POLITECHNIKA ŚLĄSKA

WYDZIAŁ INŻYNIERII BIOMEDYCZNEJ

Zastosowanie sztucznych sieci neuronowych w wykrywaniu obecności kontrastu w badaniach tomografii komputerowej

---

Sandra Wieczorek

8 lutego 2022

Spis treści

Rozdział 1

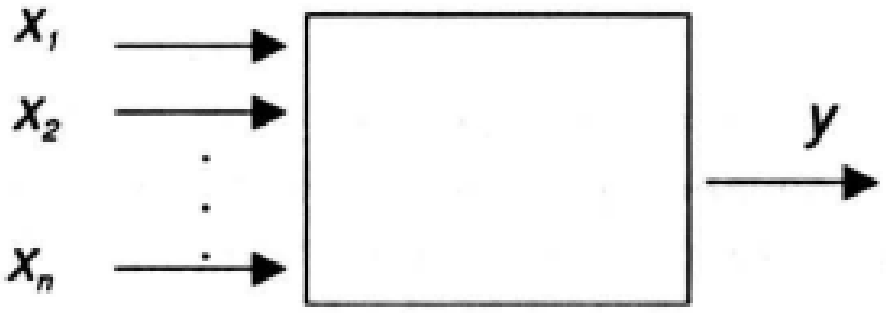
Wstęp teoretyczny

1.1 Sieci neuronowe

Sztuczna sieć nieuronowa (ang. artificial neural network, ANN) jest modelem uczenia maszynowego, który nawiązuje do biologicznych połączeń i sieci nerwowych. Sztuczne sieci neuronowe wykorzystują modelowanie matematyczne do symulacji działania neuronów w mózgu, w tym dendrytów, aksonów i synaps. Analogicznie do biologicznego neuronu, model matematyczny ma wiele wejść o określonej wadze i jedno wyjście. Ma tutaj odzwierciedlenie teoria Hebbowska, która postuluje: "nerwy, które zapalają się razem, łączą się razem". Połączenia węzłów w ANN są ważone na podstawie ich zdolności do zapewniania pożądanych rezultatów. Wynik operacji jest złożeniem obliczeń funkcji aktywacyjnej, która za argument przyjmuje sumę wag. Sieci poddaje się treningowi, w celu wyznaczenia tzw. funkcji kosztu (która reprezentuje błędy) oraz jej optymalizacji. [1]

1.1.1 Neurony

Najprościej, neurony można zdefiniować jako elementy, z których buduje się sieci. Mogą posiadać wiele wejść, ale tylko jedno wyjście.



Rysunek 1.1: Schemat neuronu, który posiada n wejść i jedno wyjście
Źródło: R. Tadeusiewicz *Sieci neuronowe*[1]

Zarówno sygnały wejściowe x_i(i = 1, 2, ..., n) jak i sygnał wyjściowy y mogą przyjmować wartości z pewnego ograniczonego przedziału. Zależność:

$$y = f(x_1, x_2, ..., x_n) \tag{1.1}$$

w uproszczeniu może być również rozpatrywana jako zależność liniowa, wyrażona wzorem:

$$y = \sum_{i=1}^{n} w_i x_i \tag{1.2}$$

gdzie: w_i to wagi synaptyczne. Mogą one podlegać modyfikacjom w procesie uczenia. Stanowią jeden z zasadniczych wyróżników sieci neuronowych jako adaptacyjnych systemów przetwarzania informacji.
Zachowanie pojedynczego neuronu będzie warunkowane pojedynczym wektorem wag W, a jego działaniem sterować będzie macierz wag W_k.[2]

1.1.2 Konwolucyjne sieci neuronowe

Konwolucyjne sieci neuronowe (ang. convolutional neural network, CNN) powstały pośrednio w celu optymalizacji procesów obliczeniowych liniowych sieci neuronowych. Można powiedzieć, że "łatają" niedoskonałości tych algorytmów.

