Techniki obrazowania medycznego

Skład osobowy:

Górak Kamil
Kozioł Bartosz
Kwater Jakub
Mermon Gabriela

Cel projektu:

Celem realizowanego projektu jest stworzenie sieci neuronowej, zdolnej do segmentacji nerek i obecnych w nich nowotworów. Projekt ten realizowany jest w języku "Python", w środowisku "Spyder". Dane oraz szczegóły dotyczące tej pracy pozyskano ze strony oraz repozytorium (GitHub) konkursu "*KiTS19 grand-challange*".

Wykorzystane metody

Zapis danych do PNG:

W celu usprawnienia pracy komputera na zadanych obrazach, zapisano je do formatu PNG, gdyż próby odczytania kilku obrazów w ich oryginalnym formacie "*.nii.gz*" skutkowały całkowitym wykorzystaniem pamięci RAM co uniemożliwiało dalszą pracę. W tym celu uprzednio utworzono odpowiednie katalogi na dane treningowe (*dataPNG\TRAIN\CT* oraz *dataPNG\TRAIN\MASK*) i testowe (*dataPNG\TEST\CT* oraz *dataPNG\TEST\MASK*).

Przed zapisem dane sprawdzano pod kątem przydatności, maski nie zawierające nerek ani komórek rakowych oraz odpowiadające im obrazy CT w większości (ok 97%) nie były zapisywane postanowiono jednak zapisać część z nich (ok 3%) w celu wprowadzenia dodatkowej różnorodności.

Wykorzystując funkcję "*load_case()*" uzyskano ilość przekrojów dla każdego przypadku.

Wykorzystane funkcje:

- "*load_case()*" - wczytanie danych w formacie "*.nii.gz*"
- "*get_fdata()"* - wczytanie ilości przekrojów i wymiarów zdjęć
- "*plt.imsave()*" - zapis zdjęć do formatu "*.PNG*"

Preprocessing:

Aby umożliwić poprawną analizę i obróbkę danych zmniejszono ich rozmiary z 512x512 na 256x256, a także sprowadzono je do skali szarości. W ten sposób przygotowane dane zostaną później wczytane do sieci neuronowej.

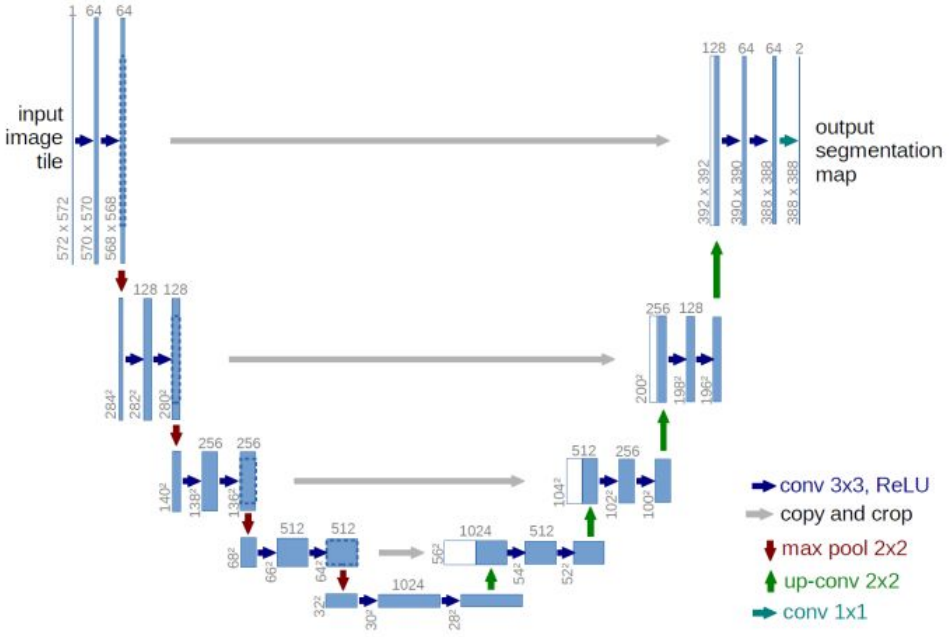
Wykorzystane funkcje:

- "*np.zeros()*" - przygotowanie macierzy o zadanych wymiarach

- "*os.listdir()*" - zwraca listę plików znajdujących się pod ścieżką wysłaną jako argument do funkcji
- "*imread()*" - odczyt zdjęcia ze ścieżki
- "*os.path.join()*" - łączy dwie zadanie ścieżki w jedną
- "*resize()*" - zmiana rozmiarów zadanej macierzy

Model sieci U-net:

Model sieci zaimplementowano z wykorzystaniem frameworku Tensorflow. Framework ten wykorzystano przez wzgląd na dobrze rozbudowaną dokumentację i liczne pomoce naukowe. Wykorzystywany model sieci składa się ze ścieżki kodującej i dekodującej i stanowi realizację poniżej przedstawionego modelu:



Rys 1. Model sieci

Wykorzystane funkcje:

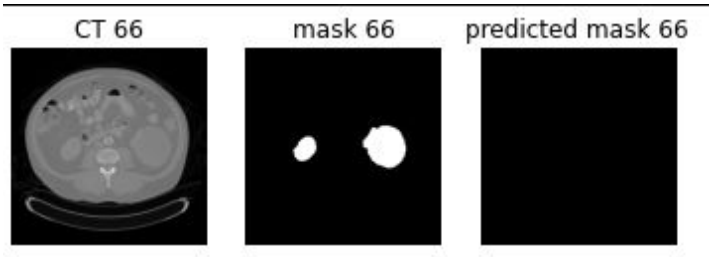
- "*tf.keras.layers.Input()*" - przyjmuje wymiary obrazów
- "*tf.keras.layers.Conv2D()*" - splot 2D
- "*tf.keras.layers.Dropout()*" - metoda zapobiegająca overfittingowi poprzez odrzucanie losowo wybranych neuronów
- "*tf.keras.layers.MaxPooling2D()*" - warstwa ta zmniejsza rozmiar danych, liczbę parametrów oraz przeprowadzanych operacji

- "*tf.keras.layers.concatenate()*" - połączenie odpowiadających sobie poziomów warstw
- "*tf.keras.layers.Model()*" - grupowanie warstw do obiektu
- "*tf.keras.layers.Model.compile()*" - konfiguracja modelu pod trening
- "*tf.keras.layers.Model.summary*" - wypisanie podsumowania sieci (liczba parametrów itd.)
- "*tf.compat.v1.ConfigProto()*", "*tf.compat.v1.Session()*" - ustawienie GPU jako urządzenia na którym mają być wykonywane obliczenia (bez powodzenia)
- "*tf.keras.layers.callbacks.EarlyStopping()*" - metoda monitorująca daną wartość i przerywająca dalszą naukę sieci jeśli owa wartość ulega pogorszeniu w zadanym zakresie tolerancji
- "*tf.keras.layers.callbacks.TensorBoard()*" - umożliwienie wizualizacji
- "*tf.keras.layers.callbacks.ModelCheckpoint()*" - zapisuje model lub jego wagi z pewną częstotliwością
- "*tf.keras.layers.Model.fit()*"- trening modelu dla zadanej ilości epok

Prezentacje wyników:

W celu otrzymania wyników do sieci podano następujące argumenty:

- liczba przypadków - 37 pacjentów gdzie 34 to zbiór treningowy, a 3 to zbiór testowy. Ze zbioru treningowego zostało wydzielone 30% zdjęć jako zbiór walidacyjny.
- liczba epok - 5
- patience (*EarlyStopping()*) - 2
- monitor ((*EarlyStopping()*) - "*val_loss*"
- optimizer -"adam"
- loss - "*binary_crossentropy*"
- funkcja aktywacji - "*relu*" (w warstwach), "*sigmoid*"(na wyjściu)
- padding - "*same*"
- kernel_initializer - "*he_normal*"



Rys 2. Wyniki predykcji

```
In [52]: eva = model.evaluate(test_CT,test_MASK)
103/103 [==============================] - 8s 81ms/sample - loss: 15.7230 - acc:
0.9684
```

Rys 3. Wyniki ewaluacji

Dyskusja i podsumowanie

W zrealizowanym projekcie nie udało się dokonać segmentacji nerki oraz guzów. Mimo, że sieć była uczona nawet przy zwiększeniu liczby przypadków do 100 (na więcej niestety nie pozwalał nam sprzęt) nie dokonywana była odpowiednia predykcja na podstawie wysyłanych zdjęć CT. Z racji na brak otrzymanych rezultatów nie jest możliwe dokonanie ewaluacji.

Prawdopodobnym jest, iż przyczyna leży w zastosowaniu wspólnej sieci zarówno dla nerki jak i guza. Implementacja drugiej sieci tak aby jedna mogła uczyć się tylko na nerkach, a druga tylko na guzach. Takie rozwiązanie mogłoby skutkować otrzymaniem odpowiednich wyników jednak są to tylko spekulacje gdyż tak naprawdę nie mamy pewności czemu zastosowana obecnie metoda nie przyniosła oczekiwanych rezultatów.

Pomimo niepowodzenia realizacja projektu pozwoliła na zapoznanie się z nowymi metodami segmentacji obrazów biomedycznych. Na laboratoriach korzystano głównie z metod algorytmicznych, dlatego podjęto próbę wykorzystania sieci neuronowych. Zmierzenie się z przedstawionym problemem pozwoliło na poznanie frameworku jakim jest '*tensorflow*'. Owa umiejętność może okazać się przydatna w przyszłej karierze.

Bibliografia

[1] https://kits19.grand-challenge.org/
[2] https://en.wikipedia.org/wiki/U-Net
[3] https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47
[4] Youtube - Python for Microscopists by Sreeni
[5] https://www.tensorflow.org/guide

Podział pracy:

Górak Kamil - zapis do PNG, preprocessing, model U-net, trening i ewaluacja sieci*

Kozioł Bartosz - research**,** zapis do PNG, preprocessing, model U-net, GitHub management*

Kwater Jakub - research, preprocessing, model U-net, raporty*

Mermon Gabriela - research, analiza danych, raporty*

*Przedstawiony podział stanowił luźne granice przyjętych obowiązków. Każdy z realizatorów projektu wykazywał równe zaangażowanie. Ostateczny kod wysyłany jest przez jedną osobę w celu zwiększenia przejrzystości przekazywanych informacji i nie namnażania zbędnych commitów. Dyktowane jest to również czynnikiem sprzętowym gdyż ostateczna wersja kodu musiała powstawać na najlepszym dostępnym nam sprzęcie.