Politechnika Wrocławska

Wydział Podstawowych Problemów Techniki

PRACA DYPLOMOWA

Zastosowanie sieci neuronowych do automatycznej segmentacji tkanki nowotworowej na obrazach histopatologicznych

Autor: Jakub Siembida nr alb. 236728
Opiekun: dr inż. Witold Dyrka

Sieć neuronowa, obraz histopatologiczny, rak płuc, głębokie uczenie

Aplikacja komputerowa oznaczająca miejsca obecności komórek nowotworowych płuc na obrazach histopatologicznych typu whole slide image przy pomocy sieci neuronowej trenowanej na zbiorze danych ACDC-Lung HP

1. CEL I ZAKRES PRACY

Nowotwory stanowią poważny problem społeczny. W ostatnich latach obserwuje się wzrost zachorowalności na te choroby. Stanowią one drugą najczęstszą przyczynę zgonów na świecie. Najczęściej diagnozowanymi przypadkami nowotworu są raki płuc. Stanowią one około 20% wszystkich zdiagnozowanych przypadków. Wpływ na tak dużą zachorowalność mają m.in. zły stan powietrza, czy palenie tytoniu. Ze względu na trudną diagnostykę, są to również najbardziej letalne nowotwory. Powodem tak dużej śmiertelności jest również nieodpowiedni dobór terapii [1][2].

W przeciągu ostatnich lat nastąpił duży wzrost zainteresowania sztuczną inteligencją, w szczególności sieciami neuronowymi. Dzięki nim możliwa jest implementacja zadań, które dla człowieka wydają się proste, lecz dla komputera są skomplikowane, np. rozpoznawanie twarzy, głosu, gra w szachy. Dzięki wykorzystaniu algorytmu uczenia maszynowego możliwe jest poprawne analizowanie przez sieć danych, które nie zostały użyte w procesie uczenia [3].

Zainteresowanie analizą danych medycznych przy pomocy algorytmów sztucznej inteligencji wzrasta z roku na rok. Spowodowane jest to m.in. brakiem dostatecznej liczby specjalistów z danej dziedziny, lub czasem trwania analizy danych przez lekarza. Opracowywane są coraz to nowsze algorytmy pozwalające na uzyskanie dokładniejszej oceny obrazów medycznych, sygnałów (np. EEG, EKG), porównywalnych z oceną specjalisty w danej dziedzinie, w krótszym czasie [4].

Celem niniejszej pracy jest stworzenie oprogramowania wspomagającego ocenę obrazów histopatologicznych płuc zapisanych jako obrazy całego preparatu histopatologicznego (ang. whole slide image, WSI) [5][6], wykorzystującego konwolucyjną sieć neuronową oznaczającą tkankę nowotworową na obrazie. Jako zestaw danych został wybrany zbiór udostępniony wraz z ACDC-Lung HP challenge [7]. Wybrano ten zestaw, gdyż składa się on z całych skanów skrawków płuc, dzięki czemu nie jest narzucony wymiar obrazu. Jako język programowania wybrano Python 3.7, który jest na dzień dzisiejszy jednym z najlepszych języków programowania do implementacji sieci neuronowej, dzięki obecności zoptymalizowanych wydajnościowo modułów do uczenia maszynowego.

2. NOWOTWÓR PŁUC

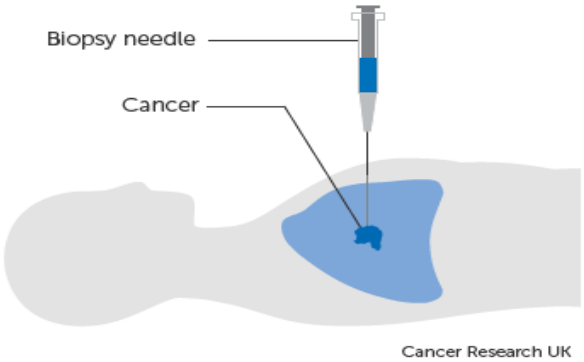
W ostatnich latach zanotowano znaczny wzrost zapadalności na nowotwory płuc. Według statystyk WHO z 2018r. stanowią drugą najczęstszą przyczynę zgonów na świecie. Najczęściej diagnozowanym i zarazem najbardziej śmiertelnym nowotworem jest rak płuc [1].

Rakiem płuc nazywamy nowotwór wywodzący się z tkanki nabłonkowej wyścielającej drogi oddechowe [8]. Czynnikami sprzyjającymi powstawaniu zmian nowotworowych w płucach są m.in. palenie tytoniu, ekspozycja na duże dawki promieniowania jonizującego, obecność azbestu w płucach [2]. W obrębie raka płuc wyróżnia się dwa typy: drobno- i niedrobnokomórkowy. Pierwszy z wymienionych rozwija się agresywniej, daje przerzuty w krótszym czasie, niż pozostałe linie. Powoduje to, iż wyleczenie pacjenta cierpiącego na ten rodzaj raka jest trudniejsze [9][10].

Nowotwór niedrobnokomórkowy płuc jest dzielony na trzy linie komórek rakowych: rak gruczołowy, płaskonabłonkowy oraz wielkokomórkowy. Pierwsza z wymienionych charakteryzuje się tworzeniem struktur podobnych do gruczołów, widocznych na obrazowaniu histopatologicznym, oraz obecnością śluzu. Rak płaskonabłonkowy jest silnie związany z paleniem tytoniu, cechuje się występowaniem stanów przedrakowych, takich jak zmiana kształtu komórek i ich funkcji. Rak wielkokomórkowy kształtem przypomina komórki pozostałych dwóch linii komórek rakowych. Tworzy wyraźne guzy, w przeciwieństwie do nowotworu drobnokomórkowego płuc. Komórki tej linii rakowej są zdecydowanie większe od komórek pozostałych dwóch linii [2][10].

3. KLINICZNA DIAGNOSTYKA RAKA PŁUC

Diagnostyka raka płuc polega na wykryciu obecności komórek nowotworowych w tkankach pacjenta. Ważną cechą diagnostyczną jest stopień rozsiania nowotworu w tkance. Aby dobrać odpowiedni typ leczenia, należy przeprowadzić badanie histopatologiczne. Jest to inwazyjna metoda polegająca na poborze tkanki i jej analizy przez lekarza histopatologa [11]. Pozwala ona na dokładne rozpoznanie choroby nowotworowej poprzez przypisanie konkretnej linii rakowej, której obecność jest stwierdzona w próbce. Pozwala to dostosować terapię do potrzeb pacjenta zmniejszając ekspozycje na toksyczne właściwości cytostatyków [12][13].



Rys. 3.1. Wizualizacja biopsji przezskórnej guza płuc. Źródło: [14]

Pobór tkanek do badań może odbyć się na drodze biopsji przezskórnej, zabiegu chirurgicznego lub podczas bronchoskopii. Pierwsza z metod została zaprezentowana na rysunku 3.1. Polega ona na nakłuciu guza igłą biopsyjną i poborze tkanki. Podczas zabiegu wykonywane jest obrazowanie CT w celu umożliwienia lokalizacji guza w płucu. Druga z metod jest używana w momencie, gdy pacjent kwalifikuje się do operacji. Polega na torakotomii (otwarciu ściany klatki piersiowej) i wycięciu chorej tkanki [14].

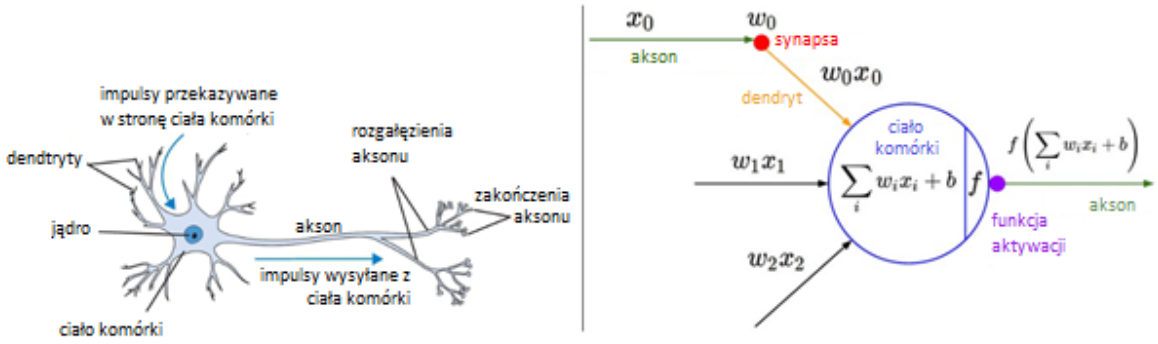
Świeżo pobrane tkanki należy wybarwić przed analizą histopatologiczną. Zastosowanie barwienia ma za zadanie ułatwić lekarzowi rozpoznawanie struktur komórkowych. Najczęściej stosowanym barwieniem jest barwienie hematoksyliną i eozyną. Pierwszy z wymienionych związków chemicznych jest utleniany przy pomocy tlenu z powietrza przy obecności światła (hematoksylina Boehmera), lub tlenku rtęci(II) (hematoksylina Harrisa) do hemateiny. Barwi ona w obecności jonów glinu jądro komórkowe i siateczkę śródplazmatyczną szorstką na kolor niebieski. Eozyna natomiast wybarwia cytoplazmę i włókna kolagenowe na kolor czerwony [13][15].

4. SIECI NEURONOWE

4.1. SIEĆ NEURONOWA A MÓZG LUDZKI

W ostatnich latach można dostrzec duże zainteresowanie dziedziną sztucznej inteligencji. Jest to spowodowane chęcią automatyzacji procesów, mającej na celu wspomaganie pracy człowieka, osiągnięcie powtarzalności wyników i zaoszczędzenie czasu. Jednym z działów sztucznej inteligencji są sieci neuronowe, które swoją strukturą przypominają system połączeń neuronów i przesyłanie sygnału w mózgu ludzkim [16].

Na rysunku 4.1. przedstawiono porównanie komórki nerwowej z modelem matematycznym stanowiącym podstawową jednostkę sieci neuronowej. Każda ze struktur otrzymuje sygnały wejściowe od dendrytów (będących danymi wejściowymi do sieci lub wzmocnieniami neuronów z poprzedniej warstwy) i tworzy sygnał wyjściowy wysyłany aksonem. W modelu matematycznym każdy dendryt ma przypisaną do siebie wagę w_x, powodującą wzmocnienie ($w_x > 0$) lub osłabienie ($w_x < 0$) sygnału. Sygnały od poszczególnych dendrytów są łączone wybraną operacją matematyczną. Wartość sygnału wyjściowego zależy od funkcji aktywacji, mapującej liczby z dziedziny liczb rzeczywistych na oczekiwany przedział (najczęściej $[0, \infty)$ lub $[0, 1]$) [16].



Rys. 4.1. Porównanie komórki nerwowej człowieka z modelem matematycznym neuronu sieci neuronowej. Źródło: [16] Adaptacja

4.2. ZASADA DZIAŁANIA SIECI

Istotną kwestią zrozumienia procesu przetwarzania danych przez sieć neuronową jest sposób przekazywania informacji pomiędzy kolejnymi warstwami oraz algorytm nauki sieci. Poprzez naukę sieci rozumie się ustawienie parametrów sieci (np. wag) na podstawie przekazywanych do sieci danych wejściowych oraz danych wzorcowych [16][17]. W celu

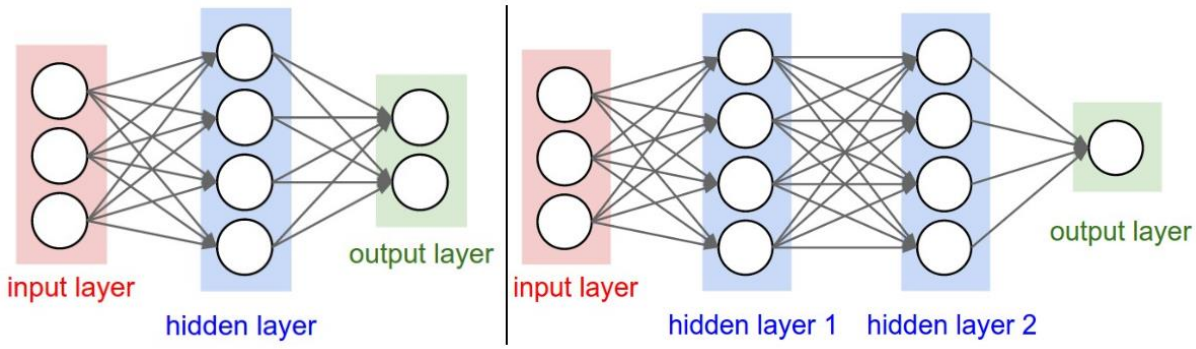
nauki modelu tworzy się zestaw danych składający się z dwóch podzbiorów – zbiór danych do nauczania (na których sieć jest uczona), oraz zbiór danych testowych (na których przeprowadzana jest walidacja). Każdy ze zbiorów zawiera pary dane wejściowe – dane wzorcowe, reprezentujące oczekiwany wynik. Celem nauczania sieci jest ustawienie jej parametrów w taki sposób, aby podobieństwo między predykcją sieci, a danymi wzorcowymi było jak największe, oraz aby dla nigdy nie przetwarzanych przez sieć danych uzyskać jak najdokładniejsza, zgodną z rzeczywistością predykcję [3][16][17].

Pierwsza warstwa neuronów, na podstawie otrzymanych danych wejściowych, wytwarza sygnały (na podstawie formuły opisanej w podrozdziale 4.1.), które są przekazywane do kolejnej warstwy jako sygnał wejściowy. Proces jest powtarzany do momentu osiągnięcia ostatniej warstwy. W tym miejscu predykcja sieci jest porównywana z danymi wzorcowymi przy pomocy określonej funkcji. Wynik porównania jest mnożony przez współczynnik uczenia, będący liczbą z zakresu [0, 1], determinującą szybkość zmian. Wyznaczana jest pochodna powyższego iloczynu i jest dodawana do wag warstwy wcześniejszej. Powyższy proces jest powtarzany dla całego zestawu danych [3].

4.3. Rodzaje sieci neuronowych

Sieci neuronowe można klasyfikować na wiele sposobów. Podział ze względu na kierunkowość przepływu danych, oraz rodzaj dokonywanych operacji to najpopularniejsze formy klasyfikacji sieci [17]. W powyższym podrozdziale zostaną opisane dwa główne typy sieci - wielowarstwowy perceptron i sieć konwolucyjna, oraz w pełni konwolucyjny model sieci U-Net.

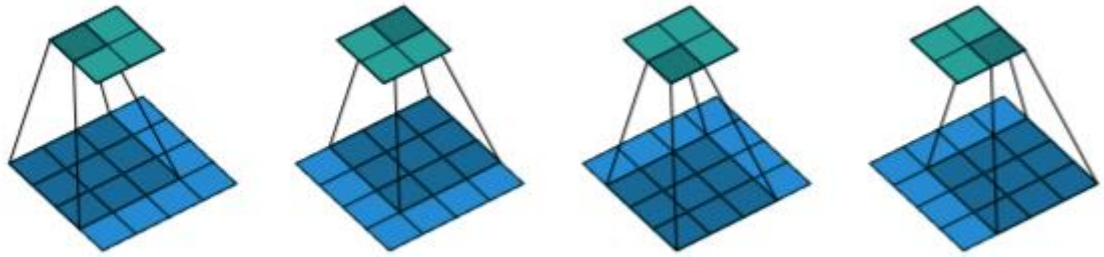
Najprostszym typem sieci jest wielowarstwowy perceptron. Składa się on z warstwy wejściowej, warstwy wyjściowej i przynajmniej jednej warstwy ukrytej (znajdującej się pomiędzy warstwą wejściową, a warstwą wyjściową). Neurony sąsiadujących warstw są połączone na zasadzie każdy z każdym, tzn. neuron a_i należący do warstwy A jest połączony z każdym neuronem z warstwy B, natomiast każdy neuron b_j należący B otrzymuje sygnał od każdego neuronu z warstwy A (Rys. 4.2.) [17].



Rys. 4.2. Dwa modele sieci typu wielowarstwowy perceptron- z jedną ukrytą warstwą (po lewej), z dwiema ukrytymi warstwami (po prawej). Źródło: [17]

Zasadniczą wadą wielowarstwowego perceptronu jest ignorowanie wzajemnego położenia danych, które często jest istotnym czynnikiem definiującym dane wejściowe (np. podczas analizy funkcji matematycznych, obrazu). W takich sytuacjach zdecydowanie lepszym wyborem jest konwolucyjny model sieci neuronowej. Zasada przejścia danych z jednej warstwy do drugiej opiera się na operacji splotu – z tego powodu sąsiadujące warstwy nie są w pełni połączone- tylko pewien region warstwy jest połączony z neuronem kolejnej warstwy (Rys. 4.3.) [17]. Funkcja splotu osiąga lokalne maksimum w sytuacji, gdy dane wejściowe są podobne do struktury wag (wektora w przypadku funkcji, macierzy w przypadku

obrazu) [3][17]. W tym typie sieci wykorzystywana jest też operacja poolingu (grupowania), polegająca na mapowaniu grupy wzmocnień neuronów warstwy A na wzmocnienie jednego neuronu warstwy B. Najczęściej stosowanym typem poolingu jest pooling maksymalny, który jako wzmocnienie neuronu następnej warstwy wybiera wartość maksymalną z grupy wzmocnień z warstwy aktualnej [3][17][18].

