Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 5 з дисципліни
«Прикладні задачі машинного навчаннях»

«Проектування та навчання штучної нейронної мережі для задач класифікації»

Варіант 7

Виконав студент: ІП-12 Васильєв Єгор Костянтинович

Перевірив: Нестерук Андрій Олександрович

Київ 2023

Лабораторна робота №5

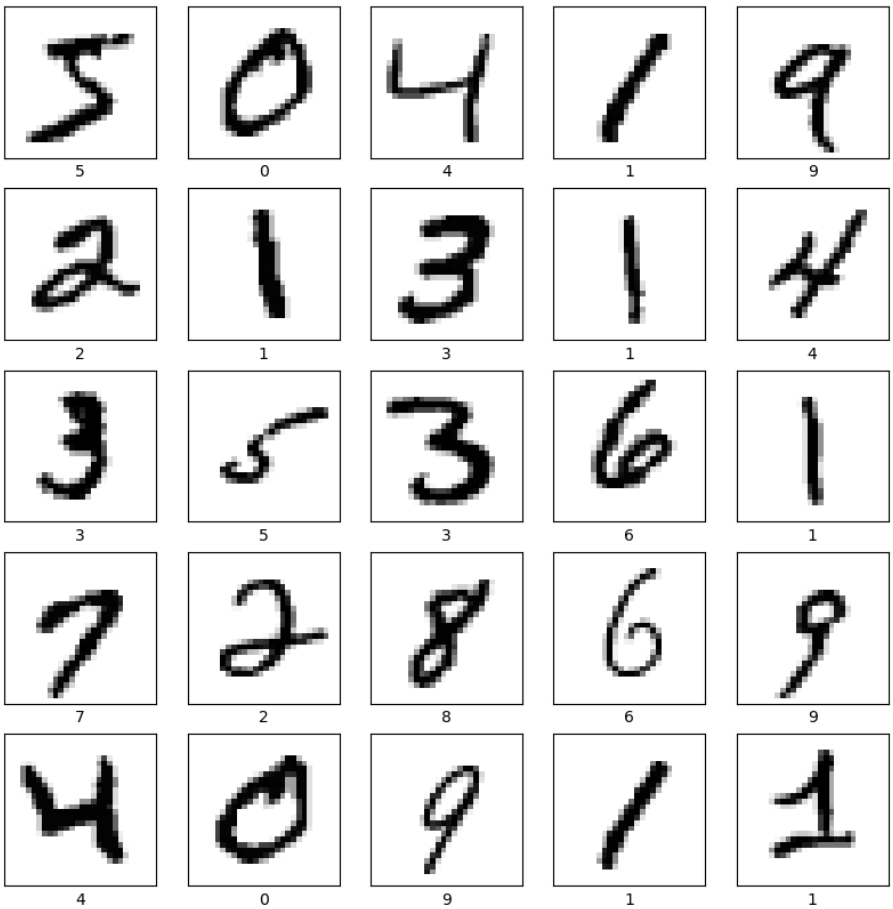
Тема: Проектування та навчання штучної нейронної мережі для задач класифікації.

Постановка завдання:

1) Візуалізація перших 25 зображень набору mnist.
2) Створення та навчання моделі на тренувальних даних.
3) Тест точності отриманої моделі.
4) Перевірка роботи моделі на довільному зображені.
5) Візуалізація перших 25 зображень набору cifar10.
6) Створення та навчання моделі на тренувальних даних.
7) Тест точності отриманої моделі.
8) Перевірка роботи моделі на довільному зображені.
9) Візуалізація перших 25 зображень набору fmnist.
10) Створення та навчання моделі на тренувальних даних.
11) Тест точності отриманої моделі.
12) Перевірка роботи моделі на довільному зображені.

Хід роботи:

- Графічне представлення частини набору даних mnist



- Навчання моделі

```
Epoch 1/5
600/600 [==============================] - 3s 5ms/step - loss: 0.2482 - accuracy: 0.9285
Epoch 2/5
600/600 [==============================] - 3s 4ms/step - loss: 0.0988 - accuracy: 0.9697
Epoch 3/5
600/600 [==============================] - 3s 4ms/step - loss: 0.0657 - accuracy: 0.9801
Epoch 4/5
600/600 [==============================] - 3s 4ms/step - loss: 0.0467 - accuracy: 0.9861
Epoch 5/5
600/600 [==============================] - 3s 4ms/step - loss: 0.0350 - accuracy: 0.9894
```

