Projekt 3

Damian Jurga 122481
Grzegorz Miebs 122453

1 lipca 2018

1 Preprocessing

Wejściowy obraz jest skalowany do rozmiarów $750x750$ a następnie konwertowany do przestrzeni barw HSV. Później wycinany jest tylko pierwszy kanał (H), natomiast pozostałe są odrzucane. Dodatkowo wykonano korekcję gamma ze współczynnikiem 0,9 i sprogowano obrazki zerując piksele o wartościach z poza przedziału $< 50, 90 >$, który wyznaczono eksperymentalnie. Na koniec wykonano otwarcie oraz zamknięcie obrazka jądrem kwadratowym o wymiarach 3x3. Powstałe obrazki identyfikowały większość nawierzchni drogi lepiej niż odpowiadające im etykiety, ale były na nich zaznaczone także na przykład podjazdy domów. Zauważono, że pozwala to dość dobrze, bo na poziomie ok. 75%, przybliżyć docelowe rozwiązanie, a więc stanowi dobry punkt startowy do nauki sieci.

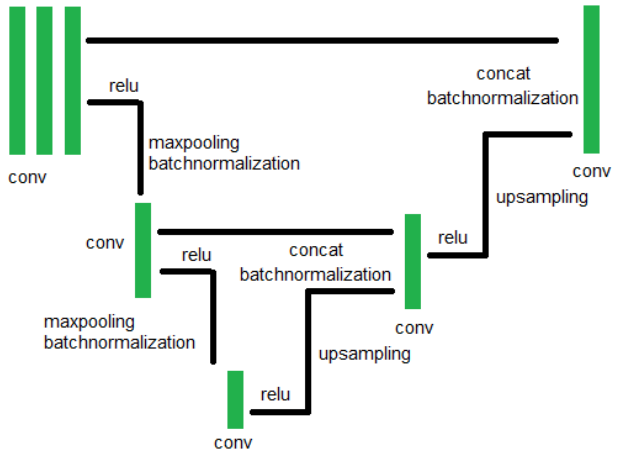
W celu uniknięcia problemów z pamięcią obraz zostaje pocięty na fragmenty o rozmiarach $85x85$, przy czym z każdej strony występuje margines szeroki na 5 pikseli, który nakłada się z kolejnym fragmentem. Na krawędziach oryginału wykonywane jest lustrzane odbicie. Dzięki temu zminimalizowane zostaje ryzyko rozdzielenia drogi na kilka osobnych fragmentów, co znacznie utrudniłoby jej rozpoznanie. Jednocześnie ten sam mechanizm mógłby zostać wykorzystany do generacji większego zbioru obrazków uczących.

Każdy fragment jest normalizowany do przedziału $< 0, 1 >$ poprzez odjęcie minimum i podzielenie przez różnicę maksymalnej i minimalnej wartości piksela, jeżeli ta jest różna od zera.

2 Architektura sieci

Wykorzystano architekturę U przedstawioną na rys.1. Jako funkcji aktywacji ostatniego splotu w grupie użyto LeakyReLU (min(1;max(x;0,001x))), rozmiar filtrów w warstwach splotowych to $5x5$. Sploty na najwyższym poziomie miały po 32 filtry (za wyjątkiem ostatniego, który musiał zwrócić pojedynczą wartość dla każdego piksela), na środkowym 64, a na najniższym 128. Schodząc na niższy poziom zawsze znajdywano maksimum z czterech sąsiednich pikseli (*Maxpooling*) i normalizowano pakiet (*Batch Normalization*), a wchodząc na

wyższy konkatenowano wynik nadmiarowego próbkowania z wynikiem ostatniej warstwy splotu na tym samym poziomie.



