AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

KiTS19 CHALLENGE

Projekt realizowany w ramach przedmiotu:

Techniki Obrazowania Medycznego

Andrzej Jasek

Michał Crosta

Krzysztof Huras

Mikołaj Dobrowolski

Kraków, 23 czerwca 2020

Abstrakt

Celem projektu jest napisanie programu automatycznie segmentującego nerki i nowotwory w obrazach z tomografii komputerowej. Dane medyczne zostały udostępnione przez „2019 Kidney Tumor Segmentation Challenge” na platformie GitHub.

1 Wstęp teoretyczny

Program automatycznie segmentujący nerki i nowotwory w obrazach z tomografii komputerowej oparto na kilku spostrzeżeniach, do których można dojść analizując obrazy z tomografii komputerowej:

1. Zarówno nerka, jak i nowotwór charakteryzują się na zdjęciach z tomografii komputerowej jednolitą barwą (z wyjątkiem obszaru miedniczki nerkowej zobrazowanej na ciemno) oraz stanowią one spójne obiekty,
2. Nerka na zdjęciach jest jednym z najjaśniejszych obiektów,
3. Nowotwór jest częścią nerki.

Dzięki spostrzeżeniu 1: segmentację nerki, jak i guza można przeprowadzić przy pomocy metody rozrostu obszaru w wersji lokalnej. Dzięki spostrzeżeniu 2: lokalizację punktu startowego dla metody rozrostu obszaru w wersji lokalnej można łatwo wyznaczyć analizując jasność pikseli obrazu. Dzięki spostrzeżeniu 3: lokalizację guza da się wyznaczyć dzięki wyznaczonej w poprzednim punkcie lokalizacji nerki.

Jednakże spostrzeżenie 2 jest dość niepewne, bowiem na zdjęciach mogą istnieć obiekty o porównywalnej jasności do nerki. Niemniej jednak na tym spostrzeżeniu można zbudować efektywny algorytm segmentujący nerkę z największym prawdopodobieństwem. Aby jednak upewnić się, że wysegmentowany obiekt jest rzeczywiście nerką, należy przeprowadzić klasyfikację. Tą klasyfikację oprzemy na odpowiednich współczynnikach kształtu i objętości w jednostkach przestrzennych dla nerki. Badany obiekt może być zaklasyfikowany jako: lewa zdrowa nerka, prawa zdrowa nerka, lewa chora nerka, prawa chora nerka, lub jako obiekt niebędący nerką. W zależności od wyniku algorytmu klasyfikującego zostaną podjęte odpowiednie kroki, które opisano w następnej sekcji.

2 Metody

Program automatycznie segmentujący nerki będzie:

1. Wyznaczał ze zdjęcia piksele o pożądanej jasności.
2. Wyznaczał na zdjęciu piksele o najniższych wartościach mieszczących się w pożądanym przedziale jasności,
3. Przeprowadzał dla tych pikseli algorytm rozrostu obszaru w wersji lokalnej,
4. Klasyfikował obiekt jako: lewa zdrowa nerka, prawa zdrowa nerka, lewa chora nerka, prawa chora nerka, lub jako obiekt niebędący nerką.
    - Jeżeli obiekt został zaklasyfikowany jako zdrowa nerka, to wysegmentowywana jest druga nerka (o ile już to nie zostało uczynione),
    - Jeżeli obiekt został zaklasyfikowany jako chora nerka, to wysegmentowywany jest guz; przeprowadzone zostaną wertykalne sieczne sprawdzające piksele nerki na ich odcinkach. Jeżeli sieczna zawiera rząd pikseli, który: składa się z nerki, składa się z obszaru o nieciemnych pikselach (ciemne piksele wskazują obszar miedniczki nerkowej) i ponownie składa się z nerki, to te nieciemne piksele stanowią potencjalne miejsce guza. Dla tych pikseli ponownie przeprowadzona zostanie metoda rozrostu obszaru w wersji lokalnej celem segmentacji guza. Ten krok oparty jest na spostrzeżeniu, że guz jest częścią nerki. Następnie wysegmentowywana jest druga nerka (o ile już to nie zostało uczynione),
    - Jeżeli obiekt został zaklasyfikowany jako nie nerka, to obiekt zostaje zaciemniony na zdjęciu i krok segmentacji zostaje powtórzony do pesymistycznego momentu, gdy całe zdjęcie stanie się czarne.
5. Odpowiednie kroki zostaną zwizualizowane na zdjęciu.