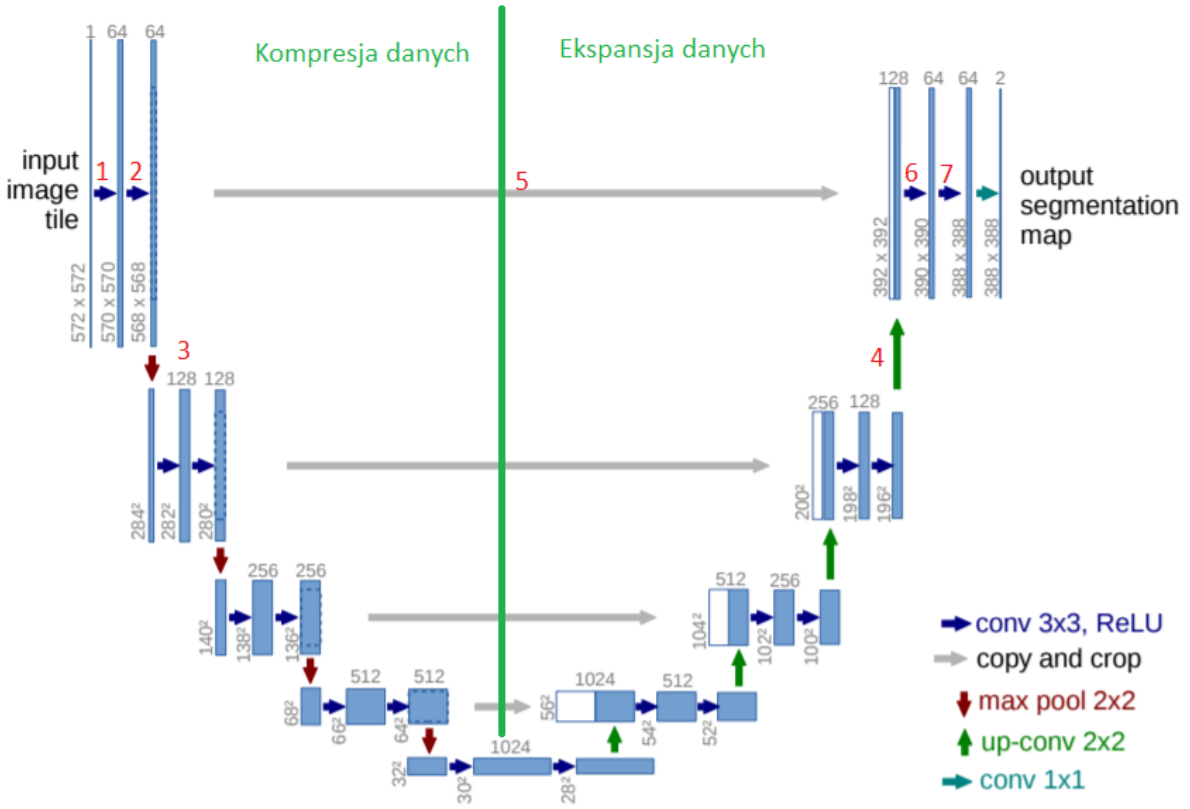


Rys. 4.3. Operacja splotu macierzą 3x3 na macierzy 4x4. Źródło: [18]

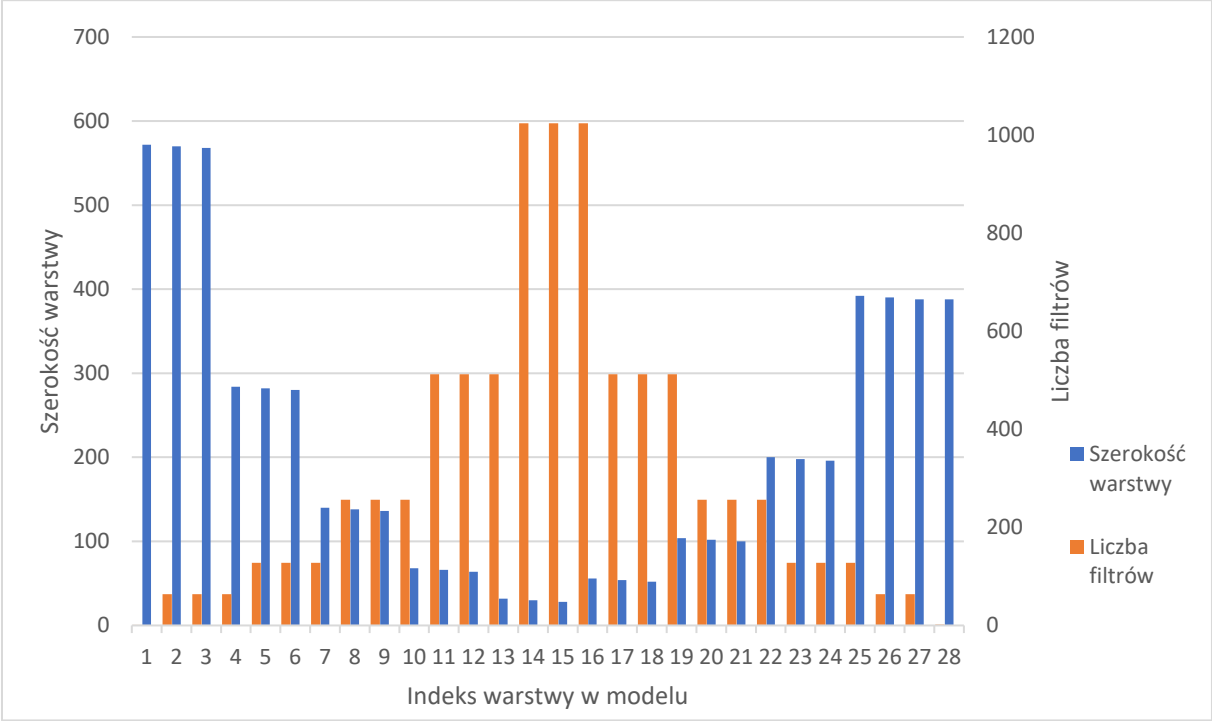
U-Net to w pełni konwolucyjny model sieci zaproponowany w 2015 roku przez Olafa Ronnebergera, Philippa Fischera i Thomasa Broxa z uniwersytetu w Freiburg w celu segmentacji obrazów medycznych [19][20]. Jego nazwa została zainspirowana pierwszym, graficznym przedstawieniem modelu, przypominającym literę „U". Został on użyty po raz pierwszy do segmentacji struktur komórek nerwowych na obrazach mikroskopii elektronowej oraz do oznaczania zmian próchnicznych zębów na obrazach rentgenowskich osiągając lepsze wyniki od innych architektur [19][21].

Przykładowy schemat modelu sieci został przedstawiony na rysunku 4.4. Jest to architektura typu auto-enkoder. Warunkami koniecznymi do klasyfikacji sieci do tej grupy jest zastosowanie kompresji danych poprzez zmniejszanie rozmiaru kolejnych warstw (na lewo od zielonej linii), dekompresja danych poprzez ponowne zwiększanie rozmiarów warstw (na prawo od zielonej linii) oraz podobieństwo między danymi wejściowym, a danymi wyjściowymi sieci. Podczas procesu kompresji, selekcjonowane są najistotniejsze informacje, pozwalające na spełnienie zaprojektowanego zadania (klasyfikacji, detekcji, czy segmentacji) [22][23].

Model U-Net jest w pełni konwolucyjny. Oznacza to, iż nie występuje w nim ani jedna w pełni połączona warstwa neuronów, a główną operacją wykonywaną na danych jest operacja splotu. Dane wejściowe podawane są w tensorze reprezentującym obraz. Przy pomocy dwóch operacji splotu macierzą 3x3 (zaznaczone na obrazie niebieską strzałką) selekcjonowane są wybrane cechy i następnie przy pomocy operacji poolingu maksymalnego macierzą 2x2 z krokiem 2 (zaznaczonej czerwonymi strzałkami) obraz jest zmniejszany czterokrotnie. W ten sposób przeprowadzana jest detekcja cech charakterystycznych. Im więcej filtrów jest obecnych w danej warstwie, tym więcej cech dana warstwa może rozróżnić. Po zmniejszeniu szerokości do żądanej wielkości, wykonywana jest operacja próbkowania w górę (zielone strzałki), polegająca na przetworzeniu jednej komórki w obrazie zadanym na cztery komórki w obrazie wynikowym, co powoduje czterokrotne zwiększenie rozmiaru obrazu. Dodatkowo-wykonywana jest operacja splotu macierzą 2x2 powodującą dwukrotne zmniejszenie liczby filtrów. Cechą wyróżniającą architekturę U-Net od innych, głębokich auto-enkoderów jest operacja dołączenia warstwy pochodzącej z procesu detekcji (szare strzałki). Celem tego zabiegu jest ponowne pozyskanie informacji o położeniu danej cechy na obrazie. Następnie wykonywane są dwie operacje splotu macierzą 3x3. Kroki opisane od próbkowania w górę są powtarzane dokładnie tyle razy, ile razy została wykonana operacja maksymalnego poolingu. Na sam koniec wykonywana jest operacja splotu w celu przetworzenia filtrów obrazu zadanego do ilości filtrów wymaganych do spełnienia zadania [19][20].



Rys. 4.4. Przykładowy schemat modelu sieci typu U-Net. Źródło: [35] Adaptacja



Rys. 4.5. Zależność szerokości warstwy oraz liczby filtrów od indeksu warstwy w modelu przedstawionym na rysunku 8.1.

Na wykresie 4.5. przedstawiono zależności liczby filtrów oraz szerokości warstwy od jej indeksu w modelu. Opisane zależności są przeciwnie monotoniczne. Wraz ze wzrostem

numeru warstwy liczba filtrów rośnie, a szerokość warstwy maleje. Obie charakterystyki osiągają ekstremum dla indeksu równemu połowie indeksu maksymalnego. Po osiągnięciu warstwy o indeksie 15, szerokość warstwy jest proporcjonalna do indeksu warstwy, a liczba filtrów – przeciwnie proporcjonalna. Charakterystyka liczby filtrów od indeksów warstwy dla przedstawionego modelu jest symetryczna, a zależność szerokości warstwy od pozycji warstwy jest asymetryczna. Powodem tego jest fakt, iż obraz wejściowy jest większy, niż obraz wyjściowy z powodu zastosowania operacji splotu.

W architekturze U-Net można wyróżnić dwie grupy operacji – pierwszą obejmującą dwie operację splotu i jedną operację poolingu, oraz drugą, obejmującą próbkowanie w górę, dołączenie warstwy i dwie operacje splotu.

4.4. METRYKI

Aby móc oceniać ilościowo poziom wyuczenia sieci, używa się funkcji dwóch zmiennych- wartości oczekiwanych i predykcji sieci, zwracająca skalar reprezentujący jakość dopasowania [24][25]. Jedną z metryk stosowanych do porównywania segmentacji obrazów jest współczynnik Sørensena-Dice'a, zwany również binarnym współczynnikiem Czekanowskiego [7]. Dla dwóch zbiorów X i Y jest zdefiniowany formułą przedstawioną na równaniu 4.1:

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{4.1}$$

gdzie |X| i |Y| są liczbami kardynalnymi zbiorów X i Y, a $|X \cap Y|$ jest liczbą kardynalną zbioru wspólnych elementów X i Y. W przypadku użytego zbioru danych analizowanymi zbiorami są zbiory pikseli. Wartość tej metryki silnie zależy od liczby elementów występujących w obu zbiorach- im więcej wspólnych elementów jest posiadają oba zbiory, tym wartość metryki jest większa. Antydziedziną współczynnika Sørensena-Dice'a jest przedział [0, 1]. Metryka ta jest też często zapisywana w procentach [26].

W celu porównania danych boolowskich, współczynnik Sørensena-Dice'a opisywany jest zależnoscią 4.2:

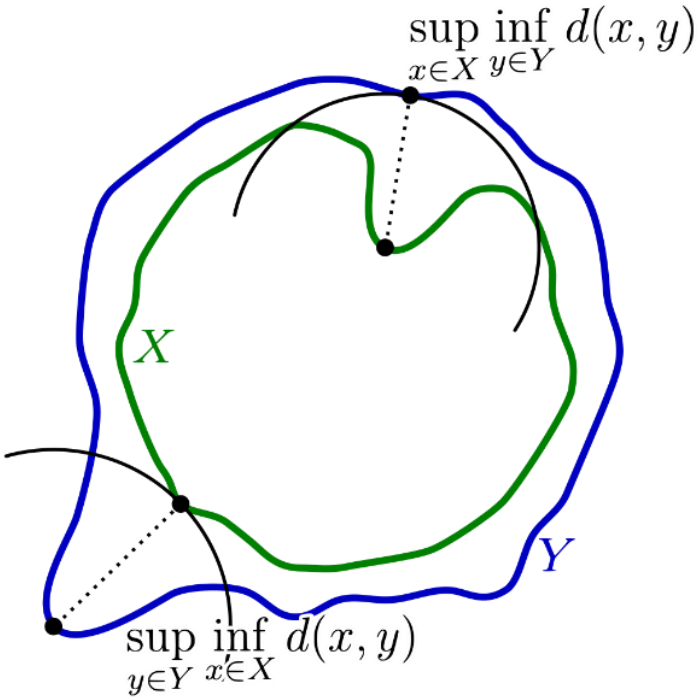
$$DSC = \frac{2TP}{2TP + FP + FN} \tag{4.2}$$

gdzie TP to liczba porównań pozytywnie dodatnich (część wspólna zbiorów wartości wzorcowych i wartości porównywanych), FP to liczba porównań fałszywie pozytywnych (elementy zbioru porównywanego niewystępujące w zbiorze odniesienia), a FN to liczba elementów oznaczonych w obu zbiorach jako brak występowania cechy [26].

Inną metryką szeroko stosowaną do porównywania jakości segmentacji obrazów jest odległość Hausdorffa. Nie analizuje ona powierzchni, jak metryka Sørensena-Dice'a, lecz obwiednie kształtów. Algorytm wyznaczenia odległości Hausdorffa dla dwóch zbiorów punktów leżących na obwiedniach kształtów można opisać w kilku punktach:

1. wyznaczenie dla każdego punktu ze zbioru X odległości do najbliższego punktu ze zbioru Y (najmniejszej odległości z obwiedni X do obwiedni Y)
2. odnalezienie największej z najmniejszych odległości ze zbioru X do zbioru Y
3. wyznaczenie dla każdego punktu ze zbioru Y odległości do najbliższego punktu ze zbioru X (najmniejszej odległości z obwiedni Y do obwiedni X)
4. odnalezienie największej z najmniejszych odległości ze zbioru Y do zbioru X
5. wynikiem jest większa z wartości z punktów 2. i 4.

$$d_H(X,Y) = \max\left\{\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\right\} \qquad (4.3)$$



Rys. 4.6. Sposób wyznaczania odległości Hausdorffa dla dwóch zbiorów X i Y. Źródło: [27]

Matematyczna definicja tej metryki dla dwóch zbiorów X i Y jest opisana równaniem 4.3., a wizualizacja wyznaczonych największych z najmniejszych odległości od obwiedni do obwiedni została przedstawiona na rysunku 4.6. Wyznaczenie maksymalnej z minimalnych odległości punktu *x* należącego do zbioru X do zbioru Y jest równoznaczne ze znalezieniem największego promienia okręgu o środku w punkcie *x*, który byłby styczny do obwiedni zbioru Y, lecz nie przecinałby jej [27].

5. ZBIÓR DANYCH

Zbiorem danych wykorzystanych w niniejszej pracy jest zbiór udostępniony wraz z konkursem „ACDC- Lung HP" na portalu www.grand-challenges.org. Celem konkursu było stworzenie algorytmu lub sieci segmentującej skrawki histologiczne płuc [7].

Zbiór danych został stworzony na podstawie próbek pobranych od 33 pacjentów, w wieku 30-90 lat, leczonych w Pierwszym Szpitalu w Changshy, w Chinach, na oddziale Onkologii płucnej. Próbki zawierały komórki nowotworowe gruczolakoraka płuc, raka płaskonabłonkowego oraz drobnokomórkowego. Pacjenci zostali zdiagnozowani klinicznie pod kątem choroby nowotworowej płuc, nie stosowano u nich radioterapii przed pobraniem próbek, nie byli poddawani transplantacji. Kryteriami odrzucającymi pacjentów były np. obecność dwóch lub więcej pierwotnych linii komórek rakowych płuc, obecność przerzutów innego typu nowotworu do płuc, zdiagnozowana trwale obniżona odporność u pacjenta [28].

