Projekt Zespołowy - Tytuł

Kacper Weiss,
Kacper Studenny,
Krystian Wojakiewicz

Kod projektu jest dostępny pod adresem:
https://github.com/KacperWeiss/ProjektZespolowyKWKWKS

18 czerwca 2019

Spis treści

Cele

- Celem projektu było stworzyć sieć neuronową głębokiego uczenia (deep learning) do rozpoznawania znamion nowotworowych na skórze człowieka.
- Kolejnym celem było stworzenie sieci opartej o binnary pattern w celu porównania efektów obu podejść rozpoznawania znamion nowotworowych.

Przygotowane Dane

- W celu wykonania projektu wykorzystano dane udostępnione na stronie kaggle.com [1], których część musieliśmy wykluczyć z projektu, ze względu na niedopasowanie do interesujących nas kategorii.
- Do wykluczonych zdjęć należały m.in. te w których znamiona były spowodowane wystąpieniem pasożyta, a więc takie które nie były ani zdrowe, ani nie przedstawiały znamion nowotworowych.
- Zdjęcia podzielono na dwie grupy: zdrowe znamiona, chore znamiona nowotworowe otrzymując w ten sposób kolejno 6000 i 2000 zdjęć. Te natomiast podzieliliśmy następująco:

Przygotowane danych - Tabela

Znamiona	Walidacja	Grupa testowa	Grupa ucząca
Chore	500	500	1000
Zdrowe	1000	1000	4000

Tablica: Podział danych na grupy

Deep learning

- Program oparty o sieć głęboką pisano w środowisku Anaconda, Jupiter Notebook i z pomocą biblioteki Keras.
- W projekcie użyliśmy sieci konwolucyjnej ze względu na jej ogólne dobre wyniki w pracy z obrazami.
- Sieć konwolucyjna (CNN lub ConvNet) jest klasą głębokich sieci neuronowych, które najczęściej stosowane są do analizy obrazów wizualnych.

Binary Pattern

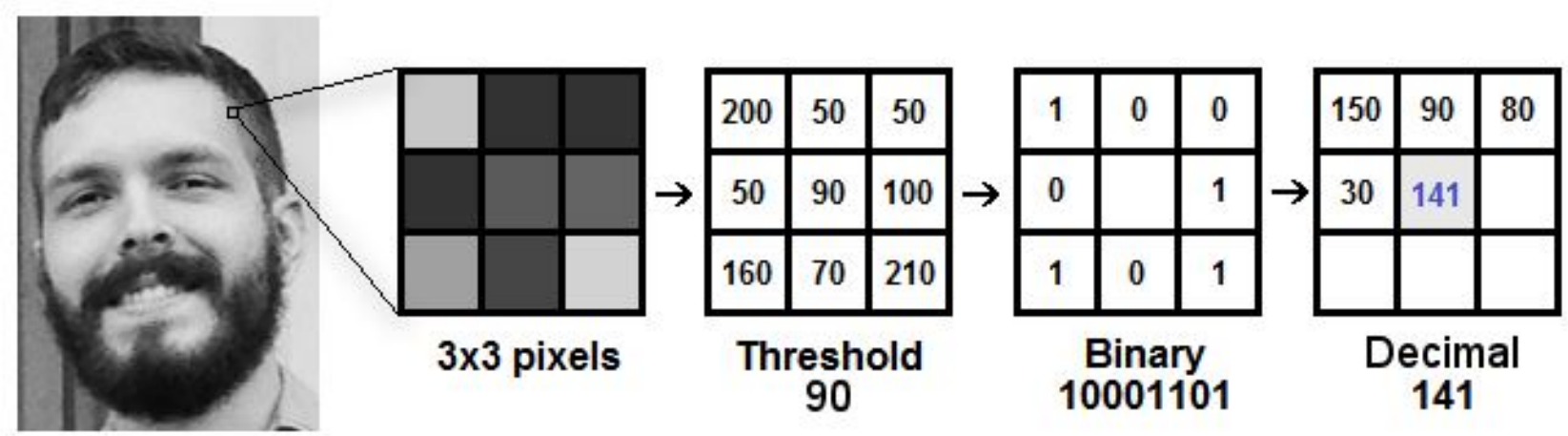
- Binary pattern (a dokładniej Local Binary Pattern - LBP) jest rodzajem deskryptora wizualnego używanego do klasyfikacji w wizji komputerowej.
- Jako że LBP jest deskryptorem wizualnym, może być wykorzystywany do zadań związanych z rozpoznawaniem cech z obrazów przedstawiających znamiona nowotworowe i zdrowe.

