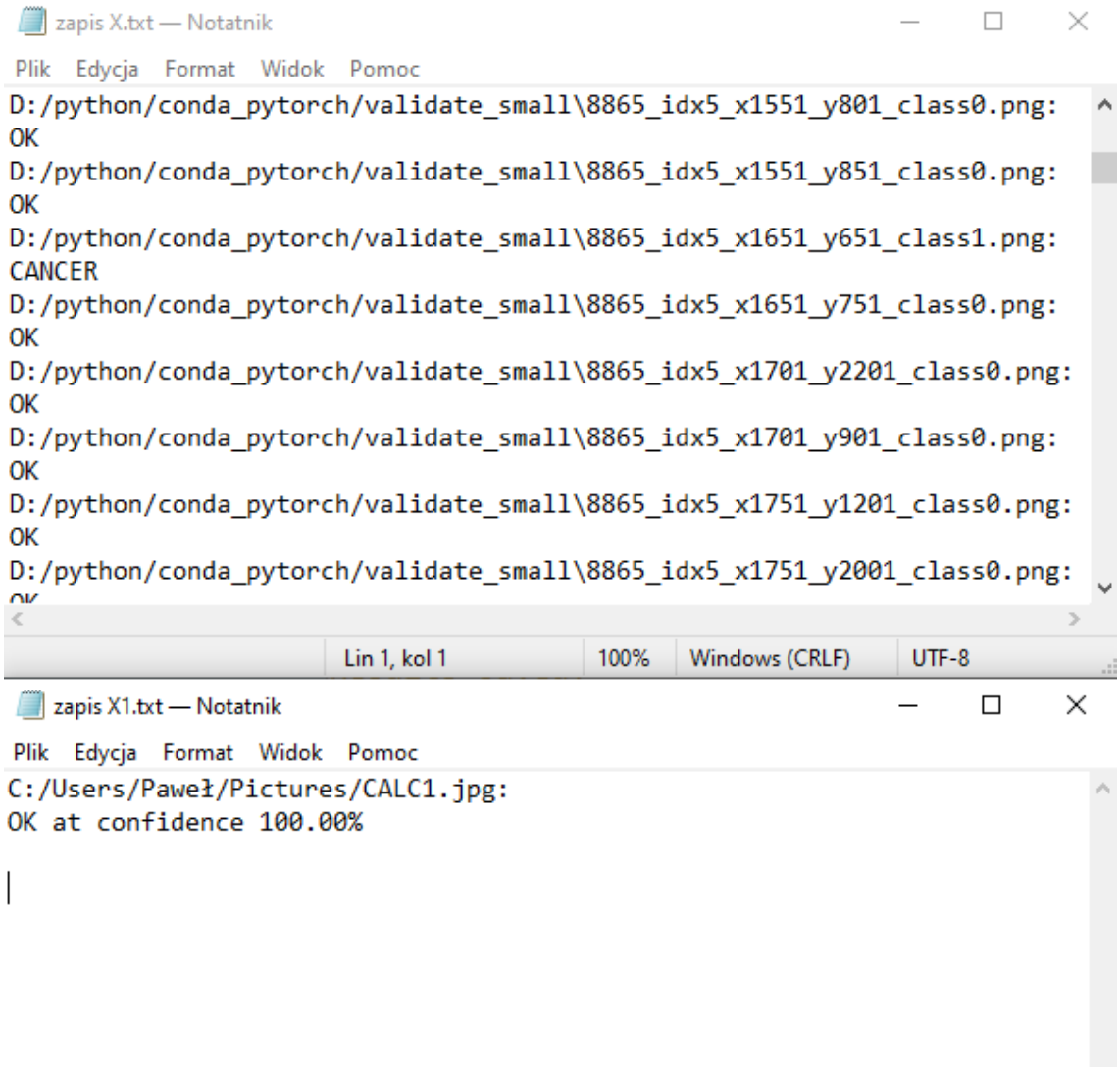
Przycisk oznaczony cyfrą 4 na rysunku 4.2 służy do zapisu wyników otrzymanych z detekcji wykonanej po wciśnięciu przycisków 1 lub 2. Wywołuje on okno systemowe widoczne na rysunku 4.7, w którym można podać nazwę pliku i go zapisać. Rysunek 4.8 przedstawia pliki tekstowe utworzone podczas zapisu wyników detekcji raka piersi w całym folderze zdjęć oraz na pojedynczym zdjęciu zestawione obok siebie.



