PRiAD zespół nr 8 J Klasyfikacja cyfr pisanych odręcznie (baza MNIST)

Andrzej Czechowski, Franciszek Wysocki, Bartosz Zakrzewski

Spis treści

Przykładowa dokumentacja

która pozwoli nam dowiedzieć się, jakie są parametry i za co odpowiadają:
https://keras.io/api/optimizers/
https://www.tensorflow.org/api_docs/python/tf/keras/activations
https://keras.io/api/metrics/

Pomysł na realizację

Badanie różnych parametrów modelu Sequential() sieci neuronowej.

1. Zaimplementowanie gotowego rozwiązania/rozwiązań.

https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d
lub / i
https://pythonprogramming.net/introduction-deep-learning-python-tensorflow-keras/

2. Następnie:

Zmiana **jednego parametru** i zapisanie jak zmieniło się działanie sieci neuronowej - przede wszystkim jak zmieniła się wartość metryki.

Sprawdzimy “accuracy” i jeżeli będzie to niewystarczające, to użyjemy też innych metryk.

“model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])”

Możemy także zmierzyć czas “trenowania” sieci neuronowej w zależności od ilości “epochów” i jak zmienia się poprawność prognoz algorytmu.

Przykład atrybutów, które można zbadać:

Różne parametry trenowania/kompilowania modelu (loss, optimizer):

```
"model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])"
```

Różne activation

```
"model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))"
```

Różna ilość neuronów

```
"model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))"
```

Różna ilość treningu:

```
"model.fit(x_train, y_train, epochs=10)"
```

Różna ilość warstw sieci neuronowej:
(ile razy `model.add`)

Przez badanie mamy na myśli tworzenie wykresów, pisanie wniosków oraz opisanie jak działa dany atrybut.

Dodatkowo możemy sprawdzić jakie zdjęcia są źle sklasyfikowane.

Plan działań

Stworzymy repozytorium na GitHubie. Będziemy pracować za pomocą narzędzia Jupyter Notebook, gdzie stworzymy kilka notatników. Wnioski i wykresy mogą zostać umieszczone w oddzielnym pliku pdf.

Jeżeli zajdzie potrzeba to podział prac zostanie dokonany, jeżeli nie, to dana osoba wybierze parametr, który chce analizować i poinformuje o tym zespół.

Rozwiązania dostępne w Internecie

Przykładowe gotowe rozwiązania zadania:
https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d
https://pythonprogramming.net/introduction-deep-learning-python-tensorflow-keras/

Większość rozwiązać w Internie opiera się na module tensorflow - czyli platformie open-source uczenia maszynowego.

Z tego modułu można pobrać bazę danych MNIST.
Mamy tam 60 000 elementów zbioru uczącego i 10 000 elementów zbioru testowego.
Do każdego zdjęcia jest przypisana etykieta, która mówi, jaka to jest cyfra.

Większość rozwiązań wybiera model Sequential - czyli prosty model sieci neuronowej - jej warstwy są liniowo ustawione.

```
model = tf.keras.models.Sequential()
```

Dodanie do niego warstw ukrytych:

```
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
```

I ostatniej warstwy:

```
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))
```

Następnie wybieramy ustawienia trenowania modelu:

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Wybieramy ile razy trenujemy:

```
model.fit(x=x_train,y=y_train, epochs=10)
```

Sprawdzamy poprawność na zbiorze testowym:

```
model.evaluate(x_test,y_test)
```

Rozwiązania zwykle dochodzą do „accuracy” równej około 99% i w zbiorze uczącym i w zbiorze testowym.

Możemy sprawdzić jak przykładowe zdjęcie jest sklasyfikowane przez nasz model:

```
image_index = 4444
pred = model.predict(x_test[image_index].reshape(1, 28, 28, 1))
print(pred.argmax())
```