Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний

інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Катедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 5 з дисципліни

«Прикладні задачі машинного навчання»

«Проектування та навчання штучної

нейронної мережi для задач класифiкацiї»

Виконав студент ІП-11 Лесів Владислав Ігорович

Перевірив Нестерук Андрій Олександрович

Київ 2023

**Лабораторна робота 5**

**Проектування та навчання штучної**

**нейронної мережi для задач класифiкацiї**

**Постановка завдання.**

1. Виконати завдання з прикладу й отримати файл з навченою моделлю для розпiзнавання рукописних цифр. У будь-якому графiчному редакторi створити файл з рукописною цифрою й розпiзнати її. Пояснити результат.
2. Спроєктувати й розробити нейронну мережу на основi таких наборiв даних, iмплементованих в Keras:
   1. Cifar10;
   2. FMNIST.

Потрiбно розробити архiтектуру нейронної мережi, навчити її на тестових прикладах i продемонструвати її роботу на кiлькох зображеннях.

**Хід роботи.**

1. Виконую завдання з прикладу.

Спочатку імпортую необхідні модулі й завантажую масиви навчальних і тестових даних набору mnist.

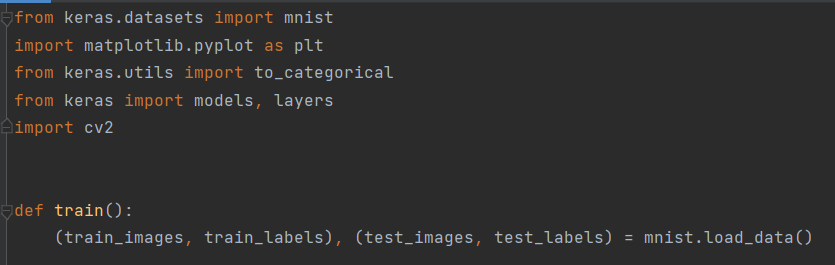


Рисунок 1 – Імпортування модулів й завантаження даних

Далі переглядаю перші 25 зображень набору даних: візуалізую відповідні дані і підписую цифри відповідними індексами. У даному наборі даних кожній цифрі відповідає індекс, власне, цієї цифри.