Sposób działania

- Na początku trzeba dobrać paramenty takie jak żadius", ńeighbors", "grid x", "grid y". W tym celu wykorzystuje się strukturę parametrów z biblioteki łbph". Następnie wywołujemy funkcje "Init"przekazując jej parametry.
- W następnej kolejności należy zacząć uczyć algorytm za pomocą funkcji "Train". Wszystkie zdjęcia muszą być tego samego rozmiaru.

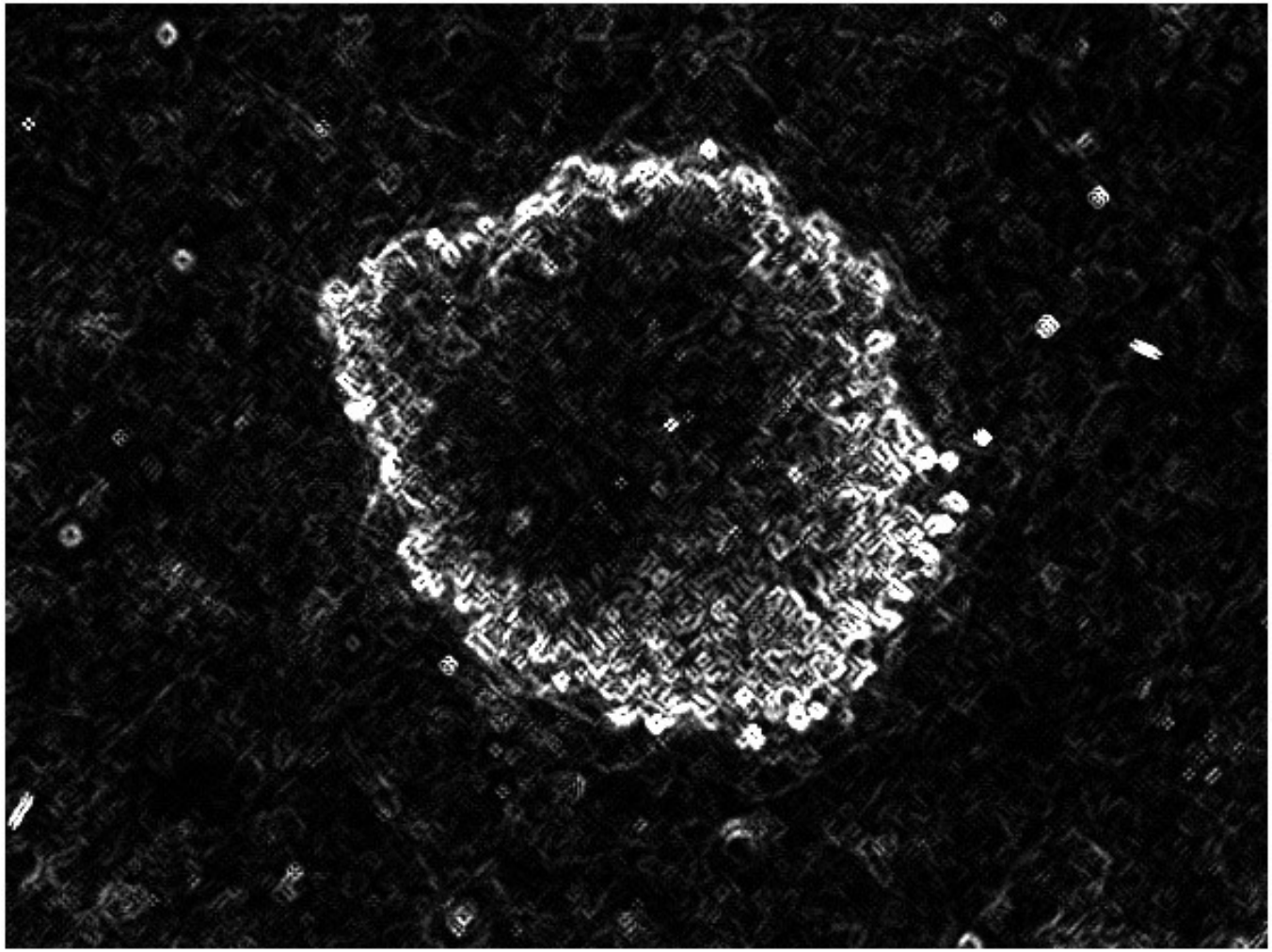
Sposób działania cz.2

- Następnie LBP przetworzy obraz na podstawie podanych jej w pierwszym kroku parametrów.

