Akademia Górniczo-Hutnicza im Stanisława Staszica
Wydział Automatyki, Elektroniki Informatyki i Inżynierii Biomedycznej

Techniki Obrazowania Medycznego

Prototyp realizacji projektu

Krystian Strzałka, Jakub Chabior, Katarzyna Urbanek, Aleksandra Augustyn
Kraków, 22 czerwca 2020

Spis treści

1 Cel **2**

2 Wstęp **2**

3 Struktura kodu **4**

4 Wyniki **6**

5 Dyskusja oraz podsumowanie **9**

6 Podział zadań **9**

1 Cel

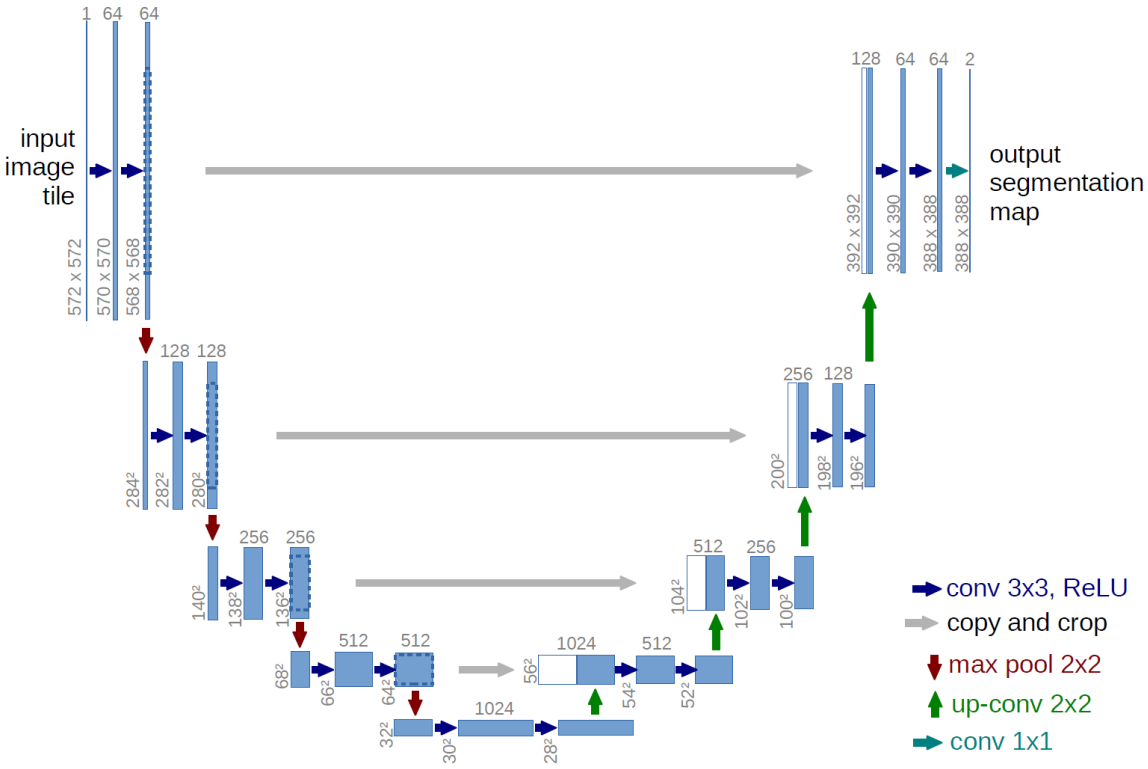
Celem projektu było opracowanie rozwiązania jednego z problemów zaprezentowanych na konferencji naukowej MICCAI 2019. Należało zaimplementować automatyczną metodę segmentacji nerek i nowotworów nerek na obrazach tomografii komputerowej oraz zaproponować sposób ewaluacji uzyskanych wyników.