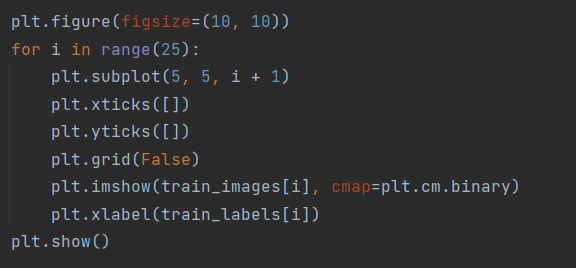


Рисунок 2 – Код для візуалізації перших 25 цифр набору

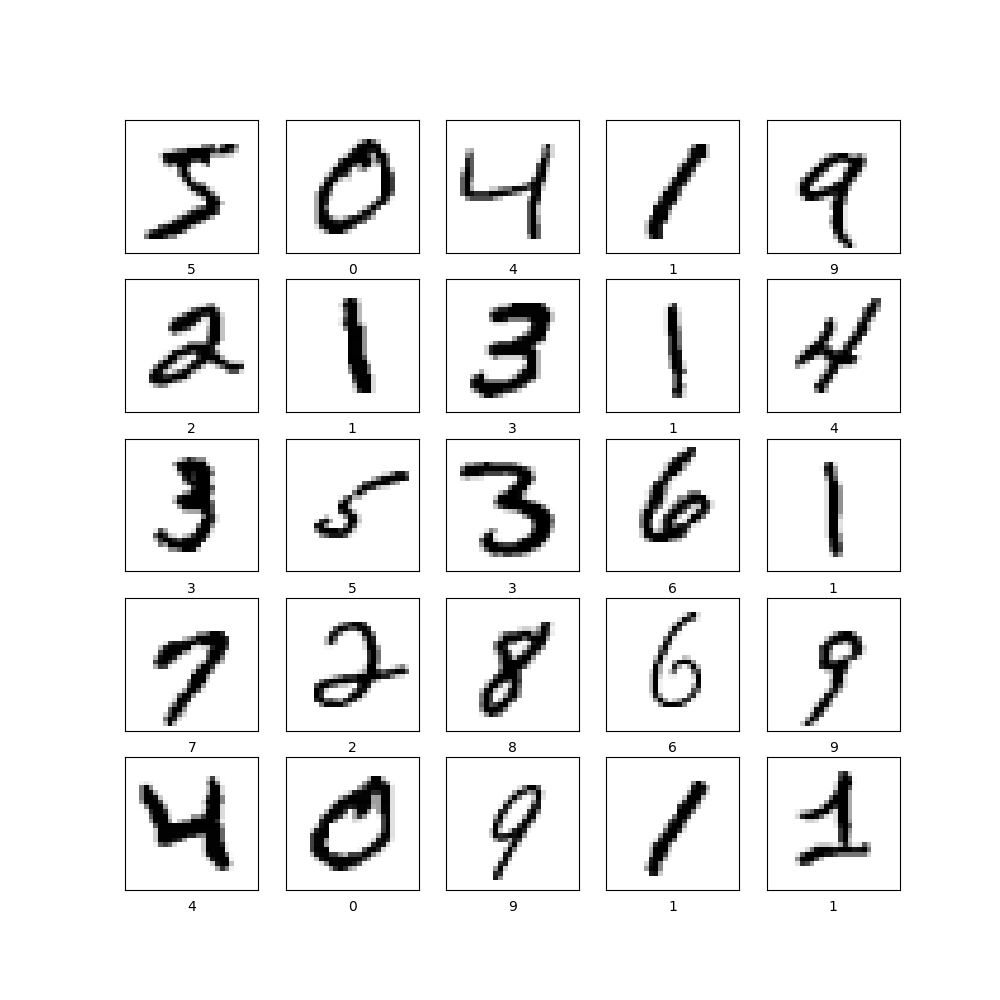


Рисунок 3 – Відповідна візуалізація перших 25 цифр набору

Далі я перетворюю тривимірний масив у двовимірний і нормалізую для того, щоб отримати значення в межах .

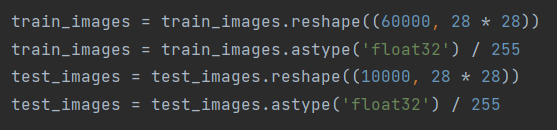


Рисунок 4 – Перетворення й нормалізація масивів даних

Після цього маємо зображення, які подаються вектором довжини 784.

Наступним кроком завантажую мітки – масиви з відгуками на зображення.

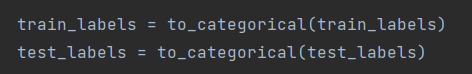


Рисунок 5 – Завантаження масивів відгуків

Створю модель. Я використовую клас Sequential і два щільні, повнозв’язні шари Dense, в яких кожен нейрон одного шару зв’язаний з кожним нейроном іншого шару. У прихованому шарі маємо 512 нейронів, у вихідному – 10, оскільки маємо 10 можливих варіантів.

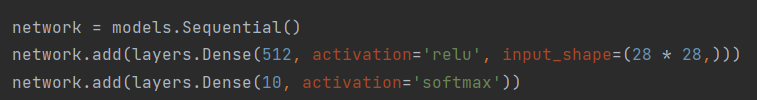


Рисунок 6 – Створення моделі

Далі скомпілюю модель. Оскільки дана модель вирішує задачу класифікації з багатьма класами, то застосовуємо оптимальне поєднання параметрів для такої класифікації.

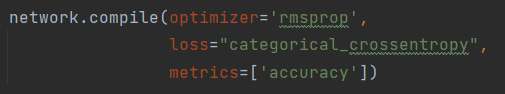


Рисунок 7 – Компіляція моделі

Для навчання викликаю метод fit().



