Politechnika Warszawska

Inteligentne Obliczenia

Projekt: Dąb czy Sosna?

Wykonali:
P.W.
Bartłomiej Guś,nr albumu 297415

Warszawa 2021/2022

Spis treści

1. Wstęp

Celem projektu było stworzenie klasyfikatora, który rozpoznawałby gatunki drzew sosny i dębu na podstawie ich kory. Aby rozwiązać powyższe zadanie użyte zostały konwolucyjne sieci neuronowe, które zostały uczone na podstawie dostarczonych zdjęć tych drzew. Przetworzone zdjęcia zostały odpowiednio podzielone na zbiór treningowy/walidujący/testowy w taki sposób, aby nie wystąpiło zjawisko Data Leakage, które mogło spowodować uzyskanie fałszywie dobrych wyników. Również podczas projektu mieliśmy możliwość zapoznać się z pojęciem rozszerzenia danych – data augmentation, jak i porównać wyniki klasyfikacji dla przetworzonych zdjęć jak i przy zastosowaniu zdjęć oryginalnych.

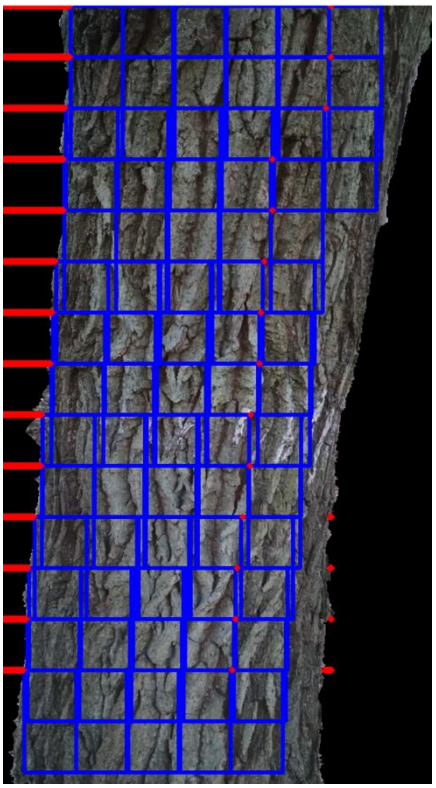
2. Przygotowanie zdjęć

W celu ułatwienia sieci neuronowej rozpoznawania tych dwóch gatunków drzew, na samym początku przygotowaliśmy program, który ze zdjęć na których na pierwszym planie jest przykładowe zdjęcie jednego z dwóch gatunków drzew a wokół niego tło (inne drzewa, liście itp.), wycinało drzewo pierwszoplanowe i następnie wycinało z już przetworzonego zdjęcia fragmenty kory, które następnie posłużą nam do nauki sieci neuronowej.

Rysunek 1 - przykładowe otrzymane zdjęcie dębu

W celu otrzymania zdjęcia, na którym będzie wycięte jedynie to drzewo pierwszoplanowe i jednocześnie automatyzacji tego procesu wykorzystywaliśmy funkcję grabCut dostępną w bibliotece OpenCV, za pomocą którego można otrzymać pierwszy plan przy podaniu na zdjęciu pewnej lokalizacji pierwszego planu oraz tła i ewentualnie miejsc, których nie jesteśmy pewni, czyli przypisać: prawdopodobnie pierwszy plan, prawdopodobnie tło. W naszym przypadku na samym początku określaliśmy, że całe zdjęcie jest prawdopodobnie tłem. Następnie na bocznych fragmentach zdjęcia określaliśmy, że w tym miejscu na pewno znajduje się tło i na środku na pewno znajduje się obiekt. Za pomocą tego schematu przetwarzania mogliśmy zautomatyzować wycinanie drzew ze zdjęcia i następnie wycinaliśmy z nich fragmenty o odpowiedniej wielkości. W naszym przypadku były to wielkości 128x128 px, które następnie wykorzystaliśmy do nauki konwolucyjnej sieci neuronowej. Powyższe zadanie wykonuje dołączony program: getPierwszeTloAutomated.