Rys. 4.6. Widok wyników detekcji przeprowadzonej na całym folderze zdjęć walidacyjnych



Rys. 4.7. Widok zapisu pliku z wynikami



Rys. 4.8. Widok plików tekstowych utworzonych przez zapisanie wyników detekcji raka piersi

5. TESTOWANIE APLIKACJI

Testowanie rozpoczęto już na etapie trenowania modelu *resnet18*. Dobór odpowiednich parametrów uczenia był kluczowy, aby uzyskać jak najlepszy model, który jest używany przez aplikację w jej głównym celu, czyli wykrywaniu raka piersi. Następnie tworzono kolejne funkcje realizujące przypadki użycia i sprawdzano na bieżąco poprawność ich pracy. Na koniec przeprowadzono testy manualne wszystkich przypadków użycia, wcielając się w użytkownika końcowego i weryfikując wyniki.

5.1. DOBÓR PARAMETRÓW UCZENIA

Aby algorytm działał skutecznie, potrzebuje być wytrenowany przy użyciu odpowiednich danych z odpowiednimi parametrami. Optymalizator, stopień uczenia (ang. learning_rate) i rozmiar porcji danych były dobierane w celu przeprowadzenia jak najefektywniejszego procesu nauki. Trening na potrzeby testów był przeprowadzany na zbiorze o rozmiarze stanowiącym 5% docelowego zbioru danych z losowo wybranymi próbkami z zachowaniem proporcji występowania raka piersi. Wykorzystano ten sposób z uwagi na czasochłonność głębokiego uczenia, które na pełnym zbiorze trwa więcej niż dobę. Trening zakłada dostrajanie wyszkolonego modelu *resnet18* na zbiorze danych ImageNet. W celach testowych trening przeprowadzany jest każdorazowo z użyciem pięciu epok.

Porcje danych tworzone podczas treningu są przechowywane w pamięci operacyjnej RAM lub pamięci karty graficznej, więc ich rozmiar jest limitowany sprzętowo. Wykorzystywany do treningu komputer umożliwiał uruchomienie treningu z rozmiarem porcji danych 64 jako maksymalny, więc taki został wykorzystany do porównania z 32 podczas testów.

5.1.1. Cele eksperymentu

Dobranie jak najlepszych parametrów treningowych uczenia głębokiego w celu uzyskania możliwie najlepszego modelu sieci *resnet18* przystosowanego do rozpoznawania raka piersi z uwzględnieniem ilości poświęconego czasu na trening. Jakość modelu będzie określana na podstawie procentowej dokładności generowanych przez niego wyników.

5.1.2. Plan eksperymentu

Testowaniu będą podlegały następujące parametry:

Tabela 5.1. Porównanie parametrów uczenia

SGD

batch size	learning rate	czas	dokładność [%]
32	0,01	4min 56s	94,6
32	0,001	4min 55s	93,3
64	0,01	4min 50s	**94,7**
64	0,001	4min 58s	93,5

Adam

batch size	learning rate	czas	dokładność [%]
32	0,01	5min 1s	89,3
32	0,001	5min 11s	93,1
64	0,01	4min 49s	88,8
64	0,001	4min 59s	93,2

— optymalizator - porównanie SGD i Adam, dwóch najbardziej popularnych metod,
— learning rate - wielkość zmian wprowadzanych w modelu podczas propagacji wstecznej w formie liczby wyrażonej jako ułamek, najczęściej 0,01 lub 0,001 i takie też wartości są porównywane,
— batch size - rozmiar porcji danych wprowadzanych równocześnie do sieci, często wykorzystywany rozmiar to 32 instancje, porównany zostanie do rozmiaru 64.