Рисунок 8 – Навчання моделі

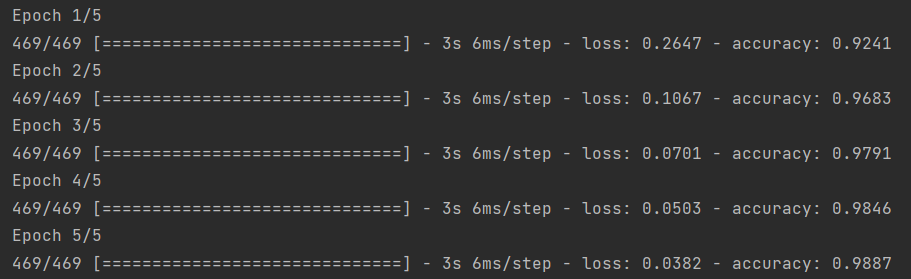


Рисунок 9 – Результати навчання моделі

Бачимо, що на навчальних даних було досягнуто точности 0.9887, тобто точність висока.

Після навчання перевіряю модель на контрольному наборі даних.

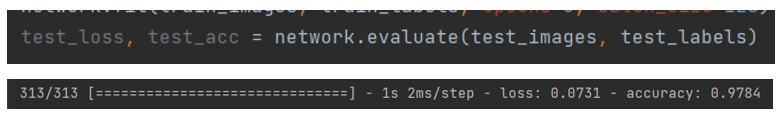


Рисунок 10 – Перевірка моделі на тестових даних

На тестових даних досягнуто точности 0.9784 – модель працює ефективно.

Тепер збережу модель у файлі «model1.h5» для подальшого використання.



Рисунок 11 – Збереження моделі у файлі

Тепер попрацюю із моделлю. Завантажую попередньо збережену модель.

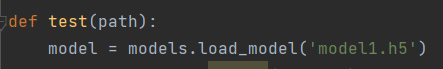


Рисунок 12 – Завантаження моделі

У графічному редакторі Paint створюю рукописні цифри. Цифри 3, 5 і 7 зберігаю у відповідних файлах “test.png”, “test1.png”, “test3.png”.

Далі такі файли завантажую в OpenCV як сіре зображення і переформатовую розмір отриманого масиву до розміру вхідних даних моделі.

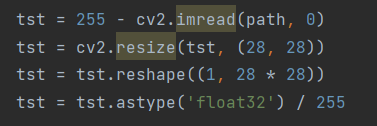


Рисунок 13 – Підготовка рукописних цифр

Ці дані передаю функції model.predict(tst). На виході отримую список списків з ймовірностями появи числа і. Обираємо позицію з максимальною ймовірністю.

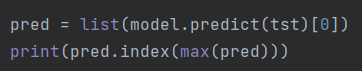


Рисунок 14 – Одержання відповідного числа

Запускаю функцію test(), яка отримує шлях до рукописної цифри і виводить передбачення, на трьох зразках.



Рисунок 15 – Рукописні цифри

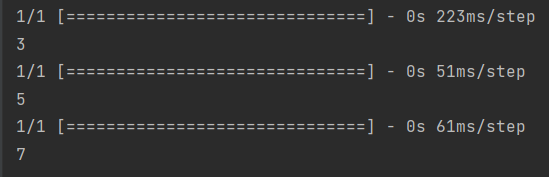


Рисунок 16 – Результат прогнозування цифр

Бачимо, що в усіх зразках модель спрацювала коректно.

1. Проєктую й розробляю нейронну мережу на основi набору даних Cifar10.

Завантажую дані з набору, створюю список назв міток, аби числові значення показувалися у форматі, доступному для людини.

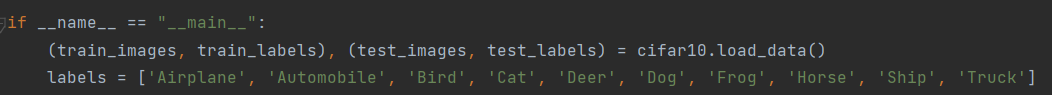


Рисунок 17 – Завантаження даних

Далі виводжу розміри вхідних і вихідних даних для кращого орієнтування.