5.4. Opis ilościowy zbioru danych

Dane zostały udostępnione jako 150 skanów skrawków do uczenia, oraz 50 skrawków do walidacji. Na każdy skrawek zbioru służącego do nauki składają się dwa pliki: o rozszerzeniu .tif (ang. tagged image format) zawierające całe skany histologiczne oraz .xml (ang. extensible markup language) zawierające oznaczenia miejsc występowania tkanki nowotworowej na obrazie. Zbiór do walidacji zawierał jedynie obrazy skrawków histologicznych. Skany zostały zapisane przy użyciu zautomatyzowanego mikroskopu Olympus VS120 [20].

Każdy skan histologiczny składa się z wielu pomniejszych skanów [7], które ułożone w kolejności zapisu oddają rzeczywistą geometrię próbki. Zapisano kilka przybliżeń próbki w jednym pliku .tif. Pozwala to na wyświetlenie obrazu w rzeczywistej skali, jak i w skali komórkowej.

5.5. Opis jakościowy zbioru danych

Wśród obrazów preparatów histologicznych można wyróżnić dwie grupy: zawierającą jeden duży skrawek oraz zawierającą kilka skanów tej samej tkanki. W drugiej grupie kluczowym jest wybranie tylko tego skanu, który został oznaczony. W przeciwnym razie dany fragment zostanie wycięty tylokrotnie, ile takich samych skanów jest obecnych na obrazie. W efekcie otrzymamy kilka obrazów tego samego fragmentu wycinka, lecz tylko jeden z nich będzie poprawnie oznaczony [7]. Zaistnienie takiej sytuacji może realnie wpłynąć na jakość predykcji sieci.

Kolejnym aspektem jest stosunek powierzchni tkanki do powierzchni całego obrazu. W pierwszej grupie tkanka wypełnia obraz w dużym procencie, co przekłada się na dużą liczbę danych do uczenia sieci pozyskanych z jednego slajdu. W drugiej grupie procent tkanki na obrazie jest zdecydowanie mniejszy - większość obrazu wypełnia białe tło. Konsekwencją tego jest pozyskanie mniejszej liczby obrazów do uczenia z jednego skanu histologicznego, niż w przypadku obrazu wypełnionego tkanką w dużym procencie.

Slajdy preparatów histologicznych zawierają liczne zabrudzenia, czy inne artefakty. Wśród nich wymienić można np. plamy mieszaniny barwiącej w miejscach, gdzie nie znajduje się tkanka, makroskopowe oznaczenia na obrazach, liczne, drobne fragmenty tkanek, czy nachodzenie na siebie dwóch warstw preparatu. Są to typowe artefakty powstające podczas pracy przy preparatach histologicznych płuc [29].

6. Narzędzia

6.1. Język programowania Python

Jako wiodący język programowania projektu został wybrany Python w wersji 3.7.3. Jest to język obiektowy, zajmujący czwarte miejsce w rankingu najczęściej używanych języków programowania oraz drugie miejsce w rankingu najbardziej lubianych języków przez programistów [30].

Główną przewagą Pythona nad innymi wysokopoziomowymi językami, takimi jak Java, C#, jest fakt, iż programy o identycznej funkcjonalności napisane w innych językach statystycznie zawierają więcej linii kodu źródłowego. Powoduje to, iż utrzymanie programu napisanego w Pythonie jest łatwiejsze od oprogramowania napisanego np. w Javie [31]. Kolejną zaletą tego języka jest obecność dobrze przetestowanych, intuicyjnych w użyciu zewnętrznych modułów do np. analizy danych, obróbki zdjęć, czy uczenia maszynowego [32].

6.2. ASAP- Automated Slide Analysis Platform

Wraz z zestawem danych dołączone zostało oprogramowanie napisane w języku C++, w którym histopatolodzy oznaczali zmiany nowotworowe – Automated Slide Analysis

Platform [33]. Pozwala ono na otwarcie slajdów preparatów histologicznych oraz wczytanie i nałożenie na obraz oznaczeń. Dostarczono również fasadę API powyższego programu w języku Python pozwalającą na odczyt slajdu, fragmentu slajdu, czy adnotacji w kodzie programu oraz na generowanie masek prawdy na podstawie adnotacji [7]. Pozwala to na dowolność przy doborze wielkości analizowanych obrazów i zastosowanie własnej obróbki danych.

6.3. Keras i Tensorflow

Jako główne narzędzie do tworzenia sieci neuronowej użyto biblioteki Keras. Jest to jedna z najpopularniejszych bibliotek do tworzenia, nauki i walidacji konwolucyjnych i rekurencyjnych modeli sieci neuronowych w języku Python. Zaletą tej biblioteki jest możliwość użycia jednego z kilku modułów matematycznych, które wykonują obliczenia niezbędne do nauki, czy walidacji sieci. Tworzenie modelów odbywa się w kodzie Pythona, bez udziału plików konfiguracyjnych. Keras zapewnia prostotę tworzenia modelu. Ilość kodu potrzebna do zaimplementowania danej sieci w tej bibliotece jest niewielka w porównaniu do alternatywnych bibliotek [34].

Tensorflow to moduł matematyczny implementujący podstawowe operacje niezbędne do stworzenia modelu sieci neuronowej, napisany w języku Python. Jest często stosowany jako silnik biblioteki Keras, lecz może służyć również jako niezależne środowisko do tworzenia modeli sieci, które oferuje większą kontrolę nad modelem, umożliwia projektowanie bardziej złożonych procesów nauki. Dzięki zastosowaniu rozszerzenia Tensorboard możliwa jest wizualizacja procesu uczenia na wykresach, jak i śledzenie zmian parametrów uczenia. Tensorflow jest używany przez wiele międzynarodowych firm, takich jak Google, Intel, Airbus [35][36].

6.4. Pozostałe użyte biblioteki

Podstawową biblioteką matematyczną, użytą na potrzeby projektu, w celu obróbki danych przechowywanych w wielowymiarowych tablicach jest NumPy. Umożliwia ona łatwe zarządzanie tablicami wielowymiarowymi. Biblioteka ta zawiera również zoptymalizowaną czasowo implementację podstawowych operacji algebry liniowej, rachunku różniczkowego i statystyki. Dostarczanie danych przechowywanych w tablicach NumPy jest wspierane przez wiele popularnych bibliotek do edycji obrazów, obróbki danych tabelarycznych, czy uczenia maszynowego [37][38][39][40].

Jako narzędzia do wizualizacji i obróbki obrazów wybrano Pyplot z biblioteki Matplotlib oraz Scikit-image. Pierwszy z wymienionych służy głównie do wyświetlania obrazów zapisanych w wielowymiarowej tablicy NumPy. Biblioteka ta umożliwia także generowanie wykresów punktowych, histogramów. Możliwe jest przedstawienie kilku wykresów, czy obrazów w jednym oknie, co znacząco wpływa na komfort analizy danych. Biblioteka Scikit-image zapewnia implementację podstawowych operacji wykorzystywanych do obróbki zdjęć, czy miar podobieństwa obrazów [39][40].

Obliczenia wykonano przy użyciu zasobów udostępnionych przez Wrocławskie Centrum Sieciowo-Superkomputerowe, grano obliczeniowy Nr 380 [41].

Kod programu został zamieszczony na repozytorium GitHub pod adresem https://github.com/doszke/tif_image_cutter.

7. Przygotowanie zbioru danych

Zbiór danych został dostarczony w postaci całych skanów histologicznych wraz z opisem fragmentów tkanek, uznawanych za nowotworowe, w pliku XML [7]. Pozwala to na

dobór dowolnej szerokości obrazów, do których całe skany histologiczne zostaną przycięte i sposobu pozyskiwania danych.

Pierwszym podjętym krokiem była konwersja XML-owych oznaczeń tkanki nowotworowej na jednokanałowe obrazy. Każdej komórce z zaznaczonego obszaru przypisano wartość 1, a pozostałym komórkom wartość 0. W wyniku otrzymano czarno-białą, jednokanałową maskę. Konwersji dokonano przy pomocy dedykowanego przez dostawcę zbioru danych oprogramowania.

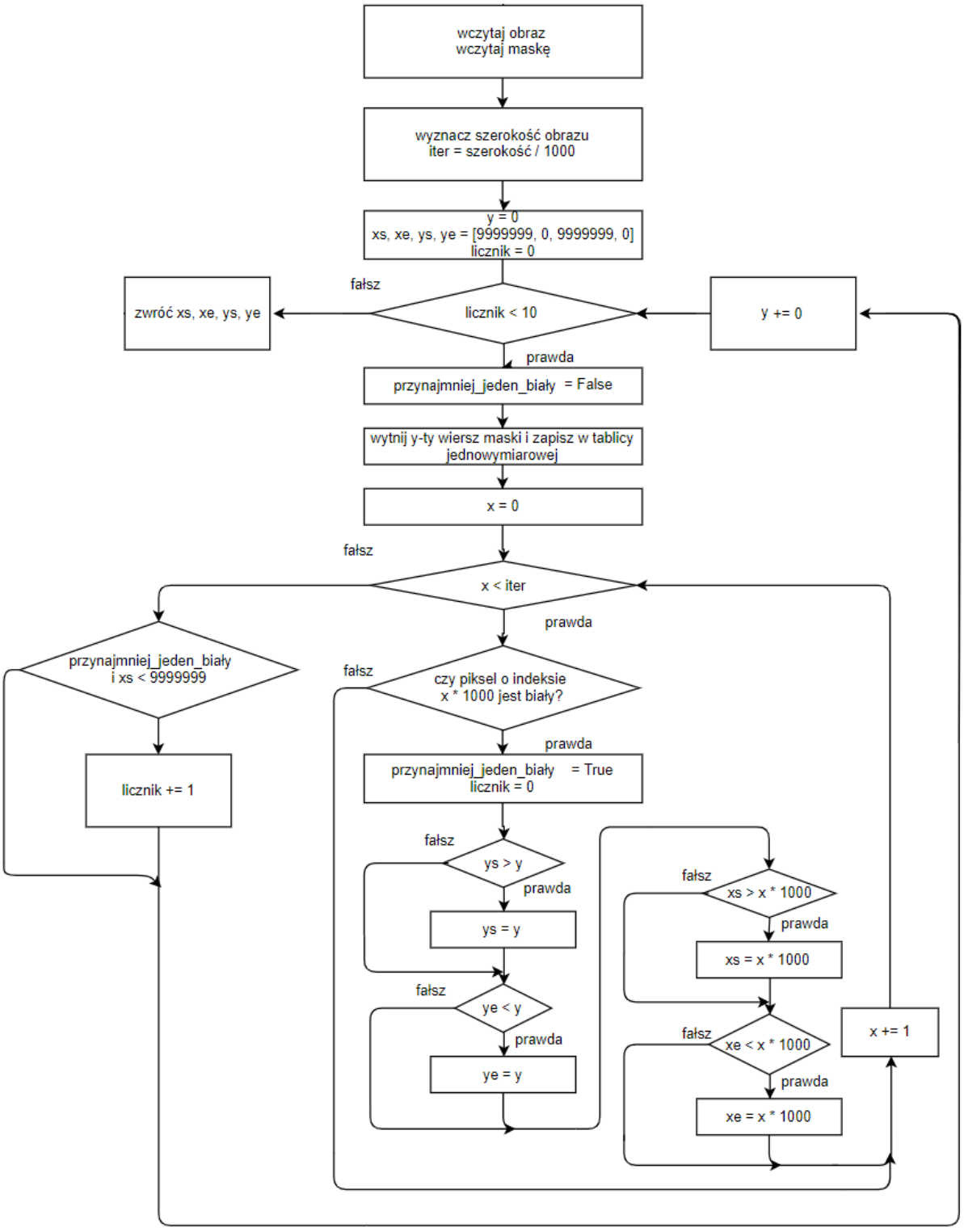
Kluczowym krokiem w obróbce zbioru danych było napisanie algorytmu selekcjonującego fragment pełnego skanu histologicznego zawierającego oznakowaną przez histopatologa tkankę. Pełne skany histologiczne cechują się wymiarami rzędu 10^5 pikseli [42]. W niektórych skanach stosunek tkanki do tła na obrazie był niewielki. W innych natomiast występowały dwie kopie tej samej tkanki, z czego tylko jedna była oznaczona. Wybór oznakowanych fragmentów podczas obróbki powodowałaby wykorzystanie pamięci i czasu procesora do przetwarzania danych, które w dużym procencie nie kwalifikowały się do wykorzystania do nauki sieci. Algorytm można podzielić na dwa odrębne zagadnienia:

- odnalezienie współrzędnych ramki zawierającej wszystkie adnotacje
- rozszerzenie współrzędnych z pierwszego podpunktu w taki sposób, aby zawierały całą oznaczoną tkankę.

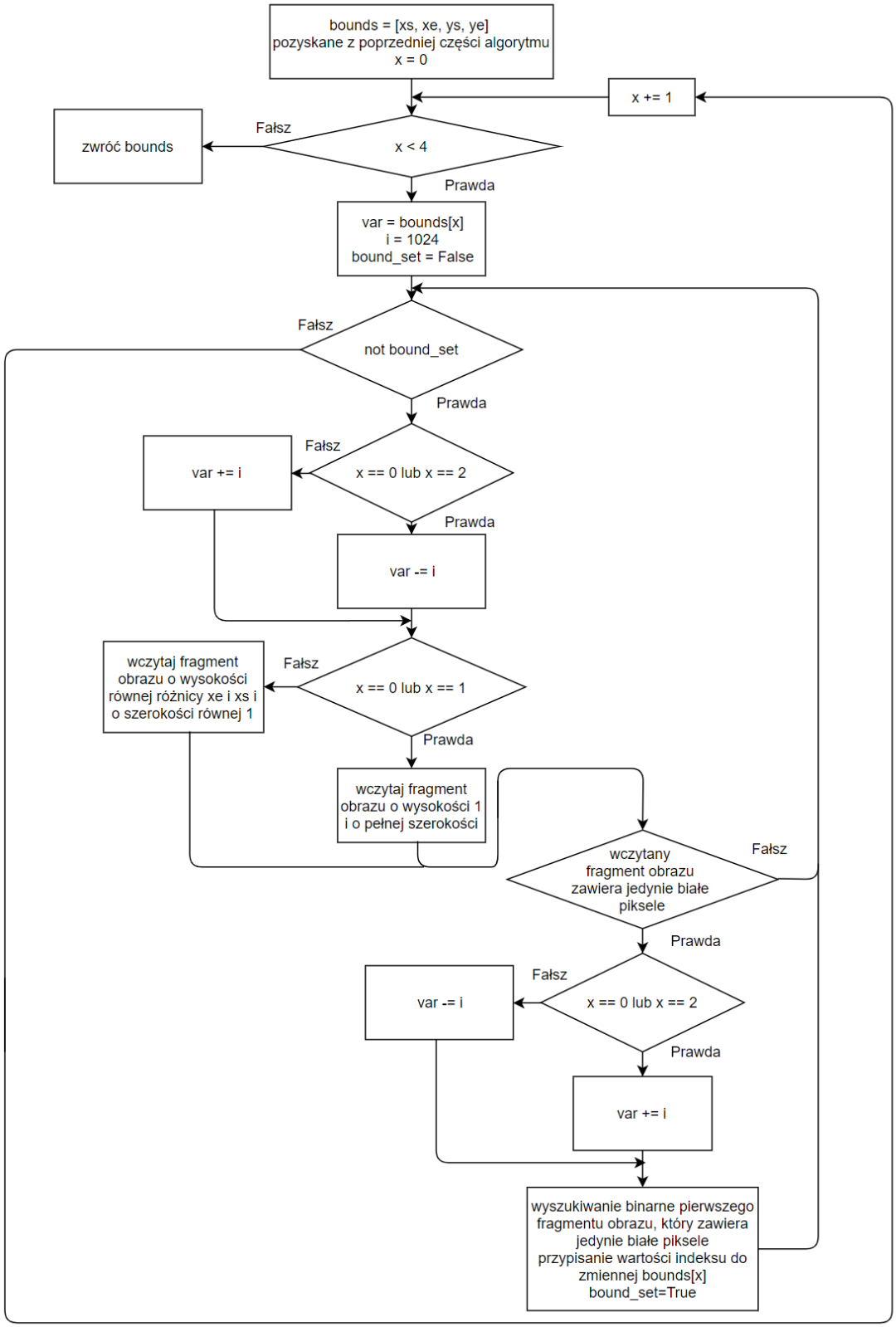
Schematy blokowe obu części algorytmu zostały przedstawiony na rysunkach 7.1. oraz 7.2. W pierwszym podpunkcie jako dana wejściowa podawana jest maska. W wyniku otrzymujemy wektor zawierający cztery współrzędne, za pomocą których można definiować punkty peryferyjne oznaczeń. Wczytywany jest pierwszy wiersz maski. Co tysięczny piksel jest porównywany do „1”- gdy porównanie zwróci *True*, porównywane są współrzędne obecnie przetwarzanego piksela z współrzędnymi z poprzednich iteracji- zmniejszane są współrzędne początkowe jeżeli współrzędne iterowanej komórki są mniejsze, zwiększane współrzędne końcowe, gdy przetwarzany piksel znajduje się w komórce o większych indeksach. Wszystkie w/w czynności są powtarzane dla co tysięcznego wiersza. Gdy przynajmniej jeden biały piksel został wcześniej odnaleziony, oraz w obecnie analizowanym wierszu nie ma ani jednej białej komórki - zwracany jest wektor współrzędnych.