2 Wstęp

Nowotwór nerki jest coraz częściej występującą chorobą, której późne wykrycie najczęściej prowadzi do śmierci pacjenta. Do obecnie stosowanych metod leczenia należy nefrektomia radykalna, która polega na całkowitym wycięciu guza wraz z nerką oraz otaczającą tkanką tłuszczową. Jednak postęp medycyny oraz dokładność obrazowania coraz częściej umożliwiają przeprowadzenie częściowej nefrektomii, która polega na wycięciu guza, ale pozostawienie nerki. Bardzo ważną rolę w diagnostyce nowotworów guza gra odpowiednia segmentacja nerek oraz guzów na obrazach tomografii komputerowej (CT). Obecnie segmentacja jest wykonywana ręcznie przez specjalistów, jednak taka metoda generuje wiele niedokładności oraz w znacznym stopniu zależy od doświadczenia osoby wykonującej segmentację. W celu usprawnienia oraz zwiększenia dokładności tego procesu, niezbędna jest automatyzacja.

Segmentacja obrazów polega na wykonaniu podziału na regiony o zbliżonych właściwości na podstawie poziomu szarości, koloru, tekstury, kontrastu czy jasności. Wykonuje się ją w celu określenia budowy anatomicznej, zlokalizowania nieprawidłowości (jak na przykład nowotwory) oraz określenia rozmiarów danej tkanki (przykładowo w celu określenia przyrostu rozmiaru guza). Segmentację obrazów CT można wykonać między innymi poprzez dobór progu na podstawie histogramu, grupowanie obszarów na podstawie ich cech, metodami obszarowymi lub krawędziowymi. Bardzo obiecującą metodą jest uczenie sieci neuronowych, które analizując obszary obrazu są w stanie rozpoznawać pewne nieskomplikowane zależności jak krawędzie. [1]

Sieć konwolucyjna U-Net (Rysunek 1) została stworzona właśnie do rozpoznawania krawędzi na obrazach biomedycznych. U-Net składa się z dwóch podstruktur - kodera (encoder) oraz dekodera (decoder). Pierwsza część ma na celu uchwycenie kontekstu na obrazie poprzez zmniejszanie odbieranych pól w trakcie przechodzenia przez warstwy, natomiast dekoder jest nazywany ścieżką symetrycznego rozrastania i ma na celu określanie precyzyjnej lokalizacji wykorzystując transponowane sploty. Sieć taka może przyjmować za dane wejściowe obraz o dowolnym rozmiarze (tak długo jak są kilkukrotnie podzielne przez 2), zwracając obraz o zbliżonej wielkości. [2]



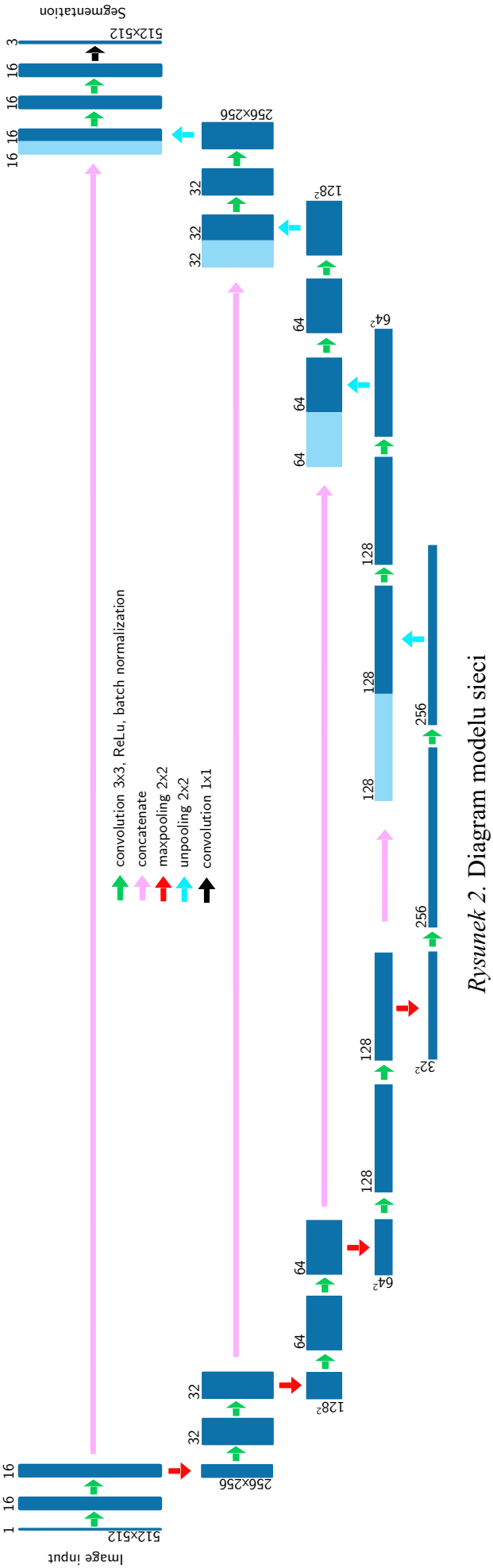
Rysunek 1. Architektura sieci U-Net [3]