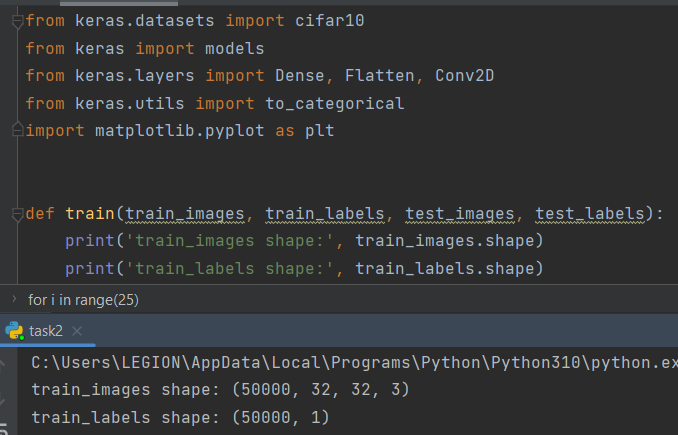


Рисунок 18 – Виведення розмірів даних

Бачимо, що вхідні дані – це RGB картинка 32\*32 пікселі, відповідно кожному пікселю задано колір у форматі RGB. Маємо 50000 навчальних значень.

Далі переглядаю перші 25 зображень набору даних: візуалізую відповідні дані і підписую їх відповідними назвами, для чого використовую створений список.