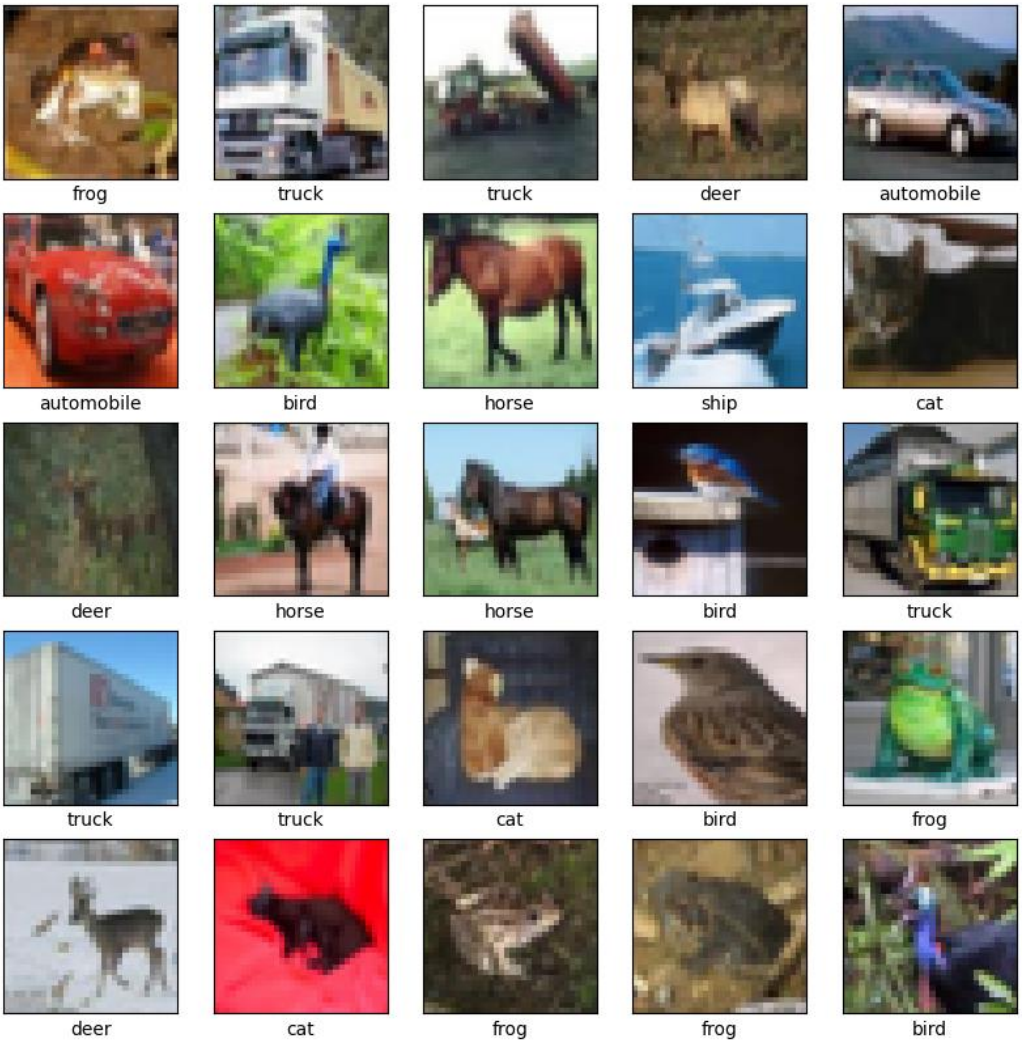
- Оцінка точності роботи моделі

```
313/313 [==============================] - 1s 1ms/step - loss: 0.0695 - accuracy: 0.9793
```

- Тест моделі на власному зображені

```
1/1 [==============================] - 0s 74ms/step
Result: 9
```