Podczas uczenia się sieci neuronowej, bardzo ważne jest zastosowanie normalizacji w celu przyspieszenia procesu, zwiększenia dokładności oraz zminimalizowania błędów. Jednym z takich błędów jest wewnętrzna zmiana współzmiennej (internal covariate shift), czyli zmiana parametrów sieci w trakcie uczenia się. W celu eliminacji tego błędu, najczęściej stosowane jest Batch Normalization po każdej warstwie konwolucyjnej. Normalizacja ta polega na ponownym centrowaniu i skalowaniu warstwy. Innymi typami normalizacji stosowanymi do eliminacji wewnętrznej zmiany współzmiennej jest Instance Normalization oraz Layer Normalization. [4]

3 Struktura kodu

Tabela 1
Opis plików w repozytorium

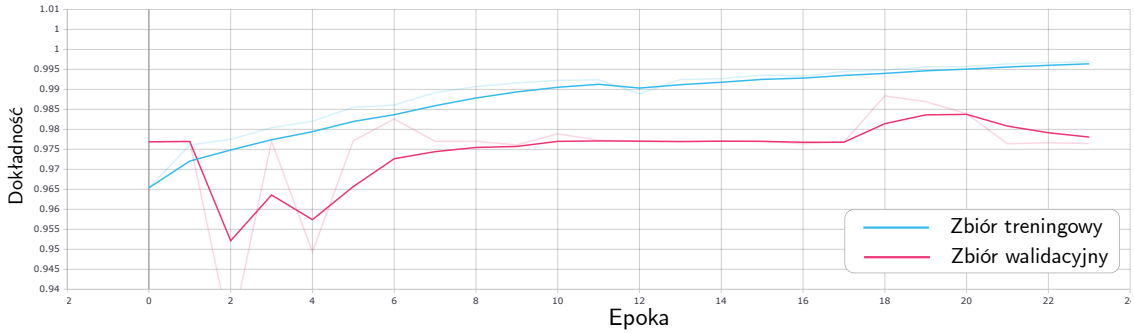
Nazwa	Opis
preprocessing.py	Przygotowanie obrazów CT do dalszej obróbki.
scale	Skalowanie obrazu od 0 do 1 poprzez odjęcie minimum i podzielenie przez 4095, czyli maksymalną wartość skanów.
find segmentation	Następuje dodanie do siebie kolejnych *slice*, w momencie, gdy suma wartości któregoś jest większa od 0, to funkcja zwraca indeks początku segmentacji, podobny proces przebiega także od końca segmentacji skanu aby uzyskać indeks końca segmentacji.
preprocess X	Obiekt niftyImage zamieniany jest na numpy array, następnie skalowany za pomocą funkcji scale. Następuje dodanie dodatkowego wymiaru ze względu na architekturę sieci. W przypadku jeśli drugi albo trzeci wymiar jest inny niż 512x512 to zostaje przeskalowany do 512x512.
preprocess Y	Również działa na obiekcie niftyImage i zamienia go na numpy array. Znajduje początek i koniec segmentacji funkcją find segmentation, a następnie zamienia na dane kategoryczne. Ewentualnie skalowanie i zwraca, numpy array y, a także początek oraz koniec fragmentu, w którym znajduje się nerka.
dataGenerator.py	Skrypt korzysta z danych zwróconych przez preprocessing.py, zawiera generatory, które przekazują sieci odpowiednie dane.
trainGenerator	Iteruje po *case* listy treningowej, wybiera fragmenty, w których znajdują się nerki, zwracaja fragmenty o rozmiarach [batch size, 512, 512, 1] dla X i [batch size, 512, 512, 3] dla y.
validation generator	Działanie identyczne jak w przypadku trainGenerator, dodatkowo wypisuje informacyjny tekst „validation in progress”.
NeuralNetwork.py	Skrypt zawierający architekturę modelu.
makeModel	Funkcja zwracająca obiekt ze skompilowanym modelem.
training.py	Wczytanie modułów, tworzenie instancji modelu. Tensorboard pozwala na podgląd parametrów w czasie uczenia modelu.
predict.py	Skrypt wczytuje zadany case oraz wyświetla okienko matplotlib ze poziomym suwakiem i trzema obrazami - wejściowym, ground truth segmentetion oraz our segmentation. W celu wyświetlenia wyniku case o danym numerze należy wpisać w konsoli na przykład „python predict.py 1” dla przypadku 1. Wymagany jest folder data z repozytorium kits19 w tym samym folderze co skrypt. Możliwe jest zapisanie segmentacji w formacie *.nii* poprzez podanie dowolnego drugiego argumentu w postaci liczby całkowitej.



