МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ

ТЕХНОЛОГІЙ

ІНДИВІДУАЛЬНА РОБОТА #4

З дисципліни

«Штучний інтелект»

Виконав:

Студент групи ПД-44

Солов'ян Арсен

Київ – 2025

Мета роботи: вивчення принципів роботи згорткових нейронних мереж (CNN) та їх реалізація на мові програмування Python для розпізнавання
зображень; розробку та тренування згорткової нейронної мережі на основі бібліотеки Keras в Python; застосування мережі для розв'язання задач класифікації та обробки зображень; оцінка ефективності моделі за допомогою
відповідних метрик, з врахуванням точності, функції втрат та інші показників
якості.

Спочатку завантажено набір даних рукописних цифр MNIST, що містить тренувальну та тестову вибірки. Значення пікселів зображень нормалізовано до діапазону від 0 до 1 шляхом ділення на 255; це важливо для стабілізації та прискорення навчання нейронних мереж, оскільки забезпечує схожий масштаб для всіх вхідних ознак. Мітки класів перетворено у категоріальний формат (one-hot encoding).

```
Крок 1: Завантаження та підготовка даних MNIST
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Розмір тренувальних даних: (60000, 28, 28, 1)
Розмір тестових даних: (10000, 28, 28, 1)
Нормалізація та перетворення міток завершено.

Крок 2: Побудова архітектури CNN
Model: "sequential"
```

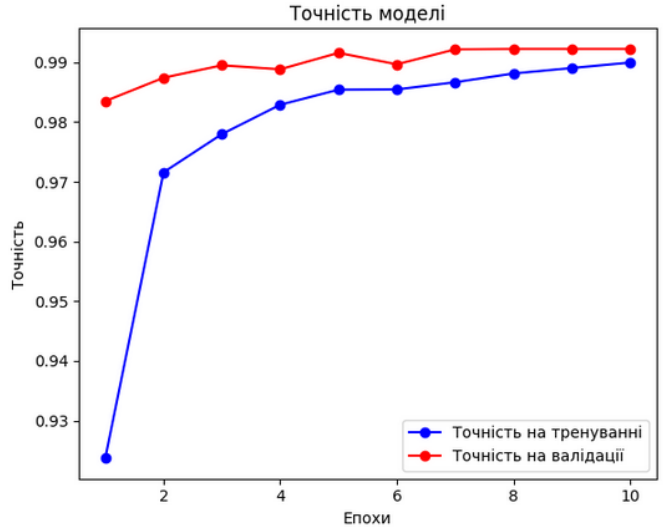
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout (Dropout)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 128)	204,928
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

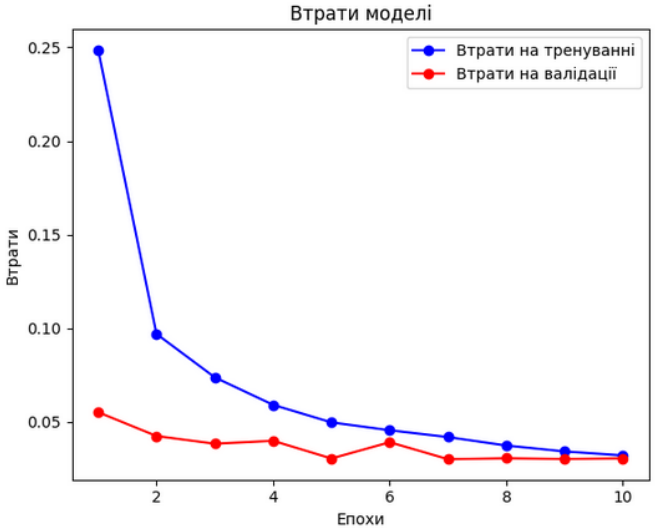
```
Total params: 225,034 (879.04 KB)
Trainable params: 225,034 (879.04 KB)
Non-trainable params: 0 (0.00 B)
Модель CNN створено.
```