W drugiej części zasada działania algorytmu jest bardzo podobna. Daną wejściową jest otrzymany z wcześniejszej części algorytmu wektor współrzędnych i cały skan histologiczny. Każda współrzędna jest rozszerzana o 1024 (od początkowej 1024 jest odejmowane, do końcowej- dodawane). Jest to powtarzane tak długo, jak średnia arytmetyczna z kanałów danego piksela jest różna od 255 (gdy piksel nie jest koloru białego). Gdy pierwsza biała komórka zostanie znaleziona, współrzędna jest cofana o iterację. Wykonywane jest wyszukiwanie binarne pierwszego piksela o kolorze białym. Pozwala ono na odnalezienie wystąpienia białej komórki z dokładnością do jednego piksela. Zapisywany jest indeks. Całość jest powtarzana dla każdej współrzędnej. Wynikiem funkcji jest wektor zawierający koordynaty najbardziej wysuniętych punktów na oznakowanym obrazie histologicznym.

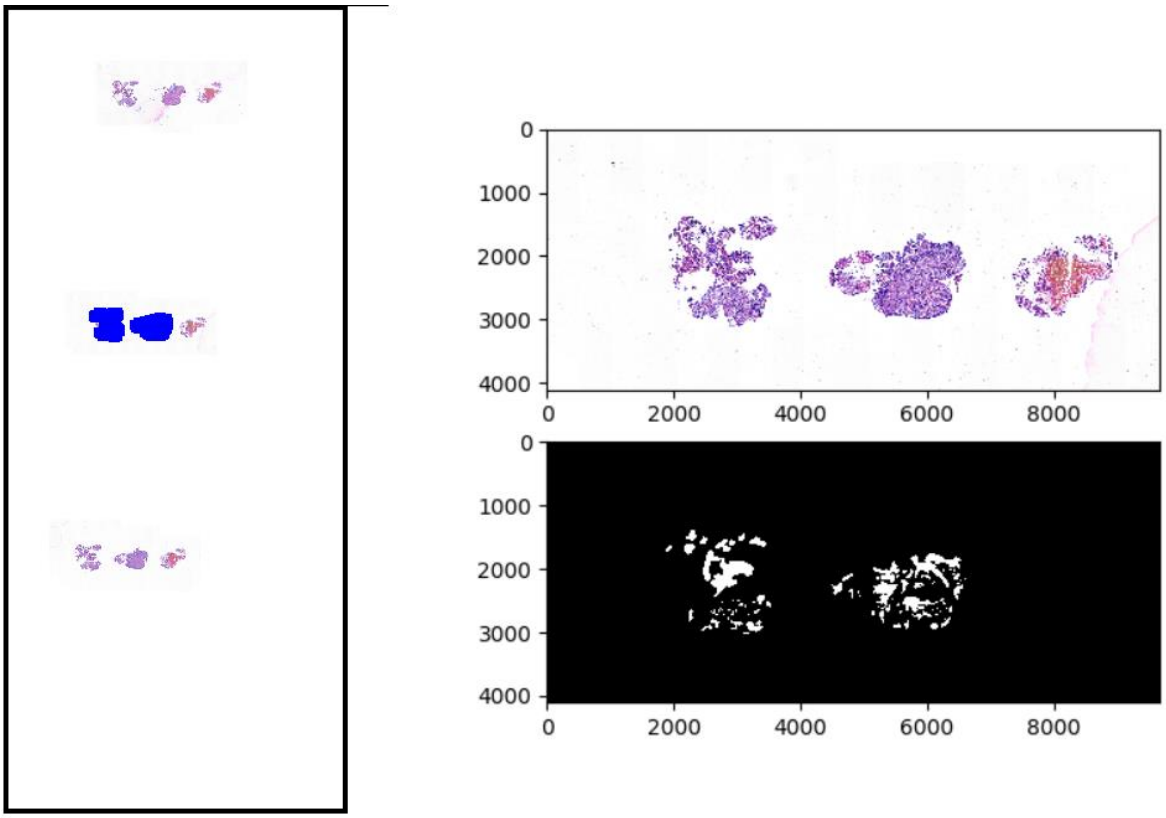
Celem pierwszej części algorytmu jest identyfikacja oznakowanej kopii obrazu. Nie zawsze pierwszy obraz, iterując od góry, jest tym oznakowanym. Dzięki temu jedynie adnotowane przez histopatologa próbki są obecne w zbiorze danych. Krok iteracji równy 1000 ma za zadanie przyspieszyć przetwarzanie obrazu. Druga część rozszerza okno o tkankę zdrową (nieoznakowaną). Gdyby ten krok pominąć, obraz nie obejmowałby całego, zaadnotowanego fragmentu tkanki, a powierzchnia zaznaczona jako występowanie komórek nowotworowych mogła by przeważać nad pozostałą powierzchnią (zdrowa tkanka i obszary niebędące tkanką) [43]. Wynik działania algorytmu przedstawiono na rysunku 7.3.



Rys. 7.1. Pierwsza część algorytmu przycinającego całe skany histologiczne.



Rys. 7.2. Druga część algorytmu przycinającego całe skany histologiczne.



Rys. 7.3. Po lewej- obraz całego skanu histologicznego, wyświetlony w programie ASAP. Po prawej- wynik działania algorytmu przycinającego zaadnotowany skrawek oraz odpowiadający jemu fragment maski.

Ostatnim krokiem było pocięcie obrazów na mniejsze, o zadanej szerokości, równej 256 pikseli. Wyeliminowano pozostałe fragmenty białego tła uśredniając intensywność obrazu i zapisując tylko te obrazy, dla których średnia wynosiła mniej, niż 240. Eliminowało to także te fragmenty, w których większość pikseli była biała, a tylko niewielka część była fragmentami tkanek.

7.1. Selekcja zbioru danych

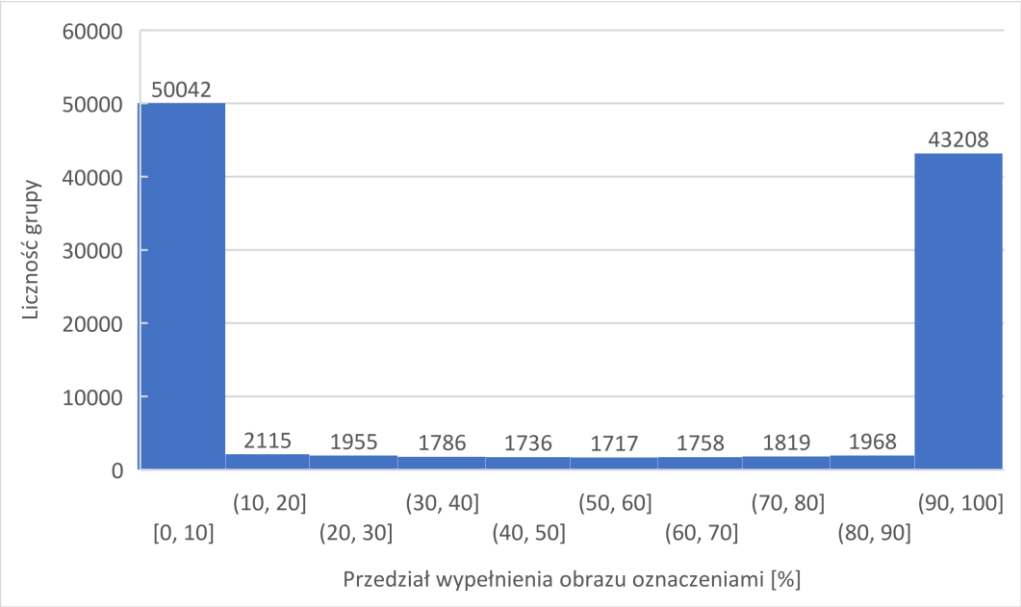
Po wykonaniu operacji opisanych w poprzednich podrozdziałach otrzymano 108 104 obrazy o wymiarach 256x256 pikseli wraz z odpowiadającymi im maskami. W celu wizualizacji rozkładu wypełnienia maski oznaczeniami sporządzono histogram procentu powierzchni zaznaczonej, przedstawiony na wykresie 7.1.

Z łatwością można zauważyć, iż przedziały 0-10% oraz 90-100% są najliczniejsze. W nich znajduje się większość masek.

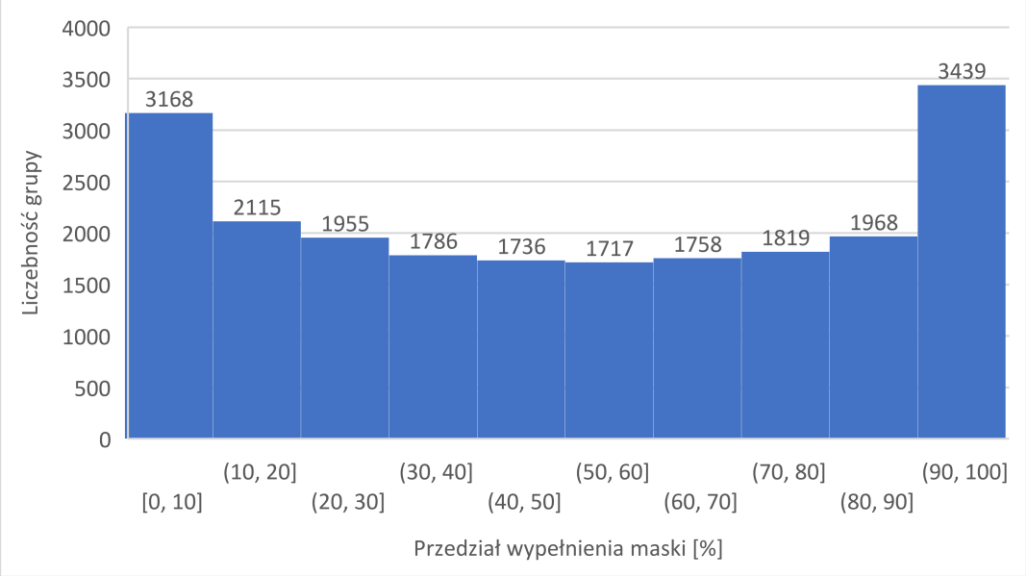
Najbardziej licznym jest przedział 0-10%, stanowi on 46.29% wszystkich obrazów. Drugim najliczniejszym jest natomiast przedział 90-100%, składający się z około 40% próbek. Żaden z pozostałych zbiorów nie stanowi więcej, jak 2% objętości danych. Najmniej liczną grupą są próbki będące zaadnotowane w 50-60%, składającą się z 1717 par masek i obrazów.

Duża dysproporcja między liczebnością poszczególnych grup może mieć realny wpływ na proces nauki sieci [43]. W celu zrównania rozkładu wypełnienia próbek, wydzielono dwie dodatkowe grupy: pierwsza zawiera obrazy tkanki, która nie została oznaczona jako nowotworowa, a druga- obrazy tkanki w całości oznaczone jako tkanka nowotworowa.

Wykres 7.2. przedstawia histogram zbioru danych z wyłączeniem próbek z obu zdefiniowanych powyżej grup.



Wykres 7.1. Histogram wypełnienia maski adnotacją.



Wykres 7.2. Histogram wypełnienia maski adnotacją z pominięciem obrazów tkanki w pełni zaadnotowanych i pozbawionych adnotacji.

Po wyizolowaniu dwóch wyżej zdefiniowanych grup, najliczniejszymi przedziałami wciąż są przedziały 0-10% oraz 90-100%. Z każdej z 12 grup wyselekcjonowano 1717 losowych obrazów wraz z adnotacjami, co daje łącznie 20 604 próbki. Liczba próbek wybieranych z każdej grupy jest równa ilości obrazów w najmniej licznym przedziale.

8. MODEL

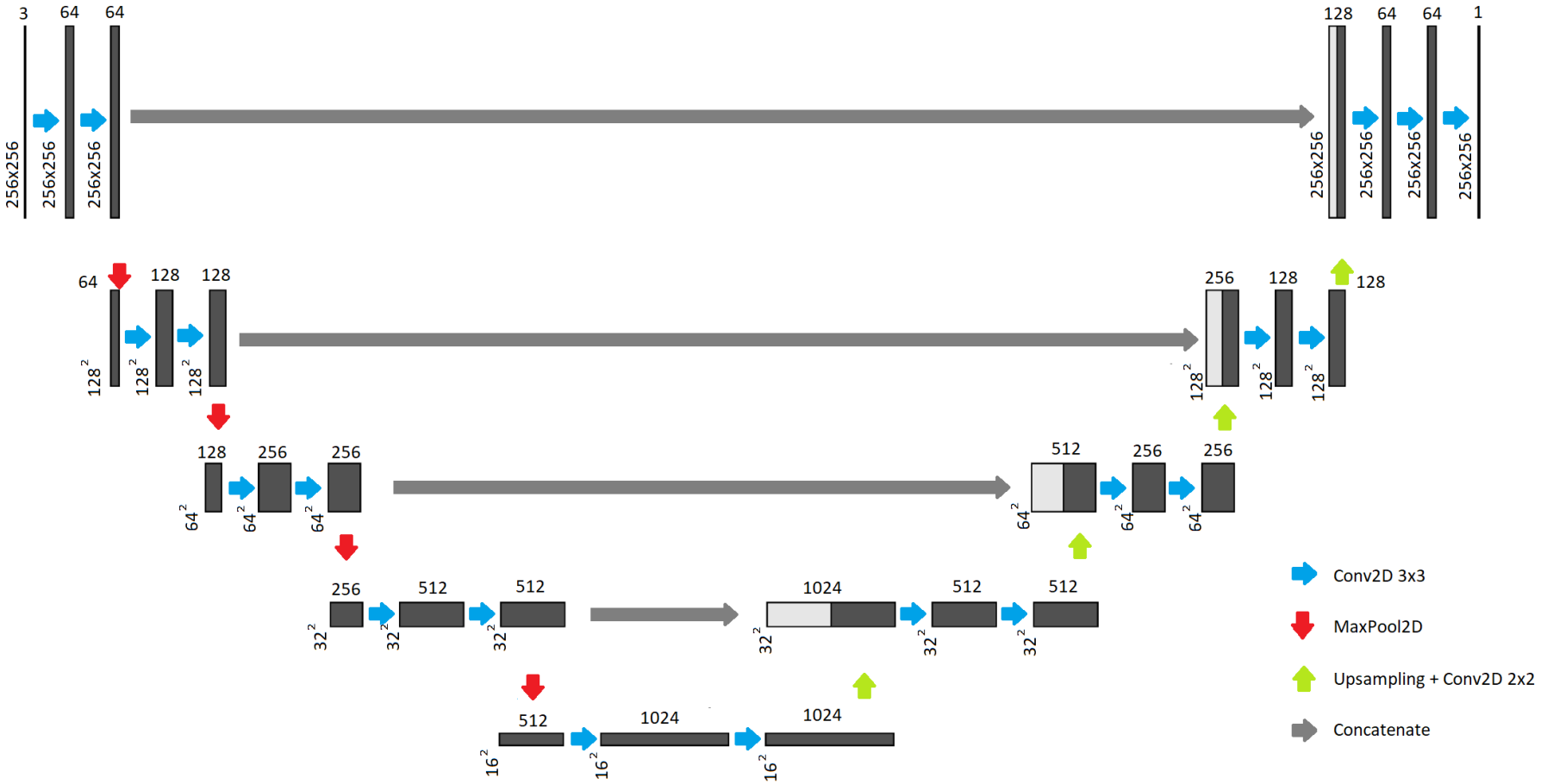
8.1. ZAPROPONOWANY MODEL SIECI

Zastosowano gotową implementację szkieletu modelu udostępnioną przez użytkownika zhixuhao na licencji MIT na portalu Github [44]. Wykorzystano dwie funkcje: o nazwach „down" i „up". Implementacja pierwszej z nich zawiera operacje niezbędne do dwukrotnego zmniejszenia szerokości obrazka i dwukrotnego zwiększenia ilości filtrów, a drugiej- do uzyskania przeciwstawnego efektu. Zastosowanie tych dwóch funkcji upraszcza tworzenie modeli o różnej głębokości. Na listingu 8.1. przedstawiono kod tworzący model sieci.

