Міністерство освіти і науки України

Національний університет «Одеська політехніка»

Кафедра комп’ютеризованих систем та програмних технологій

Протокол лабораторної роботи №5

з дисципліни «Інтелектуальні системи»

на тему «ПОБУДОВА ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ»

Виконав:

студент гр. АТ-191

Кинєв Є. О.

Перевірив:

д.т.н., проф. Фомiн О.О.

Одеса-2023

**Мета роботи:** отримання практичних навичок побудови моделей у вигляді згорткових нейронних мереж в задачах класифікації.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Keras (keras.io)

**ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ**

Для датасета <https://www.cs.toronto.edu/~kriz/cifar.html> згідно варіанту завдання побудувати модель машинного навчання. Підібрати архітектуру нейронної мережі, що забезпечує достовірність класифікації не менш 90%.

Варіанти завдання:

1. airplane / automobile / ship
2. horse / cat / dog
3. frog / bird / cat
4. truck / automobile / ship
5. deer / horse / dog
6. bird / deer / airplane
7. cat / dog / bird
8. ship / truck / dog
9. automobile / dog / deer
10. dog / frog / bird

**ХІД РОБОТИ**

1. Визначити постановку задачі класифікації для датасета згідно варіанту завдання.

2. Побудувати модель у вигляді згорткової нейронної мережі для задачі класифікації. Аргументовано підібрати кількість шарів, кількість нейронів в шарах, оптимізатор активаційні функції та функції втрат для якісного вирішення кожної задачі.

3. Візуалізувати отриману модель (функція plot\_model).

4. Візуалізувати процес навчання нейронної мережі.

5. Визначити показники якості збудованої моделі;

6. Зробити висновки по роботі.

# Встановлення необхідних бібліотек

import tensorflow as tf

from keras import layers, models

from keras.optimizers.legacy import Adam

from keras.losses import SparseCategoricalCrossentropy

from keras.metrics import SparseCategoricalAccuracy

from keras.utils import to\_categorical

from keras.callbacks import ModelCheckpoint, EarlyStopping

import matplotlib.pyplot as plt

from keras.datasets import cifar10

import numpy as np

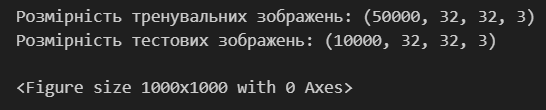
# Завантаження даних

(train\_images, train\_labels), (test\_images, test\_labels) = cifar10.load\_data()

# Перевірка розмірності даних

print(f"Розмірність тренувальних зображень: {train\_images.shape}")

print(f"Розмірність тестових зображень: {test\_images.shape}")



selected\_classes=[5,6,2]# dog / frog / bird

train\_mask= np.isin(train\_labels.flatten(), selected\_classes)

test\_mask=np.isin(test\_labels.flatten (), selected\_classes)

#Відфільтруємо дані

train\_images, train\_labels = train\_images[train\_mask],train\_labels [train\_mask]

test\_images, test\_labels = test\_images[test\_mask], test\_labels[test\_mask]

#we are doing normalization

train\_images, test\_images=train\_images/255.0, test\_images/255.0

import matplotlib.pyplot as plt

class\_names = ['dog', 'frog', 'bird']

#Візуалізація перших 9 зображень

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

class\_index = 0

for i in range (9):

    while class\_index>= len(class\_names):

        class\_index = 0 # Починаємо спочатку, якщо всі класи вже відображені

    current\_class = selected\_classes[class\_index]

    class\_images = train\_images[train\_labels.flatten() == current\_class]

    plt.subplot (3, 3, i + 1)

    plt.xticks([])

    plt.yticks([])

    plt.grid (False)

    plt.imshow(class\_images[i])

    plt.xlabel(class\_names[class\_index])

    class\_index += 1

    plt.show()

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(len(class\_names), activation='softmax'))

model.summary()

#Компіляція моделі

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

#Перетворення міток

class\_mapping={5:0, 6:1, 2:2}

train\_labels\_mapped = np.vectorize(class\_mapping.get) (train\_labels)

test\_labels\_mapped = np.vectorize(class\_mapping.get) (test\_labels)

checkpoint = ModelCheckpoint("best\_model.h5", monitor="val\_loss", save\_best\_only=True)

# Зупинка тренування, якщо немає поліпшень протягом 5 епох

early\_stopping = EarlyStopping(monitor="val\_loss", patience=5)

history = model.fit(train\_images, train\_labels\_mapped, epochs=10, validation\_data = (test\_images, test\_labels\_mapped),

                    callbacks=[checkpoint, early\_stopping])

# Візуалізація процесу тренування

plt.plot(history.history['loss'], label='Тренувальна втрата')

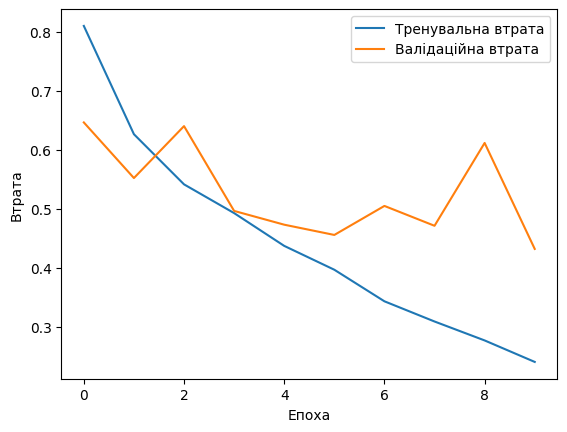
plt.plot(history.history['val\_loss'], label='Валідаційна втрата')

plt.xlabel('Епоха')

plt.ylabel('Втрата')

plt.legend()