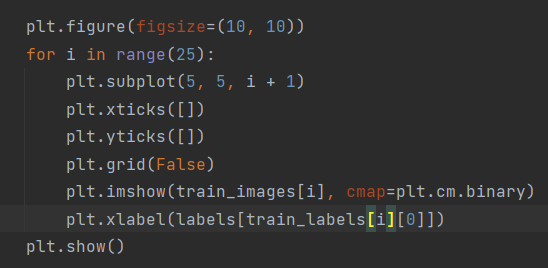


Рисунок 19 – Код для візуалізації набору даних

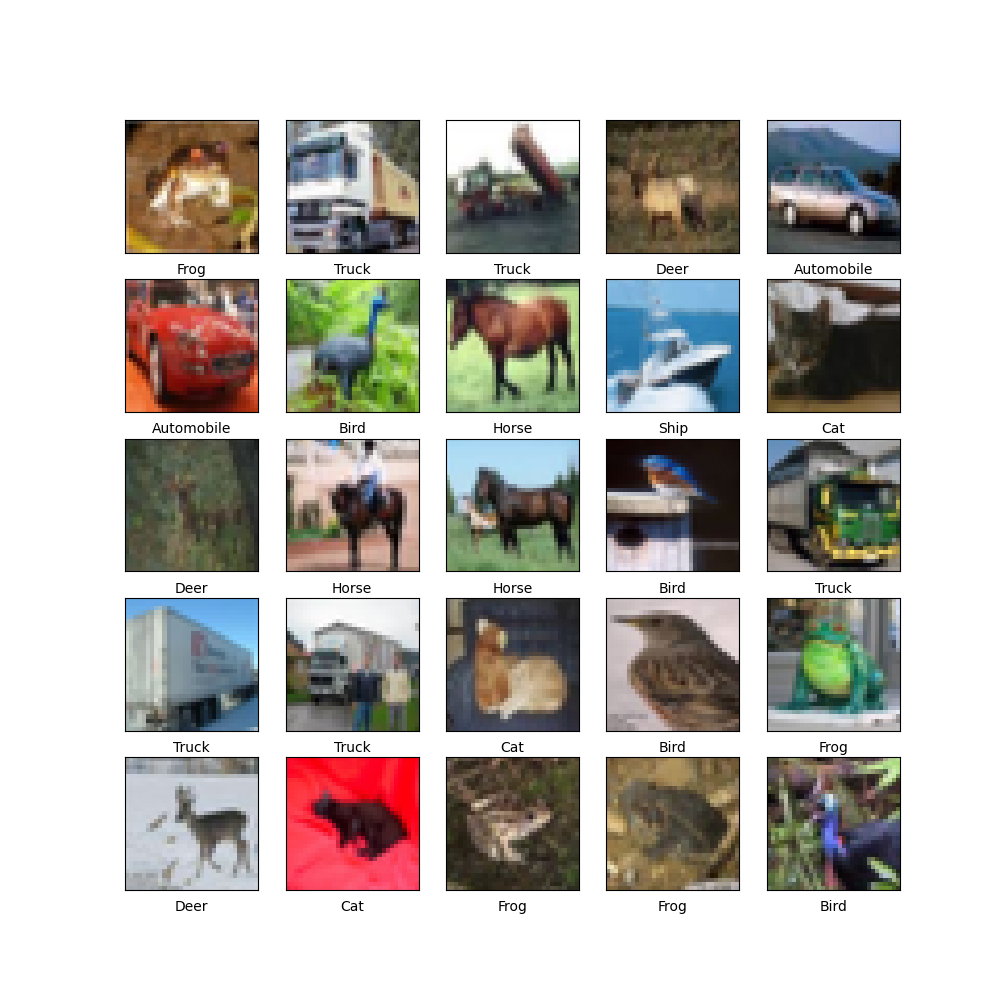


Рисунок 20 – Візуалізація набору даних

Бачимо, що зображуються дуже різні речі.

Наступним кроком нормалізую дані і завантажую мітки, тобто масиви з відгуками на зображення.

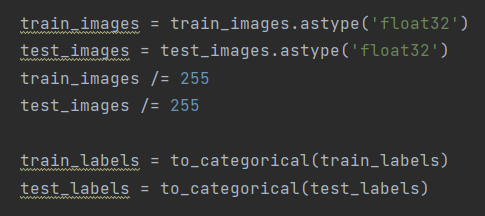


Рисунок 21 – Нормалізація й завантаження міток

Створюю модель. Я використовую клас Sequential. Першим шаром я використовую Conv2D. У даній класифікації близькі асоціації між ознаками мають значення для виявлення об'єктів. І дуже малоймовірно, що пікселі, які знаходяться далеко одне від одного, будуть якось корисними в даній ситуації. Тож для першого шару я використав саме Conv2D. Далі я використовую Flatten для того, щоб відійти від багатовимірного масиву і застосувати Dense. Відповідно ще два шари – це Dense з 512 нейронами. І заключним шаром маємо Dense з 10 нейронами, бо на вихід маємо 10 варіантів.

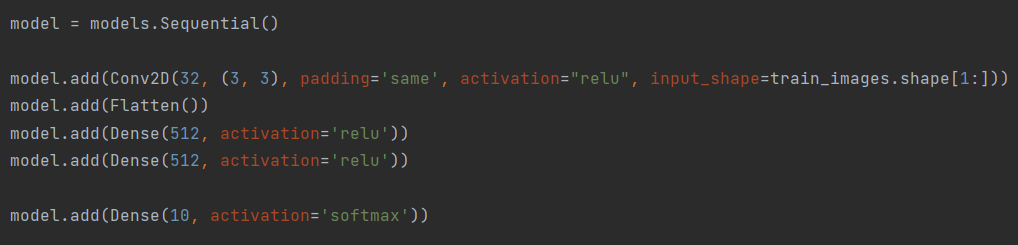


Рисунок 22 – Створення моделі

Компілюю модель і завантажую навчальні дані. Вибір параметрів такий, як і в попередньому пункті, адже маємо багато класів.

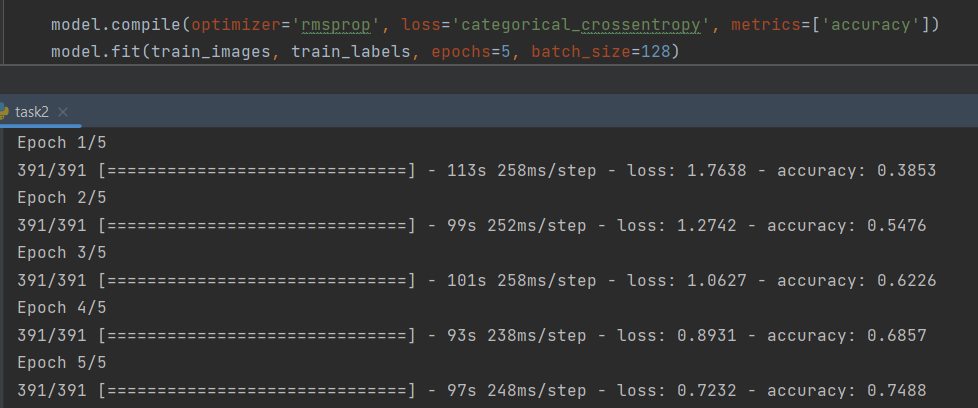


Рисунок 23 – Компіляція й завантаження навчальних даних

На навчальних даних ми досягли точности 0.7488. Після навчання перевіряю модель на контрольному наборі даних і зберігаю модель.