Listing 8.1. Kod klasy Unet służący do definiowania modeli sieci. Metody 'down' oraz 'up' pochodzą z repozytorium GitHub użytkownika zhixuhao [44].

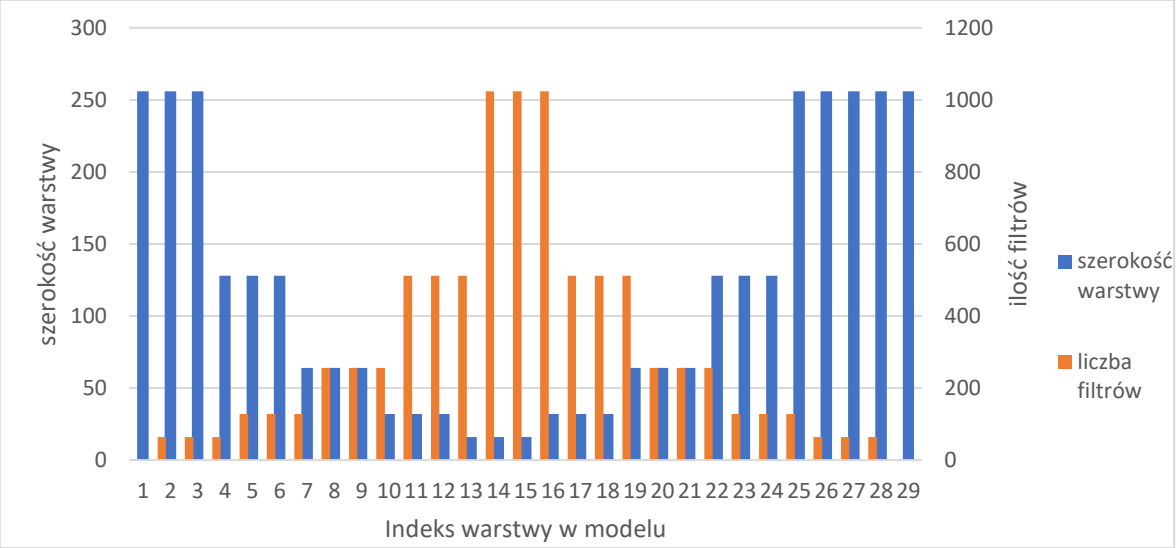
```
import numpy as np
from tensorflow.keras.layers import Conv2D, MaxPool2D, UpSampling2D, Concatenate, Input
from tensorflow.keras.models import Sequential, Model

class Unet:

    'metoda pochodzi z repozytorium: https://github.com/zhixuhao/unet'
    def down(self, input_layer, filters, pool=True, activation="softplus"):
        conv1 = Conv2D(filters, (3, 3), padding='same', activation=activation)(input_layer)
        residual = Conv2D(filters, (3, 3), padding='same', activation=activation)(conv1)
        if pool:
            max_pool = MaxPool2D()(residual)
            return max_pool, residual
        else:
            return residua

    'metoda pochodzi z repozytorium: https://github.com/zhixuhao/unet'
    def up(self, input_layer, residual, filters, activation="softplus"):
        filters = int(filters)
        upsample = UpSampling2D()(input_layer)
        upconv = Conv2D(filters, kernel_size=(2, 2), padding="same")(upsample)
        concat = Concatenate(axis=3)([residual, upconv])
        conv1 = Conv2D(filters, (3, 3), padding='same', activation=activation)(concat)
        conv2 = Conv2D(filters, (3, 3), padding='same', activation=activation)(conv1)
        return conv2

    def my_unet_model(self, filters=64, size=32, down=4, activation="softplus"):
        residuals = []
        shape = [size, size, 3]
        input_layer = Input(shape=shape)
        d = input_layer
        res = None
        for x in range(down):
            d, res = self.down(d, filters, activation=activation)
            residuals.append(res)
            filters *= 2
        d = self.down(d, filters, pool=False, activation=activation)
        print(np.shape(d))
        print(residuals)
        for x in range(down):
            d = self.up(d, residual=residuals[-x-1], filters=filters/2, activation=activation)
            filters /= 2
        out = Conv2D(filters=1, kernel_size=(1, 1), activation="sigmoid")(d)
        model = Model(input_layer, out)
        return model
```

Architektura użyta w niniejszej pracy nieznacznie różni się od zaprezentowanej w poprzednim rozdziale. Jako dane wejściowe do sieci wybrano obrazy o wymiarach 256x256px, kodowane w 3 kanałach. Zdecydowano się na zastosowanie splotu macierzą 3x3 z jednostkową wyściółką (padding) zer w celu zachowania szerokości warstwy. Pozwoliło to na uzyskanie predykcji w postaci obrazu o takiej samej wielkości, jak obraz wejściowy przy równej liczbie operacji poolingu i próbkowania w górę. Schemat sieci został zaprezentowany na rysunku 8.1.



Rys. 8.1. Schemat zaprojektowanego modelu sieci, w którym występują cztery operacje poolingu. Poziomo: liczba filtrów, pionowo: wymiary warstwy.

Na rysunku 8.2. przedstawiono zależności liczby filtrów i szerokości warstwy w zależności od jej położenia w zaproponowanym modelu. W przeciwieństwie do charakterystyk zilustrowanych na rysunku 4.5, zarówno jedna jak i druga zależność jest symetryczna względem środkowego indeksu. Spowodowane jest to zastosowaniem splotu macierzą 3x3 z jednostkowym paddingiem zerami, który gwarantuje zachowanie szerokości obrazu w trakcie operacji splotu.



Wykres 8.2. Zależność szerokości warstwy i liczby filtrów od indeksu warstwy w modelu sieci użytej w niniejszej pracy.

8.2. PREPROCESSING

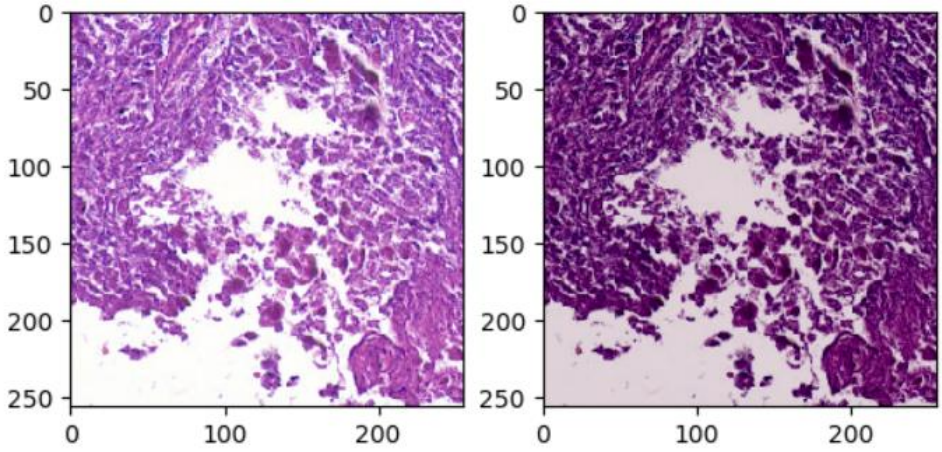
Preprocessing to szereg operacji wykonywanych na danych przed ich udostępnieniem sieci neuronowej w celu zwiększenia jakości predykcji, czy wspomagania procesu uczenia[45]. Do stworzenia modelu na potrzeby niniejszej pracy użyto transformację koloru Reinharda, filtr medianowy oraz filtr HED.

$$l = \frac{\sigma^{l}_{referencji}}{\sigma^{l}_{źródła}}\left(l_{źródła} - \mu^{l}_{źródła}\right) + \mu^{l}_{referencji} \tag{8.1}$$

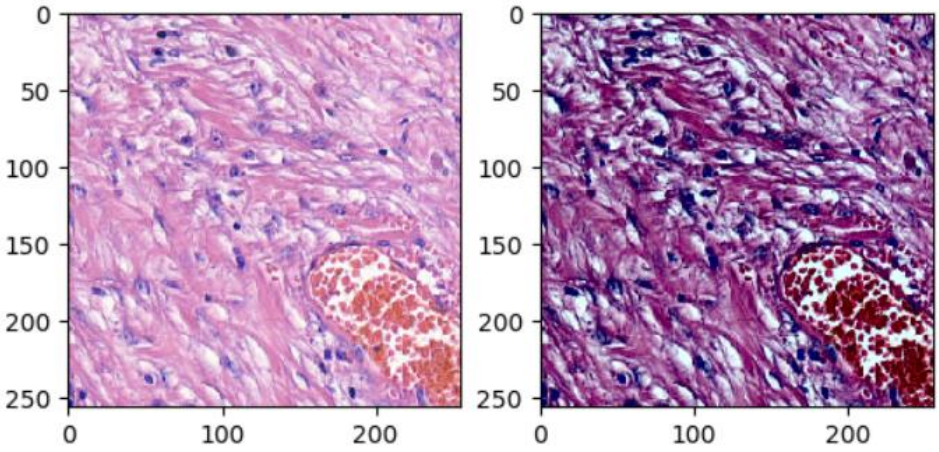
$$\alpha = \frac{\sigma^{\alpha}_{referencji}}{\sigma^{\alpha}_{źródła}}\left(\alpha_{źródła} - \mu^{\alpha}_{źródła}\right) + \mu^{\alpha}_{referencji} \tag{8.2}$$

$$\beta = \frac{\sigma^{\beta}_{referencji}}{\sigma^{\beta}_{źródła}}\left(\beta_{źródła} - \mu^{\beta}_{źródła}\right) + \mu^{\beta}_{referencji} \tag{8.3}$$

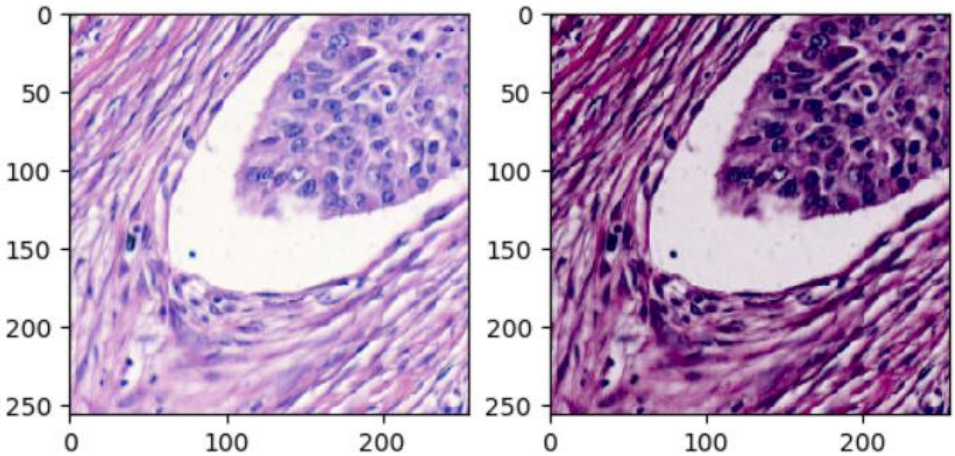
Intensywność wybarwienia próbek hematoksyliną i eozyną jest zależna np. od stężenia barwników i czasu ekspozycji [46]. Powoduje to, iż uzyskanie dwóch identycznie wybarwionych próbek jest trudne [47]. W celu znormalizowania odcieni barw zastosowano transformację koloru Reinharda[48]. Pozwala ona na zmodyfikowanie koloru obrazu na podstawie obrazu odniesienia. Dane z obu obrazów są konwertowane do przestrzeni barw $l\alpha\beta$, która minimalizuje korelacje między poszczególnymi kanałami[48]. Kolejnym krokiem jest wyznaczenie wartości średniej i odchylenia standardowego obu obrazów. Wartości kanałów $l\alpha\beta$ obrazu wynikowego opisują zależności przedstawione na równaniach 8.1-8.3. Rysunki 8.3.-8.5. przedstawiają przykłady transformacji koloru na reprezentantach próbek histopatologicznych ze zbioru danych użytego w niniejszej pracy.



Rys. 8.3. Po lewej- obraz przed zastosowanie preprocessingu, po prawej- obraz po



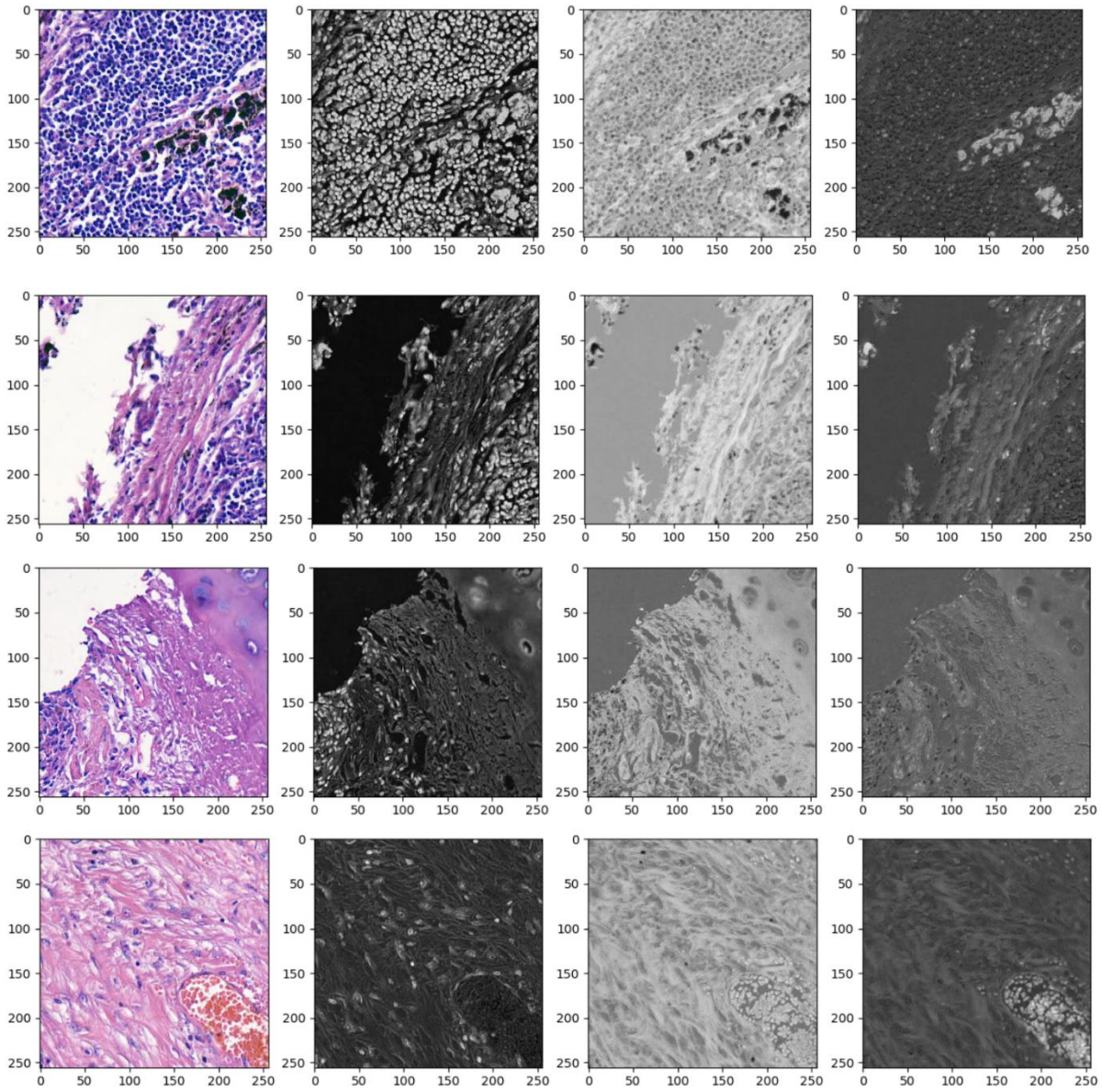
Rys. 8.4. Po lewej- obraz przed zastosowanie preprocessingu, po prawej- obraz po



Rys. 8.5. Po lewej- obraz przed zastosowanie preprocessingu, po prawej- obraz po zastosowaniu transformacji koloru Reinharda.

Informacja o wybarwionych strukturach biologicznych jest rozproszona pomiędzy trzema kanałami w kodowaniu RGB. W celu ułatwienia nauki, wykonano konwersję obrazu

z przestrzeni barw RGB do przestrzeni HED [50]. Konwersja powoduje zmianę kanałów kolorów na kanały przechowujące informacje o występowaniu trzech podstawowych barwników histopatologicznych- hematoksyliny, eozyny i diaminobenzydyny [39][50].