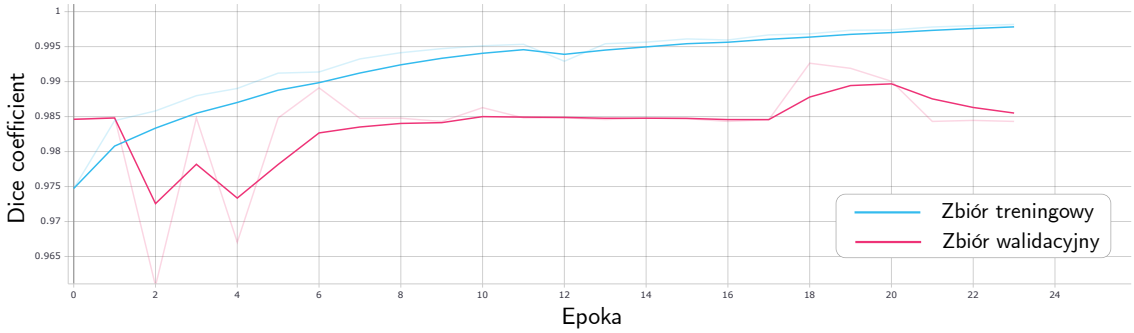
Rysunek 2. Diagram modelu sieci

4 Wyniki

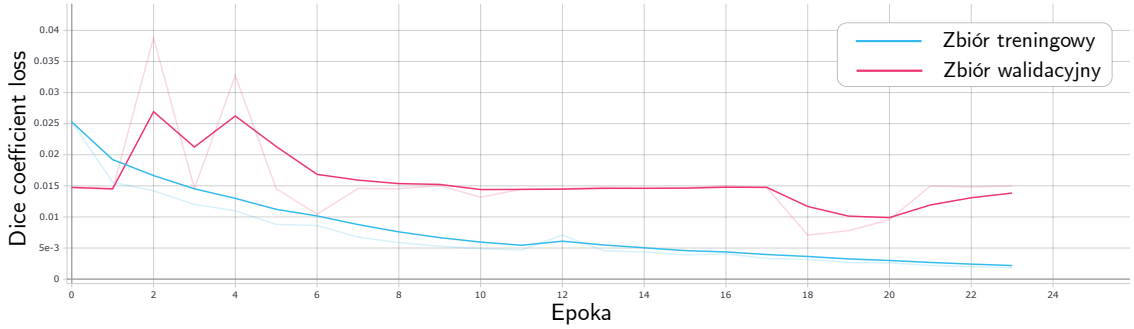
Model uczył się na przypadkach od 0 do 148 a ewaluacja przebiegła na przypadkach od 149 do 209 z wyłączeniem przypadków 158, 159, 170, 202 ponieważ były one uszkodzone. Użyta została tylko funkcja kosztu na bazie współczynnika Dice'a ponieważ do wykorzystania dwóch funkcji kosztu wymagane byłby dwa wyjścia sieci. Jako metryki wykorzystano dokładność oraz współczynnik Dice'a. Model jest w stanie prawidłowo rozpoznać nerki, lecz wyłącznie w niektórych przypadkach jest w stanie określić dokładne położenie komórek nowotworowych. Podczas trenowania ewaluowano model przy pomocy dokładności oraz współczynnika Dice'a. Wyniki przedstawiono na rysunkach 3, 4 oraz 5.