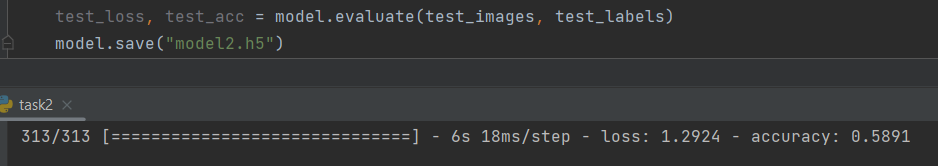


Рисунок 24 – Перевірка моделі на тестових даних

На тестових даних було досягнуто точности 0.5891. Модель працює з помірною ефективністю.

Тепер попрацюю із моделлю. Завантажую попередньо збережену модель.

Обираю порядковий номер зображення з тестових даних, переформатовую його під формат вхідних даних і роблю передбачення за допомогою model.predict(). Візуалізую зображення і підписую так, як передбачила мережа, а також так, як воно є насправді.

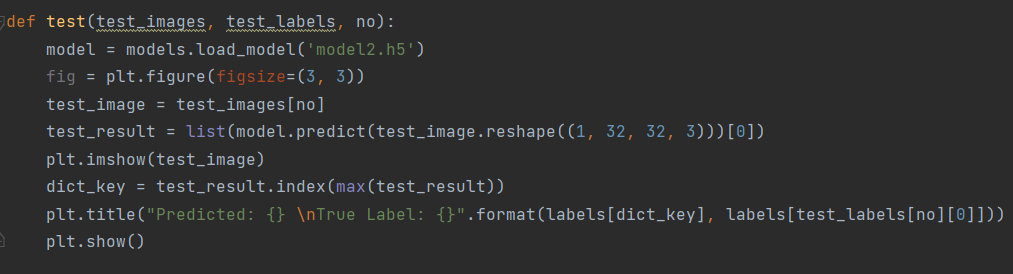


Рисунок 25 – Код до завантаження та тестування моделі

Обираю три випадкові зображення, наприклад, за номерами 203, 204, 205.

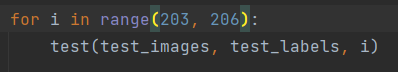


Рисунок 26 – Тестування моделі

Отримуємо такі результати.

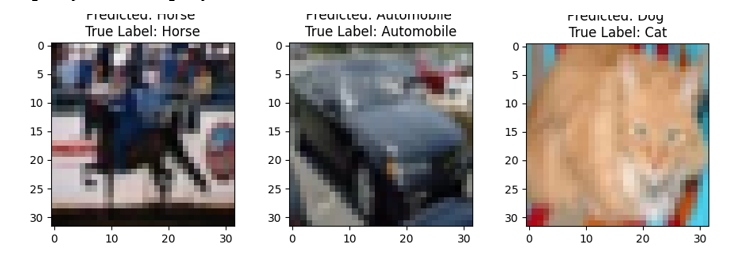


Рисунок 27 – Отримані результати передбачення

Бачимо, що у двох з трьох випадків передбачення було правильним, але кішку модель передбачила як собаку. Це може бути пов’язано з тим, що це дві домашні тварини, які мають схожі ознаки, тож розрізнити між кішкою і собакою важче, ніж між кішкою й автомобілем.

1. Проєктую й розробляю нейронну мережу на основi набору даних FMNIST.

Аналогічно до попереднього пункту, завантажую дані з набору, створюю список назв міток, аби числові значення показувалися у форматі, доступному для людини.

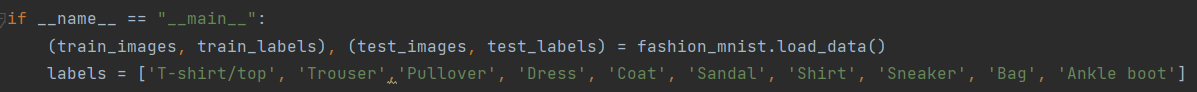


Рисунок 28 – Завантаження даних

Далі виводжу розміри вхідних і вихідних даних для кращого орієнтування.