Rysunek 2 – wycięte drzewo pierwszoplanowe



Rysunek 3 - zaznaczone fragmenty wycięte z drzewa pierwszoplanowego

Na Rysunku 3 możemy zauważyć zaznaczone wycięte fragmenty drzewa - są to obszary niebieskie. Aby zwiększyć liczbę pozyskanych zdjęć z jednego drzewa, kolejny domniemany fragment był analizowany po przesunięciu okna wycinania o pół szerokości lub wysokości tego okna. Czerwone kropki oznaczają miejsce, w których nie został wycięty fragment, ponieważ zawiera on piksel (0,0,0) - w taki sposób zaznaczaliśmy tło na zdjęciu z już tylko wyciętym drzewem pierwszoplanowym. Wprowadzenie do sieci zdjęć zawierających fragmenty czarnego tła mogłoby zaburzyć proces nauki sieci neuronowej.

Rysunek 4 - przykładowo otrzymane zdjęcia gotowe do nauki

Okazało się, że nasze podejście pozwoliło nam na uzyskanie wyciętych fragmentów z ponad 80% dostarczonych nam zdjęć. Głównym powodem brakiem otrzymania fragmentów z pozostałych 20% zdjęć jest fakt, że drzewa te były chudsze niż 128 pikseli albo bardzo nieforemne.

Finalnie, otrzymaliśmy 2002 zdjęć dębu oraz 2960 zdjęć sosny gotowych do nauki. Dysproporcja pomiędzy nimi wynika z większej ilości dostarczonych zdjęć sosny niż dębu.

3. Rozszerzenie danych

W celu możliwości osiągnięcia zadowalających wyników działania naszej sieci neuronowej musimy posiadać zbiór danych o możliwie jak największej liczności. Niestety w niektórych przypadkach/dziedzinach, w szczególności w medycynie, ale nie tylko, nie jesteśmy w stanie pozyskać większej ilości danych. W takiej okoliczności możemy skorzystać z rozszerzenia danych – data augmentation, czyli za pomocą jednego zdjęcia możemy uzyskać wiele innych zdjęć dzięki zastosowaniu różnego rodzaju operacji takich jak: rotacja, przesunięcie, przeskalowanie, odbicie. Dzięki takiemu zabiegowi możemy powiększyć nasz wejściowy zbiór danych i otrzymać bardziej satysfakcjonujące wyniki.

Rysunek 5 - przykładowe zdjęcie wejściowe

Rysunek 6 - przykładowe zdjęcia otrzymane z Rysunku 5 dzięki data augmentation

4. Konwolucyjne sieci neuronowe

W celu rozpoznawania gatunku drzewa na podstawie dostarczonych zdjęć wybraliśmy konwolucyjne sieci neuronowe o architekturze Lenet. Założyliśmy, że podział na zbiór trenujący – walidujący – testowy wykonamy w proporcji 70 – 15 – 15 %. Rzeczywista proporcja podziału na te zbiory była lekko inna i wynikała z tego, że jeśli wczytywaliśmy wycinki do zbioru trenującego lub walidującego z danego zdjęcia, w którym jeszcze pozostały niewczytane wycinki to w celu uniknięcia zjawiska DataLeakage wczytywaliśmy pozostałe wycinki do danego zbioru w którym znajdowały się już wycinki tego zdjęcia. Jako optimizer wybrano: Adamax, a jako funkcja strat: binary crossentropy. Było to spowodowane tym, że chcemy rozpoznawać dwa gatunki drzew. Kod tej części został umieszczony w pliku IOB.

Rzeczywisty podział dla dębu wyniósł:

```
Liczba zdjec w zbiorze uczacym: 1402
Liczba zdjec w zbiorze walidujacym: 302
Liczba zdjec w zbiorze testowym: 298
W sumie zdjec: 2002
Jaki procent w zbiorze uczacym: 70.02997002997003
Jaki procent w zbiorze walidujacym: 15.084915084915085
Jaki procent w zbiorze testowym: 14.885114885114886
```

