Segmentacja komórek na zdjęciach mikroskopowych skóry z użyciem głębokich sieci neuronowych.

Michał Tracewicz

2020-12-25

Spis treści

1 Wstęp

W codziennej pracy wielu lekarzy poświęca czas na manualne liczenie zafarbowanych komórek na zdjęciach. Jest to czasochłonne i monotonne zadanie, nie wymagające sześcioletnich studiów. Zadanie takie jak to idealnie nadaje się do automatyzacji. W ostatnich latach bardzo mocno rozwija się użycie uczenia maszynowego do rozwiązywania tej klasy problemów. W ramach mojej pracy powstał zestaw narzędzi pozwalający na łatwe testowanie różnych architektur oraz funkcji strat dla sieci neuronowych w tym problemie. Dodatkowo przeprowadziłem testy dla architektury u-net oraz pewnych wariacji na jej temat. Całość dopełnia moduł pozwalający na obróbkę zdjęć wejściowych oraz wyjściowych.

2 Sztuczne sieci neuronowe

Sztuczne sieci neuronowe są próbą stworzenia programów, których działanie jest inspirowane naszym rozumieniem mózgu. Chociaż zagadnienie to podejmowano już od bardzo długiego czasu, to dopiero od niedawna posiadamy wystarczającą moc obliczeniową aby realizować tworzone od wielu dziesięcioleci algorytmy.

2.1 Ogólne zasady działania

Głębokie sieci neuronowe składają się z warstwy wejściowej, wyjściowej oraz pewnej liczby warstw ukrytych. Liczba oraz typ tych ostatnich w dużej mierze determinuje działanie sieci. Omówienie zaczniemy od tego w jaki sposób sieć przeprowadza obliczenia zwracające odpowiedź, a następnie zajmiemy się procesem uczenia. Każda z warstw zawiera pewną ilość neuronów. Neurony kolejnych warstw są ze sobą połączone (zajmę się tu jedynie warstwami, które posiadają połączenia neuronów każdy z każdym, czyli tak zwanymi warstwami typu "dense"). Połączenia te posiadają wagi. Aby obliczyć wynik zwracany przez sieć rozpoczynamy obliczenia od warstwy wejściowej i obliczmy wartości neuronów w kolejnych warstwach. Wartość dla neuronu w warstwie k-tej to wartość funkcji aktywacji na sumie wartości wszystkich neuronów z warstwy poprzedniej przemnożonych przez odpowiednie im wagi połączeń.

$$x_{k,j} = \sigma(\sum_{i=0}^{n-1} w_i x_{(k-1),i})$$

Gdzie (k>=1):

- σ - funkcja aktywacji
- $x_{k,j}$ - wartość j-tego neuronu w k-tej warstwie
- n - liczba neuronów w warstwie k-1
- w_i - i-ta waga wchodząca do neuronu

2.2 Porównanie do klasycznych metod programowania

Przy standardowych paradygmatach programowania (programowanie zorientowane obiektowo, programowanie proceduralne, programowanie funkcyjne) jako programista piszemy kod, który instruuje komputer na temat tego w jaki sposób ma on przekształcić otrzymane dane wejściowe. Natomiast sieci neuronowe są zdecydowanie bardziej zbliżone do programowania deklaratywnego. Jako programista podajemy dane wejściowe, architekturę i oczekiwane dane wyjściowe i na ich podstawie, chcemy aby komputer samemu ńauczył się"tego jaką drogą ma dojść aby uzyskać oczekiwany efekt (jakie wartości mają przyjmować wagi).

3 Sieci splotowe

Sieci splotowe to takie głębokie sieci neuronowe, które zawierają jedną lub więcej warstw splotowych.

3.1 Operacja splotu

3.2 Sieci w pełni splotowe

Sieciami w pełni splotowymi nazywamy te głębokie sieci neuronowe, które jako wszystkie warstwy posiadają warstwy splotowe. Sieci tego typu służą głównie do przekształcania obrazów.

4 Problem segmentacja

Zaznaczenie konturów komórek na zdjęciach mikroskopowych jest problemem segmentacji instancji. W celu zrozumienia tego pojęcia przeanalizujemy składające się na niego mniejsze problemy z kategorii widzenia komputerowego.

4.1 Klasyfikacja

Jest to najprostsze zagadnienie w dziedzinie widzenia komputerowego. Polega ono na tym, że dla otrzymanego na wejście obrazu sieć ma zwrócić klasę do której przedstawiony na zdjęciu obiekt należy. Za dobry przykład może nam posłużyć "Problem MNiST", polegający na rozpoznawaniu odręcznie napisanych cyfr. Jako wejście sieć otrzymuje czarno-biały obrazek rozmiaru 16x16 pikseli na, którym widnieje odręcznie napisana cyfra. Natomiast na wyjściu sieć zwraca nam predykcję jaka cyfra widnieje na zdjęciu. Ograniczeniem w tym typie problemu jest fakt, że na danym zdjęciu może znajdować się jedynie pojedyńczy obiekt.