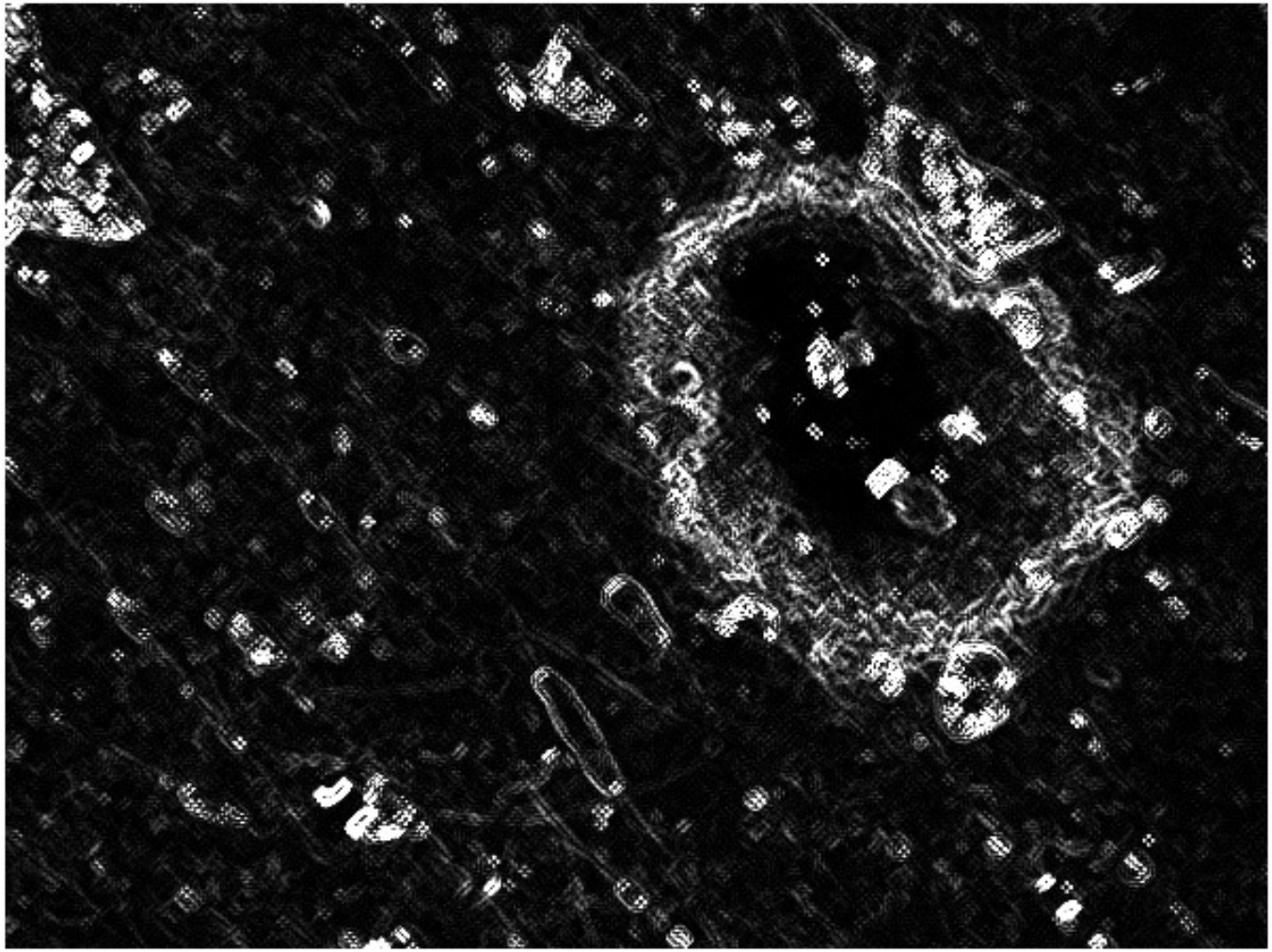


Obrazy przetworzone przez LBP

Obrazy przetworzone przez LBP



Obrazy przetworzone przez LBP



Teoria

- Sieci konwolucyjne takie jak CNN są regularnymi wersjami wielowarstwowych perceptronów, które to odnoszą się do w pełni połączonych sieci, gdzie każdy neuron w jednej warstwie jest połączony ze wszystkimi neuronami w następnej warstwie. Czyni to je podatnymi na przeładowanie danych.
- Typowe sposoby regularyzacji obejmują dodanie pewnej wielkości pomiaru wagi do funkcji straty. CNN przyjmują jednak inne podejście do regularyzacji: wykorzystują hierarchiczny wzór danych i tworzą bardziej złożone wzorce, wykorzystując mniejsze i prostsze wzorce. Dlatego w skali powiązań i złożoności CNN znajdują się na niższym poziomie.

Projekt sieci

- Tworząc model sieci głębokiej potrzeba wiele parametrów którym należy znaleźć najbardziej optymalnych ustawień. Wymagało to od nas testów na zasadzie prób i błędów szukając odpowiedniej wielkości sieci.

Projekt sieci

```
model = models.Sequential()
#1
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2,2)))
#2
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
#3
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
#4
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
#5
model.add(layers.Flatten())
#model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
```

Rysunek: Proponowany model sieci

Projekt sieci cz.2