Działanie CNN można uprościć do wyróżnienia dwóch typów warstw tworzących sieć. Pierwsza z nich, warstwa stricte konwolucyjna, przekształca fragmenty obrazu tak, aby wydobyć konkretne cechy. Najczęściej wyszukuje się ich w najbliższym sąsiedztwie. Początkowe warstwy rozpoznają kontury czy plamy barw, a te głębsze np. kształt prostokąta czy elipsy.
Konwolucja jest operacją matematyczną nazywaną również "splotem". Wykorzystuje przekształcenia macierzowe fragmentów np. zdjęcia, które mają za zadanie wykrycie cech charakterystycznych obrazu.
Drugim typem warstw jest warstwa łącząca. Redukuje ona wymiary cech konwolucyjnych, które zostały stworzone w poprzedniej grupie warstw. Jako rezultat, otrzymujemy zmniejszoną macierz, której poszczególne elementy są maksymalnymi wartościami macierzy wejściowej, po filtracji.
W zdecydowanej większości, sieci konwolucyjne są sieciami ze sprzężeniem w przód - oznacza to, że nie posiadają zapętleń. Warstwy neuronów ułożone są tak, aby najpierw obliczać odpowiedzi pierwszej warstwy, potem następnej i tak dalej. Warstwy konwolucyjne zawsze są tej samej wymiarowości, co dane wejściowe, jednak jej rozmiary są zmniejszone. Wynika to z faktu, że każdy neuron z warstwy $n+1$ podłączony jest do wielu neuronów z warstwy n lub bezpośrednio do danych wejściowych. Najczęściej używane jest okno prostokątne o wymiarach PxQ. Dopuszczalne jest użycie wielu podwarstw neuronów, które tworzą tzw. mapę cech.
Ważnym zagadnieniem przytaczanym w kontekście sieci konwolucyjnych jest **perceptron**. Nazwą tą określa się warstwę w pełni połączoną, czyli taką, która posiada wszystkie możliwe połączenia między jej neuronami, a wyjściami neuronów warstwy poprzedniej. W CNN warstwa ta występuje jako jedna z warstw końcowych. Jej wartości wejściowe to cechy zwracane przez poprzedzające ją warstwy konwolucyjne.[3]
Dzięki gromadzeniu danych o cechach wspólnych CNN znakomicie radzą sobie w analizie obrazów medycznych większości modalności: TK, MRI, PET czy RTG. Najczęściej stosuje się je do segmentacji interesujących diagnostę obszarów, klasyfikacji chorób czy detekcji zmian patologicznych.
Tworząc algorytm konwolucyjnej sieci neuronowej należy zdefiniować najważniejsze parametry:

- Kernel size (rozmiar jądra) - nazywany również rozmiarem filtra. Określa wielkość przesuwanego okna nad wejściem. Jego wybór ma ogromny wpływ na klasyfikację obrazu. Małe jądra wydobywają znacznie więcej informacji z obrazu niż duże. Prowadzą również do zmniejszenia wy-

miarów warstw, dzięki czemu architektura sieci jest głębsza. Duże jądra dobrze sprawdzają się w przypadku większych elementów.

- Padding (wypełnienie) - parametr ten definiuje, w jaki sposób obramowanie próbki jest przetwarzane. Umożliwia otrzymanie rozmiaru wejścia takiego samego jak wyjścia. Dodaje na krawędziach sztuczne wagi (zazwyczaj równe 0).
- Strides (kroki) - parametr określa liczbę kroków (pikseli) przesunięcia jądra w jednej iteracji. Dla warstw splotowych najczęściej wynosi 1, czyli okno 3x3 przesuwane jest o jeden piksel w każdej iteracji.
- Pooling layer (warstwa łącząca) - służy do zmniejszenia rozmiaru i ilości parametrów do wytrenowania. Dzięki temu, algorytm działa szybciej (mniej cech do przeanalizowania), zużywa mniej zasobów i chroni przez "przeuczeniem" sieci.
- Dropout layer (warstwa porzucenia) - zeruje losowe krawędzi neuronów ukrytych jednostek. Zapobiega to uczeniu się sieci "na pamięć" - co epokę, poprzez zerwanie niektórych połączeń, delikatnie zmienia się architektura sieci.
- Flatten layer (warstwa spłaszczająca) - przekształcenie wielowymiarowej warstwy w wektor jednowymiarowy. Warstwa używana jest do sklasyfikowania, wyodrębnionych w poprzednich warstwach, cech.