Rys. 8.6. Przykłady konwersji obrazu histopatologicznego płuc z przestrzeni barw RGB do przestrzeni barw HED. Kolejno od lewej: oryginalny obraz, kanał hematoksylinowy, kanał eozynowy, kanał diaminobenzydynowy.

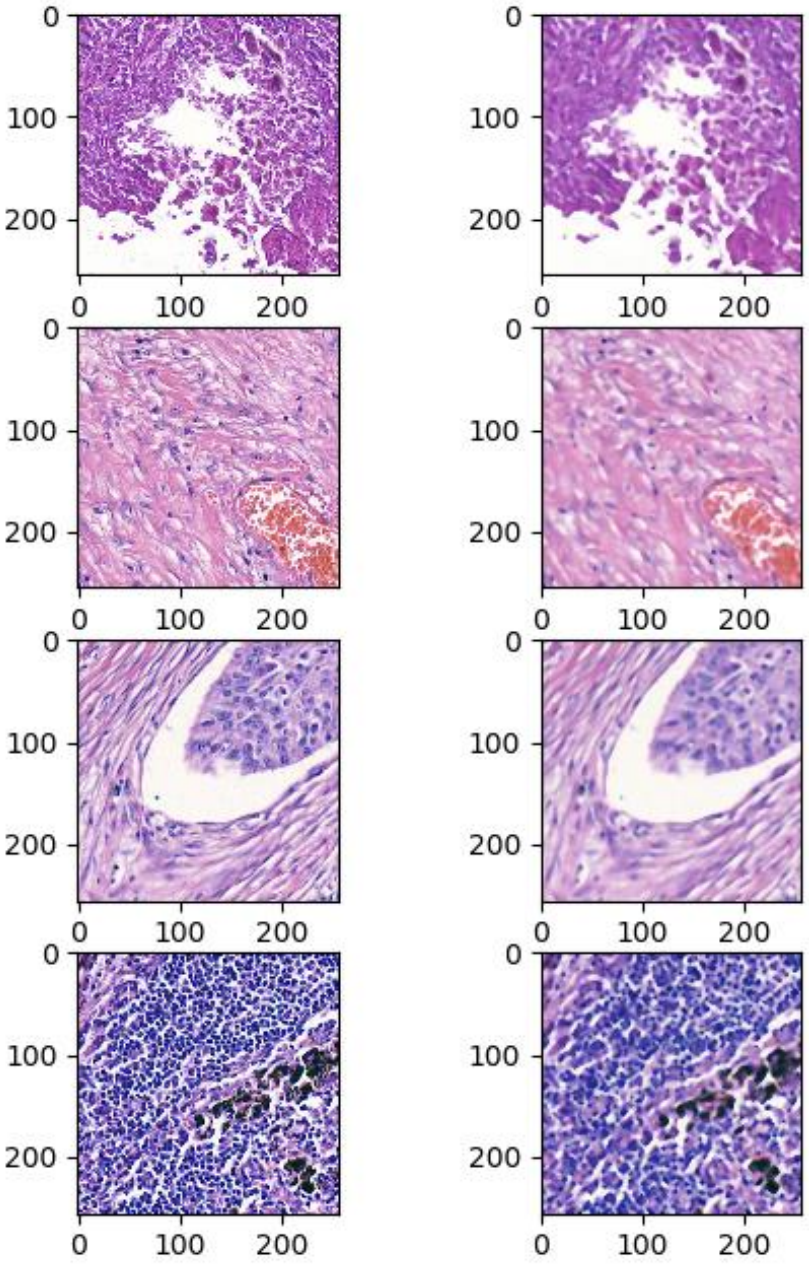
Przejście z jednego systemu barw do drugiego zachodzi na drodze konwolucji obrazu z macierzą:

$$\begin{bmatrix} 1.88 & -0.07 & -0.60 \\ -1.02 & 1.13 & -0.48 \\ -0.55 & -0.13 & 1.57 \end{bmatrix}$$

Kolumny reprezentują kolejne kanały systemu RGB, a wiersze- systemu HED. Macierz ta jest ortogonalna do znormalizowanej macierzy gęstości optycznych poszczególnych kanałów.

Na rysunku 8.6. przedtsawiono wynik konwersji obrazów histopatologicznych do przestrzeni barw HED.

Zastosowanie konwersji barw do systemu HED pozwoliło na oddzielenie struktur barwionych różnymi odczynnikami na poszczególne kanały. Na kanale hematoksylinowym najwyższe wartości zostały przypisane miejscom występowania jąder komórek na obrazie oryginalnym. Najwyższe wartości dla kanału eozynowego występują dla cytoplazmy. Pomimo faktu, iż diaminobenzydyna nie została użyta do wybarwiania próbek z zestawu danych, pozostawiono ten kanał w obrazie wynikowym.



Rys. 8.7. Zastosowanie filtru medianowego do preprocessingu obrazów histopatologicznych. Po lewej – oryginalny obraz, po prawej – obraz z zastosowanym filtrem medianowym.

Filtr medianowy jest jednym z podstawowych narzędzi służących wygładzaniu obrazów. Dla każdego piksela wyznaczana jest mediana w odniesieniu do sąsiednich pikseli, najczęściej w kwadratowym oknie o długości boku podanej przez użytkownika, gdzie analizowany piksel

jest położony centralnie. Wartość median zastępuje oryginalną wartość pikseli powodując wygładzenie obrazu, redukcję szumów, ale też jego rozmycie [51].

Na rysunku 8.7. przedstawiono zastosowanie filtru medianowego na przykładowych obrazach histopatologicznych. Zastosowanie filtru medianowego pozwoliło na wygładzenie krawędzi. Spowodowało to również zatarcie granic pomiędzy poszczególnymi strukturami komórkowymi. Zastosowanie tego rodzaju preprocessingu może spowodować, że struktury o wyraźnych krawędziach, tj. włókna kolagenowe, nie będą brane pod uwagę przez sieć.

9. ANALIZA WYNIKÓW

W celu maksymalizacji jakości predykcji sieci funkcja aktywacji i krotność zmniejszenia szerokości obrazka zostały dobrane eksperymentalnie. Sprawdzono również wpływ preprocessingu na jakość segmentacji obrazów. W tym celu wylosowano 900 zdjęć ze zbioru danych, służących do nauki sieci, oraz 100 zdjęć do walidacji modelu. Parametry sieci neuronowej próby kontrolnej oraz parametry jej uczenia zostały przedstawione w tabeli 9.1.

Tabela 9.1. Zestawienie parametrów sieci kontrolnej oraz parametrów uczenia.

Rozmiar danych wejściowych	256x256x3
Rozmiar danych wyjściowych	256x256x1
Głębokość	4
Funkcja aktywacji	Softplus
Ilość filtrów po pierwszej operacji splotu	64
Optimizer	Adam
Funkcja strat	binarna entropia krzyżowa (ang. binary crossentropy)
Miara jakości segmentacji	współczynnik Sorensena-Dice'a
Liczba epok	50
Liczba danych uczących	900 obrazów i masek
Liczba danych walidujących	100 razów i masek

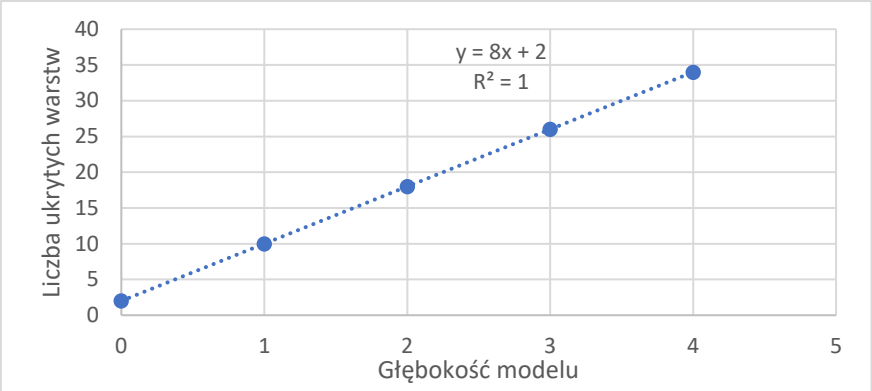
9.1. DOBÓR GŁĘBOKOŚCI SIECI

Jednym z istotnych parametrów sieci, którego zmiana może mieć znaczący wpływ na proces nauki, jest głębokość sieci. Miarą tej cechy jest liczba warstw pośrednich w modelu sieci [52]. Na potrzeby porównania głębokości architektur typu U-Net w niniejszej pracy zdefiniowano jednak inną miarę, której matematyczny zapis przedstawia równanie 9.1.

$$x = \log_2 \frac{a}{b} \tag{9.1}$$

gdzie *a* to szerokość obrazu podanego, a *b* to najmniejsza szerokość warstwy występującą w danym modelu. Ułamek a/b reprezentuje największą krotność zmniejszenia obrazu, a zastosowanie logarytmu przy podstawie 2 umożliwia określenie ilokrotnie zastosowano operację dwukrotnego zmniejszenia obrazu.

Głębsze architektury typu U-Net składają się z powtarzających się operacji. Na wykresie 9.1. zwizualizowano zależność głębokości sieci według zaproponowanej wyżej miary od liczby warstw pośrednich w modelu w celu znalezienia korelacji.



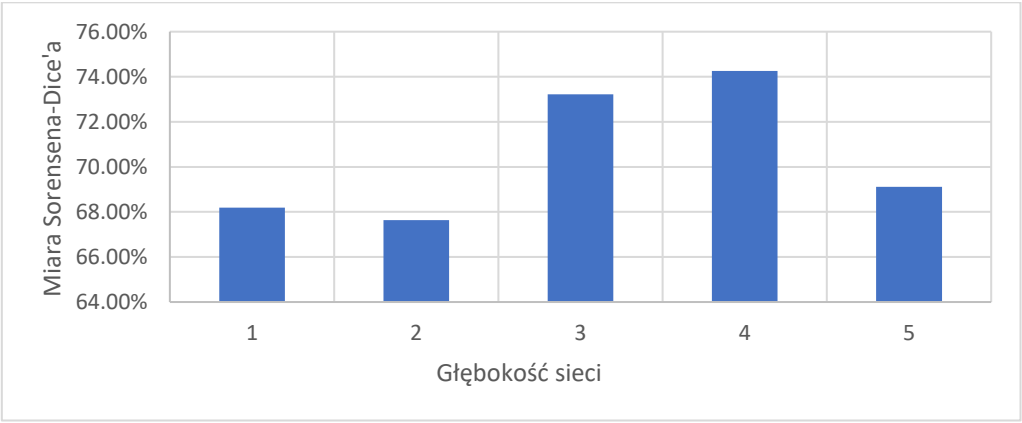
Rys. 9.1. Zależność ilości ukrytych warstw od miary głębokości sieci.

Opisana zależność jest liniowa. Oznacza to, że z każdym dwukrotnym zmniejszeniem obrazu, liczba warstw zwiększa się o 8. Dla głębokości równej 0, model składa się z dwóch warstw pośrednich, wykonujących operację splotu. Otrzymane równanie prostej umożliwia wyznaczenie ilości warstw w zaprezentowanej w niniejszej pracy architekturze U-Net na podstawie zdefiniowanej w tym rozdziale miary.

Tabela 9.2. przedstawia porównanie wyników predykcji sieci, opisanych metryką Sorensena Dice'a, różniących się głębokością, uczonych przez 50 epok. Na wykresie 9.2. została przedstawiona zależność średniej miary Sorensena Dice'a od głębokości modelu.

Tabela 9.2. Porównanie wyników predykcji sieci różniących się głębokością modelu.

Głębokość [-]	1	2	3	4	5
Wyniki cząstkowe	68.80%	72.61%	73.89%	76.45%	74.58%
	68.64%	67.83%	72.45%	77.87%	67.77%
	65.69%	66.93%	73.36%	73.88%	67.55%
	69.61%	63.19%	73.16%	69.31%	66.55%
Średnia	**68.19%**	**67.64%**	**73.22%**	**74.25%**	**69.11%**
Odchylenie standardowe	1.72%	3.87%	0.60%	3.79%	3.68%



Rys. 9.2. Zależność średnich miar Sorensena-Dice'a dla modeli różniących się głębokością.

Analizując wartości metryki Sorensena-Dice'a dla poszczególnych prób, można zauważyć spore różnice między kolejnymi próbami (rzędu kilku punktów procentowych).

Oznacza to, iż uczenie kilku modeli o takich samych parametrach nie jest w 100% powtarzalne- dla każdej epoki zbiór uczący jest podawany w różnej kolejności. Powoduje to, iż każdy uczony model po przejściu jednej epoki będzie miał inny zestaw wag.

Na podstawie charakterystyki przedstawionej na rysunku 9.2. można zauważyć, iż wraz ze wzrostem głębokości do 4 wzrasta również dokładność predykcji sieci. Dla głębokości 2 uzyskano lokalne minimum. Jest to najprawdopodobniej spowodowane losowym podawaniem obrazów do nauki. Kolejnym powodem może być mała ilość prób przeprowadzonych do wyznaczenia każdego z wyników. Należałoby zwiększyć ilość przeprowadzonych prób w celu dokładniejszej weryfikacji. Sieć o głębokości 5 również osiągnęła niższy wynik, niż sieć o głębokości 4 lub 3. Powodem tego jest sama złożoność sieci. Model o głębokości 5 według miary zaproponowanej w niniejszej pracy zawiera aż 42 warstwy pośrednie. Każda dodatkowa warstwa w modelu zwiększa możliwości sieci (np. ilość cech możliwych do zidentyfikowania), lecz wymaga większej ilości epok, aby dobrze wytrenować model [52].

9.2. Dobór funkcji aktywacji

Poprzez pojęcie funkcja aktywacji rozumie się operację matematyczną wykonywaną na danych przed ich przejściem do kolejnej warstwy w celu odfiltrowania odpowiednich informacji, lub mapowania wartości na żądany przedział [23][24][52]. Celem przeprowadzanego eksperymentu było wyłonienie funkcji aktywacji, dzięki której wytrenowana sieć osiągnie najlepszy wynik. Z racji na fakt, iż architektura U-Net jest w pełni konwolucyjna [7], do eksperymentów użyto aktywatorów zdefiniowanych dla operacji splotu: funkcji ReLU, PReLU oraz softplus.

Prostownik (ang. ReLU, Rectified Linear Unit) [53] jest funkcją, która dla argumentów ujemnych zwraca 0, a dla pozostałych argumentów zwraca wartość argumentu. Wzór 9.2. przedstawia zapis matematyczny tej formuły.

$$ReLU(x) = \begin{cases} 0 \; dla \; x < 0 \\ x \; dla \; x \geq 0 \end{cases} \quad (9.2)$$

ReLU nie jest różniczkowalna na całym przedziale. Jej pochodna dla wartości dodatnich wynosi 1, a dla wartości ujemnych 0. Dla wartości $x = 0$ pochodna nie istnieje. Z racji na fakt, iż pochodna funkcji aktywacji jest używana w procesie uczenia maszynowego, programiści często używają wartości z przedziału [0, 1] jako odpowiednik pochodnej w zerze dla ReLU [52]. Wadą całkowitego wygaszania ujemnych wartości jest problem „martwego ReLU" (ang. „dying ReLU" problem). Jest to sytuacja, w której przeważający procent wag ReLU jest ujemny, co powoduje, iż dla każdej danej wejściowej neuron zwróci wartość 0. Skutkuje to pomijaniem tych neuronów w procesie nauki, co w rezultacie wydłuża czas niezbędny do nauki sieci [54].

Sparametryzowany prostownik (ang. PReLU, Parameterized ReLU) to funkcja aktywacji, która dla nieujemnej liczby zwraca ją samą, a dla ujemnej- jej przekształcenie liniowe. Jej formuła została przedstawiona we wzorze 9.3.