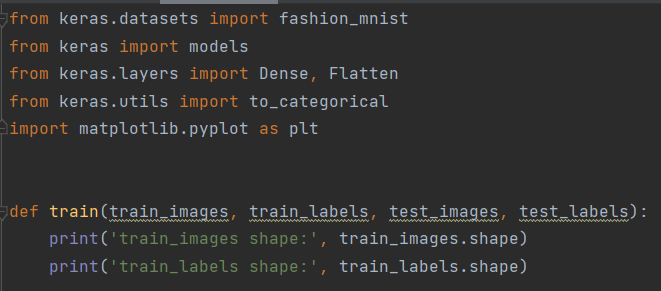


Рисунок 29 – Виведення розмірів даних

Бачимо, що вхідні дані – це картинка 28\*28 пікселі. Маємо 60000 навчальних значень.

Далі переглядаю перші 25 зображень набору даних: візуалізую відповідні дані і підписую їх відповідними назвами, для чого використовую створений список.

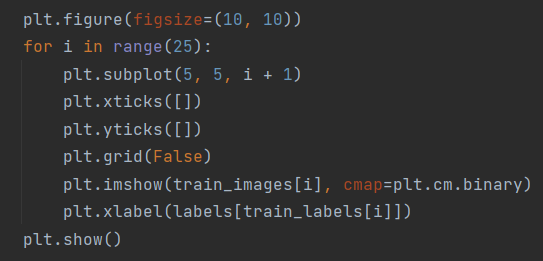


Рисунок 30 – Код для візуалізації набору даних



Рисунок 31 – Візуалізація набору даних

Бачимо, що зображуються дуже різні речі.

Наступним кроком нормалізую дані і завантажую мітки, тобто масиви з відгуками на зображення.

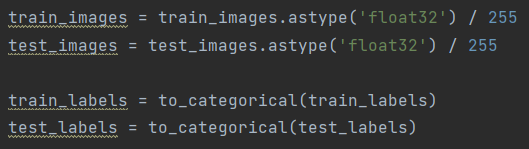


Рисунок 32 – Нормалізація й завантаження міток

Створюю модель. Я використовую клас Sequential. Спочатку я використовую Flatten для того, щоб відійти від багатовимірного масиву до одновимірного і застосувати Dense. Відповідно ще один шар – це Dense з 512 нейронами. І заключним шаром маємо Dense з 10 нейронами, бо на вихід маємо 10 варіантів.

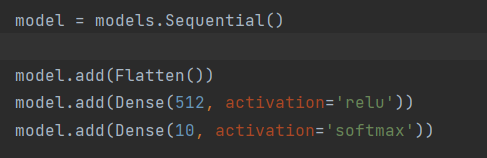


Рисунок 33 – Створення моделі

Компілюю модель і завантажую навчальні дані. Вибір параметрів такий, як і в попередньому пункті, адже маємо багато класів.

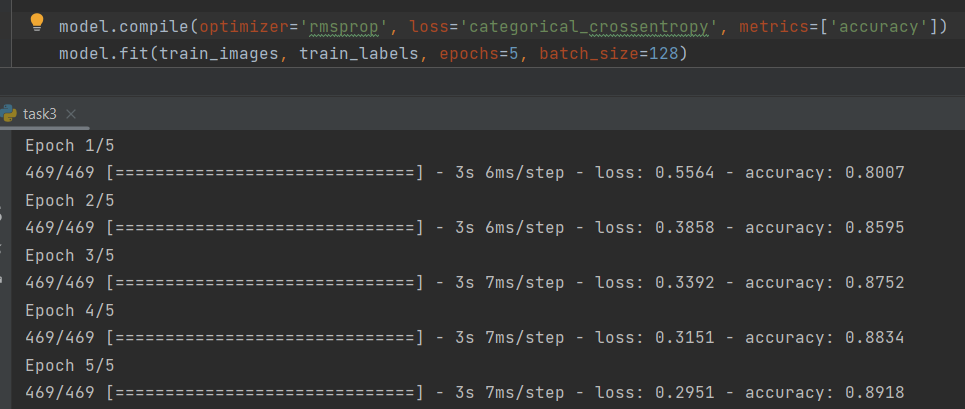


Рисунок 34 – Компіляція й завантаження навчальних даних

На навчальних даних ми досягли точности 0.8918. Після навчання перевіряю модель на контрольному наборі даних і зберігаю модель.