[4],[5]

Rozdział 2

Cel pracy i założenia

2.1 Cel pracy

Celem projektu było zaprojektowanie algorytmu sztucznej sieci neuronowej do klasyfikacji obrazów tomografii komputerowej. Sieć miała dzielić obrazy na dwie grupy: badanie z kontrastem i bez środka kontrastowego.

2.2 Założenia projektu

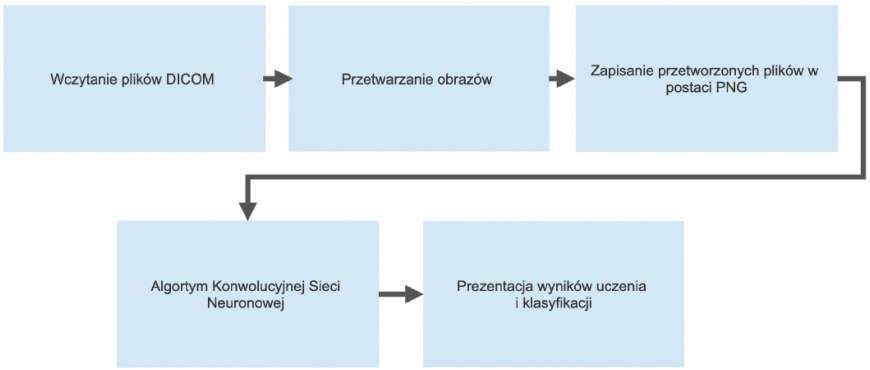
- Program wczytuje bazę obrazów medycznych o rozmiarach 512x512 w formacie DICOM.
- Obrazy są wstępnie przetwarzane przy użyciu filtra pasmowo-przepustowego. Następnie, obrazy są zapisywane w formacie .png z trzydziestodwubitowym koloru kodowaniem pikseli.
- Przetworzona baza zostaje wczytana do skryptu z siecią jako nowa baza danych do uczenia i testowania sieci.
- Zbiór danych dzielony jest na dwie grupy: testowa i treningowa. Zbiory wchodzą na wejście sieci, która zostaje poddana treningowi.
- Po procesie uczenia, sieć zwraca osobno dwie grupy obrazów. Wyświetlana jest informacja odnośnie funkcji straty (czyli błędu pomiędzy wynikiem referencyjnym, a otrzymanym) podczas treningu i walidacji.

Rozdział 3

Metodologia i specyfikacja projektu

3.1 Metodologia

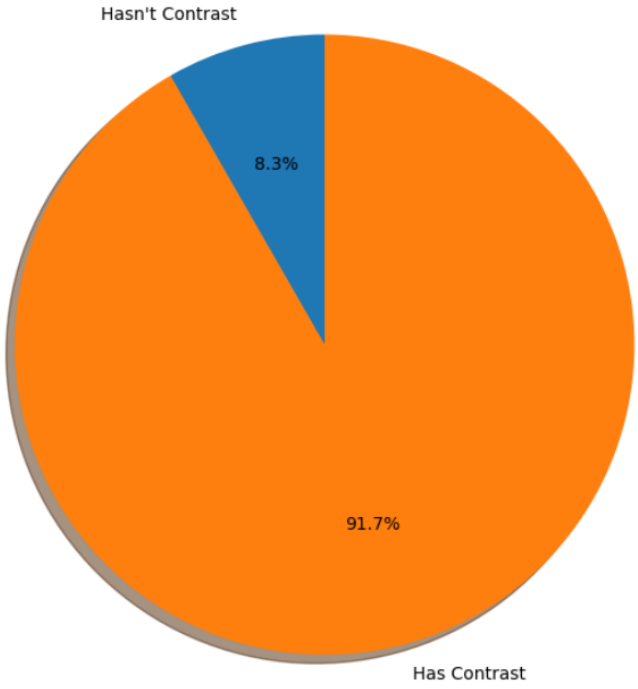
W celu zrealizowania przyjętych wcześniej założeń, stworzono program w języku Python. Realizuje on projekt, według schematu na rys. 3.1