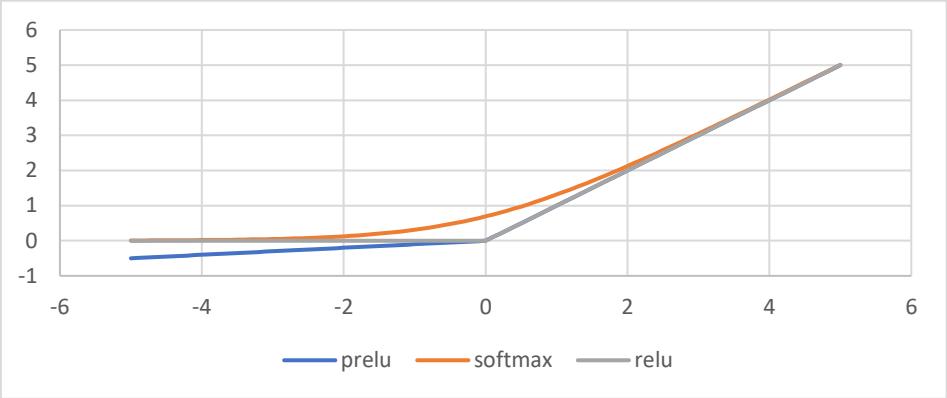
$$PReLU(x) = \begin{cases} 0 \; dla \; x < 0 \\ ax \; dla \; x \geq 0, a \in \mathbb{R} \end{cases} \quad (9.3)$$

Dzięki zastosowaniu dodatniego współczynnika kierunkowego prostej, gradient PReLU dla ujemnych wartości a wynosi -1, co pozwala przeciwdziałać problemowi „martwego ReLU" [54].

Funkcja softplus, w przeciwieństwie do dwóch wyżej przedstawionych funkcji aktywacji, jest ciągła i różniczkowalna w dziedzinie liczb rzeczywistych. Jej antydziedziną jest przedział (0, +∞). Wzór 9.4. przedstawia formułę funkcji softplus.

$$softplus(x) = \ln(1 + e^x) \quad (9.4)$$

Wykres 9.3. przedstawia graficzną wizualizację charakterystyk funkcji aktywacji w przedziale [-5, 5].



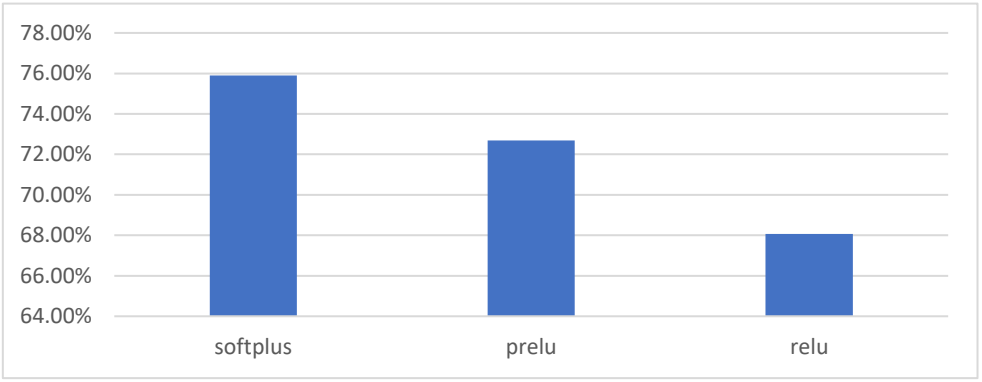
Rys. 9.3. Charakterystyki funkcji aktywacji w przedziale [-5, 5].

Asymptotami funkcji softplus są proste definiujące funkcję ReLU. Dla dużych wartości dodatnich argumentu wynik działania każdej z funkcji jest tożsamy, a dla dużych ujemnych wartości- funkcje softplus i ReLU zwracają tożsame wartości.

Tabela 9.3. przedstawia wartości metryki Sorensena-Dice'a dla sieci neuronowych o głębokości 4, różniących się funkcjami aktywacji, uczonych przez 50 epok na 900 obrazach, walidowanych na 100 obrazach. Na wykresie 9.4. zostały przedstawione średnie metryki Sorensena-Dice'a dla poszczególnych funkcji aktywacji. Parametrowi *a* funkcji PreLU przypisano wartość 0.01.

Tabela 9.3. Zestawienie wyników uczenia sieci różniących się funkcją aktywacji.

Funkcja aktywacji	softplus	PreLU	ReLU
Wyniki cząstkowe	76.45%	76.33%	69.15%
	77.87%	71.95%	62.70%
	73.38%	69.79%	72.35%
Średnia	**75.90%**	**72.69%**	**68.07%**
Odchylenie standardowe	2.39%	3.33%	4.92%



Rys. 9.4. Zestawienie średnich wyników nauczania sieci dla modeli różniących się funkcjami aktywacji.

Najlepsze wyniki sieci osiągnięto dla funkcji softplus, a najmniejszą dokładnością cechowały się sieci o funkcji aktywacji ReLU. Powodem sukcesu pierwszej z funkcji może być jej

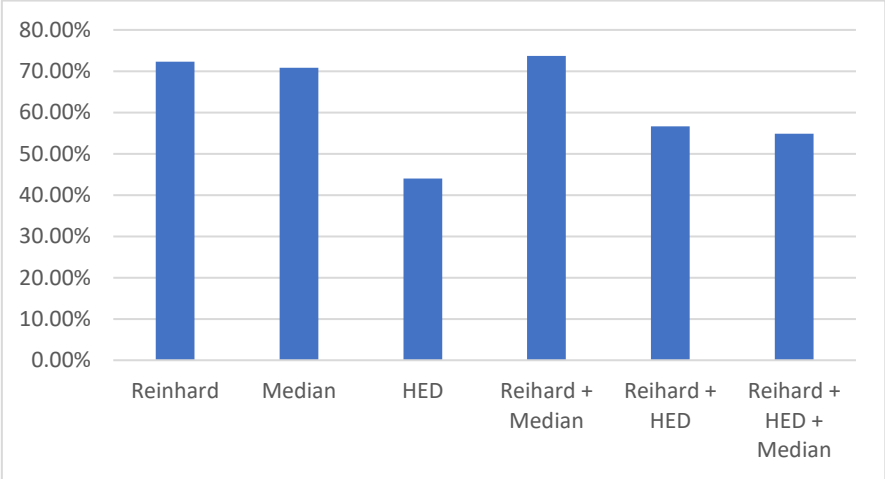
różniczkowalność w całym zbiorze liczb rzeczywistych, oraz fakt, iż softplus, w przeciwieństwie do dwóch pozostałych funkcji, nie ma stałego gradientu. Mniejszy wynik dla funkcji ReLU może być spowodowany problemem „martwego ReLU". Istotnym czynnikiem, mogącym mieć wpływ na wynik, jest obecność stałego gradientu, wynoszącego 0 dla liczb ujemnych, oraz 1 dla liczb nieujemnych. Sieci neuronowe używające funkcji aktywacji PReLU uzyskały większą średnią metryk Sorensena Dice'a od sieci używających funkcji ReLU. Obecność gradientu o wartości -0.01 dla ujemnych liczb mogła przeciwdziałać problemowi martwych neuronów powodując aktualizację wartości wag tych neuronów, które przy użyciu funkcji ReLU nie byłyby uczone.

9.3. Dobór preprocessingu

Wybrane techniki preprocessingu zostały opisane w podrozdziale 8.4. Tabela 9.4. przedstawia metryki Sorensena Dice'a dla modeli sieci neuronowych typu U-Net o głębokości 4, funkcji aktywacji softplus, uczonych przez 50 epok na 900 obrazach, walidowanych na 100 obrazach, na których zastosowano różne techniki preprocessingu. Na wykresie 9.5. zwizualizowano średnie metryk Sorensena Dice'a dla poszczególnych technik preprocessingu.

Tabela 9.4. Zestawienie metryk Sorensena-Dice'a dla sieci uczonych na obrazach, na których zastosowano różne rodzaje preprocessingu.

Preprocessing	Reinhard	Median	HED	Reihard + Median	Reihard + HED	Reihard + HED + Median
Wyniki cząstkowe	70.23%	72.42%	61.45%	74.12%	57.48%	53.29%
	73.63%	72.79%	61.57%	73.39%	60.58%	53.53%
	73.06%	67.28%	8.98%	73.52%	51.90%	57.70%
Średnia	**72.31%**	**70.83%**	**44.00%**	**73.68%**	**56.65%**	**54.84%**
Odchylenie standardowe	1.82%	3.08%	30.33%	0.39%	4.40%	2.48%



Rys. 9.5. Wykres słupkowy średnich metryk Sorensena-Dice'a modeli uczonych na danych poddanych różnemu preprocessingowi.

Techniką preprocessingu o największej średniej wartości metryki Sorensena-Dice'a jest zastosowanie transformacja koloru Reinharda i filtru medianowego, z wynikiem 73.68%. Jest to również preprocessing osiągający najbardziej powtarzalne wartości metryki dla sieci.

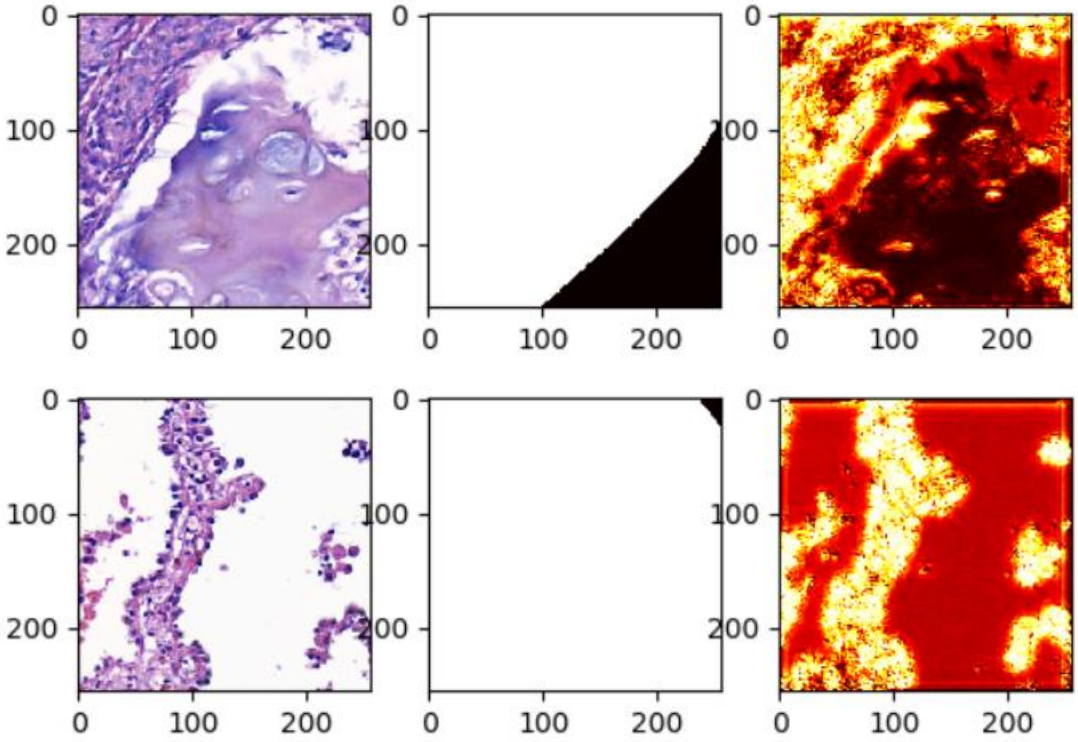
Najmniejszy wynik średniej został uzyskany dla sieci uczonej na obrazach poddanych transformacji barw z systemu RGB do systemu HED. Powodem tego może być mały przedział wartości, na który kanały RGB były przetransformowywane, a co za tym idzie – konieczność przeskalowania tych wartości do przedziału [0, 1]. Preprocessing ten okazał się również najmniej powtarzalny – odchylenie standardowe poszczególnych prób wyniosło 30.33%. Powodem tego jest fakt, iż dla tej serii został uzyskany najmniejszy wynik cząstkowy, równy 8.98%.

Średnia wartość metryki Sorensena-Dice'a dla modelu uczonego na danych, na których nie zastosowano preprocessingu, wyniosła 75.90% (tabela 8.3, kolumna „softplus"). Żaden z zaproponowanych rodzajów obróbki danych nie pozwolił na uzyskanie większej wartości metryki.

9.4. Analiza predykcji sieci o najwyższej wartości metryki

Modelem sieci o największym współczynniku Sorensena Dice'a jest model kontrolny, którego architektura została przedstawiona na rysunku 8.1. Średnia wartość tej metryki dla czterech powtórzeń wyniosła 74.25%, a dla najlepszego powtórzenia – 77.87% (tabela 9.2). Sieć, która zwyciężyła w konkursie „ACDC- Lung HP" uzyskała wynik 83.73%. Najwyższy wynik dla sieci o architekturze typu U-Net wyniósł 79.68% [7].

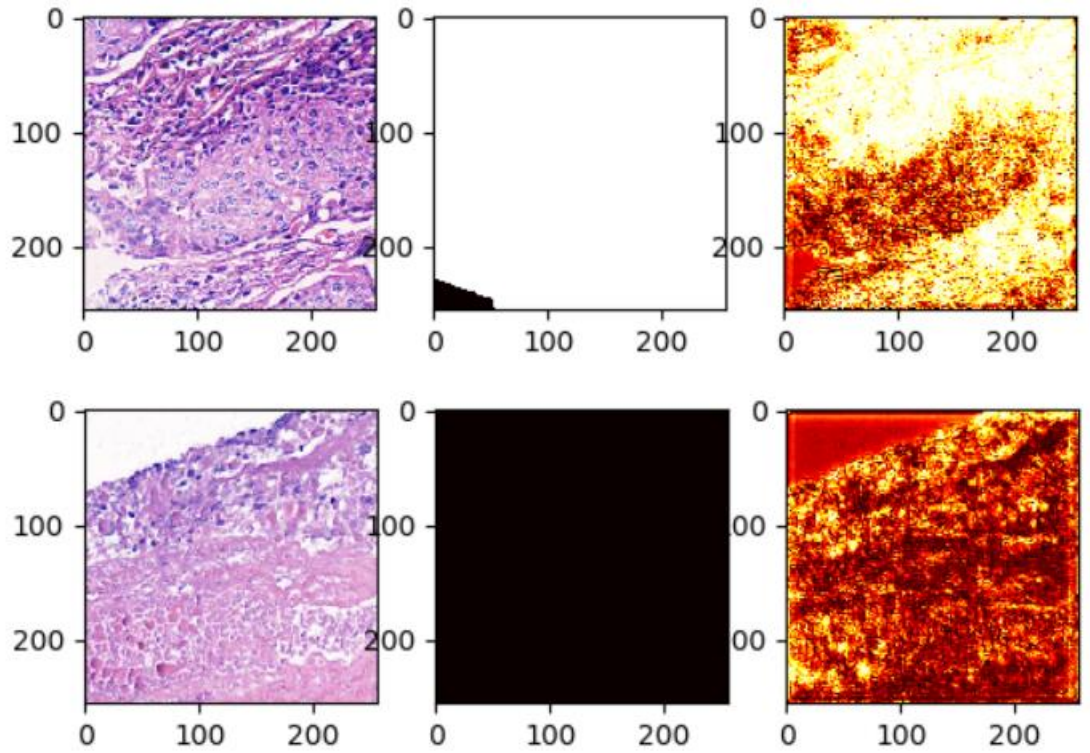
Na rysunku 9.6. przedstawiono poprawne oznaczenia tkanek wyznaczone przez sieć neuronową o największej wartości metryki Sorensena Dice'a.



Rys. 9.6. Przykłady poprawnego oznaczenia tkanek. Po lewej: fragmenty skanów histologicznych. Po środku: maska prawdy dostarczona wraz ze zbiorem danych (kolor biały – występowanie tkanki nowotworowej, kolor czarny – brak tkanki nowotworowej). Po prawej: wynik wygenerowany przez sieć. Żółto-białe fragmenty maski reprezentują fragmenty tkanki traktowanej przez siec jako nowotworowa.

Pierwszy obraz przedstawia tkankę nowotworową oraz tkankę chrzęstną. Wartości generowane przez sieć dla tkanki nowotworowej są wysokie (jasne, żółto-białe odcienie), a dla tkanki chrzęstnej – niskie (czerwono-czarne odcienie). Na drugim obrazie znajdują się fragmenty tkanki nowotworowej na białym tle. Pomimo faktu, iż dostarczona wraz ze zbiorem danych maska błędnie zaznacza tło jako tkankę nowotworową, wartości generowane przez sieć dla białego tła są zdecydowanie niższe, niż dla tkanki nowotworowej.

Rysunek 9.7. przedstawia częściowo poprawne oznaczenia tkanki nowotworowej, wyznaczone przez sieć neuronową.



Rys. 9.7. Przykłady częściowo poprawnego oznaczenia tkanek. Po lewej: fragmenty skanów histologicznych. Po środku: maska prawdy dostarczona wraz ze zbiorem danych (kolor biały – występowanie tkanki nowotworowej, kolor czarny – brak tkanki nowotworowej). Po prawej: wynik wygenerowany przez sieć. Żółto-białe fragmenty maski reprezentują fragmenty tkanki traktowanej przez siec jako nowotworowa.

Wynik działania sieci dla pierwszego obrazu jest podobny do maski dostarczonej wraz ze zbiorem danych. Fragment tkanki, który jest bledszy, nie został oznaczony przez sieć. Drugi obraz przedstawia zdrową tkankę. Na masce wygenerowanej przez sieć można zauważyć lokalne zwiększenie wartości na krawędzi tkanka – tło oraz w lewym, dolnym rogu.

