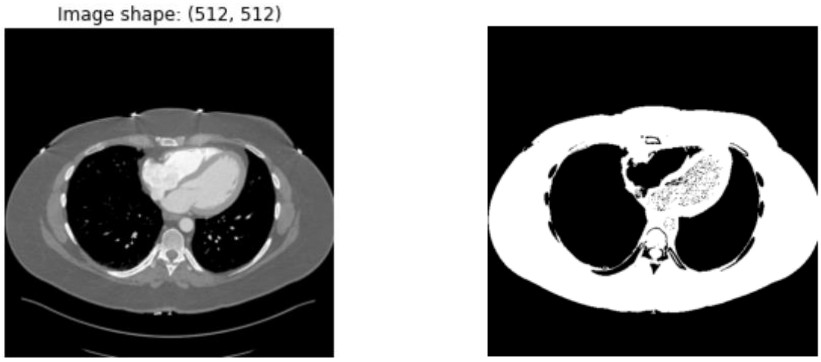
AGH	**Grupa 1:** Ceglarz Anna Karolina Barczyk Justyna Tokarz Natalia Wasik	**Wydział:** EAIiIB **Rok:** 2019/2020
	Automatyczna segmentacja	

W trakcie pracy nad projektem napotkano pewne problemy. Pierwotna wersja projektu zakładała segmentację poprzez region-growing. Pojawił się problem odnośnie wyboru kryterium jednolitości.
W takiej sytuacji postanowiono poszukać innego rozwiązania, w którym zastosowano semantyczną segmentację z wykorzystaniem sieci U-net.
Pierwszą rzeczą jest pre-processing poprzez binaryzację obrazu, następnie jest znajdowany obraz, który zawiera wystarczającą liczbę cech. Na końcu planowana jest właśnie zastosowanie semantycznej segmentacji za pomocą sieci U-net.
Współpracowano nad projektem w chmurowym notatniku colab. Poniżej pokazano etapy ewaluacji naszego projektu:

Pre-processing
a) Zwizualizowano pliki z case'ów:

```
%cd kits19
from starter_code.visualize import visualize
visualize(3, "/content/image_visualize")
```

b) Następnie wykonano normalizację oraz binaryzacje:



Rys. 1. Efekt po normalizacji (po lewej) i po binaryzacji (po prawej)

Wybranie obrazu z najlepszymi cechami

Do wybrania obrazu z najlepszymi cechami skupiono się na użyciu adaptive global maximum cluster. Wytworzony tu zostaje histogram z automatyczną liczbą lokalnych maksimów - liczba ta wskazuje nam liczbę różnych regionów na obrazie. I program wybiera największe maksimum.

Model sieci U-net

Zbiór danych podzielono na dwa zestawy: Train_Data i Test_Data.

W architekturze sieci oparto się up-samplingu i transponowanej konwolucji. Kolejne 3 warstwy łączono na bazie splotu połączonego z aktywacją ReLu(), czyli funkcją max(0,x) nie wpływającą na rozmiar obrazu oraz z operacją łączenia (max pooling) polegającą na próbkowaniu przestrzennym względem wysokości i szerokości, i wyznaczeniu maksimum, co skutkuje wynikowym wymiarem. Sam max pooling ma na celu zmniejszenie kosztu obliczeniowego poprzez zmniejszenie ilości parametrów - dyskretyzacja oparta na próbkach. Przy tworzeniu sieci skorzystano również z biblioteki TensorFlow, po uprzednim zdegradowaniu jej do wersji 1.0, co umożliwiło korzystanie z funkcji takich jak przykładowo `tf.nn.relu, tf.cast,tf.nn.softmax` czy `tf.nn.conv2d`.

Podczas implementacji nałożono *ground truth mask*, dzięki której wydzielono pierwszy i drugi plan obrazu oraz *generated mask* na oryginalną maskę.

Planowo algorytm miał modyfikować wagi błędów, jednak przy obliczaniu wartości kosztu korzystając z połączenia błędu średniokwadratowego (znajdując średnią kwadratową różnice między wartością przewidywaną a rzeczywistą) oraz zaobserwowano wystąpienie błędu:

```
Iter:  1  Cost:  nan
----------------------
Iter:  2  Cost:  nan
----------------------
Iter:  3  Cost:  nan
----------------------
Iter:  4  Cost:  nan
----------------------
Iter:  5  Cost:  nan
----------------------
Iter:  6  Cost:  nan
```

Rys 2. Próba treningu sieci - błędne wyniki

Ze względu na występujące problemy, kod będzie poddany dalszej ewaluacji. Prawdopodobnie konieczne będzie poprawienie architektury sieci.

Literatura:

https://arxiv.org/abs/1505.04597

https://kits19.grand-challenge.org/?fbclid=IwAR2QNYMlT1KFrfEdbxCGj_T_LMxkQ3meWmCi5jmnpPEqUj8SjnPsASSEeyw

https://medium.com/activating-robotic-minds/up-sampling-with-transposed-convolution-9ae4f2df52d0