4.2 Klasyfikacja z lokalizacją

Klasyfikacja z lokalizacją to dodanie do poprzedniego problemu zagadnienia odnalezienia klasyfikowanego obrazu na zdjęciu. Przykładowo tworząc sieć rozpoznającą gatunki zwierząt domowych, otrzymując zdjęcie kota siedzącego na kanapie otrzymamy nie tylko informację, że jest to kot ale również informację o tym gdzie na tej kanapie się znajduje. Najczęściej zwracane są w takim wypadku oprócz klasy współrzędne pikseli w których zaczyn i kończy się obiekt. Następnie możemy na tej podstawie narysowanie naokoło obiektu ramkę z podpisem zawierającym nazwę klasy.

4.3 Wykrywanie obrazów

Kolejnym krokiem jest wykrywanie obrazów. Jest to klasyfikacja z lokalizacją dla wielu elementów jednocześnie. Kontynuując poprzedni przykład z rozpoznawaniem zwierząt domowych. W wypadku zdjęcia, które zawiera zarówno psa i kota dostaniemy zestaw współrzędnych oraz odpowiadających im klas.

4.4 Segmentacja

Segmentacja to przypisanie każdemu pikselowi na obrazie wejściowym klasy. Za przykład może nam posłużyć zdjęcie drogi w mieście. Celem naszej sieci neuronowej będzie podział obrazu na klasy: droga, człowiek, budynek, drzewo, niebo etc. Na wyjściu możemy na przykład otrzymać ten sam obrazek z tym, że zależnie od klasy danego piksela będzie miał on różny kolor (czerwony - droga, zielony - człowiek etc.)

4.5 Segmentacja instancji

Segmentacja instancji to dodanie do segmentacji ograniczenia, że każda jednostka ma być widocznie oddzielona. Dla przykładu gdy mamy zdjęcie czterech przytulających się osób, w wyniku segmentacji wszystkie cztery zostaną zakolorowane na ten sam kolor oraz podpisane raz jako człowiek. Natomiast gdy przeprowadzimy segmentację instancji to w wyniku otrzymamy każdą osobę pokolorowaną na własny kolor, oraz etykiety ćzłowiek 1.", ćzłowiek 2.ętc.

5 Architektura U-Net

5.1 Historia i zastosowania

5.2 Obrazy wejściowe i wyjściowe

W opracowywanym przeze mnie zagadnieniu jako obrazy wejściowe użyte zostały zdjęcia mikroskopowe skóry. Wszystkie zdjęcia są zrobione z przybliżeniem x40 oraz posiadają rozmiar 1200x1600px z trzema kanałami. Obrazy wyjściowe zostały przygotowane za pomocą programu dostarczonego mi, przez dr. Wiśniewskiego. Są to obrazy tego samego rozmiaru jednak zawierają cztery kanały zamiast trzech (RGBA zamiast RGB). Możemy na nich zobaczyć czerwoną otoczkę na około komórek, natomiast cała reszta obrazu przyjmuje kolor czarny.

6 Przetestowane podejścia

6.1 Bazowa sieć u-net

6.2 Preprocessing obrazów

6.2.1 Cięcie i sklejanie obrazów

6.2.2 Wybór jednego kanału - czerwony/alfa

6.2.3 Rozmycie - gausian blur

Kolejny pomysłem było zastosowanie na obrazach docelowych przekształcenia gausian blur przed pokazaniem ich sieci. Ideą za tym stojącą było zwiększenie ilości czerwonych pikseli na obrazku w celu zredukowania wartości dokładności dla czysto czarnych obrazów zwracanych przez sieć.

6.2.4 Redukcja do obrazów trzy kanałowych i redukcja kolorów

W celu ułatwienia uczenia przeprowadziłem redukcję wymiarowości danych wyjściowych oraz zmniejszyłem ilość kolorów w obrazie. Zależnie od wartości kanałów czerwonego i alpha zaokrągliłem kolor piksela do czystego czerwonego (255, 0, 0) lub czystego czarnego (0, 0, 0).

6.3 Funkcja strat - Współczynnik Sørensena

6.4 Własna funkcja strat

6.5 Zapis wyniku w postaci distance map

7 Powstałe narzędzia

W trakcie prac powstał szereg narzędzi znacznie ułatwiających pracę nad tym zagadnieniem. Kod wraz z historią jest dostępny na platformie GitHub. Wyniki są prezentowane za pomocą platformy GitHub Pages i są dostępne pod adresem:.