- Графічне представлення частини набору даних cifar10



- Навчання моделі

```
Epoch 1/15
782/782 [==============================] - 27s 34ms/step - loss: 1.6764 - accuracy: 0.3942
Epoch 2/15
782/782 [==============================] - 26s 33ms/step - loss: 1.3595 - accuracy: 0.5170
Epoch 3/15
782/782 [==============================] - 21s 27ms/step - loss: 1.2222 - accuracy: 0.5702
Epoch 4/15
782/782 [==============================] - 22s 28ms/step - loss: 1.1317 - accuracy: 0.6012
Epoch 5/15
782/782 [==============================] - 22s 28ms/step - loss: 1.0666 - accuracy: 0.6277
Epoch 6/15
782/782 [==============================] - 23s 29ms/step - loss: 1.0069 - accuracy: 0.6472
Epoch 7/15
782/782 [==============================] - 28s 35ms/step - loss: 0.9616 - accuracy: 0.6659
Epoch 8/15
782/782 [==============================] - 23s 29ms/step - loss: 0.9146 - accuracy: 0.6828
Epoch 9/15
782/782 [==============================] - 21s 27ms/step - loss: 0.8784 - accuracy: 0.6936
Epoch 10/15
782/782 [==============================] - 20s 26ms/step - loss: 0.8462 - accuracy: 0.7095
Epoch 11/15
782/782 [==============================] - 20s 26ms/step - loss: 0.8214 - accuracy: 0.7165
Epoch 12/15
782/782 [==============================] - 21s 27ms/step - loss: 0.7908 - accuracy: 0.7259
Epoch 13/15
782/782 [==============================] - 22s 28ms/step - loss: 0.7670 - accuracy: 0.7335
Epoch 14/15
782/782 [==============================] - 22s 28ms/step - loss: 0.7508 - accuracy: 0.7401
Epoch 15/15
782/782 [==============================] - 22s 28ms/step - loss: 0.7363 - accuracy: 0.7434
```

- Оцінка точності роботи моделі

```
313/313 [==============================] - 1s 3ms/step - loss: 0.9026 - accuracy: 0.6952
```

- Тест моделі на власному зображені

- Графічне представлення частини набору даних fmnist

- Навчання моделі

```
Epoch 1/10
235/235 [==============================] - 2s 6ms/step - loss: 0.6129 - accuracy: 0.7791
Epoch 2/10
235/235 [==============================] - 1s 6ms/step - loss: 0.4112 - accuracy: 0.8487
Epoch 3/10
235/235 [==============================] - 1s 6ms/step - loss: 0.3595 - accuracy: 0.8670
Epoch 4/10
235/235 [==============================] - 1s 6ms/step - loss: 0.3299 - accuracy: 0.8771
Epoch 5/10
235/235 [==============================] - 1s 6ms/step - loss: 0.3075 - accuracy: 0.8843
Epoch 6/10
235/235 [==============================] - 1s 6ms/step - loss: 0.2878 - accuracy: 0.8921
Epoch 7/10
235/235 [==============================] - 1s 6ms/step - loss: 0.2722 - accuracy: 0.8974
Epoch 8/10
235/235 [==============================] - 1s 6ms/step - loss: 0.2597 - accuracy: 0.9007
Epoch 9/10
235/235 [==============================] - 1s 6ms/step - loss: 0.2487 - accuracy: 0.9062
Epoch 10/10
235/235 [==============================] - 1s 6ms/step - loss: 0.2372 - accuracy: 0.9096
```

- Оцінка точності роботи моделі

```
313/313 [==============================] - 1s 2ms/step - loss: 0.3769 - accuracy: 0.8736
```

- Тест моделі на власному зображені


```
Result: Shirt , confidence: 0.8029477
```

Висновок

Було досліджено бібліотеку keras, створену для побудови та навчання нейронних мереж; було побудовано три різні моделі для задач класифікації зображень з трьох різних наборів даних; було ознайомлено з різними типами прихованих шарів та методами компіляції мереж; було досягнуто точності моделей: 97,9% для mnist датасету, 69,5% для cifar10 та 87,3% для fmnist; загалом досягти прийнятної точності складніше за все для моделі, що тренується на кольорових зображеннях, але досліджуючи вплив різних параметрів на кінцевий результат, вдалось досягти і цього; також, було протестовано моделі на власних зображеннях, перетворивши їх у необхідний формат і завантаживши в моделі для прогнозування.

Вихідний код

mnist.py

```
import keras
from keras import layers
from keras.datasets import mnist
from keras.utils import to_categorical
import matplotlib.pyplot as plt

(train_imgs, train_labels), (test_imgs, test_labels) = mnist.load_data()

plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(255 - train_imgs[i], cmap='gray')
    plt.xlabel(train_labels[i])
plt.show()

train_imgs = train_imgs.reshape((60000, 28 * 28))
train_imgs = train_imgs.astype('float32') / 255
test_imgs = test_imgs.reshape((10000, 28 * 28))
test_imgs = test_imgs.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

network = keras.models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
network.fit(train_imgs, train_labels, epochs=5, batch_size=100)

test_loss, test_acc = network.evaluate(test_imgs, test_labels)

network.save(r'F:\Egor\Уроки\Машинне навчання\Ла65\digits_model.h5')
```