Rysunek 3. Dokładność zbioru treningowego oraz walidacyjnego



Rysunek 4. Współczynnik Dice'a zbioru treningowego oraz walidacyjnego

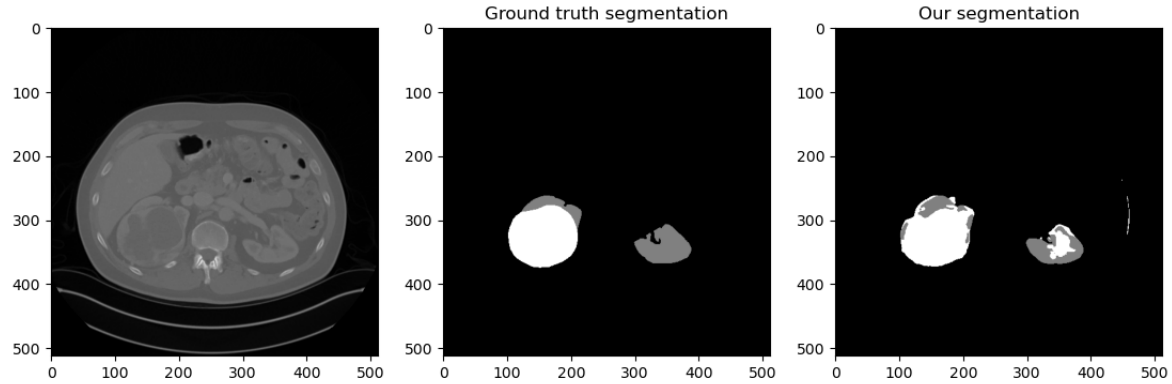


Rysunek 5. Funkcja kosztu bazowana na współczynniku Dice'a dla zbioru treningowego oraz walidacyjnego

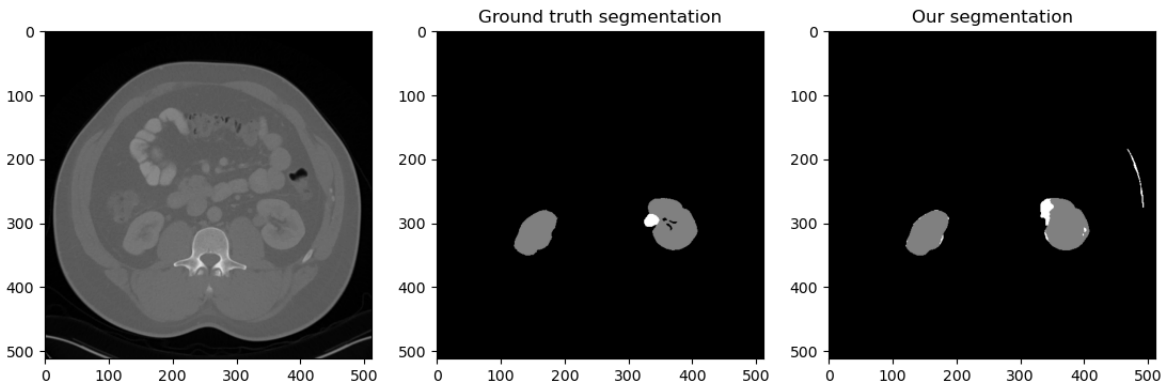
Tabela 2
Wartości ewaluacji dla modelu (epoka 18)

	Dokładność	Dice	Dice koszt.
Zbiór treningowy	0.9948	0.9968	0.003177
Zbiór walidacyjny	0.9884	0.9926	0.007071

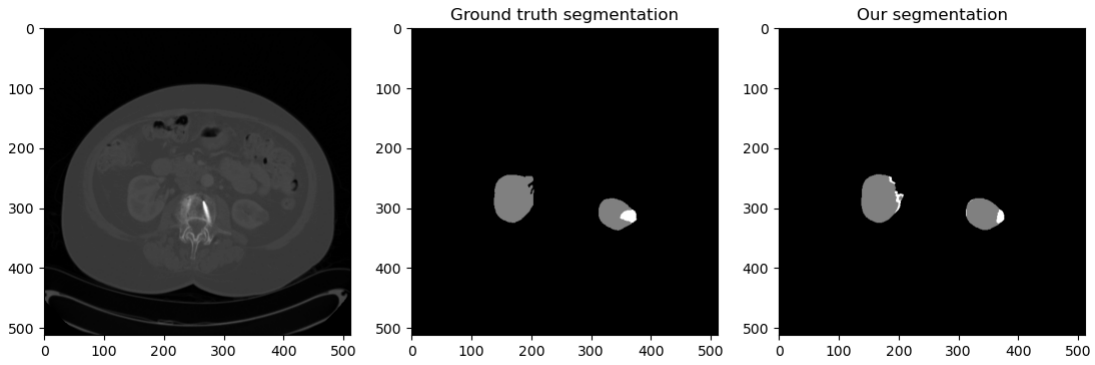
Poniżej umieszczono porównanie rzeczywistego oraz przewidzianego położenia nowotworu dla wybranych przypadków.