Planowane jest przetestowanie każdej kombinacji i wybranie jednej, która przynosi najlepsze rezultaty. Dla każdego optymalizatora przetestowany zostanie kolejno rozmiar porcji 32 i 64 oraz obydwa stopnie uczenia na każdy z rozmiarów. Wyniki zostaną przedstawione w tabeli.

5.1.3. Wyniki eksperymentu

Tabela 5.1 przedstawia wyniki eksperymentu. Najlepszą kombinacją okazało się użycie optymalizatora SGD ze stopniem uczenia 0,01 i wielkością porcji danych 64. Taka kombinacja została użyta do finalnego uczenia głębokiego pod kątem wykrywania raka piersi na pełnym zbiorze danych. Dodatkowym atutem porcji 64 jest ukończenie procesu szybciej, co ma ogromny wpływ na trening na pełnym zbiorze danych trwający kilkadziesiąt godzin.

5.2. KLASYFIKACJA RAKA PIERSI NA ZBIORZE WALIDACYJNYM

Po wytrenowaniu algorytmu i stworzeniu interfejsu użytkownika aplikacji oraz logiki dotyczącej przetwarzania wskazanych przez użytkownika zdjęć dokonano testów manualnych weryfikujących poprawność działania aplikacji i uzyskiwanych wyników na zbiorze danych odseparowanym podczas przygotowywania środowiska eksperymentalnego. Sprawdzenie aplikacji to finalne potwierdzenie jej skuteczności i szukanie ewentualnych błędów. Na tym etapie testowano skrajne przypadki i próbowano zdestabilizować działanie aplikacji w celu szukania braków w zabezpieczeniach. Testy te pomogły również dopracować wizualny

aspekt aplikacji. Pozwoliły one również zweryfikować poprawność wytrenowania modelu *resnet18*.

5.2.1. Cele eksperymentu

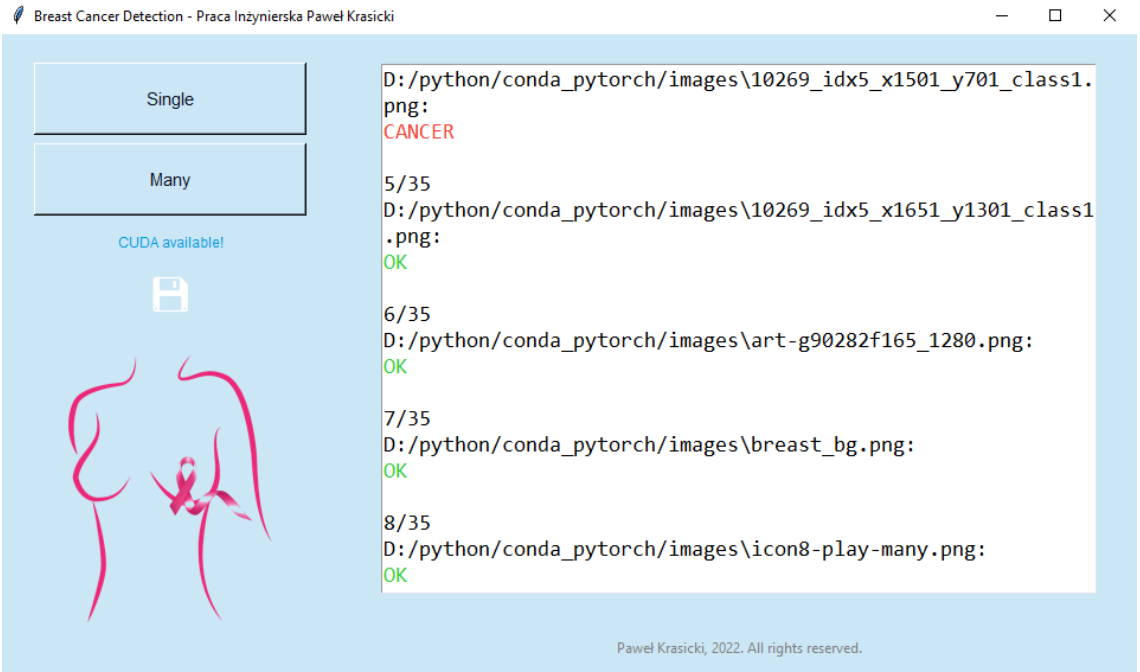
Zweryfikowanie poprawności działania aplikacji i poprawa ewentualnych błędów przed oddaniem produktu użytkownikowi końcowemu oraz sprawdzenie jakości klasyfikacji.