Rysunek 3.1: Schemat blokowy projektu

3.1.1 Wczytanie plików DICOM

Zbiór testowy i uczący zawiera się w 616 zanonimizowanych plikach w formacie DICOM. Są to badania jamy brzusznej i klatki piersiowej z lub bez kontrastu. Obrazy wczytane są do listy, a następnie przetworzone na tabelę trójwymiarową: 616x512x512, gdzie wartość x to każde kolejne zdjęcie, a pozostałe (y i z) odpowiadają wymiarowi obrazu. Każdemu pikselowi została przypisana wartość z zakresu -2000;2000. Podział ilościowy obrazów z kontrastem i bez przedstawiony jest na wykresie na rys. 3.2

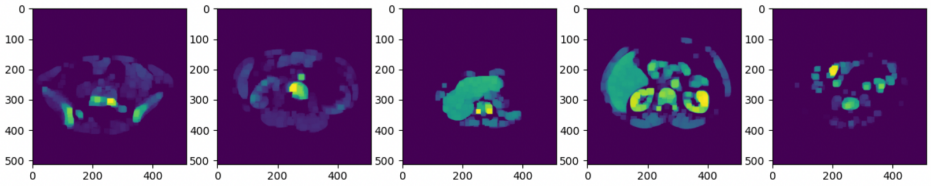


Rysunek 3.2: Podział kategorii obrazów

3.1.2 Przetwarzanie obrazów

Po poprawnym wczytaniu obrazów, program dokonuje preprocessingu na bazie obrazów. Najpierw, użyty jest filtr pasmowo-przepustowy. Wyciszone zostają wartości mniejsze od 950 i większe od 1850. Pozwala to na wyzerowanie sygnału delikatnych tkanek miękkich o niskim współczynniku pochłaniania

promieniowania rentgenowskiego oraz kości, dla których wartości pochłaniania się bardzo wysokie. W celu poprawy jakości obrazów po preprocessingu, dodana została operacja otwarcia przy użyciu okna o wymiarach 15x15. Przykładowe obrazy po preprocessingu przedstawia rys. 3.3



Rysunek 3.3: Przykładowe obrazy po operacjach przetwarzania

3.1.3 Zapisanie przetworzonych plików w formacie PNG

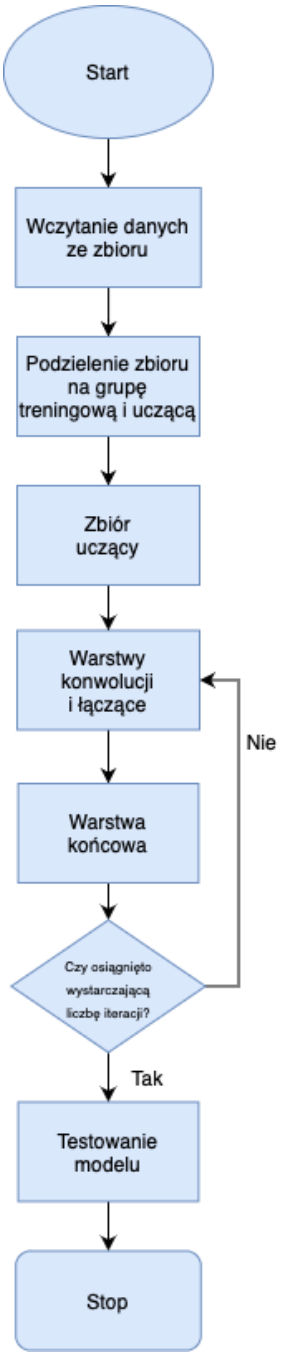
Przetworzone obrazy zostały zapisane w folderze w formacie PNG przy zachowaniu trzydziestodwubitowego kodowania koloru. Jest to jedyny format pozwalający na optymalne przechowywanie danych do uczenia sieci: dzięki bezstratnej kompresji nie powoduje utraty jakości, a wartości kolorów pozostają zachowane.

