НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Проектування та дослідження систем із штучним інтелектом

ЗВІТ

до лабораторної роботи №5

Виконав ІП-41мн Пашковський Євгеній Сергійович

Київ 2025

Завдання

Написати програму що реалізує згорткову нейронну мережу Inception V3 для розпізнавання об’єктів на зображеннях. Створити власний дата сет з папки на диску, навчити нейронну мережу на цьому датасеті розпізнавати породу Вашої улюбленої собаки чи кота. Навчену нейронну мережу зберегти на комп’ютер та написати програму, що відкриває та аналізує зображення.

Хід роботи

Код програми

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from keras import utils

dataset_path = "./dataset"


ds  =  utils.image_dataset_from_directory("./dataset",  image_size=(299,
299), interpolation="lanczos5", seed=12151251, batch_size=None)
class_names = ds.class_names

ds_count = sum(1 for _ in ds)

train_size = 0.7
valid_size = 0.2

train_count = int(ds_count * train_size)
valid_count = int(ds_count * valid_size)
test_count = int(ds_count - train_count - valid_count)


print(f"Split: {train_count}, {valid_count}, {test_count}")

train_ds = ds.take(train_count).batch(32)
valid_ds = ds.skip(train_count).take(valid_count).batch(32)
test_ds = ds.skip(train_count + valid_count).batch(32)

for i in range(len(class_names)):
  class_names[i] = class_names[i].split("-")[-1]
```

```
    plt.figure(figsize=(10, 10))
    for images, labels in train_ds.take(1):
      for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

```
    import tensorflow as tf

    target_classname = "Walker_hound"
    assert target_classname in class_names

    target_index = class_names.index(target_classname)

    train_ds = train_ds.map(lambda x, y: (x, tf.cast(tf.equal(y,
target_index), tf.int32)))
    valid_ds = valid_ds.map(lambda x, y: (x, tf.cast(tf.equal(y,
target_index), tf.int32)))
    test_ds = test_ds.map(lambda x, y: (x, tf.cast(tf.equal(y,
target_index), tf.int32)))

    class_names = ['Other', target_classname]

    plt.figure(figsize=(10, 10))
    i = 0
    for img, label in train_ds.unbatch().filter(lambda x, y: tf.equal(y,
1)).take(3):
      ax = plt.subplot(3, 3, i + 1)
      plt.imshow(img.numpy().astype("uint8"))
      plt.title(class_names[label])
      plt.axis("off")
      i += 1

    for img, label in train_ds.unbatch().filter(lambda x, y: tf.equal(y,
0)).take(6):
      ax = plt.subplot(3, 3, i + 1)
      plt.imshow(img.numpy().astype("uint8"))
      plt.title(class_names[label])
      plt.axis("off")
```

```
    i += 1
```

```
from keras import models, layers, applications


def MyNet(input_shape=(299, 299, 3)):
    inputs = layers.Input(shape=input_shape)
    x = layers.Resizing(299, 299)(inputs)
    x = layers.Rescaling(1./255)(x)
    x = layers.RandomColorJitter(
      value_range=(0, 1),
      brightness_factor=0.2,
      contrast_factor=0.2,
      saturation_factor=0.2,
      hue_factor=0.2
    )(x)
    x = layers.RandomFlip(
      mode="horizontal_and_vertical"
    )(x)

        inceptionv3 = applications.InceptionV3(include_top=False,
input_tensor=x)

    x = inceptionv3.output
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(4096, activation="relu")(x)
    outputs = layers.Dense(1, activation="sigmoid")(x)

    model = models.Model(inputs=inputs, outputs=outputs)

    return model, inceptionv3
```

```
from keras import optimizers

epochs = 10

initial_learning_rate = 1e-3
final_learning_rate = 1e-6
```

```
    learning_rate_decay_factor = (final_learning_rate / initial_learning_rate) ** (1 / epochs)

    steps_per_epoch = train_count

    learning_rate = optimizers.schedules.ExponentialDecay(
      initial_learning_rate=initial_learning_rate,
      decay_steps=steps_per_epoch,
      decay_rate=learning_rate_decay_factor
    )

    model, base_model = MyNet(input_shape=(299, 299, 3))

    model.compile(optimizer=optimizers.Adam(learning_rate=learning_rate), loss='binary_crossentropy', metrics=['accuracy'])

    model.summary()
```

```
    for layer in base_model.layers:
      layer.trainable = False

    class_weight = {0: 1, 1: 20}

    history = model.fit(train_ds, epochs=epochs, validation_data=valid_ds, class_weight=class_weight)
```

```
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.xlabel('Епохи')
    plt.ylabel('Функція втрат')
    plt.title('Графік функції втрат')
    plt.legend()
    plt.grid()
    plt.show()

    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.xlabel('Епохи')
    plt.ylabel('Точність')
    plt.title('Графік точності')
    plt.legend()
```

```
    plt.grid()
    plt.show()
```

```
    model.evaluate(test_ds)
```

```
    y_true = []
    y_pred = []

    for x_batch, y_batch in test_ds:
        predictions = model.predict(x_batch)
        pred_labels = np.array(predictions).round()
        y_true.extend(y_batch.numpy())
        y_pred.extend(pred_labels)

    y_pred
```

```
    from sklearn.metrics import confusion_matrix

    conf_matrix = confusion_matrix(y_true, y_pred)
    fig, ax = plt.subplots(figsize=(30, 8))
    cax = ax.matshow(conf_matrix, cmap="Blues")

    plt.colorbar(cax)

    class_labels=range(len(class_names))

    ax.set_xticks(range(len(class_labels)))
    ax.set_yticks(range(len(class_labels)))
    ax.set_xticklabels(class_labels, fontsize=7)
    ax.set_yticklabels(class_labels, fontsize=7)

    plt.title("Confusion Matrix", pad=20)
    plt.xlabel("Predicted", labelpad=10)
    plt.ylabel("Actual", labelpad=10)

    for (i, j), value in np.ndenumerate(conf_matrix):
         ax.text(j, i, f'{value}', ha='center', va='center', color='black',
fontsize=7)

    plt.tight_layout()
    plt.show()
```

```
model.save("./model.keras")
```

```
import keras

model = keras.models.load_model("./model.keras")
```

```
from PIL import Image
import numpy as np


imgs = []

trueImg = np.array(Image.open("./test/true.jpg").resize((299, 299), resample=Image.Resampling.LANCZOS))
falseImg = np.array(Image.open("./test/false.jpg").resize((299, 299), resample=Image.Resampling.LANCZOS))

imgs.append(trueImg)
imgs.append(falseImg)

imgs = np.array(imgs)

np.array(model.predict(imgs)).round()
```