Побудовано архітектуру згорткової нейронної мережі (CNN) з використанням Keras Sequential API. Вона включає два згорткових шари з активацією ReLU та фільтрами 3x3 (32 і 64 фільтри відповідно), після кожного з яких іде шар максимального підсемплювання (MaxPooling 2x2) для зменшення розмірності карт ознак та підвищення інваріантності до зсувів. Перший шар Dropout (0.25) додано після другого шару підсемплювання для регуляризації, зменшуючи перенавчання шляхом випадкового обнулення частини виходів шару. Далі йде шар Flatten для перетворення 2D карт ознак у 1D вектор, повнозв'язний шар (Dense) зі 128 нейронами та активацією ReLU для вивчення комбінацій виявлених ознак, другий шар Dropout (0.5) для подальшої регуляризації повнозв'язного шару, та вихідний повнозв'язний шар з 10 нейронами (по одному на клас цифри) та активацією Softmax для отримання ймовірностей приналежності до кожного класу. Згорткові шари виявляють локальні патерни, MaxPooling зменшує розмірність, зберігаючи важливу інформацію, Flatten готує дані для повнозв'язних шарів, Dense шари виконують класифікацію на основі вивчених ознак, Dropout запобігає перенавчанню, а Softmax видає ймовірнісний розподіл по класах.

```
Крок 3: Навчання моделі
Початок навчання...
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 63s 36ms/step - accuracy: 0.8422 - loss: 0.4962 - val_accuracy: 0.9835 - val_loss: 0.0552
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 76s 32ms/step - accuracy: 0.9683 - loss: 0.1059 - val_accuracy: 0.9874 - val_loss: 0.0424
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 52s 34ms/step - accuracy: 0.9787 - loss: 0.0710 - val_accuracy: 0.9895 - val_loss: 0.0383
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 81s 34ms/step - accuracy: 0.9827 - loss: 0.0583 - val_accuracy: 0.9888 - val_loss: 0.0399
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 81s 33ms/step - accuracy: 0.9858 - loss: 0.0475 - val_accuracy: 0.9916 - val_loss: 0.0304
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 48s 32ms/step - accuracy: 0.9848 - loss: 0.0454 - val_accuracy: 0.9897 - val_loss: 0.0391
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 84s 34ms/step - accuracy: 0.9873 - loss: 0.0400 - val_accuracy: 0.9922 - val_loss: 0.0301
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 80s 32ms/step - accuracy: 0.9885 - loss: 0.0345 - val_accuracy: 0.9923 - val_loss: 0.0306
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 84s 34ms/step - accuracy: 0.9893 - loss: 0.0326 - val_accuracy: 0.9923 - val_loss: 0.0302
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 81s 33ms/step - accuracy: 0.9900 - loss: 0.0322 - val_accuracy: 0.9923 - val_loss: 0.0305
Навчання завершено.
```

```
Крок 4: Візуалізація результатів навчання
```





Модель скомпільовано з оптимізатором Adam, функцією втрат categorical_crossentropy, яка підходить для багатокласової класифікації з one-hot мітками, та метрикою точності. Навчання проведено протягом 10 епох з розміром батчу 32, використовуючи 20% тренувальних даних для валідації під час навчання. Історію навчання, що включає значення втрат і точності на тренувальних та валідаційних даних для кожної епохи, збережено.

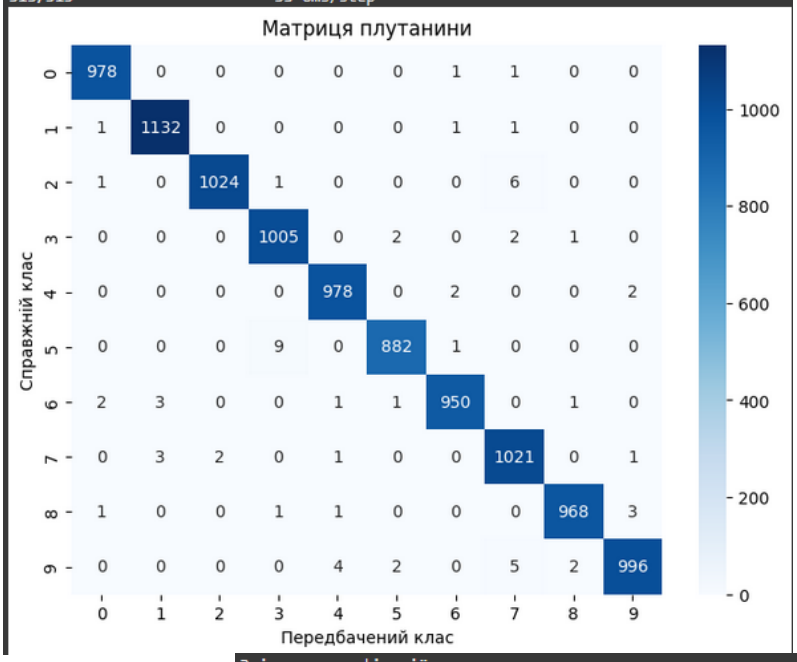
Побудовано графіки зміни точності та втрат протягом епох навчання для тренувальної та валідаційної вибірок. Аналіз цих графіків дозволяє оцінити процес навчання: зростання точності та зменшення втрат на обох вибірках свідчить про успішне навчання, тоді як значне розходження між кривими тренування та валідації (особливо якщо валідаційні втрати починають зростати) вказує на перенавчання моделі.