3.1.4 Algorytm Konwolucyjnej Sieci Neuronowej

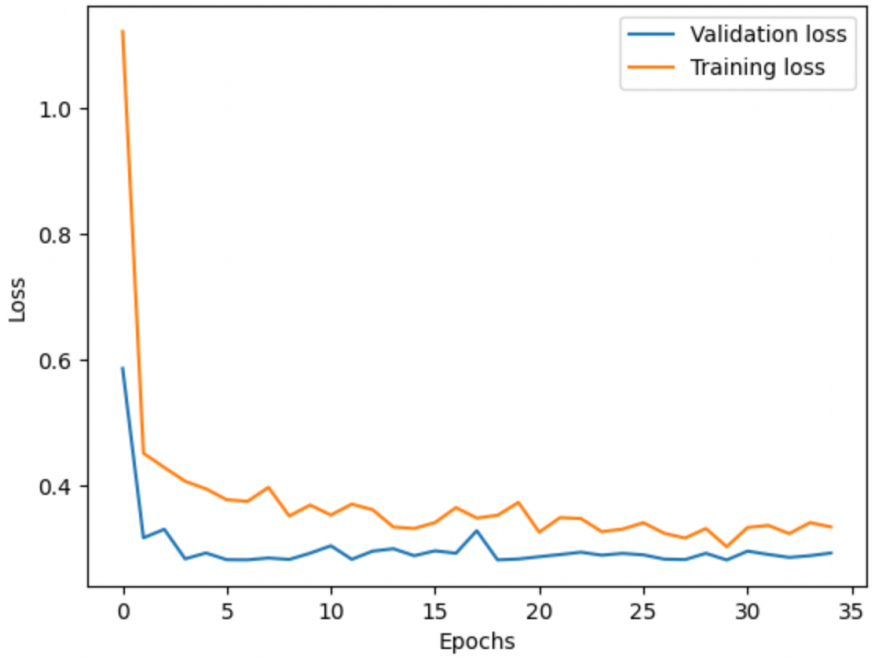
Sieć neuronowa miała za zadanie nauczyć się klasyfikacji obrazów, w zależności od tego, czy do badania został użyty środek kontrastowy czy też nie. Algorytm działa zgodnie ze schematem na rys. 3.4

3.1.5 Prezentacja wyników uczenia i klasyfikacji

W czasie uczenia, wyświetlana jest informacja na temat funkcji straty treningowej i walidacyjnej. Po zakończeniu wszystkich epok, pokazywana jest informacja o dokładności klasyfikacji sieci po procesie uczenia. Przygotowana została również zmienna przechowująca liczbę poprawnie sklasyfikowanych obrazów.
Wykres na rys. 3.5 pokazuje, jak zmieniały się parametry uczenia podczas kolejnych epok.



Rysunek 3.4: Schemat blokowy algorytmu



Rysunek 3.5: Krzywe uczenia sieci

3.2 Specyfikacja projektu

3.2.1 Wybór środowiska programistycznego

Projekt został zrealizowany w języku Python, przy użyciu oprogramowania PyCharm. Decydującym czynnikiem przy jego wyborze była umiejętność posługiwania się nim na dobrym poziomie. Ponadto, język ten bardzo dobrze radzi sobie zarówno z szeroko pojętym przetwarzaniem obrazów jak i machine learningiem.