Отриманий результат

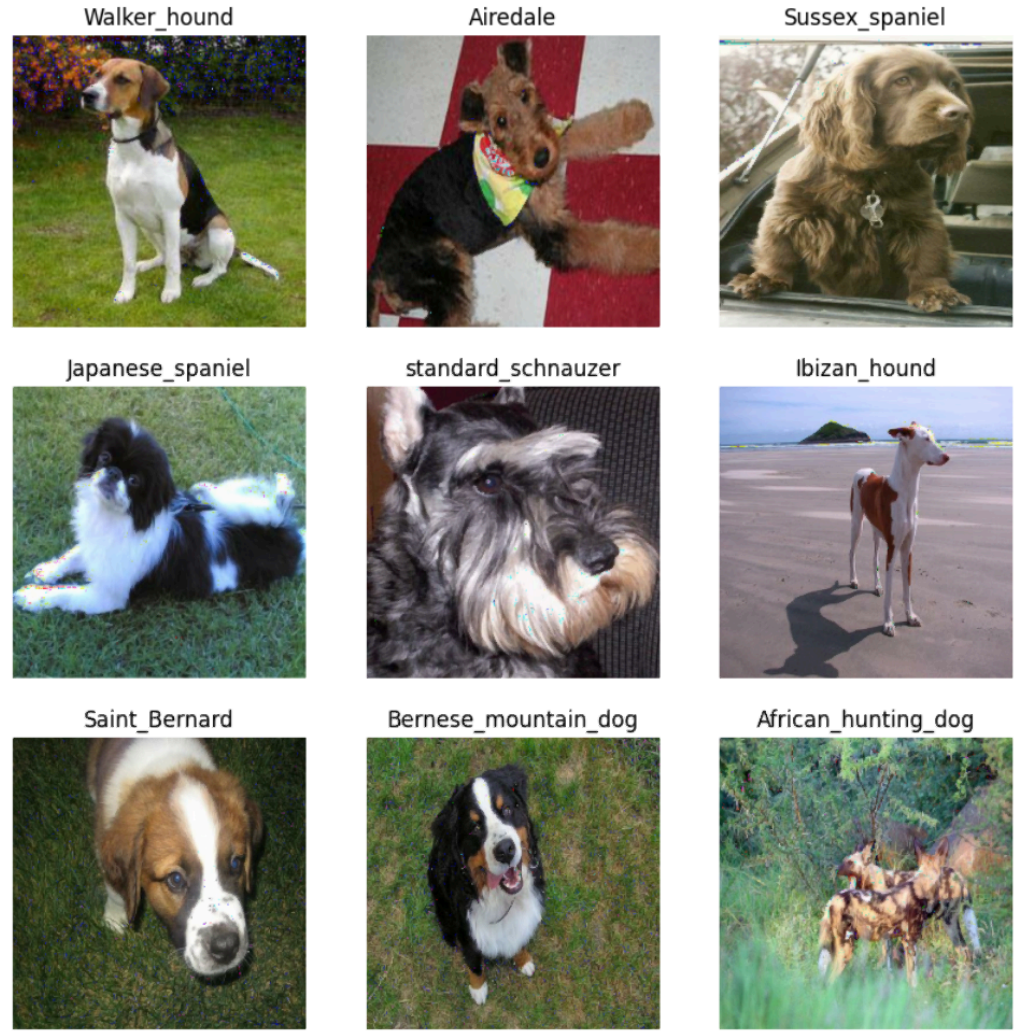

Рисунок 1 – Приклад зображень датасету

```python
def MyNet(input_shape=(299, 299, 3)):
    inputs = layers.Input(shape=input_shape)
    x = layers.Resizing(299, 299)(inputs)
    x = layers.Rescaling(1./255)(x)
    x = layers.RandomColorJitter(
      value_range=(0, 1),
      brightness_factor=0.2,
      contrast_factor=0.2,
      saturation_factor=0.2,
      hue_factor=0.2
    )(x)
    x = layers.RandomFlip(
      mode="horizontal_and_vertical"
    )(x)

    inceptionv3 = applications.InceptionV3(include_top=False, input_tensor=x)

    x = inceptionv3.output
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(4096, activation="relu")(x)
    outputs = layers.Dense(1, activation="sigmoid")(x)

    model = models.Model(inputs=inputs, outputs=outputs)

    return model, inceptionv3
```

Рисунок 2 – Архітектура нейронної мережі

```
model.evaluate(test_ds)
✓ 20.6s
67/68 ——————————— 0s 37ms/step - a
2025-04-21 22:39:19.862443: I external/loca

2025-04-21 22:39:19.870661: I external/loca

2025-04-21 22:39:20.135128: I external/loca

2025-04-21 22:39:20.274731: I external/loca

68/68 ——————————— 21s 136ms/step -

[0.02283058501780033, 0.9916897416114807]
```