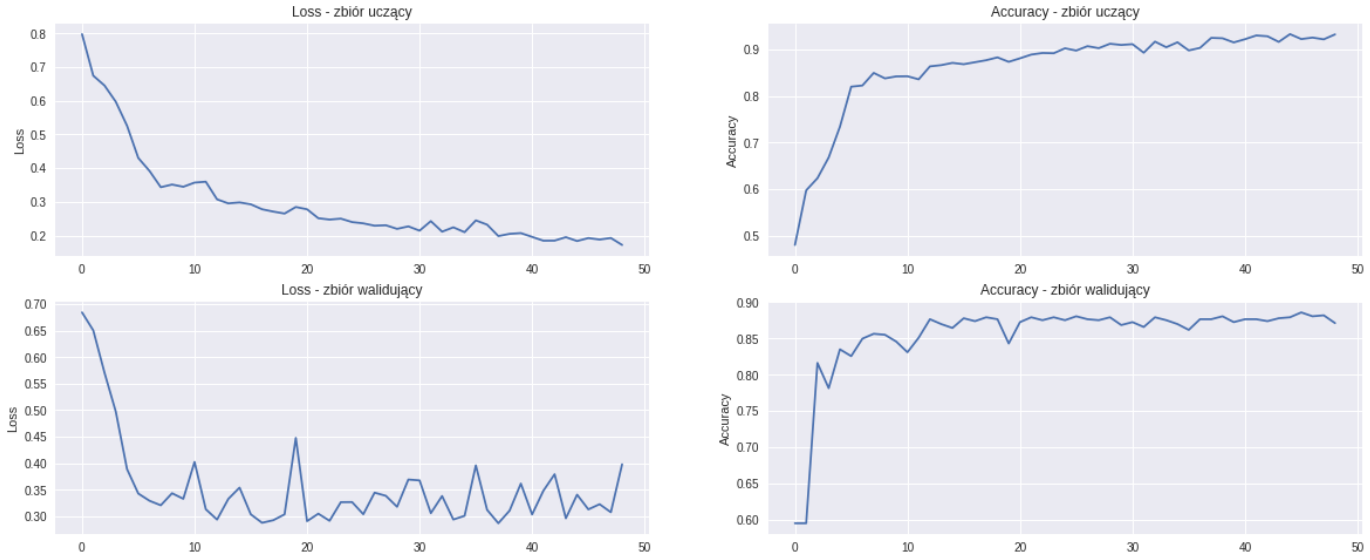
Natomiast dla sosny:

```
Liczba zdjec w zbiorze uczacym: 2082
Liczba zdjec w zbiorze walidujacym: 444
Liczba zdjec w zbiorze testowym: 434
W sumie zdjec: 2960
Jaki procent w zbiorze uczacym: 70.33783783783784
Jaki procent w zbiorze walidujacym: 15.0
Jaki procent w zbiorze testowym: 14.662162162162161
```

A architektura stworzonej sieci prezentuje się następująco:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_66 (Conv2D)          (None, 128, 128, 6)       456

 average_pooling2d_44 (Avera  (None, 64, 64, 6)        0
 gePooling2D)

 conv2d_67 (Conv2D)          (None, 60, 60, 16)        2416

 average_pooling2d_45 (Avera  (None, 30, 30, 16)       0
 gePooling2D)