3.2.2 Użyte biblioteki

W celu realizacji projektu zostały zaimportowane następujące biblioteki:

- numpy,
- os,
- matplotlib,
- pydicom,
- tqdm,
- PIL,
- glob,
- cv2,
- pandas,
- torch,
- torchvision.

3.2.3 Użyte metody

Metody wbudowane

- *listdir* - wczytanie ścieżki do folderu z plikami,
- *pydicom* - wczytanie plików DICOM z folderu,

- *pixel_array* - wydobycie tagu *Pixel Data* z pliku DICOM i zapisanie go jako tablicę liczb,
- *clip* - wartości w tablicy poza zadanym przedziałem przytjmują wartości skrajne przedziału,
- *erode* - przeprowadzenie operacji erozji,
- *dilate* - przeprowadzenie opecjacji dylatacji,
- *imsave* - zapisanie tablicy jako pliki PNG,
- *read_csv* - wczytanie pliku .csv,
- *device* - przeniesienie obliczeń na CPU,
- *print* - wyświetlenie tekstu w konsoli.

Metody zaimplementowane

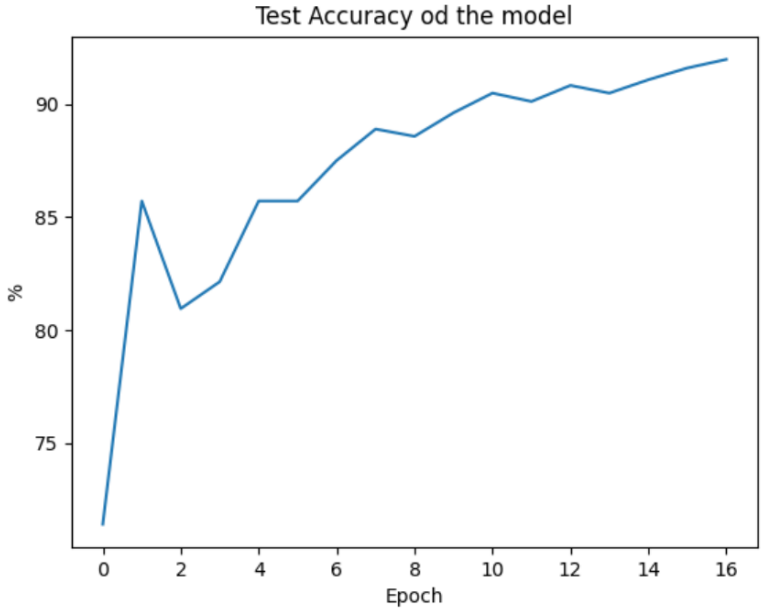
- *muteValues* - wyzerowanie wartości spoza zadanego zakresu,
- *bandpassFilter* - implementacja filtra pasmowo-przepustowego,
- *customDataset* - stworzenie zbioru danych na podstawie przygotowanego zbioru obrazów,
- *CNN* - implementacja algorytmu Konwolucyjnej Sieci Neuronowej.

Rozdział 4

Wnioski i podsumowanie

4.1 Testowanie

Przetestowano działanie modelu dla 123 obrazów testowych. Dokładność nauczonego modelu została przedstawiona na rys. 4.1.