Testing Digits Model.py

```
import numpy as np
from keras.models import load_model
from PIL import Image

model = load_model(r'F:\Egor\Уроки\Машинне навчання\Ла65\digits_model.h5')

test_digit = Image.open(r'F:\Egor\Уроки\Машинне навчання\Ла65\9.png').convert('L')
test_digit = test_digit.resize((28, 28))
# noinspection PyTypeChecker
test_digit = 1 - np.array(test_digit) / 255
test_digit = test_digit.reshape((1, 28 * 28))

prediction = list(model.predict(test_digit)[0])
print('Result:', prediction.index(max(prediction)), ', confidence:', max(prediction))
```

cifar10.py

```python
import keras
from keras import layers
from keras.datasets import cifar10
from keras.utils import to_categorical
import matplotlib.pyplot as plt

(train_imgs, train_labels), (test_imgs, test_labels) = cifar10.load_data()
objects = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(train_imgs[i])
    plt.xlabel(objects[train_labels[i][0]])
plt.show()

train_imgs = train_imgs.astype('float32') / 255
test_imgs = test_imgs.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

network = keras.models.Sequential()
network.add(layers.Conv2D(filters=32, kernel_size=(4, 4), input_shape=(32, 32, 3), activation='relu'))
network.add(layers.MaxPool2D(pool_size=(2, 2)))
network.add(layers.Dropout(0.25))  # Drop 25% of the units from the layer.
network.add(layers.Conv2D(filters=32, kernel_size=(4, 4), input_shape=(32, 32, 3), activation='relu'))
network.add(layers.MaxPool2D(pool_size=(2, 2)))
network.add(layers.Dropout(0.25))
network.add(layers.Flatten())
network.add(layers.Dense(256, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
network.fit(train_imgs, train_labels, epochs=15, batch_size=64)

test_loss, test_acc = network.evaluate(test_imgs, test_labels)

network.save(r'F:\Egor\Уроки\Машинне навчання\Ла65\cifar10_model.h5')
```

Testing Cifar10 Model.py

```python
import numpy as np
from keras.models import load_model
from PIL import Image

model = load_model(r'F:\Egor\Уроки\Машинне навчання\Ла65\cifar10_model.h5')

objects = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

test_image = Image.open(r'F:\Egor\Уроки\Машинне навчання\Ла65\horse.jpeg')
test_image = test_image.resize((32, 32))
# noinspection PyTypeChecker
img_array = np.asarray(test_image, dtype='float32')
img_array = img_array[:, :, :3]
img_array = img_array / 255.0

# Add a new dimension to the array to represent the batch size (1)
img_array = np.expand_dims(img_array, axis=0)

prediction = list(model.predict(img_array)[0])
print('Result:', objects[prediction.index(max(prediction))], ', confidence:', max(prediction))
```

fmnist.py

```python
import keras
from keras import layers
from keras.datasets import fashion_mnist
from keras.utils import to_categorical
import matplotlib.pyplot as plt

(train_imgs, train_labels), (test_imgs, test_labels) = fashion_mnist.load_data()
objects = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(255 - train_imgs[i], cmap='gray')
    plt.xlabel(objects[train_labels[i]])
plt.show()

train_imgs = train_imgs.reshape((60000, 28 * 28))
train_imgs = train_imgs.astype('float32') / 255
test_imgs = test_imgs.reshape((10000, 28 * 28))
test_imgs = test_imgs.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

network = keras.models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(128, activation='relu'))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
network.fit(train_imgs, train_labels, epochs=10, batch_size=256)

test_loss, test_acc = network.evaluate(test_imgs, test_labels)

network.save(r'F:\Egor\Уроки\Машинне навчання\Ла65\fashion_model.h5')
```

Testing Fashion Model.py

```python
import numpy as np
from keras.models import load_model
from PIL import Image

model = load_model(r'F:\Egor\Уроки\Машинне навчання\Ла65\fashion_model.h5')

objects = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

test_img = Image.open(r'F:\Egor\Уроки\Машинне навчання\Ла65\T-shirt.jpg').convert('L')
test_img = test_img.resize((28, 28))
# noinspection PyTypeChecker
test_img = 1 - np.array(test_img) / 255
test_img = test_img.reshape((1, 28 * 28))

prediction = list(model.predict(test_img)[0])
print('Result:', objects[prediction.index(max(prediction))], ', confidence:', max(prediction))
```