Prototyp rozwiązania projektu realizowanego w ramach przedmiotu Techniki Obrazowania Medycznego 2020

1. Przygotowanie danych

W celu przygotowania danych do zastosowania uczenia maszynowego każdy obraz standaryzowano do skali Hounsfielda w celu pozbycia się artefaktów niekorzystnych do uczenia modelu, odcięcie ustawiono na wartości [-700,700] w ten sposób usuwając większość niechcianych elementów na obrazie. Zdecydowaliśmy zapisać tak przetworzone obrazy w osobnych plikach .png w celu ułatwienia późniejszego wczytywania danych do modelu.

Dane zostały podzielone na zestawy train, development oraz test odpowiednio zawierające 80%, 10%, 10% danych, razem licząc 210 obiektów. Dla każdego obrazu zapisywaliśmy również odpowiadający plik .png który zawierał dostarczone wyniki segmentacji w którym można rozróżnić trzy wartości 0 oznaczające tło, 1 nerkę oraz 2 dla nowotworu nerki.

2. Architektura modelu

W naszym projekcie zastosowaliśmy sieć UNet, która została zaprojektowana do segmentacji obrazów biologicznych. Architekturę sieci można podzielić na dwie większe ścieżki. Pierwsza to tak zwany koder, służący do wyłapywania kontekstu w obrazie, a druga to symetryczny do kodera, dekoder. Druga część ścieżki służy do precyzyjnej lokalizacji przy użyciu transponowanych warstw.

W naszym modelu koder jest zbudowany z 5 bloków. Każdy blok jest zbudowany z 2 warstw konwolucyjnych (Conv2D 3x3) o takich samych rozmiarach filtra i z warstwy MaxPooling. Wraz z głębokością sieci ilość filtrów rośnie od 32 do 512 reprezentując o kolejne potęgi dwójki. Po 5 blokach znajdują się 2 warstwy konwolucyjne z 1024 filtrami.

Następnie przechodzimy do dekodera. Zbudowanego z 5 bloków, w których występują identyczne pary warstw konwolucyjnych, z tą różnicą, że ilość filtrów maleje. Każdy blok rozpoczyna się od warstwy transponowanej, która wykorzystuje informacje zebrane przez sieć w części kodera.

Wszystkie przedstawione warstwy wykorzystywały jako funkcję aktywacyjną ReLu. Jedynie ostatnia warstwa, która decyduje o przynależności pixeli do jednej z trzech klas (tło, nerka, nowotwór nerki) oparta jest na funkcji softmax.

3. Pobieranie danych

W celu wykorzystania wszystkich obrazów wykorzystano klasę ImageDataGenerator, która umożliwia pobieranie danych ze ścieżki w zadanych porcjach. Wykorzystując tą klasę dokonano normalizacji danych. Obrazy CT podzielono na 255 w celu uzyskania wartości 0-1, a obrazy segmentacji podzielono przez 127 zapewniając postać całkowitą w celu otrzymania wartości 0 dla tła, 1 dla nerki i 2 dla nowotworu nerki. ImageDataGenerator pozwala pobierając dane od razu zmienić ich wymiar. Obrazy przesyłane do sieci mają wymiary 256 x 256. Gdy były większe zbierały za dużo pamięci, a

gdyby je jeszcze zmniejszyć mogłaby nastąpić utrata pewnych informacji. Ze względu na możliwości pamięci użyto batch_size = 16, co mocno spowalnia model, ale pozwala na wykorzystanie wszystkich dostępnych danych.

4. Optymalizacja modelu

Do optymalizacji modelu wybrano optymalizator Adam o standardowych wartościach $\beta_1 = 0.9$ oraz $\beta_2 = 0.999$. Ustawiono wskaźnik uczenia na $\alpha = 3 \cdot 10^{-4}$, a spadek szybkości uczenia na $decay = 0.2$

5. Uczenie modelu

Do uczenia modelu wykorzystano funkcję kosztu dice loss, bardzo popularną dla problemu segmentacji segmentacji. Najpierw obliczono wartości dice coefficient dla każdej klasy osobno, a następnie obliczono ich średnią ważoną. Wagi nadano ręcznie najmniejszą dla tła - 0.1, gdyż na obrazach tła jest zdecydowanie najwięcej, a jego rozpoznawanie najmniej nas interesuje. Klasie 1, czyli nerce nadano wagę 0.3. Klasie 2, nowotworowi nerki, przypisano wagę 0.6, ponieważ jest to najmniejszy obszar na obrazach i najtrudniejszy do segmentacji. Podczas uczenia zastosowano Early Stopping oraz 30 epok.

6. Ewaluacja modelu

Do ewaluacji modelu wykorzystano średnią ważoną wartości dice coefficient dla wszystkich klas. Końcowy wynik projektu to ewaluacja dla zbioru testowego, wcześniej nie widzianego przez sieć. Dla zbioru train set uzyskaliśmy wartość dice score 57,% , dla zbioru dev set 48%, a dla najbardziej istotnego zbioru test set 46%. Pobieranie danych, uczenie modelu i ewaluacja jest realizowana w programie training_network.py.