2.1 Algorytm klasyfikujący zdjęcie

Istotną częścią analizy zdjęć nerek jest wykorzystanie kodu do rozpoznania czy wysegmentowane fragmenty zdjęć należą do założonych kategorii. Są nimi chora lewa nerka, chora prawa

nerka, zdrowa lewa nerka oraz zdrowa prawa nerka. Ponadto piąta kategoria jest przydzielana w przypadku, gdy na fragmencie nie ma nerki.

Aby dokonać takiej klasyfikacji wykorzystano bibliotekę tensorflow. Za pomoca funkcji **image_narrowing** oraz **segment_from_image** z danych medycznych otrzymano obrazy przedstawiające przekroje nerek. Jest to konieczny etap do umożliwienia trenowania sieci neuronowych, aby umożliwić rozpoznanie do jakiej kategorii należą nerki na zdjęciach testowych.

Utworzono funkcję **get_training_set** która przyjmuje na wejście ilość elementów w każdej z wybranych kategorii, oraz przypadki „cases" z danych repozytoryjnych. Dzięki temu zostają utworzone listy zdjęć nerek należących do wybranych kategorii. Dzięki segregowaniu zdjęć w taki sposób, możliwe jest utworzenie danych treningowych do metody sieci neuronowych.

W metodzie sieci neuronowych z wykorzystaniem Tensorflow są potrzebne dwie dane wejściowe: macierz zdjęć (danych treningowych) oraz macierz etykiet (label) odpowiadające kategoriom do których należą zdjęcia. Obie macierze muszą odpowiadać sobie kolejnością, wtedy program jest w stanie "nauczyć się" rozpoznawania elementów na zdjęciach.

Metoda wykorzystana w projekcie stosuje model sieci keras należący do biblioteki Tensorflow. Tworzone zostają warstwy umożliwiające uczenie maszynowe (konkretnie deep learning).

- Warstwa flatten tworzy z dwuwymiarowych macierzy danych(zdjęć) macierze jednowymiarowe. Jest to w uproszczenie spłaszczenie danych wejściowych,
- Warstwy typu Dense korzystają z 128 neuronów i zwracają macierz wynikową z 5 elementami (każdym odpowiadającym jednej kategorii)

Przed wykorzystaniem modelu wykorzystano opcję compile aby dostosować odpowiednie opcje programu:

- Loss - mierzy dokładność modelu podczas treningu,
- Optimizer -umożliwia aktualizowanie modelu na podstawie danych oraz funkcji Loss,
- Metrics-stosowane do monitorowania kroków podczas treningu oraz testowania. W projekcie użyto opcji Accuracy, czyli części obrazów które zostały poprawnie sklasyfikowane.

Trenowanie wykorzystywanego modelu sieci neuronowej przebiega następująco:

1. Dane treningowe(train_labels i train_images) zostają wprowadzone do modelu,

2. Model uczy się kojarzyć obrazy i etykiety,
3. Wprowadzane zostają dane testowe wcześniej przygotowane,
4. Model wykonuje rozpoznawania elementów wprowadzonych w postaci macierzy zdjęć testowych,
5. Sprawdzana zostaje poprawność rozpoznawania.

Model.fit umożliwia wprowadzenie danych treningowych do modelu i rozpoczęcie trenowania sieciami neuronowymi. Aby zwiększyc poprawność rozpoznawania została zastosowana wysoka ilość cykli epoch.

Zostały wytworzone dane testowe w których wykorzystuje się tą samą metodę obróbki zdjęć co w danych treningowych. Skorzystano z ilości danych testowych proporcjonalnie mniejszej do ilości danych treningowych. Następnie wykorzystano dodatkową warstwę Softmax pozwalającą na określenie prawdopodobieństwa poprawności odpowiedzi.