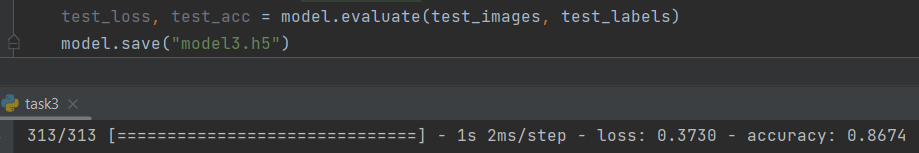


Рисунок 35 – Перевірка моделі на тестових даних

На тестових даних було досягнуто точности 0.8674. Модель працює ефективно.

Тепер попрацюю із моделлю. Завантажую попередньо збережену модель.

Аналогічно до попереднього пункту, обираю порядковий номер зображення з тестових даних, переформатовую його під формат вхідних даних і роблю передбачення за допомогою model.predict(). Далі візуалізую зображення і підписую так, як передбачила мережа, а також так, як воно є насправді.

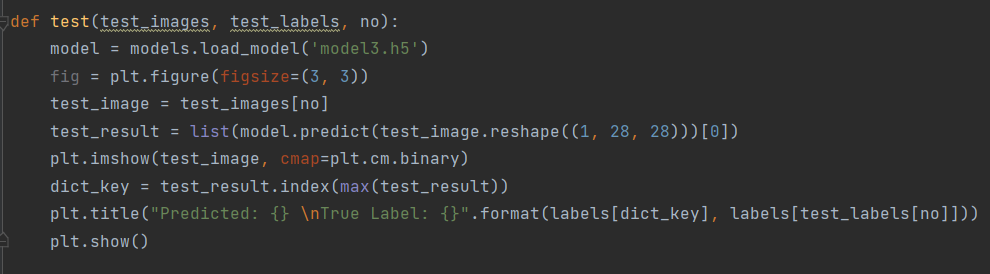


Рисунок 36 – Код до завантаження та тестування моделі

Обираю три випадкові зображення, наприклад, за номерами 100-102.

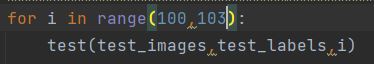


Рисунок 37 – Тестування моделі

Отримуємо такі результати.

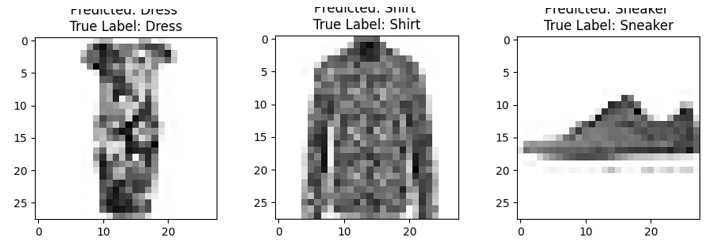


Рисунок 38 – Отримані результати передбачення

Бачимо, що у всіх трьох випадках передбачення було правильним. Тож можна зробити висновок, що модель дійсно працює доволі ефективно.

**Висновок.**

Отже, у цій роботі я навчився проєктувати та навчати штучні нейронні мережi для задач класифiкацiї.

У результаті лабораторної роботи було виконано завдання зі створення моделі для розпiзнавання рукописних цифр. У графiчному редакторi Paint було створено файли з рукописними цифрами. У результаті мережа розпізнала усі цифри коректно, тож можна стверджувати, що розроблена мережа працює ефективно. Також було спроєктовано й розроблено нейронні мережі на основi наборiв даних Cifar10 та FMNIST. Було розроблено архiтектуру нейронної мережi, навчено її на тестових прикладах i продемонструвано її роботу на кiлькох зображеннях. Отримали, що мережа на основі даних Cifar10 працює з ефективністю майже 59%, а на основі FMNIST – майже 87%. Тобто обидві моделі у більшості випадків здатні розпізнавати потрібні дані.

Використовуючи програмний засіб PyCharm, мову програмування Python з модулем Keras, отримуємо коректний результат.