5.2.2. Plan eksperymentu

Testowaniu będą podlegały następujące funkcjonalności aplikacji:

— wykrywanie raka piersi na pojedynczym zdjęciu
— wykrywanie raka piersi na zbiorze obrazów we wskazanym katalogu
— wyświetlanie i odczytywanie wyników
— zapisywanie wyników analizy

Każdy z testowanych przypadków powinien być sprawdzony wielokrotnie przy wprowadzaniu prawidłowych i nieprawidłowych danych, przy próbie wykonania akcji wtedy, kiedy jest to możliwe i wtedy kiedy nie powinno się tego robić. Przy wykryciu nieprawidłowości należy je poprawić, a następnie przetestować aplikację ponownie aż do momentu, w którym nie zostanie wykryty żaden problem. Gdyby okazało się, że algorytm źle klasyfikuje raka piersi, należałoby poprawić trening.



Rys. 5.1. Test masowej analizy zdjęć - wyniki zgodne z oczekiwanymi

5.2.3. Wyniki przeprowadzonego wykrywania raka piersi

Po przeprowadzeniu wszystkich założonych testów aplikacji wykryto, że zapis wyników jest dostępny po uruchomieniu programu, czyli wtedy kiedy nie ma czego zapisywać, więc błąd ten wyeliminowano. Pozostałe aspekty aplikacji działały bez zarzutu. Ustawiona początkowo wartość 16 rozmiaru porcji danych analizowanych naraz przez model podczas przetwarzania całego folderu obrazów została zmieniona na 4, aby zajętość pamięci karty graficznej spadła poniżej 1 GB, a płynność zwracania informacji użytkownikowi aplikacji wzrosła. Model okazał się dobrze wytrenowany i we wszystkich zdjęciach weryfikujących poprawność rozpoznawania raka piersi nie pomylił się ani razu. To też pokazuje i udowadnia potrzebę i zaletę przeprowadzenia testów treningu.

5.3. WNIOSKI

Przeprowadzenie testów zarówno na poziomie uczenia głębokiego sieci CNN, jak i po wykonaniu aplikacji było kluczowe do stwierdzenia poprawności jej działania. Dzięki testom wykonywanym podczas treningu dobrano najlepsze parametry i po pełnym uczeniu sieci była ona gotowa do działania, bez konieczności dalszych popraw. Przetestowanie pozostałych aspektów aplikacji pozwoliło na uniknięcie niepożądanych błędów, które niedoświadczony użytkownik mógłby wywołać nieumiejętnym korzystaniem z aplikacji.

6. UŻYTKOWANIE APLIKACJI

Rozdział ten przedstawia skróconą instrukcję użytkowania aplikacji dla osób, które pierwszy raz mają z nią styczność. Dostęp do aplikacji rozpoczyna się poprzez otworzenie pliku aplikacji z rozszerzeniem *.exe* o nazwie *bcd.exe* będącej skrótem od *Breast Cancer Detection*. Aplikacja została sprawdzona pod kątem prawidłowego działania na systemach Windows 10 i 11.

Po uruchomieniu aplikacji istnieje możliwość przejścia na tryb pełnoekranowy poprzez naciśnięcie klawisza F11. Wyjście z trybu pełnoekranowego odbywa się klawiszem Escape. Zmianę motywu aplikacji na ciemny można wywołać klawiszem b. Wciskając go ponownie, można przejść na motyw jasny.

Uwaga! Nie należy przenosić pliku *bcd.exe* z lokalizacji, w której się znajduje. Aplikacja przestanie działać z powodu braku potrzebnych plików, które wykorzystuje. W przypadku chęci przeniesienia aplikacji np. na pulpit należy utworzyć skrót do pliku *bcd.exe* i przenieść go w wybrane miejsce. Nie zaleca się też zamykania aplikacji podczas wykonywania operacji wykrywania raka piersi na obrazach.