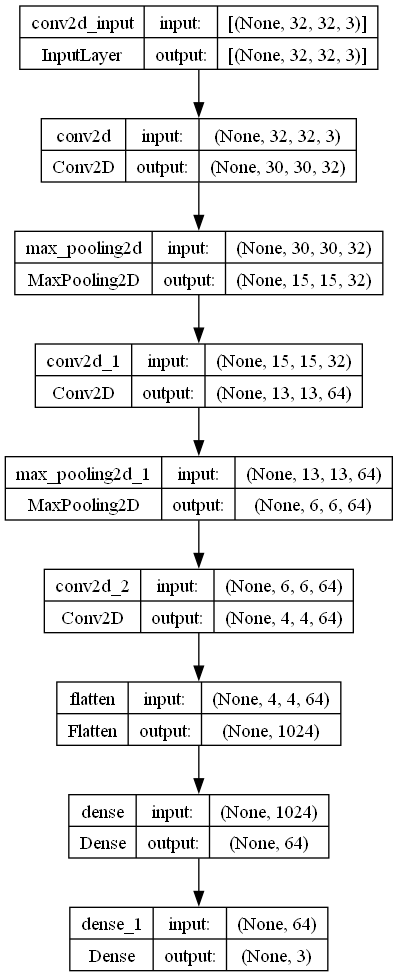
plt.show()



import matplotlib.pyplot as plt

from keras.utils import plot\_model

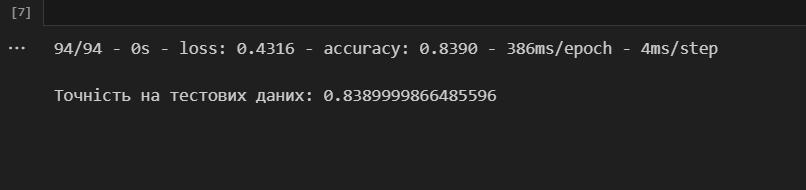
plot\_model(model, to\_file='model\_plot.png', show\_shapes=True, show\_layer\_names=True)



# Визначення показників якості

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels\_mapped, verbose=2)

print(f'\nТочність на тестових даних: {test\_acc}')



# Отримання передбачень для тестових зображень

predictions = model.predict(test\_images)

# Функція для отримання індексу максимального значення у векторі ймовірностей

def get\_predicted\_class(probabilities):

    return np.argmax(probabilities)

# Візуалізація перших 9 зображень та їх передбачених класів

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

for i in range(9):

    plt.subplot(3, 3, i + 1)

    plt.xticks([])

    plt.yticks([])

    plt.grid(False)

    # Отримання передбаченого та правильного класів для поточного зображення

    predicted\_class = get\_predicted\_class(predictions[i])

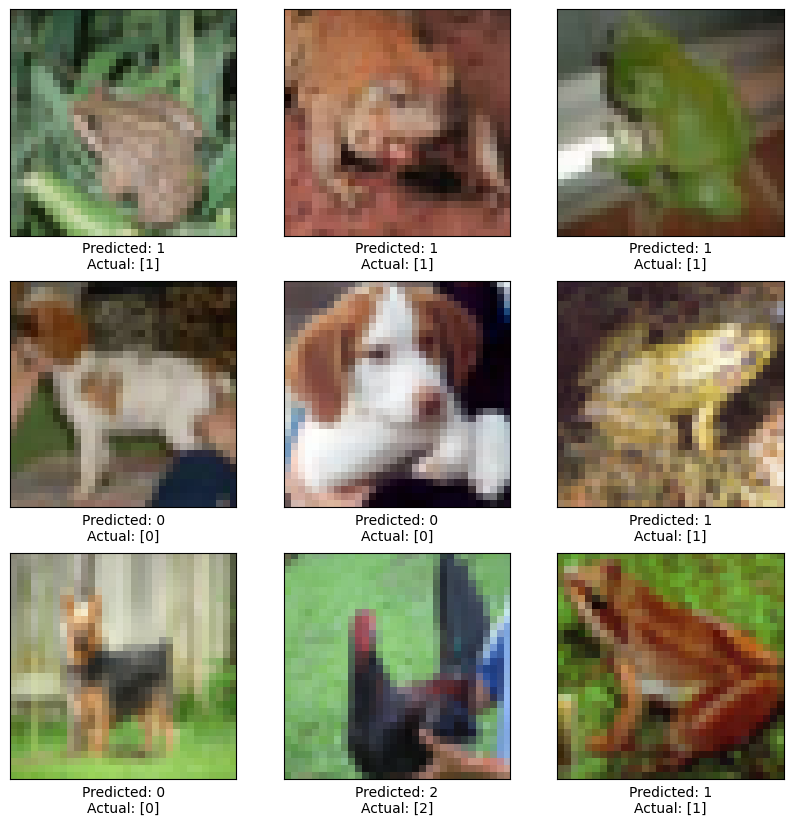
    actual\_class = test\_labels\_mapped[i]

    # Відображення зображення та передбаченого класу

    plt.imshow(test\_images[i])

    plt.xlabel(f'Predicted: {predicted\_class}\nActual: {actual\_class}')

plt.show()



**Висновок:**

У ході виконання лабораторної роботи була проведена практична реалізація побудови та оптимізації згорткових нейронних мереж для задач класифікації на прикладі вибраного датасету зображень тварин. Вибрали варіант "dog / frog / bird" для класифікації зображень собак, жаб та птахів. Перенавчання та використання валідаційних даних допомагають у підтримці стабільності моделі. Експерименти з різними конфігураціями дозволяють здобути навички у виборі оптимальних параметрів. Отримані практичні навички з побудови згорткових нейронних мереж покращать розуміння та навички в галузі машинного навчання та комп'ютерного зору.