## **Державний університет інформаційно-комунікаційних технологій**

**Індивідуальна робота №3**

**з дисципліни: Технології штучного інтелекту**

**Тема: Глибоке навчання Deep Learning: основи згорткових нейронних мереж**

Виконав:  
студент ДУІКТ  
Тертишний В.Ю.

група: ШІДМ-51

м.Київ

**Мета:** 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.

**Код №1:**

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense

from tensorflow.keras.utils import to\_categorical

from sklearn.metrics import confusion\_matrix

import seaborn as sns

# Завантаження та підготовка даних

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

x\_train, x\_test = x\_train / 255.0, x\_test / 255.0 # Нормалізація даних

x\_train = x\_train.reshape((-1, 28, 28, 1))

x\_test = x\_test.reshape((-1, 28, 28, 1))

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# Архітектура CNN

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D((2, 2)),

Dropout(0.25),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(10, activation='softmax')

])

# Компіляція моделі

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Навчання моделі

history = model.fit(x\_train, y\_train, epochs=10, batch\_size=32, validation\_split=0.2)

# Візуалізація результатів навчання

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.title('Training and Validation Loss')

plt.show()

# Оцінка моделі на тестових даних

test\_loss, test\_accuracy = model.evaluate(x\_test, y\_test)

print(f"Test Accuracy: {test\_accuracy \* 100:.2f}%")

# Матриця плутанини

y\_pred = np.argmax(model.predict(x\_test), axis=1)

y\_true = np.argmax(y\_test, axis=1)

conf\_matrix = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(10, 8))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.title("Confusion Matrix")

plt.show()

# Функція для прогнозу та візуалізації

def predict\_digit(image):

image = image.reshape(1, 28, 28, 1)

pred = model.predict(image)

return np.argmax(pred)

# Приклад прогнозування

samples = np.random.choice(len(x\_test), 5, replace=False)

for idx in samples:

plt.imshow(x\_test[idx].reshape(28, 28), cmap='gray')

plt.title(f"Predicted: {predict\_digit(x\_test[idx])}")

plt.show()

# Збереження моделі

model.save("mnist\_cnn\_model.h5")

print("Модель збережена як 'mnist\_cnn\_model.h5'")

**Робота коду №1:**   
(**>Скріни сюди<**)

**Код №2:**

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, UpSampling2D

from tensorflow.keras.datasets import cifar10

from skimage.color import rgb2lab, lab2rgb

# Завантаження CIFAR-10 та підготовка даних

(x\_train, \_), (x\_test, \_) = cifar10.load\_data()

x\_train, x\_test = x\_train[:5000] / 255.0, x\_test[:1000] / 255.0 # Використання меншого набору

# Перетворення у простір Lab

def batch\_rgb2lab(batch):

return np.array([rgb2lab(img) for img in batch])

x\_train\_lab = batch\_rgb2lab(x\_train)

x\_test\_lab = batch\_rgb2lab(x\_test)

x\_train\_L = x\_train\_lab[..., 0:1] / 100

y\_train\_ab = (x\_train\_lab[..., 1:] + 128) / 255

x\_test\_L = x\_test\_lab[..., 0:1] / 100

y\_test\_ab = (x\_test\_lab[..., 1:] + 128) / 255

# Архітектура CNN для розфарбовування

color\_model = Sequential([

Conv2D(32, (3, 3), activation='relu', padding='same', input\_shape=(32, 32, 1)),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu', padding='same'),

MaxPooling2D((2, 2)),

Conv2D(128, (3, 3), activation='relu', padding='same'),

UpSampling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu', padding='same'),

UpSampling2D((2, 2)),

Conv2D(2, (3, 3), activation='tanh', padding='same')

])

# Компіляція моделі

color\_model.compile(optimizer='adam', loss='mse')

# Навчання моделі

color\_history = color\_model.fit(

x\_train\_L, y\_train\_ab,

epochs=3, # Менше епох

batch\_size=32,

validation\_split=0.1

)

# Тестування на випадковому зображенні

def predict\_and\_display(image\_L):

pred\_ab = color\_model.predict(image\_L.reshape(1, 32, 32, 1))[0]

lab\_image = np.concatenate((image\_L \* 100, pred\_ab \* 255 - 128), axis=-1)

rgb\_image = lab2rgb(lab\_image)

return rgb\_image

idx = np.random.randint(0, len(x\_test\_L))

predicted\_image = predict\_and\_display(x\_test\_L[idx])

# Візуалізація результатів

plt.figure(figsize=(8, 4))

plt.subplot(1, 2, 1)

plt.imshow(x\_test\_L[idx].reshape(32, 32), cmap='gray')

plt.title("Original Grayscale")

plt.axis("off")

plt.subplot(1, 2, 2)

plt.imshow(predicted\_image)

plt.title("Colorized")

plt.axis("off")

plt.tight\_layout()

plt.show()

**Робота коду №2:**   
(**>Скріни сюди<**)

**Висновки**

У процесі виконання роботи було досягнуто поставлену мету, а саме – вивчено принципи роботи згорткових нейронних мереж (CNN) та їх реалізація на мові програмування Python для розпізнавання зображень. Було розроблено та натреновано модель згорткової нейронної мережі за допомогою бібліотеки Keras, що дозволило застосувати її для розв’язання задач класифікації та обробки зображень. Для класифікації було використано набір даних MNIST для розпізнавання рукописних цифр, а для обробки зображень – набір CIFAR-10 для розфарбовування чорно-білих зображень. Модель продемонструвала високу точність на тестових даних, що підтверджує її ефективність у задачах класифікації. Оцінка ефективності моделі була проведена з урахуванням метрик точності та функції втрат. Результати навчання показали стабільне зниження втрат на тренувальних та валідаційних даних, що свідчить про успішне навчання та узагальнення моделі. У задачі обробки зображень модель також показала задовільні результати, що підтверджує можливість використання CNN для реконструкції та колоризації зображень.