- Wprowadzamy do pierwszej warstwy o 32 neuronach obraz rozmiaru 150x150 o 3 głębiach koloru, gdyż pracujemy tutaj na plikach typu .jpg, a niewielkie rozmiary obrazów pozwolą na znacznie szybsze uczenie sieci. Mamy tu wykorzystaną czterokrotnie funkcję Conv2D [3] naszej sieci konwolucyjnej.
- Pojawia się czterokrotnie MaxPooling2D, której ogólne działanie tutaj polega na dwukrotnym zmniejszeniu rozmiaru obrazu. W ostatnich fazach wprowadzamy funkcję Flatten, które w prosty sposób ujmując „wciska" dane do jednowymiarowej tablicy. Dense to kolejna warstwa sieci neuronowej.
- Ponieważ opracowywany problem obiera się na klasyfikacji binarnej to nasza ostatnia warstwa jest pojedynczym wyjściem z funkcją aktywacyjną typu sigmoid[4].

Efekty przyjętego projektu

- Otrzymana w ten sposób sieć pozwoliła na rozpoznawanie znamion nowotworowych na poziomie skuteczności rzędu 82%. Możliwe że dalsze konfigurowanie sieci polepszyłoby ten wynik. Być może ilość danych jakie mieliśmy dostępne nie wystarczyła do osiągnięcia lepszych wyników, albo sieć była zbyt mała.

Nauka sieci głębokiej

- Uznaliśmy że jedna z klas posiada niewielką ilość danych i zaimplementowaliśmy odpowiednią modyfikację zdjęć tj. przesunięcie, obrót, zbliżenie, przycinanie obrazu, odbicie w płaszczyźnie poziomej.
- Pozwoliło to na polepszenie modelu poprzez zmniejszenie strat zbioru walidacyjnego i treningowego.

Nauka sieci głębokiej - kod

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=-.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,)

test_datagen = ImageDataGenerator(rescale=1./255)