Проведено оцінку навченої моделі на відкладеній тестовій вибірці, визначено фінальну точність та втрати. Обчислено та візуалізовано матрицю плутанини, яка показує кількість правильних та неправильних класифікацій для кожного класу. Це дозволяє ідентифікувати цифри, які модель найчастіше плутає між собою (наприклад, 4 і 9, або 3 і 5), що вказує на найскладніші для розпізнавання класи. Шляхи покращення можуть включати збір більше даних для складних класів, використання аугментації, зміну архітектури або гіперпараметрів.

Створено функцію для прогнозування цифри на окремому зображенні. Функція приймає зображення, перетворює його у потрібний формат (28x28x1, нормалізоване) та повертає передбачену моделью цифру. Роботу функції продемонстровано на кількох випадкових зображеннях з тестового набору, візуалізуючи саме зображення, його справжню мітку та мітку, передбачену моделлю.

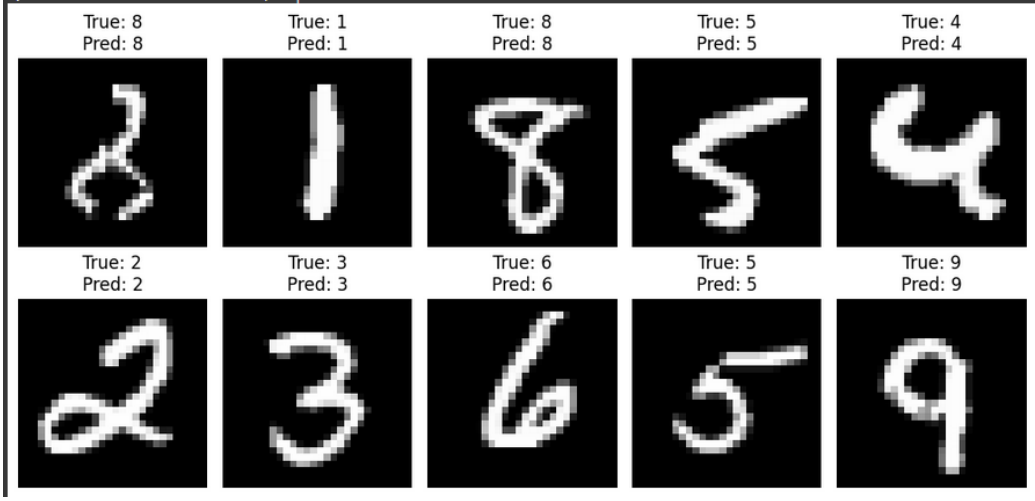
Для покращення моделі застосовано техніку аугментації даних за допомогою ImageDataGenerator, що включає випадкові зсуви, обертання та масштабування тренувальних зображень. Це штучно збільшує розмір та різноманітність тренувального набору, допомагаючи моделі стати більш стійкою до варіацій у даних та зменшуючи перенавчання. Проведено повторне навчання моделі на аугментованих даних та оцінено її точність на тій самій тестовій вибірці. Як правило, аугментація покращує точність моделі на тестових даних, хоча може вимагати більшої кількості епох для досягнення оптимальних результатів.

```
Крок 5: Оцінка моделі на тестових даних
Тестові втрати: 0.0229
Тестова точність: 0.9934
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step
```



