ROZPOZNAWANIE KOTKÓW OD PIESKÓW

Michał Skibiński (260352)

30.05.2022

1 Wstęp

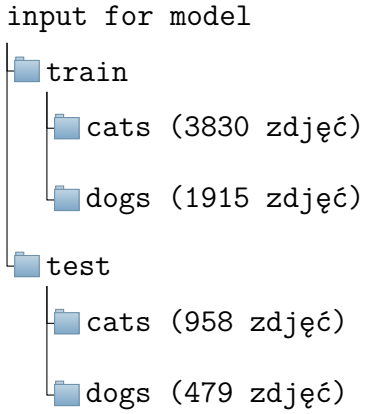
Przedmiotem projektu jest zadanie klasyfikacji binarnej (klasyfikacji kotów oraz psów). do rozwiązania problemu posłużyłem się **konwolucyjną siecią neuronową (CNN)**. Ponieważ sieci neuronowe operują na macierzach, są one szczególnie wydajne przy pracy z obrazami, które są macierzą pikseli.
Rozwiązanie zaprojektowałem w języku *python* używając biblioteki *tensorflow*.
Zdecydowałem się na język programowania *python*, ze względu na jego prostotę, niezawodność oraz wielorakość bibliotek ogólno-dostępnych.
Bibliotekę *tensorflow* wybrałem ze względu na możliwość wykonywania złożonych operacji na sieciach neuronowych, przy użyciu stosunkowo niewielkiej ilości kodu. Oprócz tego biblioteka ta działa bardzo efektywnie, oraz została do niej napisana bogata dokumentacja.
Dane dla mojego modelu pobrałem ze strony: https://www.kaggle.com/datasets/zippyz/cats-and-dogs-breeds-classification-oxford-dataset

2 Model

2.1 przygotowanie danych

aby przygotować dane kolejno:

1. napisałem algorytm do odczytywania zdjęć z zadanego folderu i tworzeniu folderu plików z podziałem na **dane testowe i treningowe** oraz klasy, tak by strukrura wyglądała następująco:

   ilość danych testowych ustawiłem na 20 procent.

2. usunąłem różnicę w ilości fotografii kotów i psów (ważne aby sieć neuronowa miała tyle samo danych z każdej z klas)

3. dokonałem **normalizacji danych** - przeskalowałem wartości pikseli tak by pochodziły z zakresu [0,1)

4. ustandaryzowałem dane - wszystkie zdjęcia przeskalowałem do rozmiarów [224 x 224] px.

5. zrezygnowałem z użycia **PCA** ponieważ redukcja wymiarów wiązała się ze zbyt dużą utratą danych.

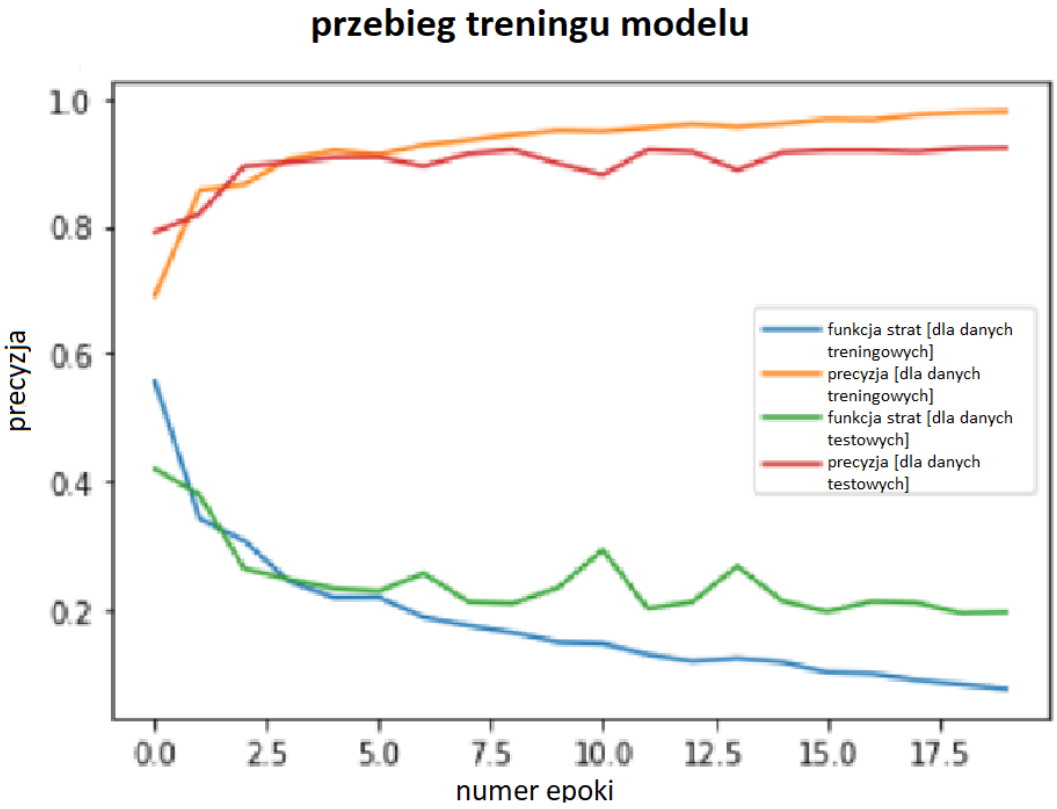
2.2 budowa modelu

własności modelu:

- wejście dla modelu: macierz o rozmiarach (224 x 224 x 3) - gdzie 3 wynika z reprezentacji pikseli poprzez RGB
- liczba warstw ukrytych: 20
- wyjście: wartość 0 lub 1 (0 - dla psa, 1 - dla kota)
- metoda optymalizacji: **stochastyczny spadek gradientu(SGD)** z prędkością nauki = 0.0001

- funkcja strat: binarna entropia krzyżowa
- ilość generacji(epok): 20
- populacja w każdej generacji: 60
- **funkcja regularyzacji**: l2 (bez regularyzacji występowało **przetrenowanie modelu**)

2.3 przebieg treningu

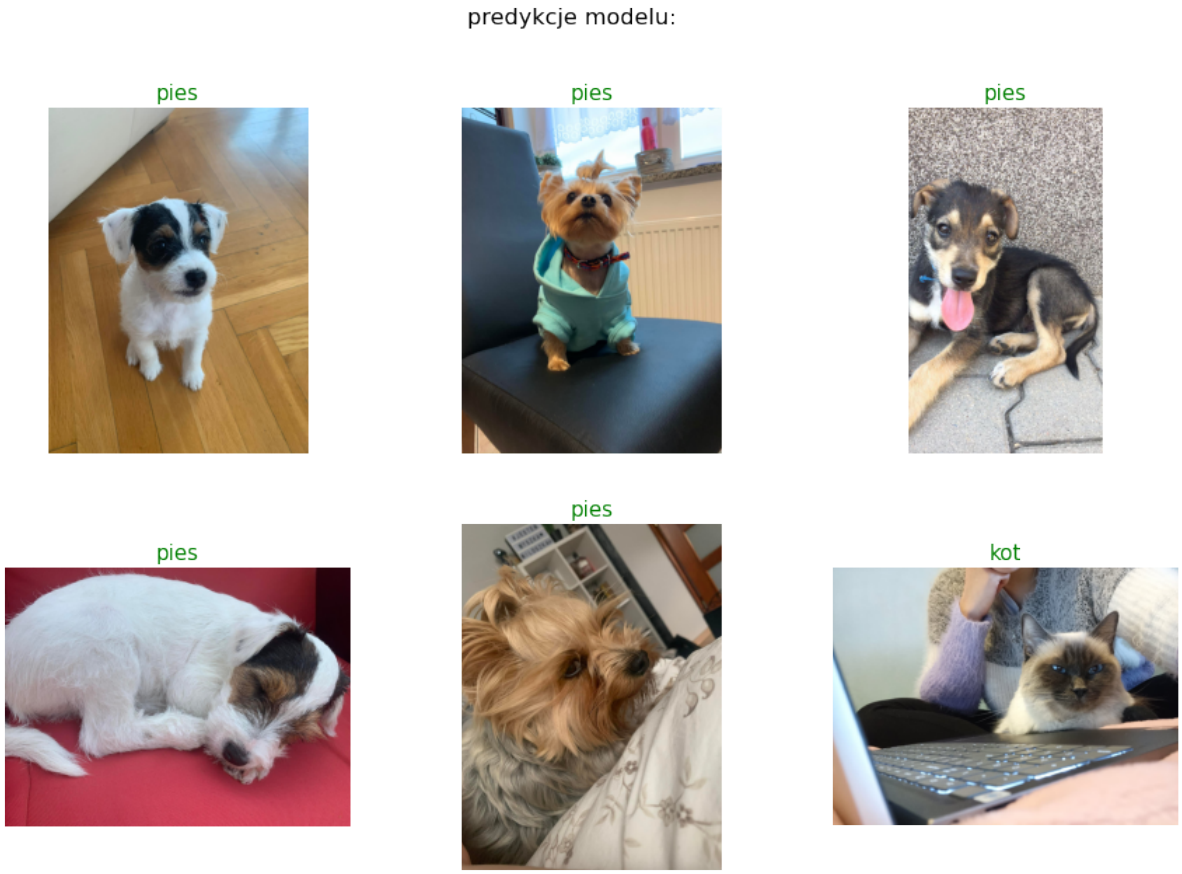


3 Wyniki

3.1 jakość modelu

po skończonym treningu precyzja:

- dla danych treningowych wynosiosła: **98 procent**
- dla danych testowych: **93 procent**



aby zweryfikować poprawność modelu uruchomiłem go dla zdjęć zrobionych przeze mnie oraz moich znajomych, tak aby mieć pewność że zdjęcia są różnych formatów, rozmiarów, różnej jakości i nie były wcześniej w żaden sposób przerobione

3.2 przyczyny takiej a nie innej jakość modelu

model okazał się stosunkowo skuteczny, przy klasyfikacji zdjęć nie spreparowanych skuteczność na poziomie 93 procent, możemy uznać za zadowalającą.

dlaczego model nie ma skuteczność́ci 100 procent?

1. ponieważ dysponowałem urządzeniem o niewielkiej możliwości obliczeniowej, a operacje na zdjęciach, są skomplikowane, trenowanie sieci skończyłem gdy jej dokładość miała tendencje rosnące.
2. niektóre ze zdjęć mogły być nieostre, lub przedstawiać zwierzę które jest mieszanką genetyczną psa i kota. przykład takich zdjęć:

4 wnioski