10. WNIOSKI

Stworzenie modelu sieci o wysokiej dokładności segmentacji nie jest prostym zadaniem. Należy wziąć pod uwagę wiele czynników, takich jak architektura modelu, jego głębokość, rodzaj funkcji aktywacji aby zmaksymalizować jakość predykcji.

Zbiór danych w postaci całych skanów histologicznych pozwala na dowolny dobór rozmiaru danych wejściowych, oraz liczby próbek, na które skany zostaną pocięte. Wiąże się to jednak z koniecznością opracowania metod do ich obróbki, których implementacja jest czasochłonna. Istotnym elementem systemu generowania zbioru danych dla sieci neuronowej

jest algorytm selekcjonujący fragmenty skanu histologicznego posiadającego jedynie zaadnotowany przez histopatologa skrawek histologiczny. Precyzyjne wyznaczenie położenia oznakowanej tkanki na slajdzie zapobiega obecności nieoznaczonych przez histopatologa fragmentów. Akceptowanie obrazów, których średnia intensywności po wszystkich kanałach jest mniejsza niż 240 pozwoliło wykluczyć ze zbioru uczącego próbki składające się głównie z białego tła, drobnych, rozproszonych skrawków komórek oraz plam po hematoksylinie i eozynie. Znacznym ulepszeniem metodologii tworzenia zbioru danych na podstawie całych skanów histologicznych mógłby się okazać algorytm wykrywający niedoskonałości barwienia hematoksyliną i eozyną, takich jak fragmenty komórek, czy nakładania się dwóch warstw tkanki na siebie.

Transformacja koloru Reinharda umożliwia wyrównanie różnic w wybarwieniu próbek hematoksyliną i eozyną. Istotną kwestią jest dobór obrazu, z którego transfer koloru ma następować- próbka powinna być dobrze wybarwiona, pozbawiona skaz, oraz powinna zawierać zarówno elementy barwione hematoksyliną, jak i eozyną. Filtr medianowy natomiast umożliwia na wygładzenie wyraźnych krawędzi na obrazie.

Użycie przestrzeni barw HED do zapisu obrazów spowodowało zmniejszenie jakości predykcji sieci. Przedział liczb poszczególnych kanałów po konwersji z przestrzeni RGB do przestrzeni HED dla obrazów ze zbioru danych użytego w niniejszej pracy był rzędu 10^{-4}. Przeskalowanie go do przedziału [0, 255] dla każdego obrazu indywidualnie nie przyniosło poprawy jakości predykcji. Należało by zastosować inny moduł umożliwiający konwersję w celu sprawdzenia, czy powodem pogorszenia jakości predykcji jest użyta implementacja konwersji RGB do HED.

Głębokość sieci umożliwia wyuczenie się przez model rozpoznawania większej ilości cech. Wraz z każdą dodaną do modelu warstwą zwiększa się również ilość wag. Zastosowanie głębszego modelu wymaga jednak większej ilości epok, aby w pełni wykorzystać dodatkowe zmienne modelu.

Funkcja aktywacji użyta w modelu jest istotnym parametrem sieci. Zastosowanie ReLU powodowało pojawienie się problemu „martwego ReLU", który powodował niewrażliwość części neuronów na zmianę podawanych wartości. Użycie sparametryzowanego prostownika (PReLU) pozwoliło na zredukowanie problemu „martwego ReLU". Funkcją aktywacji, która osiągnęła najlepszy wynik, jest softplus, posiadająca pochodną dla każdego argumentu ze zbioru liczb rzeczywistych. Gradient tej funkcji zmienia się wraz z argumentem, w przeciwieństwie do dwóch pozostałych funkcji aktywacji.

Zastosowanie uczenia transferowego mogłoby przyspieszyć proces nauki sieci. Metoda ta polega na zainicjalizowaniu wag modelu przed nauką przy pomocy wag pochodzących z modelu o tej samej architekturze, uczonego na innym zbiorze danych, pomijając wagi ostatnich kilku warstw. Sieć poddana uczeniu transferowemu często osiąga lepsze wyniki, niż sieć o takich samych parametrach, uczona tradycyjnie przy takiej samej liczbie epok [55].

11. LITERATURA

[1] Fact sheets: Cancer [online]. Światowa Organizacja Zdrowia (WHO), 2013 (dostęp z dnia 20.05.2019). Dostępny w Internecie: https://www.who.int/news-room/fact-sheets/detail/cancer

[2] Nowotwory płuca i opłucnej [online]. Krajowy rejestr nowotworów (dostęp z dnia 20.05.2019) Dostępny w Internecie: http://onkologia.org.pl/nowotwory-pluca-oplucnej-tchawicy/

[3] Chollet, F. (2018). Deep Learning with Python: Wydawnictwo Manning, ISBN-13 978-1617294433

[4] Grand challenges in Biomedical Image Analysis [online] (dostęp z dnia 05.06.2019). Dostępny w Internecie: https://grand-challenge.org/

[5] Innowacyjne metody archiwizacji, prezentacji i udostępniania preparatów histologicznych na przykładzie funkcjonowania Centrum Archiwizacji Obrazów Morfologicznych i Cyfrowej Bazy Danych Obrazów Mikroskopowych Uniwersytetu Medycznego w Poznaniu [online]. Katedra i Zakład Histologii i Embriologii, Uniwersytet Medyczny w Poznaniu (dostęp z dnia 12.10.2019). Dostępny w internecie: http://www.script.home.pl/pbkom/roczniki/pdf2011/pbk%2011-3/s475-490.pdf

[6] Jagielska, J. (2014). Statystyczna analiza obrazów morfologicznych i jej możliwości zastosowania w mikroskopii wirtualnej [online]. Zakład Bioinformatyki i Biologii Obliczeniowej, Katedra Patomorfologii Klinicznej Uniwersytetu Medycznego w Poznaniu (dostęp z dnia 2.12.2019). Dostępny w Internecie: https://docplayer.pl/25857891-Statystyczna-analiza-obrazow-morfologicznych-i-jej-mozliwosci-zastosowania-w-mikroskopii-wirtualnej.html

[7] Konkurs „ACDC – Lung HP" (2019). Dostęp z dnia 05.06.2019. Dostępny w Internecie: https://acdc-lunghp.grand-challenge.org/

[8] Lung Cancer: Forms of lung cancer. Columbia University, Herbert Irving Comprehensive Cancer Center (dostęp z dnia 20.11.2019). Dostępny w Internecie: https://cancer.columbia.edu/lung-cancer-forms-lung-cancer

[9]Lung cancer risk factors (2019). American Cancer Society (dostęp z dnia 17.10.2019). Dostępny w Internecie: https://www.cancer.org/cancer/lung-cancer/prevention-and-early-detection/risk-factors.html

[10] „Small Cell Lung Cancer – What is it?" (2017). Harvard Health Publishing (dostęp z dnia 12.05.2019). Dostępny w Internecie: https://www.health.harvard.edu/a_to_z/small-cell-lung-cancer-a-to-z

[11] Collins, L., Haines, C. (2007). Lung Cancer: Diagnosis and Management [online] (dostęp z dnia 07.05.2019). Dostępny w Internecie: https://www.researchgate.net/profile/Lauren_Collins2/publication/6576120_Lung_Cancer_Diagnosis_and_Management/links/00b4952d3f9c7b8434000000/Lung-Cancer-Diagnosis-and-Management.pdf

[12] Hammerschmidt, S. (2009). Lung Cancer: Current Diagnosis and Treatment [online] (dostęp z dnia 12.04.2019). Dostępny w Internecie: ttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC2797332/

[13] What is H&E? The Histology Guide, University of Leeds [online] (dostęp z dnia 12.04.2019). Dostępny w Internecie: https://www.histology.leeds.ac.uk/what-is-histology/H_and_E.php

[14] Lung Cancer. Biopsy through the skin (2019). Cancer Research UK [online] (dostęp z dnia 07.05.2019). Dostępny w Internecie: https://www.cancerresearchuk.org/about-cancer/lung-cancer/getting-diagnosed/tests/biopsy-through-skin

[15] Barwienie hematoksyliną i eozyną. Kolchem (dostęp z dnia 27.05.2019). Dostępny w Internecie: http://www.kolchem.pl/instrukcje/he_instrukcja.pdf

[16] CS231n Convolutional Neural Networks for Visual Recognition. Dostęp z dnia 07.05.2019. Dostępny w Internecie: http://cs231n.github.io/neural-networks-1/
[17] Ketkar, N. (2017). Deep Learning with Python - A Hands-on Introduction. Wydawnictwo Apress, ISBN-13 978-1-4842-2766-4
[18] Dumoulin V., Visin F. - A guide to convolution arithmetic for deep learning[online] (dostęp z dnia 27.05.2019). Dostępny w Internecie: https://arxiv.org/pdf/1603.07285.pdf
[19] Ronnenberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation (dostęp z dnia 07.11.2019). Dostępny w Internecie: https://arxiv.org/pdf/1505.04597.pdf
[20] U-Net (Biomedical Image Segmentation). Towards Data Science (dostęp z dnia 07.11.2019). Dostępny w Internecie: https://towardsdatascience.com/review-u-net-biomedical-image-segmentation-d02bf06ca760
[21] Strona Katedry Nauk Komputerowych Wydziału Inżynierii na Uniwersytecie w Freiburg [online] (dostęp z dnia 10.11.2019). Dostępny w Internecie: https://lmb.informatik.uni-freiburg.de/people/ronneber/isbi2015/
[22] Sparse autoencoder [online]. Uniwersytet w Stanford (dostęp z dnia 21.11.2019). Dostępny w Internecie:
http://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf
[23] Zaccone, G., Karim, R. (2018). Deep Learning With Tensorflow, Second Edition. Wydawnictwo Packt. ISBN 978-1-78883-110-9
[24] Liu, Y. (2019). Python Machine Learning By Example. Wydawnictwo Packt. ISBN 978-1-78961-672-9
[25] Moore, S. (2018). Deep Learning for Computer Vision. Wydawnictwo Packt. ISBN 978-1-78829-562-8
[26] Meyer, A., Garcia, A., Souza, A. (2003). Comparison of similarity coefficients used for cluster analysis with dominant markers in maize. Dostępny w Internecie: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.198.3853&rep=rep1&type=pdf
[27] Hausdorff distance [online]. *Wikipedia: The Free Encyclopedia*. Dostęp z dnia 2.12.2019. Dostępny w Internecie:
https://en.wikipedia.org/w/index.php?title=Hausdorff_distance&oldid=930212269
[28] Zhang, L., Zheyu, H., Jiaolong, X., Tao, T., Hui, C. i inni. Computer-aided diagnosis of lung carcinoma using deep learning – a pilot study. Dostępny w Internecie: https://arxiv.org/ftp/arxiv/papers/1803/1803.05471.pdf
[29] Rolls, G., Farmer, N., Hall, J. Artifacts in Histological and Cytological Preparations (dostęp z dnia 07.12.2019). Dostępny w Internecie:
https://www.leicabiosystems.com/fileadmin/academy/Artifacts_Handbook.pdf
[30] Developer Survey Results 2019 [online]. StackOverflow (dostęp z dnia 21.10.2019). Dostępny w Internecie: https://insights.stackoverflow.com/survey/2019
[31] Comparing Python to Ohther Languages [online]. Python Software Foundation Site (dostęp z dnia 21.10.2019). Dostępny w Internecie:
https://www.python.org/doc/essays/comparisons/
[32] Applications for Python [online]. Python Software Foundation Site (dostęp z dnia 21.10.2019). Dostępny w Internecie: https://www.python.org/about/apps/
[33] Repozytorium GitHub programu „Automated Slide Analysis Platform" [online] (dostęp z dnia 15.10.2019). Dostępny w Internecie:
https://github.com/computationalpathologygroup/ASAP
[34] Keras: The Python Deep Learning library [online]. Keras Documentation (dostęp z dnia 21.10.2019). Dostępny w Internecie: https://keras.io/
[35] Strona główna biblioteki TensorFlow [online] (dostęp z dnia 21.10.2019). Dostępny w Internecie: https://www.tensorflow.org/about

[36] Get started with TensorBoard [online] (dostęp z dnia 21.10.2019). Dostępny w Internecie: https://www.tensorflow.org/tensorboard/get_started
[37] Strona główna biblioteki NumPy [online] (dostęp z dnia 21.10.2019). Dostępny w Internecie: https://numpy.org/
[38] Pandas Documentation [online] (dostęp z dnia 21.10.2019). Dostępny w Internecie: https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html
[39] Scikit-image: Image processing in Python [online] (dostęp z dnia 21.10.2019). Dostępny w Internecie: https://scikit-image.org/
[40] Dokumentacja biblioteki Matplotlib [online] (dostęp z dnia 21.10.2019). Dostępny w Internecie: https://matplotlib.org/contents.html
[41] Strona główna Wrocławskiego Centrum Sieciowo-Superkomputerowego [online] (dostęp z dnia 01.12.2019). Dostępny w Internecie: https://www.wcss.pl/
[42] Kohlberger, T., Yun, L., Moran, M. i inni. Whole-Slide Image Focus Quality: Automatic Assessment and Impact on AI Cancer Detection [online] (dostęp z dnia 19.11.2019). Dostępny w Internecie: https://arxiv.org/ftp/arxiv/papers/1901/1901.04619.pdf
[43] Małowidzki, M., Bereziński, P., Mazur, M. (2015). Network Intrusion Detection: Half a Kingdom for a Good Dataset.
[44] Repozytorium GitHub użytkownika zhixuhao: unet [online] (dostęp z dnia 17.09.2019). Dostępny w Internecie: https://github.com/zhixuhao/unet
[45] Shrivastava, H., Sridharan, S. (2013). Conception of data preprocessing and partitioning procedurę for machine learning algorithm [online] (dostęp z dnia 16.11.2019). Dostępny w Internecie: https://www.academia.edu/9517738/CONCEPTION_OF_DATA_PREPROCESSING_AND_PARTITIONING_PROCEDURE_FOR_MACHINE_LEARNING_ALGORITHM
[46] Sampias, C. H&E Basics Part 4: Troubleshooting H&E [online] (dostęp z dnia 21.11.2019). Dostępny w Internecie: https://www.leicabiosystems.com/knowledge-pathway/he-basics-part-4-troubleshooting-he/
[47] Guidelines for hematoxylin & eosin staining. Strona National Society of Histotechnology [online] (dostęp z dnia 21.11.2019). Dostępny w Internecie: http://nsh.org/sites/default/files/Guidelines_For_Hematoxylin_and_Eosin_Staining.pdf
[48] Reinhard, E., Ashikhmin, M., Grooch, B., Shirley, P. (2001). Color transfer between images [online] (dostęp z dnia 04.12.2019). Dostępny w Internecie: https://www.cs.tau.ac.il/~turkel/imagepapers/ColorTransfer.pdf
[49] Chung-Ming, W., Yao-Hsien, H., Ming-Long, H. (2006). An effective algorithm for image sequence color transfer [online] (dostęp z dnia 4.12.2019). Dostępny w Internecie: https://www.sciencedirect.com/science/article/pii/S089571770600032X#fd1
[50] Ruifrok, A., Johnston, D. (2001). Quantification of Histochemical Staining by Color Deconvolution [online] (dostęp z dnia 04.12.2019). Dostępny w Internecie: https://pdfs.semanticscho lar.org/e0a0/dc42714dd8add295cb43c15110d86af95cf9.pdf
[51] Drakos, N. (1996). Median Filter [online]. Computer Based Learning Unit, University of Leeds (dostęp z dnia 04.12.2019). Dostępny w Internecie: http://fourier.eng.hmc.edu/e161/lectures/smooth_sharpen/node2.html
[52] Vasilew, I., Slater, D., Spacagna, G., Roelants, P., Zocca, V. (2019). Python Deep Learning, Second Edition. Wydawnictwo Packt. ISBN 978-1-78934-846-0
[53] Zwonarz, W. Wstępne uwagi o sieci neuronowej. W: Algorytmy wysokiej dokładności śledzenia trajektorii robota przemysłowego [online] (dostęp z dnia 14.12.2019). Dostępny w Internecie: https://www.eaiib.agh.edu.pl/files/3175/Zwonarz_Praca_doktorska_calosc.pdf
[54] Lu, L., Yeonjong, S., Yanhui, S., Karniadakis, G. (2019). Dying ReLU and Initialization: Theory and Numerical Examples [online] (dostęp z dnia 13.12.2019). Dostępny w Internecie: https://arxiv.org/pdf/1903.06733.pdf

[55] Bonaccorso, G. (2018). Mastering Machine Learning Algorithms. Wydawnictwo Packt. ISBN 978-1-78862-111-3