6.1. ANALIZA JEDNEGO ZDJĘCIA POD KĄTEM WYSTĘPOWANIA RAKA PIERSI

W celu przeanalizowania posiadanego zdjęcia i zweryfikowania czy zawiera ono raka piersi trzeba nacisnąć pierwszy od góry przycisk na panelu sterowania, służący do rozpoczęcia procesu dokładnej analizy pojedynczych zdjęć. Następnie należy wskazać zdjęcie i je zatwierdzić. Proces analizy rozpocznie się automatycznie, a po jego zakończeniu na panelu wynikowym pojawią się stosowne komunikaty i wynik detekcji. W tym samym czasie otworzy się widok zdjęcia na wykresie z podpisem w postaci wyniku i jego pewności. Aby powrócić do tekstowego panelu wynikowego, należy zamknąć widok wykresu, naciskając symboliczny przycisk "X" w prawym górnym rogu wykresu. Trzeba uważać, żeby nie pomylić go z systemowym przyciskiem "X" zamykającym aplikację.

6.2. ANALIZA WIELU ZDJĘĆ POD KĄTEM WYSTĘPOWANIA RAKA PIERSI

Masowa analiza zdjęć na dużym zbiorze danych rozpoczynana jest kliknięciem drugiego przycisku i wybraniem pożądanego folderu. Proces analizy rozpocznie się automatycznie, a na panelu wynikowym będą pokazywały się kolejne wyniki dla poszczególnych zdjęć.

Proces zakończy się, kiedy wszystkie zdjęcia zostaną przeanalizowane, a na panelu wynikowym zostanie wypisane podsumowanie informujące, ile zdjęć zawierało raka piersi. Analiza masowa nie tworzy widoku wykresu ze zdjęciem.

6.3. ZAPIS WYNIKÓW

Zapis wyników możliwy jest po skończonym przetwarzaniu jednego bądź wielu zdjęć. Możliwość zapisu zostanie zakomunikowana graficznie poprzez zmianę przycisku z lekko przezroczystego na mocno widoczny. Po jego kliknięciu pojawi się okno systemowe, w którym będzie możliwość wyboru folderu. Po wybraniu pożądanej lokalizacji należy wpisać nazwę pliku i zatwierdzić.

Uwaga! Rozpoczęcie kolejnej analizy masowej bez zapisania dotychczasowych wyników spowoduje ich utratę.

6.4. UWAGI KOŃCOWE

Warto zapamiętać, że:

— w wybranym folderze do detekcji masowej może znajdować się nie więcej niż 1000 zdjęć,
— proces analizy zdjęć obsługuje formaty obrazów *.png*, *.jpeg*, *.jpg* oraz *.gif*,
— plik wynikowy zapisywany jest w formacie *.txt* i próby zmiany tego formatu mogą skutkować niemożliwością późniejszego odczytania wyników,
— detekcja raka piersi na obrazach jest procesem czasochłonnym, więc należy wygospodarować odpowiednią ilość czasu.

Ograniczenie dotyczące maksymalnie tysiąca zdjęć analizowanych naraz jest celowe i podyktowane rozsądkiem z uwagi na przechowywanie wyników w pamięci operacyjnej komputera i czasochłonności procesu analizowania obrazów.

PODSUMOWANIE

Podczas prac nad aplikacją zrealizowano wszystkie wymienione założenia i dodano kilka funkcjonalności bonusowych, takich jak prezentacja analizowanego obrazu na wykresie oraz ciemny i jasny motyw aplikacji. Wyuczony model okazał się spełniać swoje zadanie, a pomogły w tym wielokrotne testy wykonane na etapie treningu. Skończona aplikacja z wytrenowanym modelem *resnet18* stanowi idealne narzędzie do wykrywania raka piersi na obrazach histopatologicznych.

Podczas realizacji projektu problemem okazało się początkowo wykorzystywanie do obliczeń samego procesora, z użyciem którego trenowanie można było szacować na kilkadziesiąt dni. Problem rozwiązano przez zainstalowanie najnowszych sterowników CUDA. Finalnie trening sieci zajął nieco ponad 3 dni. Kluczowe zadania potrzebne do stworzenia modelu sieci oraz aplikacji przebiegły pomyślnie i bez żadnych problemów.