7.1 Skrypty pozwalające na preprocessing obrazów uczących

W ramach prac stworzyłem cztery skrypty pozwalające na edycję zdjęć wejściowych/wyjściowych. Powstałe skrypty pozwalają na:

- zastosowanie filtra Gausian Blur z wybranym parametrem na pojedyńczym obrazie lub katalogu obrazów
- pocięcie obrazów na mniejsze nachodzące się na siebie obrazki
- zespolenie obrazków utworzonych poprzednim skryptem z powrotem w całość
- zredukowanie kanałów obrazów z RGBA do RGB i zaokrąglenie wartości pikseli do czerwonego (255, 0, 0) i czarnego (0, 0, 0)

7.2 Skrypt do uczenie sieci, poprzez wstrzyknięcie architektury sieci/funkcji strat

7.3 Skrypt pozwalające na testowanie zapisanych modeli

Kolejny skrypt przyjmuje na wejście ścieżkę do zapisanego modelu oraz ścieżki do katalogów zawierających dane wejściowe i wyjściowe. A następnie przeprowadza na nich test sprawdzający jak dobrze model sobie na nich poradził.

7.4 Skrypt pozwalające na predykcję dla obrazu/folderu obrazów

Stworzyłem również skrypt, który otrzymując na wejście ścieżkę do modelu, którego chcemy użyć oraz ścieżkę do obrazu wejściowego/katalogu obrazów wejściowych zapiszę do pliku/plików nowy obraz będący predykcją modelu.

7.5 Testy jednostkowe

W celu wykazania poprawności działania oraz umożliwienia szybszej modyfikacji bez wprowadzania błędów do modułu pozwalającego na preprocessing zdjęć zostały dodane testy jednostkowe. Zostały napisane przy pomocy modułu PyTest a ich uruchamianie jest obsługiwane, przez moduł Tox. Testy są automatycznie uruchamiane przy wypchnięciu lokalnych commitów, do publicznego repozytorium na platformie GitHub. Funkcjonalność ta została zrealizowana za pomocą

platformy CircleCi, pozwala ona na uruchomienie testów po otrzymaniu informacji od portalu GitHub. Status tych testów możemy zobaczyć w łatwy sposób poprzez dodany do pliku README.md obraz. Jest on każdorazowo przy wyświetlaniu wyrenderowanego pliku README pobierany z serwera CircleCi i zależnie od tego czy wszystkie testy jednostkowe z ostatniego commita zostały poprawnie ukończone czy nie przybiera odpowiednio kolor zielony/czerwony.

7.6 Automatyczne renderowanie i publikacja wyników

Dodatkowo powstał notatnik Jupyter[1] zawierający prezentację uzyskanych wyników. W celu interaktywnego przejrzenia go należy pobrać repozytorium. Jednakże jest on również dostępny jako strona internetowa w postaci z pokazanymi wyjściami z wszystkich komórek. Strona ta jest uaktualniana za każdym razem gdy do repozytorium na serwerze GitHub zostanie wysłany nowy commit zawierający modyfikację notatnika. Rozwiązanie to tak samo jak automatyczne testy jednostkowe zostało zrealizowane przy użyciu platformy CircleCi. Otrzymuje ona informację od serwisu GitHub o nowym commicie następnie sprawdza, czy plik notatnika został zmieniony. Jeżeli tak to wywoła polecenie eksportujące go do pliku w formacie HTML a następnie sama stworzy commit i wyśle go do repozytorium na serwisie GitHub. Gdy ten go otrzyma opublikuje nowo utworzony plik na platformie GitHub Pages. Dzięki tej integracji wszystkie zmiany których dokonam w notatniku prezentującym wyniki moich prac są automatycznie publikowane i dostępne do wglądu.

7.7 Kontener developerski

Stworzyłem również skrypty, które pozwalają na uruchomienie kontenera, przy użyciu technologii Docker zawierającego wszystkie wymagane biblioteki. Rozwiązanie to w szczególności w znaczący sposób upraszcza dostęp do bibliotek firmy NVIDIA wymaganych do uruchomienia biblioteki TensorFlow na karcie graficznej tegoż producenta. Dzięki temu osoba zainteresowana uruchomieniem aplikacji z wykorzystaniem karty graficznej w celu przyśpieszenia obliczeń musi jedynie posiadać na swoim urządzeniu aktualne sterowniki oraz Docker.

[1] interaktywne środowisko uruchomieniowe. Pozwalające na łączenie kodu oraz języka znaczników markdown. Dostępne pod adresem: https://jupyter.org/

8 Podsumowanie i możliwe następne kroki

Testy, które przeprowadziłem wskazują na to, że dla tych danych wejściowych/wyjściowych zastosowanie architektury U-Net nie sprawdza się. Jako, że jest to najbardziej rozpowszechniona architektura do segmentacji obrazów medycznych może to również sugerować, że użycie sieci neuronowych do tego zadania na ten moment nie przyniesie żądanych wyników. Jednakże dzięki powstałym w trakcie testów narzędziom pojawiła się możliwość łatwego testowania nowych podejść, które jak wskazują na to ostatnie lata pojawiają się w dziedzinie uczenia maszynowego i głębokich sieci neuronowych bardzo często.

9 Bibliografia