Лабораторна робота №3. Кисляк Марини

Варіант 3. MNIST

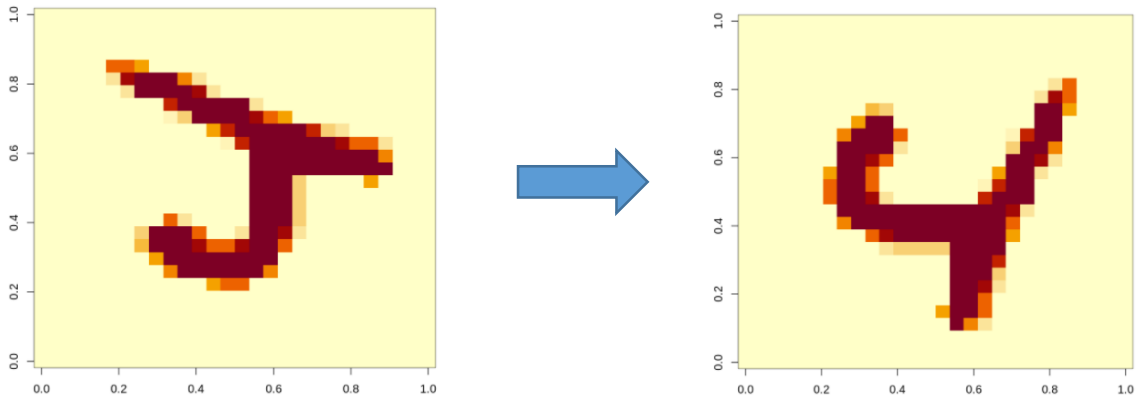
Для виконання даної роботи використовувся Google Colab.

1) Завантажте датасет, підготуйте його для входу в нейронну мережу.

Датасет MNIST - це набір даних із 60 000 зображень у форматі grayscale розміром 28x28 із 10 цифр, а також тестовий набір із 10000 зображень.

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

Якщо ми візьмемо довільний елемент нашого датасету і виведемо як зображення- це буде зображення повернуте на 90 градусів вліво. Тому, щоб отримати правильне зображення, елемент (який представлений у вигляді матриці) потрібно транспонувати та «прочитати» справа наліво, а потім намалювати функцією `image()`.



Елемент 41097

2) Будуємо щільну нейронну мережу для класифікації.

Тут при epochs = 15, batch_size=128 отримали наступні результати:

Train
accuracy
0.992233335971832
Test
accuracy
0.973699986934662

Як бачимо, accuracy на тестовій вибірці менша ніж на тренувальній. Тому, можливо тут мало місце перенавчання, але різниця впринципі не суттєва.

3)Будуємо згорткову нейронну мережу.

Train
accuracy
0.997399985790253
Test
accuracy
0.990100026130676

Для згорткові нейронної мережі accuracy краща, ніж для щільної нейронної мережі. Також бачимо, що для тестової та тренувальної вибірки значення майже одинакові. Тому модель працює нормально і перенавчання немає.

4) Тепер спробуємо додати шари BatchNormalization та Dropout для покращення результатів моделі.

```
inputs = layer_input(shape=c(28,28,1))
z = layer_conv_2d(inputs, filters=32, kernel_size=c(3,3), activation="selu
")
z = layer_max_pooling_2d(z, pool_size = c(2, 2))
z = layer_conv_2d(z, filters = 64, kernel_size = c(3, 3), activation = "re
lu")
z = layer_max_pooling_2d(z, pool_size = c(2, 2))
z = layer_conv_2d(z, filters = 64, kernel_size = c(3, 3), activation = "re
lu")
z = layer_dropout(z, rate = 0.3)
z = layer_batch_normalization(z)
outputs = z
model = keras_model(inputs, outputs)
```

accuracy
0.992699980735779

Як бачимо, при додаванні додаткових шарів, якість класифікації на тестовій вибірці дорівнює майже 100%.

Висновок. При використанні щільної нейронної мережі, результат можна покращити за допомогою підбору чи зміни epochs чи batch_size. В нашому випадку, при збільшені цих значень до 15 та 128 відповідно, модель показала гарний результат. Результати згорткової нейронної мережі можна покращити за допомогою додавання додаткових шарів, таких як BatchNormalization чи Dropout. В даному випадку перенавчання моделі не відбулося.