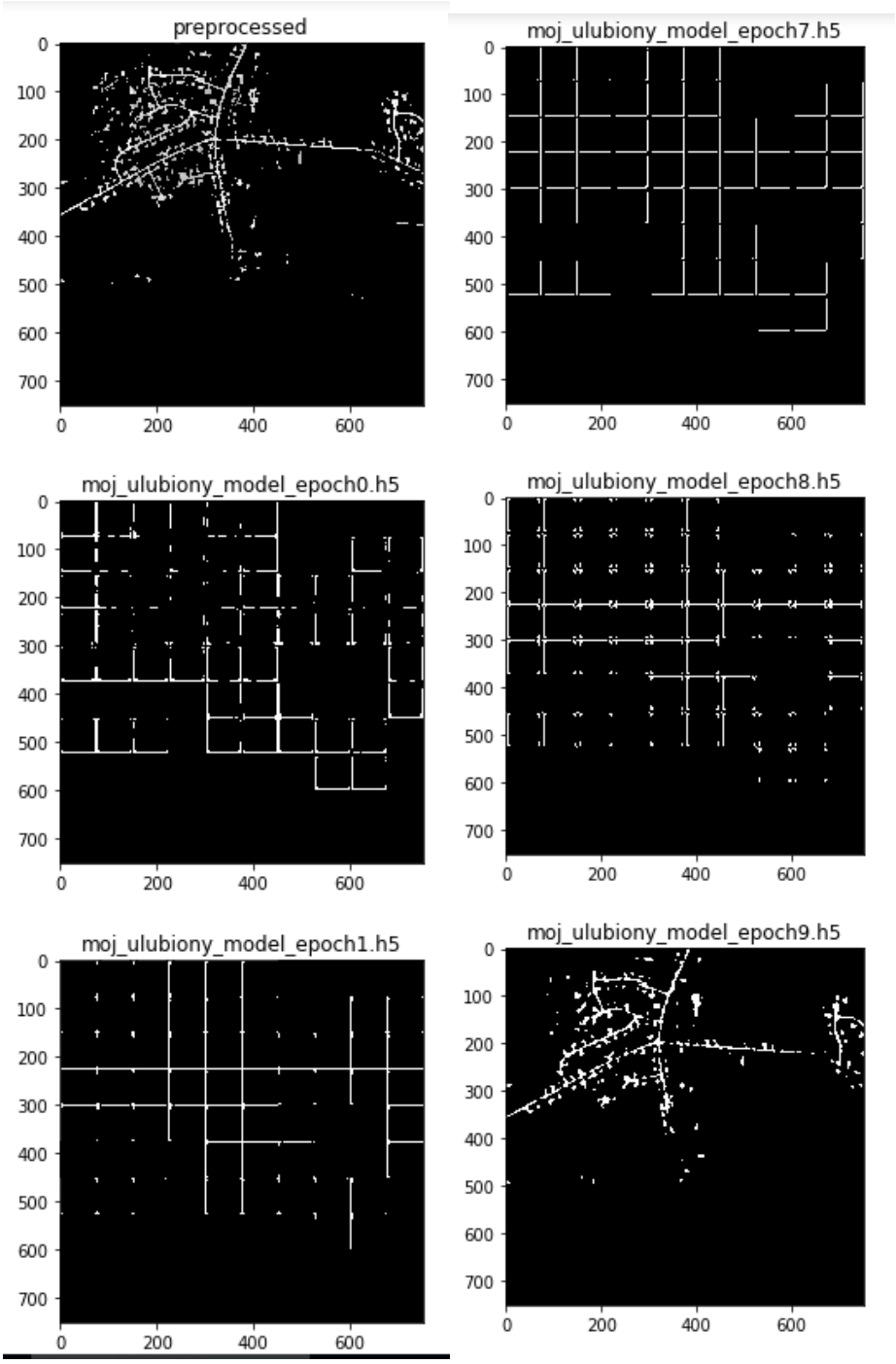
Rysunek 1: Architektura sieci

3 Sposób uczenia

Uczono na obrazkach, na których po wstępnym przetworzeniu choć jeden piksel był różny od średniej wartości piksela na tym obrazku, innymi słowy tych, na których prawdopodobnie rzeczywiście była droga. Było to konieczne ponieważ zaobserwowano, że na wielu etykietach droga jest błędnie zaznaczona.

Wiedząc, że drogi stanowią znacząco mniejszy ułamek obrazków niż tło (ok. 5%), jako funkcję straty zastosowano ważony błąd średnio-kwadratowy poprzez przemnożenie elementów macierzy błędów kwadratowych przez odpowiadające im elementy macierzy $w_1 Y_T + w_0(1 - Y_T)$, gdzie w_0 jest wagą piksela z tłem, w_1 wagą piksela z drogą, a Y_T macierzą etykiet.

Zastosowano optymalizator Adam.


preprocessed
moj_ulubiony_model_epoch7.h5
moj_ulubiony_model_epoch0.h5
moj_ulubiony_model_epoch8.h5
moj_ulubiony_model_epoch1.h5
moj_ulubiony_model_epoch9.h5

4 Postprocessing

Wyjście sieci jest zniekształcone ze względu na podjazdy, które także są klasyfikowane jako drogi, oraz różnego rodzaju zniekształcenia pojawiające się na drogach (dziury, pasy itd.). Aby zmniejszyć to zjawisko wyjściowy obraz rozmywany jest rozmyciem gaussowskim, następnie dokonywana jest binaryzacja obrazu według progu 0.5 a później kolejno erozja, otwarcie i domknięcie.

5 Wyniki

Uzyskano trafność binarną na poziomie 91,87% na zbiorze walidacyjnym utworzonym z 30% przykładów po upływie 10 epok.

6 Inne podejścia

Do rozwiązania tego problemu zastosowano wiele zróżnicowanych podejść. Niestety większość z nich skutkowała tym, że sieć zwracała niemalże identyczne wartości dla każdego z pikseli interpretując wszystko jako tło. Problem ten starano się rozwiązać np poprzez zmianę funkcji straty na binarną entropię krzyżową, a później nawet na ważoną binarną entropię krzyżową, jednak nie przyniosło to spodziewanego rozwiązania problemu a wyniki nadal pozostawał taki sam.

Manipulowano też funkcjami aktywacji, ponieważ podejrzewano, że błędne wyniki są spowodowane tym, że funkcje ReLU weszły na swoją "płaską" część, przez co nie uczą się dalej i zwracają te same wyniki. W związku z tym jako funkcję aktywacji ostatniej warstwy zastosowano funkcję liniową, a w pozostałych warstwach ELU. Jedyna zmiana jaką zauważono po zastosowaniu tych funkcji to dłuższy czas potrzebny sieci do osiągnięcia stanu w którym zwraca stałe wyjście, jednak nie poprawiło to w żaden sposób trafności.

Kolejnym podejściem była zmiana architektury sieci na kilka $(2-6)$ następujących po sobie warstw konwolucyjnych wraz z BatchNormalization, po których następowały odpowiadające warstwy dekonwolucyjne. Sieć ta była znacznie prostsza od pierwotnego modelu i posiadała mniej parametrów, jednak nadal występował z nią ten sam problem związany z niezbalansowanymi danymi.