Рисунок 3 – Кінцева похибка та точність моделі

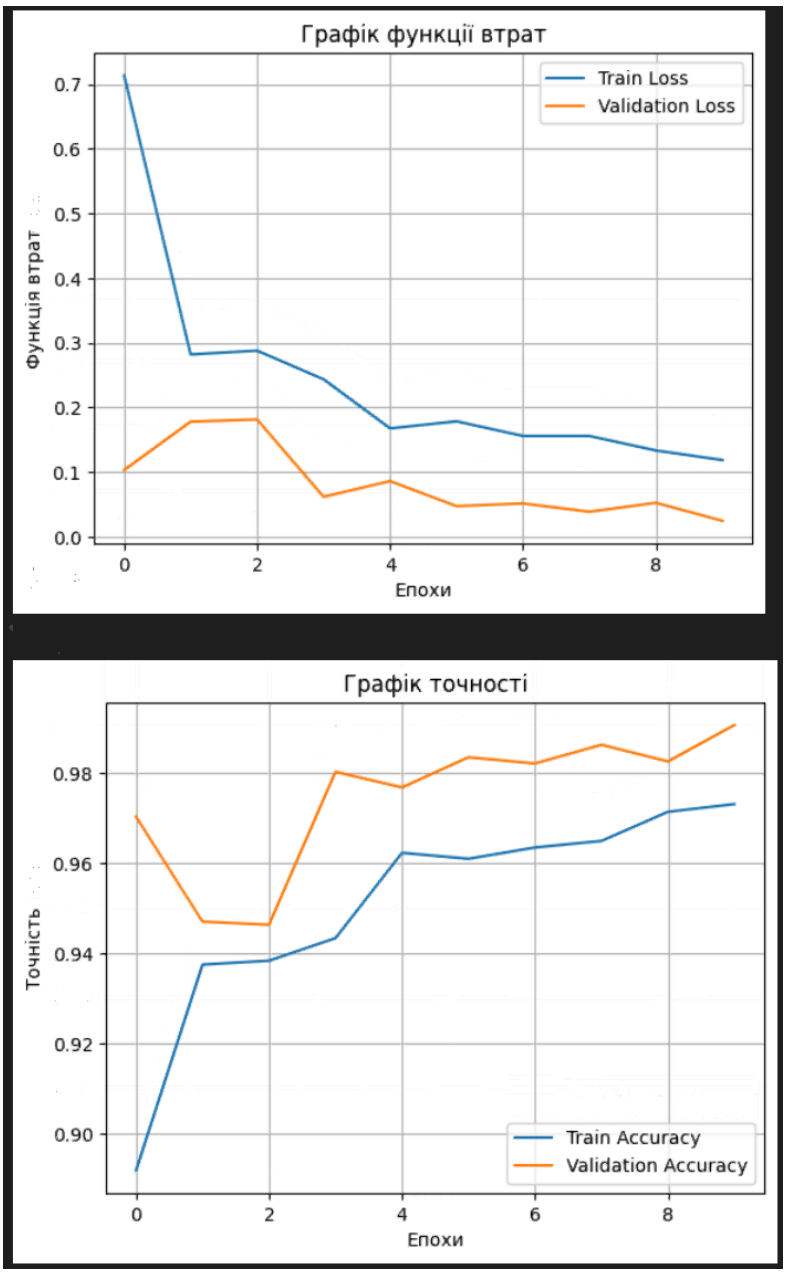

Рисунок 4 – Графіки функцій втрат та точності

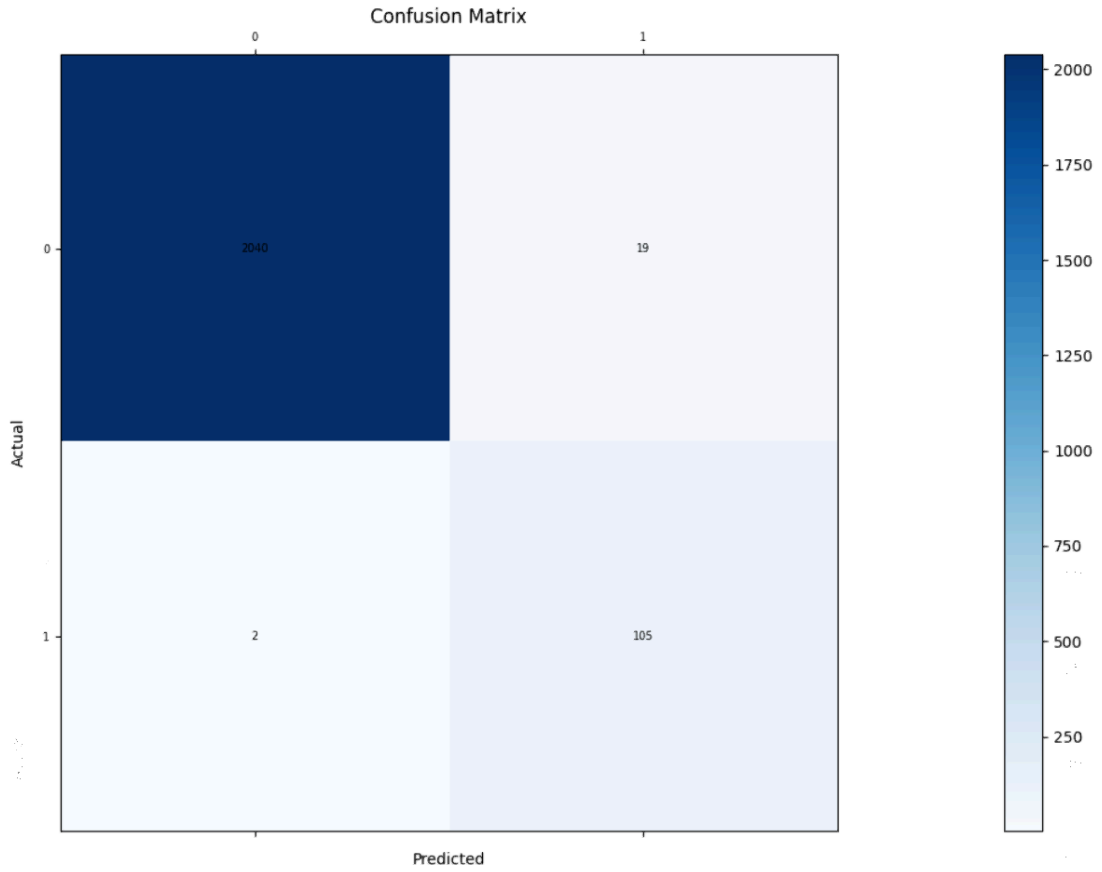

Рисунок 5 – Confusion matrix результатів від отриманої моделі

Висновки

В рамках цієї лабораторної роботи було успішно побудовано, навчено та проаналізовано модель на основі згорткової нейронної мережі InceptionV3 для розпізнавання певної цільової породи собак (Walker Hound) на зображеннях.