3 Realizacja projektu (kroki, trudności, rozwiązania)

Do zobrazowania plików dostarczonych przez zespół KIT2019 Challenge posłużono się kodem startowym, który również został udostępniony przez nich udostępniony. Wprowadzono kilka modyfikacji ograniczając zakres zapisywanych obrazów w celu oszczędzenia miejsca na dysku. Skupiono się na wyodrębnieniu plastrów zawierających nerki wraz z otoczeniem. Starano się usunąć projekcje nie wnoszące zmian (tj.: płuc oraz miednicy), a powodujące wydłużony czas działania programu i wydłużające cały proces rozpoznawczy. Działania testowe prowadzono na kilku dowolnych pacjentach, korzystając z zdjęć CT oraz wysegmentowanych nerek. Problematyczne okazało się odpowiednie wykorzystanie danych wolumetrycznych, tak aby wykorzystać ich parametry do „poruszania się” po obrazach CT. Pomimo udostępnionego kodu, natrafiono na szereg komplikacji w skład których wchodziły: wstępne trudności z zapoznaniem się z środowiskiem Colab oraz jego podstawową biblioteką służącą chociażby do kopiowania repozytorium, problemy z kompilacją oraz częste czyszczenie wirtualnego dysku Drive celem uwolnienia pamięci. Kod źródłowy działa w oparciu o kilka bibliotek m.in. Matplotlib, NiLearn oraz NiBabel, następca PyNifTI. Korzystając klas tych bibliotek uzyskano szerszy dostęp do informacji i danych obrazów CT. Obrazy wyświetlono w celu zapoznania się z tematyką. Wykorzystano możliwości klas i zmodyfikowano

kolorystykę prezentowanych obrazów tak, aby była on przyjazna lub aby odzwierciedlała faktyczne dane otrzymane z tomografu komputerowego. Wizualizacje oparto na obrazach pochodzących z płaszczyzny strzałkowej ze względu na lepszą widoczność guza jak i ograniczenie, jakim może być zła interpretacja prezentowanych danych.

Ewaluację algorytmu klasyfikującego przeprowadzimy na podstawie współczynnika Dice Coefficient. Jego wzór jest następujący:

$$\frac{2 \cdot |X \bigcap Y|}{|X| + |Y|}$$

Jego ocenę dokonuje się na podstawie masek (takiej, jaką powinniśmy otrzymać, oraz takiej, jaką otrzymaliśmy). Liczy się część wspólną masek oraz sumy poszczególnych masek. Dla wyliczenia dice coefficent napisano poniższą funkcję.

4 Wyniki, dyskusja

Algorytm nie zwraca wyniku nawet dla małej ilości danych. Podejrzewa się, że ilość interacji po kolejnych, jaśniejszych pikselach jest ogromna. Równocześnie sprawdzenie tego, czy wysegmentowany obiekt stanowi nerkę (z guzem, czy bez) algorytmem klasyfikującym zajmuje sporo czasu.

Tensorflow - Trudności podczas wykorzystywania modelu sieci neuronowych:

1. Mała ilość zdjęć treningowych. Optymalna ilość mieściła by się w granicach tysięcy, jednak brak pamięci ram nie pozwolił na wykorzystanie zbyt dużego zbioru danych,
2. Różnorodność rozdzielczości i rozmiarów zdjęć w danych treningowych. Zastosowana została funkcja resize biblioteki cv2, jednak w przypadku małych zdjęć (w okolicach 7 pikseli) wyniki skalowania tym sposobem nie są w pełni zadowalające.

5 Podsumowanie

- Rozmiar danych wejściowych jest duży. Jeden przypadek może zajmować 2,3GB. Dlatego projekt segmentacji wymaga wydajnych maszyn.
- Nerka jest widoczna w CT w 43% przekrojów.
- Dzięki dostarczonych danych jesteśmy w stanie automatycznie utworzyć zbiór treningowy.

- Algorytm rozrostu obszaru przy dużej ilości małych obiektów jest nieefektywny w wykrywaniu poszczególnych obiektów.
- Zwiększanie zbioru treningowego zwiększa dokładność algorytmu klasyfikującego.
- Zwiększanie ilości kroków uczenia modelu zwiększa precyzję algorytmu klasyfikującego.
- Algorytm rozrostu obszaru w wersji lokalnej sprawdza się wyłącznie w miejscach o małym gradiencie pikseli lub gdzie obraz jest binarny. Tu ze względu na dużą ilość obiektów nie sprawdził się. Dlatego należy szukać nowych rozwiązań.

6 Wkład osób w realizację

Nad projektem pracowaliśmy we czterech. Dużą część funkcji napisał Andrzej. Wczytał on wstępnie dane, opracował zbiory treningowe oraz wymyślił plan projektu. Nad resztą pracowaliśmy wspólnie (Michał, Krzysztof, Mikołaj). Opracowaliśmy algorytm rozrostu obszaru w wersji lokalnej, uczenie algorytmu sztucznej inteligencji, przeprowadziliśmy (z pomocą Andrzej) segmentację oraz przeprowadziliśmy ewaluację projektu (ewaluacja algorytmu klasyfikującego oraz dice coefficent). Kiedy zaszła potrzeba albo problem w którymś momencie realizacji projektu pomagaliśmy sobie wzajemnie co zaowocowało projektem, jaki został umieszczony na GitHubie.