PROJEKT Z PRZEDMIOTU

Techniki Obrazowania Medycznego

Inżynieria Biomedyczna 2019/2020

Jakub Kowalski
Michał Komala
Sophia Krupnik

Prototyp rozwiązania

1. Pobieranie i zapoznanie się z danymi

   Zestaw danych, które zostaną użyte do projektu stanowią 300 zbiorów obrazów, w których skład wchodzą przekroje ciała uzyskane przy użyciu tomografii komputerowej. Pierwsze 210 zbiorów zawiera oprócz obrazów CT także maski segmentujące nerki i nowotwory nerek. W masce wartość 0 jest przypisana dla tła, 1 dla nerki, a 2 dla nowotworu. Zestaw danych pobrano z oficjalnego repozytorium *2019 Kidney and Kidney Tumor Segmentation Challenge*. Korzystając z funkcji load_case, która została zawarta w repozytorium załadowano dane do notatnika Google Colab.

2. Przygotowanie danych

   Do poniższych obliczeń użyto następujących bibliotek: tensorflow, matplotlib, numpy. Kolejnym krokiem było pobranie wybranej liczby zestawów danych. Dzięki temu utworzono dwie listy: jedna zawierająca odpowiadające maski, a druga obrazy przekrojów ciała. Następnie wydzielono z listy z maskami jedynie obrazy, na których występowały posegmentowane nerki lub nowotwory. Jednocześnie stworzono dwa zestawy masek. Jeden zestaw zawierał jedynie maski zawierające segmentacje nerek, natomiast drugi zestaw - segmentacje nowotworów.

   Kolejnym etapem było dopasowanie danych do stworzonej sieci neuronowej. Stworzono macierze zawierające: maski nowotworów, maski nerek, obrazy CT na których występowały wcześniej wymienione maski. Dodatkowo dokonano normalizacji wybranych obrazów CT.

3. Przygotowanie architektury U-Net

   Obraz wejściowy o zadanych rozmiarach (512,512,1)

   Wykorzystano funkcje z biblioteki tensorflow.keras.layers, 4 ścieżki skurczu, w każdym z nich zwiększamy głębię obrazu o połowę: conv2D_1 -> dropout -> conv2D_2 -> max_pooling. Analizowany jest piksel po pikselu w celu jego klasyfikacji. Po osiągnięciu najniższego poziomu budowane są dwie warstwy splotowe w najmniejszej skali conv2D_4 -> dropout -> conv2D_5. Kolejny krok to 4 ścieżki, w każdej obraz jest zwiększany o połowę do pierwotnego rozmiaru poprzez jego powiększenie i łączenie z odpowiednikiem z ścieżki kurczenia: conv2D_5 -> concatenate -> dropout ->conv2D_6. Jako funkcję utraty użyliśmy polecaną do naszego przypadku “binary cross entropy”, a za optimizer obraliśmy powtarzającego się w wielu artykułach “Adam optimizer”.

4. Trenowanie sieci

   W tym celu wykorzystamy funkcję model.fit(). Planujemy użycie dwóch sieci: jedną do wykrywania nowotworów, a drugą do wykrywania nerek. Jako argumenty podajemy przygotowane wcześniej do treningu macierze. W przypadku sieci uczącej się o nerkach będą to dwie macierze : w pierwszej obrazy CT, w drugiej wyodrębnione maski prezentujące umiejscowienie nerek. Dla sieci uczącej się o nowotworach druga macierz zostanie zastąpiona wyodrębnionymi maskami nowotworów. Przyjęliśmy, że 10% podanych danych posłuży jako validation data. Oprócz tego dodaliśmy parametry pozwalająca zakończyć uczenie w przypadku overfittingu. Checkpoint oraz Earlystopping pozwalają nam przerwać naukę w przypadku, gdy wartość validation loss zacznie się powiększać.

5. Planowana ocena wyników

   Zamierzamy ocenić wyniki na podstawie dokładności modeli otrzymanej podczas testowania zbiorów. Oprócz tego zamierzamy również dokonać samodzielnej oceny. Część nieużytych do treningu obrazów ze znanymi maskami zostanie podana do funkcji model.predict(). Następnie otrzymane przy jej użyciu wyniki zostaną porównane ze znanymi maskami. Wydrukujemy je obok siebie, a także pokażemy występujące różnice na przykład poprzez odejmowanie od siebie otrzymanego i prawidłowego obrazu. Zastosowane to zostanie zarówno w przypadku sieci wykrywającej nerki jak i tej wykrywającej nowotwory.

6. Spostrzeżenia/efekty dotychczasowej pracy

   Podstawową wadą obranej przez nas metody jest bardzo duże zapotrzebowanie pamięci RAM. Żaden z komputerów, które posiadamy nie był w stanie wykonywać wymaganych obliczeń bez zacinania się. Stworzyliśmy arkusz Google Collab, na którym wykorzystujemy procesor TPU, jednak często również dla niego podawane przez nas macierze mają zbyt duże wymiary. W przypadku, gdy chcemy uczyć model często dochodzi do crushowania Collaba na skutek przekroczenia dostępnego RAMu. Próbujemy omijać to na różne sposoby, jednak uczone przez nas modele nie będą mogły przyjmować macierzy składających się ze zbyt dużej ilości obrazów CT.

   Planujemy w najbliższym czasie napisać metody, które zapisywałyby przekroje jako obrazy w formacie PNG. Liczymy, że zmniejszy to w dużym stopniu zużycie pamięci RAM i umożliwi bardziej efektywne uczenie. Oprócz tego wciąż zagłębiamy się w literaturze, chcąc opracować sposób uczenia jedynie jednej sieci. Uczenie dwóch z nich będzie dużo bardziej czasochłonne.