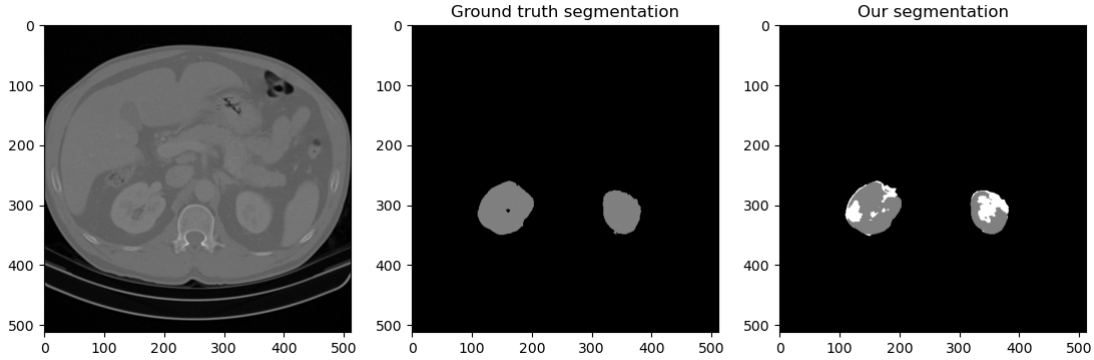
Rysunek 6. Predykcja położenia nowotworu w przypadku nr 55



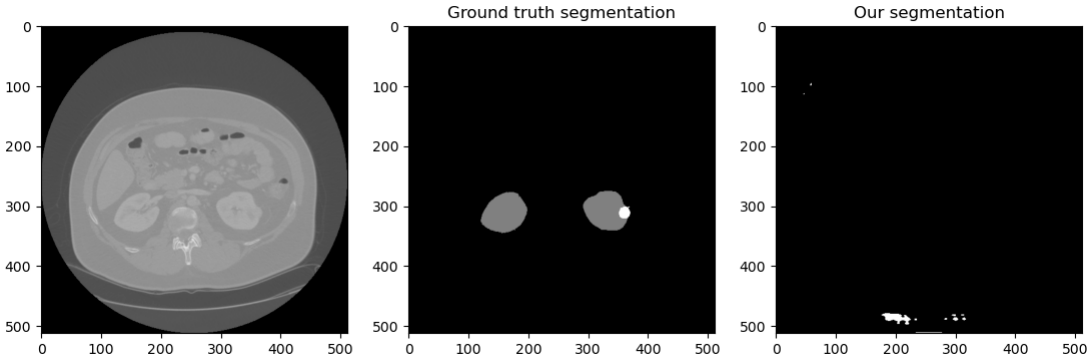
Rysunek 7. Predykcja położenia nowotworu w przypadku nr 209