train_dir = r'C:\Users\Kacper\Documents\ANACONDA\Base\train'
train_generator = train_datagen.flow_from_directory(
            train_dir,
            target_size=(150, 150),
            batch_size=32,
            class_mode='binary')
```

Rysunek: Nauka sieci głębokiej

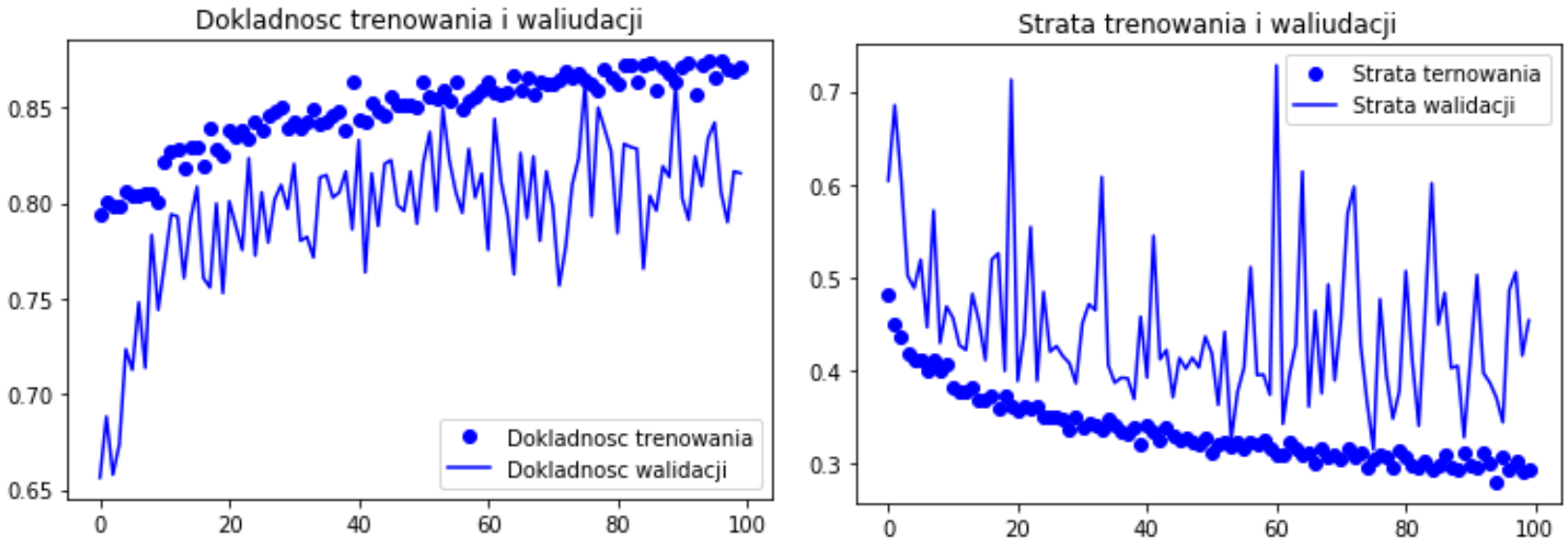
Nauka sieci głębokiej - kod cz.2

```
validation_dir = r'C:\Users\Kacper\Documents\ANACONDA\Base\validation'
validation_generator = test_datagen.flow_from_directory(
            validation_dir,
            target_size=(150, 150),
            batch_size=32,
            class_mode='binary')

history = model.fit_generator(
        train_generator,
        steps_per_epoch=100,
        epochs=100,
        validation_data=validation_generator,
        validation_steps=32)
#40 epochs wystarczy
```

Rysunek: Nauka sieci głębokiej cz.2

Wnioski (sieć głęboka)



Rysunek: Rezultaty z sieci głębokiej

Wnioski (sieć głęboka)

- Z wykresów wywnioskowaliśmy, że warto zaprzestać nauki po 40 epoce. Dodawanie kolejnych zwiększało skuteczność sieci jak i długoś jej uczenia. Sieć konwolucyjna spisała się wyjątkowo dobrze w tym zadaniu osiągając skuteczność rzędu 82%.
- Pomimo napotkanych przeszkód tj. bardzo zróżnicowane przypadki nowotworów skórnych raptem na 2000 zdjęć i 6000 zdjęć. Funkcja strat – krzyżowa entropia okazała się skuteczna dla tego zastosowania.
- Ponadto walidacja okazała się być istotną kwestią w uczeniu sieci pozwalając na bieżąco sprawdzać skuteczność sieci, a zestaw testowy ostatecznie potwierdzić to.

Bibliografia


[1] kaggle.com - zbiór zdjęć znamion na skórze
https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000


[2] keras.io - opis wykorzystywanej funkcji Conv2D
https://keras.io/layers/convolutional/


[3] „Deep Learning Praca z językiem Python i biblioteką Keras" - Francois Chollet


[4] cs.toronto.edu - RMS
http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf


[5] Wikipedia - informacje ogólne o sieci neuronowej
https://en.wikipedia.org/wiki/Convolutional_neural_network

Dziękuje za uwagę!