6.5. TRENDY ROZWOJOWE

W przyszłości aplikację można rozbudować o dodatkową funkcjonalność rozpoznawania raka piersi na zdjęciach rentgenowskich, które są zdecydowanie bardziej powszechne i częściej wykonywane. W tym celu należałoby wytrenować wykorzystywany model dodatkowym zbiorem uczącym zawierającym obrazy rentgenowskie. Aplikację dałoby się również rozbudować o bazę danych przechowującą wyniki przeprowadzonych wykrywań raka piersi i konta użytkowników, którzy mogliby śledzić historię swoich dokonań bezpośrednio w aplikacji.

6.6. WNIOSKI

Tematyka uczenia głębokiego jest bardzo rozwojowym działem informatyki, przez co praca z nią sprawia wrażenie obcowania z przyszłościową technologią. Sieci konwolucyjne to potężne narzędzie do wykrywania obiektów na obrazach, a branża medyczna jest doskonałym miejscem na ich wykorzystanie. Aplikacja wykrywająca raka piersi jest przydatnym i funkcjonalnym narzędziem potrafiącym wykonywać pracę lekarza. Rozpoznawanie raka piersi na zdjęciach jest również ciekawe, a dzięki aplikacji na tyle proste, że dowolny użytkownik powinien wykonać ten proces bez odczucia zmęczenia i konieczności posiadania wiedzy specjalistycznej.

BIBLIOGRAFIA

[1] *Breast histopathology images from kaggle*, `https://www.kaggle.com/datasets/paultimothymooney/breast-histopathology-images`.

[2] Aggarwal, C.C., *Data Classification* (Springer International Publishing, Cham, 2015), str. 285–344.

[3] Albawi, S., Mohammed, T.A., Al-Zawi, S., *Understanding of a convolutional neural network*, w: *2017 international conference on engineering and technology (ICET)* (Ieee, 2017), str. 1–6.

[4] Antczak, K., *Uczenie głębokie w diagnostyce medycznej*, Symulacja w Badaniach i Rozwoju. 2016, tom 7, 3-4.

[5] Bayramoglu, N., Kannala, J., Heikkilä, J., *Deep learning for magnification independent breast cancer histopathology image classification*, w: *2016 23rd International conference on pattern recognition (ICPR)* (IEEE, 2016), str. 2440–2445.

[6] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S., *Quantum machine learning*, Nature. 2017, tom 549, 7671, str. 195–202.

[7] Carbonell, J.G., Michalski, R.S., Mitchell, T.M., *1 - an overview of machine learning*, w: *Machine Learning*, pod red. R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Morgan Kaufmann, San Francisco (CA), 1983), str. 3–23.

[8] Chmielik, E., Łuczyńska, E., *Biopsja gruboigłowa piersi–wytyczne diagnostyczne*, Polish Journal of Pathology Supplement. 2012, tom 2011, 4, str. 13–19.

[9] Fisher, C., Egan, M., Smith, P., Wicks, K., Millis, R., Fentiman, I., *Histopathology of breast cancer in relation to age*, British journal of cancer. 1997, tom 75, 4, str. 593–596.

[10] Goodfellow, I., Bengio, Y., Courville, A., *Deep learning* (MIT press, 2016).

[11] He, H., Garcia, E.A., *Learning from imbalanced data*, IEEE Transactions on knowledge and data engineering. 2009, tom 21, 9, str. 1263–1284.

[12] He, K., Zhang, X., Ren, S., Sun, J., *Deep residual learning for image recognition*, w: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), str. 770–778.

[13] Imambi, S., Prakash, K.B., Kanagachidambaresan, G., *Pytorch*, w: *Programming with TensorFlow* (Springer, 2021), str. 87–104.

[14] Janssens, J.P., Vandeloo, M., *Rak piersi: bezpośrednie i pośrednie czynniki ryzyka związane z wiekiem i stylem życia*, Nowotwory. Journal of Oncology. 2009, tom 59, 3, str. 159–159.