Rysunek 4.1: Krzywa testowania modelu

Stworzono macierz pomyłek dla modelu. Uzyskano następujące wyniki (rys. 4.2):

Predicted values	Actual values		
		True(1)	False(0)
	True(1)	28.0	36.0
	False(0)	24.0	35.0

Rysunek 4.2: Macierz pomyłek

4.2 Wnioski

- Program poprawnie wczytuje pliki DICOM. Przepisanie wartości z tagu *Pixel array* daje bardzo dobre wyniki, zbieżne z zasadą pochłaniania promieniowania.
- Operacja otwarcia przy preprocessingu bardzo dobrze uwidacznia cechy charakterystyczne obrazu, co jest pomocne w procesie uczenia sieci.
- Zapisanie plików w formacie PNG było niezbędne do prawidłowego wczytania plików do zbioru uczącego i testowego. Jest to format korzystający z bezstratnej kompresji, dzięki czemu nie zostały utracone żadne istotne dane.
- Dodanie drugiej warstwy konwolucji znacznie przyspieszyło obliczenia i proces uczenia w porównaniu do sieci z jedną warstwą.
- Ze względu na małą dostępność zbiorów danych zanonimizowanych obrazów medycznych w formacie DICOM, zbiory uczące i testowe były niewielkie. Mimo tego, uzyskano całkiem dobrą dokładność algorytmu, która wynosi 91.9%.
- Niestety, wartości czułości i specyficzności nie dają tak dobrych wyników. Uzyskano następujące rezultaty:
  - czułość: 53,85%,
  - specyficzność: 49,29%

Można poprawić powyższe wyniki rozbudowując bazę o nowe obrazy, poprzez dodanie nowych warstw do sieci lub zmieniając jej parametry.

4.3 Podsumowanie

Ze względu na specyfikę przetwarzanych danych, zbiory wykorzystane w projekcie nie były duże. Obrazy medyczne są bardzo dobrze chronionymi typami danych i nie są udostępniane na szeroką skalę. Dodatkową trudność sprawiał fakt, że biblioteka *PyDicom* nie dopuszczała do sczytania pliku, któremu brakowało tagów. Dostęp do większej liczby danych z pewnością poprawiłby wyniki klasyfikacji. Poprawę skuteczności o 3 punkty procentowe uzyskano przy zwiększeniu zbioru z 430 plików do 616.
Dobrą praktyką, przy małych zbiorach, jest augmentacja danych. Polega to na lekkiej modyfikacji np. obrazów w zbiorze. Edycja może polegać na obróceniu pliku lub zmianie wartości paru pikseli. W przypadku tego projektu, obrócono parę obrazów o 90 stopni w lewo.
Program może być wykorzystywany do testowania wprowadzonych danych do systemu PACS. Zdarzają się sytuacje, gdzie badanie oznaczone jako kontrastowe, tak na prawdę odbyło się bez jego użycia. Algorytm mógłby wychwytywać takie błędy i przekazywać odpowiednie informacje do lekarza opisującego.
Projekt można również w przyszłości rozszerzyć i dodać nowe funkcjonalności. Konwolucyjna sieć neuronowa o zaproponowanej architekturze nie będzie mieć problemów z rozpoznaniem modalności badania czy ustaleniem badanego obszaru ciała.

Bibliografia

[1] R. Tadeusiewicz *Problemy Biocybernetyki*, Wydawnictwo naukowe PWN, Warszawa 1994,

[2] J. Cytowski, J. Gielecki, A. Gola *Cyfrowe przetwarzanie obrazów medycznych, Algorytmy. Technologie. Zastosowania*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2008,

[3] J. Kufel, I. Paszkiewicz, S. Adamski, K. Irlik *Czym jest deep learning, uczenie maszynowe i sztuczne sieci neuronowe? - Przykładeowe zastosowania w medycynie*, Innowacje w medycynie. Przegląd wybranych technologii XXI w., tom II, ArchaeGraph Wydawnictwo Naukowe, Łódź 2021,

[4] https://miroslawmamczur.pl/jak-dzialaja-konwolucyjne-sieci-neuronowe-cnn/

[5] R. Ashraf, M. A. Habib, M. Akram, M. A. Latif, M. A. S. Malik, M. Awais, S. H. Dar, T. Mahmood, M. Yasir, Z. Abbas *Deep Convolution Neural Network for Big Data Medical Image Classification*, IEEE Access vol. 8, 2020, p. 105659 - 105670