 conv2d_68 (Conv2D)          (None, 26, 26, 120)       48120

 flatten_22 (Flatten)        (None, 81120)             0

 dense_44 (Dense)            (None, 84)                6814164

 dense_45 (Dense)            (None, 2)                 170

=================================================================
Total params: 6,865,326
Trainable params: 6,865,326
Non-trainable params: 0
_________________________________________________________________
```

Poniżej zamieszczono przebieg nauki:



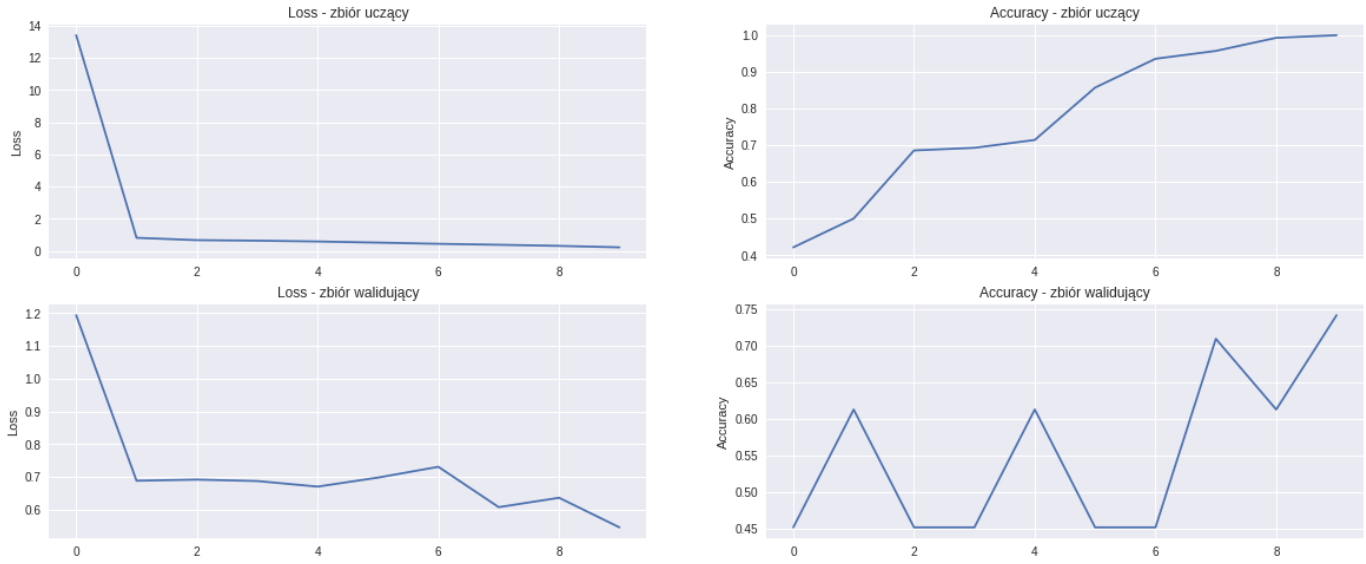
Jak możemy zauważyć, wraz z nauką spadała wartość funkcji strat a zwiększała się dokładność, co jest pożądanym efektem. Dla zbioru uczącego w ostatniej epoce uzyskano ponad 93% dokładności a dla zbioru walidującego ponad 88% dokładności. W przypadku zbioru testowego uzyskano ponad 74% dokładności oraz wartość funkcji strat to około 0.8392. Powyższe informacje utwierdzają nas w tym, że uzyskaliśmy bardzo dobre wyniki.

```
23/23 [==============================] - 0s 8ms/step - loss: 0.8392 - accuracy: 0.7459
[0.8392277359962463, 0.7459016442298889]
```

5. Czy warto wykonać przygotowanie danych?

W celu oceny sensowności włożonego trudu w wykonanie przygotowania danych, powtórzyliśmy naukę sieci neuronowych dla tej samej architektury i przy założeniu tego samego podziału na zbiór trenujący – walidujący – testowy. W tym przypadku natomiast jako dane wejściowe użyliśmy oryginalnych zdjęć. Jedno z tych zdjęć możemy zauważyć na Rysunku 1. Dodatkowo zmieniliśmy liczbę neuronów z 84 do 50 w pierwszej warstwie Dense ze względu na to, że liczba parametrów sieci była już zbyt duża, aby przeprowadzić obliczenia na komputerze osobistym, co było spowodowane większym rozmiarem zdjęcia wejściowego. Sieć z poprzedniego rozdziału posiadała jedynie ok. 7 milionów parametrów. Natomiast w tym przypadku liczba ta wzrosła do ok. 280 milinów parametrów. Kod z powyższego fragmentu zwiera się w pliku IOB2.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 1024, 768, 6)      456

 average_pooling2d (AverageP  (None, 512, 384, 6)      0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 508, 380, 16)      2416

 average_pooling2d_1 (Averag  (None, 254, 190, 16)     0
 ePooling2D)

 conv2d_2 (Conv2D)           (None, 250, 186, 120)     48120

 flatten (Flatten)           (None, 5580000)           0

 dense (Dense)               (None, 50)                279000050

 dense_1 (Dense)             (None, 2)                 102

=================================================================
Total params: 279,051,144
Trainable params: 279,051,144
Non-trainable params: 0
_________________________________________________________________
```



Ze względu na małą liczbę danych wejściowych sieć bardzo dobrze nauczyła się zbioru uczącego, ponieważ dokładność uzyskana to aż 100%. Natomiast w przypadku zbioru testowego dokładność ta wyniosła zaledwie 50%, co jest 25% gorszym wynikiem niż w przypadku, gdy wykonaliśmy preprocessing. Wniosek z tego jest zatem taki, że warto przygotować dane przed rozpoczęciem nauki sieci.

```
1/1 [==============================] - 3s 3s/step - loss: 0.8264 - accuracy: 0.5000
[0.8264327049255371, 0.5]
```