[15] Jassem, J., Krzakowski, M., *Rak piersi*, Onkologia w Praktyce Klinicznej-Edukacja. 2018, tom 4, 4, str. 209–256.

[16] Jassem, J., Krzakowski, M., Medica, W.V., *Rak piersi: praktyczny przewodnik dla lekarzy* (Via Medica, 2009).

[17] Ketkar, N., Moolayil, J., *Introduction to pytorch*, w: *Deep learning with python* (Springer, 2021), str. 27–91.

[18] LeCun, Y., Bengio, Y., Hinton, G., *Deep learning*, nature. 2015, tom 521, 7553, str. 436–444.

[19] Liu, T., Fang, S., Zhao, Y., Wang, P., Zhang, J., *Implementation of training convolutional neural networks*, arXiv preprint arXiv:1506.01195. 2015.

[20] Liu, X.Y., Wu, J., Zhou, Z.H., *Exploratory undersampling for class-imbalance learning*, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). 2008, tom 39, 2, str. 539–550.

[21] O'Shea, K., Nash, R., *An introduction to convolutional neural networks*, arXiv preprint arXiv:1511.08458. 2015.

[22] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., *Pytorch: An imperative style, high-performance deep learning library*, w: *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., 2019), str. 8024–8035.

[23] Pointer, I., *Programming pytorch for deep learning: Creating and deploying deep learning applications* (O'Reilly Media, 2019).

[24] Ponikowska, M., *Procedury pobierania materiału i przygotowywania preparatów do badań histopatologicznych*, Życie Weterynaryjne. 2020, tom 95, 02.

[25] Tabor, J., Śmieja, M., Struski, Ł., Spurek, P., Wołczyk, M., *Głębokie uczenie: wprowadzenie*. 2022.

[26] Targ, S., Almeida, D., Lyman, K., *Resnet in resnet: Generalizing residual architectures*, arXiv preprint arXiv:1603.08029. 2016.

[27] Tkaczuk-Włach, J., Sobstyl, M., Jakiel, G., *Rak piersi–znaczenie profilaktyki pierwotnej i wtórnej*, Menopause Review/Przegląd Menopauzalny. 2012, tom 11, 4, str. 343–347.

[28] Veta, M., Pluim, J.P., Van Diest, P.J., Viergever, M.A., *Breast cancer histopathology image analysis: A review*, IEEE transactions on biomedical engineering. 2014, tom 61, 5, str. 1400–1411.

[29] Weiss, K., Khoshgoftaar, T.M., Wang, D., *A survey of transfer learning*, Journal of Big data. 2016, tom 3, 1, str. 1–40.

[30] Werbos, P.J., *Backpropagation through time: what it does and how to do it*, Proceedings of the IEEE. 1990, tom 78, 10, str. 1550–1560.

[31] Wu, Z., Shen, C., Van Den Hengel, A., *Wider or deeper: Revisiting the resnet model for visual recognition*, Pattern Recognition. 2019, tom 90, str. 119–133.

[32] Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K., *Convolutional neural networks: an overview and application in radiology*, Insights into imaging. 2018, tom 9, 4, str. 611–629.

SPIS OZNACZEŃ I POJĘĆ

— CNN - konwolucyjne sieci neuronowe
— patomorfologa - dziedzina medycyny zajmująca się analizowaniem zdjęć tkanek pod kątem występowania chorób
— obrazy histopatologiczne - obrazy tkanki piersiowej pochodzące z badania pod mikroskopem wyciętego materiału biologicznego
— tensor - wielowymiarowa macierz będąca reprezentacją zdjęcia wykorzystywaną na potrzeby uczenia głębokiego
— CUDA - sterowniki kart graficznych Nvidia umożliwiające wykonywanie obliczeń na GPU
— pooling - warstwa redukcji wymiaru poprzez łączenie
— stride - wielkość przemieszczenia okna filtra lub poolingu
— GUI - graficzny interfejs użytkownika
— kernel - filtr używany podczas operacji konwolucji

SPIS RYSUNKÓW

SPIS LISTINGÓW

SPIS TABEL