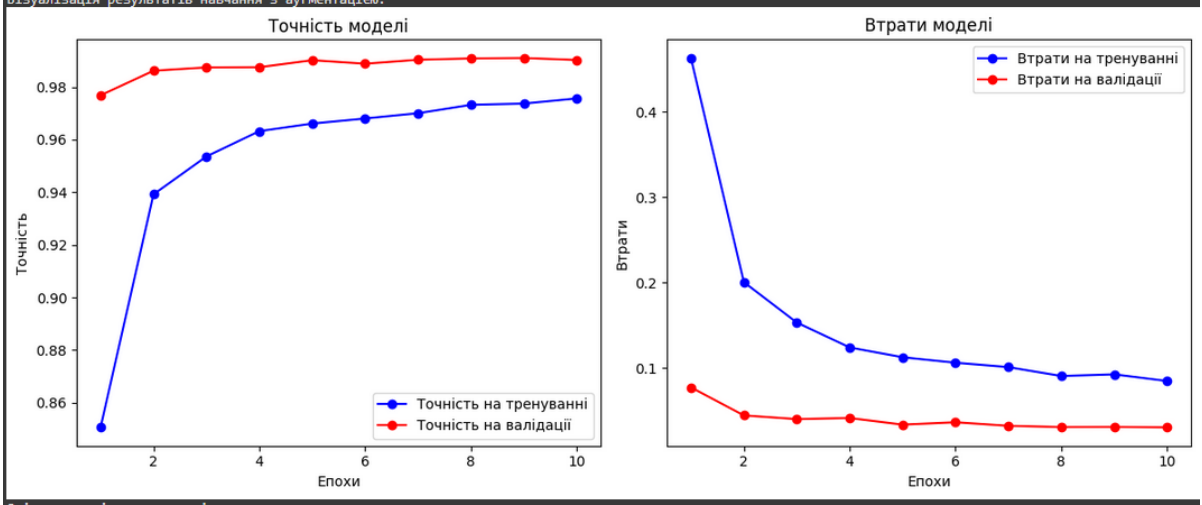

```
Звіт по класифікації:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       980
           1       0.99      1.00      1.00      1135
           2       1.00      0.99      1.00      1032
           3       0.99      1.00      0.99      1010
           4       0.99      1.00      0.99       982
           5       0.99      0.99      0.99       892
           6       0.99      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       1.00      0.99      0.99       974
           9       0.99      0.99      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

```
Крок 6: Прогноз на тестових даних
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
```



Проведено експеримент зі зміною архітектури шляхом модифікації функції побудови моделі для зміни кількості фільтрів у згорткових шарах. Зазначено, що зміна архітектури (наприклад, збільшення кількості фільтрів або шарів) може потенційно покращити точність за рахунок збільшення ємності моделі, але також підвищує ризик перенавчання та обчислювальну складність.

```
Крок 7: Покращення моделі (Аугментація даних)
Початок навчання з аугментацією...
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `Py
  self._warn_if_super_not_called()
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 72s 46ms/step - accuracy: 0.7273 - loss: 0.8053 - val_accuracy: 0.9768 - val_loss: 0.0771
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 78s 44ms/step - accuracy: 0.9326 - loss: 0.2209 - val_accuracy: 0.9862 - val_loss: 0.0442
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 81s 43ms/step - accuracy: 0.9517 - loss: 0.1606 - val_accuracy: 0.9874 - val_loss: 0.0399
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 65s 43ms/step - accuracy: 0.9643 - loss: 0.1206 - val_accuracy: 0.9875 - val_loss: 0.0412
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 65s 44ms/step - accuracy: 0.9651 - loss: 0.1185 - val_accuracy: 0.9902 - val_loss: 0.0334
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 82s 44ms/step - accuracy: 0.9674 - loss: 0.1073 - val_accuracy: 0.9888 - val_loss: 0.0362
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 66s 44ms/step - accuracy: 0.9700 - loss: 0.1017 - val_accuracy: 0.9903 - val_loss: 0.0320
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 66s 44ms/step - accuracy: 0.9741 - loss: 0.0886 - val_accuracy: 0.9908 - val_loss: 0.0306
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 66s 44ms/step - accuracy: 0.9728 - loss: 0.0936 - val_accuracy: 0.9910 - val_loss: 0.0308
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 65s 43ms/step - accuracy: 0.9749 - loss: 0.0847 - val_accuracy: 0.9902 - val_loss: 0.0302
Навчання з аугментацією завершено.
Візуалізація результатів навчання з аугментацією:
```




```
Оцінка моделі з аугментацією на тестових даних:
Тестові втрати (аугментація): 0.0211
Тестова точність (аугментація): 0.9923
Порівняння точності: Базова=0.9934, Аугментована=0.9923
Крок 8: Експерименти з архітектурою
Приклад: Збільшення кількості фільтрів.
Створено модель зі збільшеною кількістю фільтрів.
Для порівняння потрібно провести повне навчання цієї моделі (пропускається для швидкості).
Запустіть model_more_filters.fit(...) та model_more_filters.evaluate(...) для отримання результатів.
Наприклад, точність може зрости, але ризик перенавчання та час навчання також збільшаться.

Крок 9: Збереження та завантаження моделі
Модель збережено у файл: mnist_cnn_model.keras
Модель завантажено.
Перевірка завантаженої моделі на тестових даних:
Тестові втрати (завантажена): 0.0211
Тестова точність (завантажена): 0.9923
Приклад передбачення завантаженою моделлю:
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
Випадкове зображення: Справжня мітка=7, Передбачена мітка=7
```