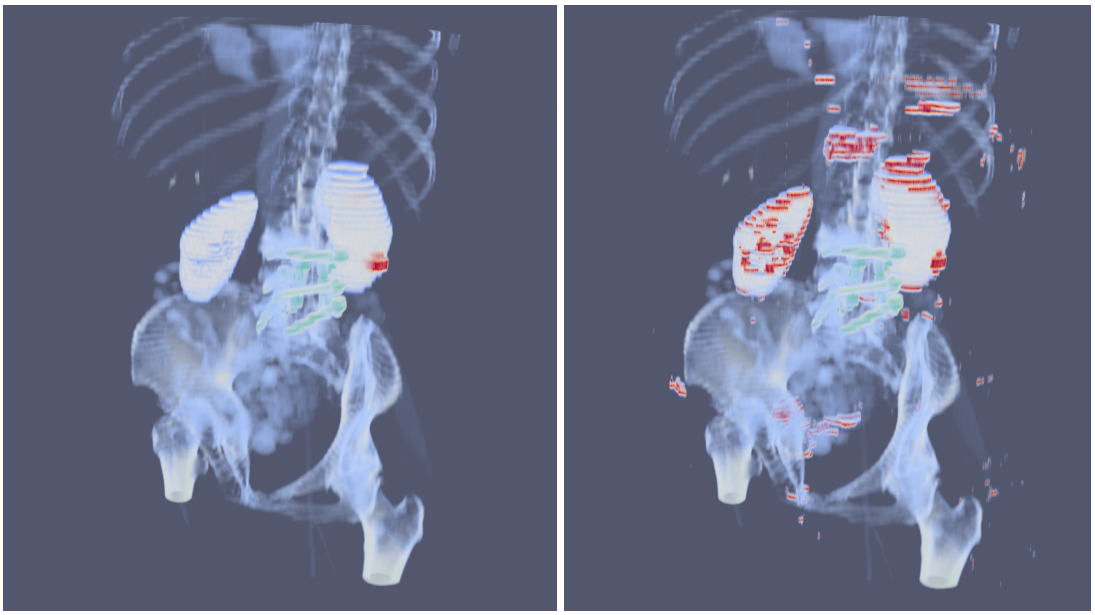
Rysunek 8. Predykcja położenia nowotworu w przypadku nr 205



Rysunek 9. Predykcja położenia nowotworu w przypadku nr 190



Rysunek 10. Predykcja położenia nowotworu w przypadku nr 123



Rysunek 11. Wolumetryczne przedstawienie, wygenerowanej segmentacji. Po lewej segmentacja wykonana przez lekarza a po prawej segmentacja wykonana przez model. Łatwo zauważyć że model jest zbyt czuły jeśli chodzi o zmiany nowotworowe lecz prawidłowo segmentuje sam narząd.

5 Dyskusja oraz podsumowanie

Stworzony model nie daje przewidywalnych wyników, dla niektórych przypadków zwraca dokładną segmentację nowotworu, natomiast dla innych generuje błędne wyniki prawdopodobnie przez zbyt dużą czułość. Oznacza to, że model nadal wymaga dopracowania oraz pewnych korekcji. W przyszłości planowane byłoby stworzenie bazy danych w formacie *.hdf*, oraz wykonanie generatora w postaci obiektu *tf.keras.utils.Sequence* aby zapewnić wielowątkowość i przyspieszyć generowanie wsadów ponieważ podczas procesu uczenia GPU nie było w pełni zutylizowane przez wolny proces wczytywania danych. Lepsze wyniki mógłby dać lepiej dobrany optymalizator sieci ponieważ po wykorzystanym modelu z epoki 18, model zaczął zbaczać z trajektorii i przestał generować jakiekolwiek segmentacje.

Na podstawie załączonych przykładowych predykcji położenia nowotworu widać, że w przypadku numer 55, 209 oraz 205 nasza segmentacja była dokładna. Natomiast w zdjęciu z przypadku 190 nowotwór został oznaczony mimo jego braku, a w przypadku numer 123 w ogóle nie doszło do segmentacji nerki.

6 Podział zadań

Augustyn Aleksandra - dokumentacja, pomoc przy researchu
Chabior Jakub - pomoc w pisaniu kodu i rozwiązywaniu problemów
Strzałka Krystian - pisanie kodu, trenowanie sieci, opracowanie architektury programu, wizualizacje i grafy do dokumentacji
Urbanek Katarzyna - research, pomoc przy ewaluacji

Literatura

[1] Sharma N, Aggarwal LM. Zautomatyzowane techniki segmentacji obrazów medycznych. J Med Phys. 2010.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

[3] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/.

[4] Xiao-Yun Zhou and Guang-Zhong Yang. Normalization in training u-net for 2d biomedical semantic segmentation, 2018.

[5] Nicholas Heller, Niranjan Sathianathen, Arveen Kalapara, Edward Walczak, Keenan Moore, Heather Kaluzniak, Joel Rosenberg, Paul Blake, Zachary Rengel, Makinna Oestreich, Joshua Dean, Michael Tradewell, Aneri Shah, Resha Tejpaul, Zachary Edgerton, Matthew Peterson, Shaneabbas Raza, Subodh Regmi, Nikolaos Papanikolopoulos, and Christopher Weight. The kits19 challenge data: 300 kidney tumor cases with clinical context, ct semantic segmentations, and surgical outcomes, 2019.