



Міністерство освіти і науки України
Національний технічний університет України „КПІ
імені Ігоря Сікорського ”
Факультет інформатики та обчислювальної техніки

Практична робота

З дисципліни « Програмування інтелектуальних інформаційних
систем »

Перевірив:

Катін П.Ю.

Виконала:

Гр. ЗПІ-зп01

Климко А.І.

Вступ

Одна з найперспективніших наук про комп'ютери та програми – комп'ютерний зір. Його сенс полягає у здатності ПК до розпізнавання та визначення суті картинки. Це найважливіша область у штучному інтелекті, що включає відразу кілька дій: розпізнавання вмісту відео, визначення предмета та його класифікація чи генерація. Пошук об'єктів на зображенні, швидше за все, є найважливішою областю комп'ютерного зору.

Визначення речей або живих істот на фотографії та відео активно використовується у таких сферах:

- Пошук автомобілів;
- Система розпізнавання людей;
- Пошук та підрахунок кількості пішоходів;
- Посилення системи безпеки;
- Створення безпілотних автомобілів тощо.

Технологія справді перевернула уявлення про штучний інтелект. Надалі вона стала основою наступних методів R-CNN, Fast-RCNN, Faster-RCNN, RetinaNet. Серед них і високоточні, швидкі методи – SSD та YOLO. Для застосування перерахованих алгоритмів, в основі яких глибоке навчання, потрібна наявність глибоких знань у математиці та досконале розуміння фреймворків.

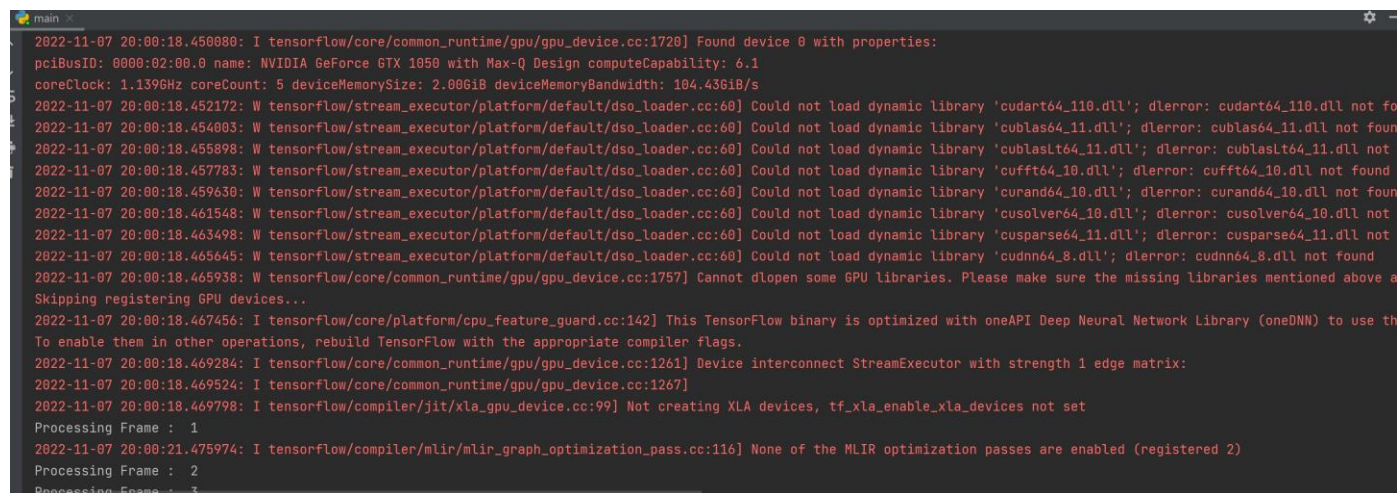
ImageAI має підтримку багатьох налаштувань для пошуку об'єктів. Наприклад, можна налаштувати вилучення всіх знайдених об'єктів під час обробки зображення. Клас пошуку здатний створити окрему папку з назвою image, а потім витягти, зберегти та повернути масив шляхом до всіх об'єктів.

У цьому додатку ми створимо розпізнавання об'єктів за допомогою Python та ImageAI.

Код програми:

```
from imageai.Detection import VideoObjectDetection
import os
execution_path = os.getcwd()
detector = VideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath( os.path.join(execution_path , "yolo.h5"))
detector.loadModel()
video_path = detector.detectObjectsFromVideo(
    input_file_path=os.path.join(execution_path, "Красивая дорога
по лесу.mp4"),
    output_file_path=os.path.join(execution_path, "detected"),
    frames_per_second=20,
    log_progress=True
)
print(video_path)
```

Виконання програми:



```
main
2022-11-07 20:00:18.450080: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1720] Found device 0 with properties:
pciBusID: 0000:02:00.0 name: NVIDIA GeForce GTX 1050 with Max-Q Design computeCapability: 6.1
coreClock: 1.1396Hz coreCount: 5 deviceMemorySize: 2.00GiB deviceMemoryBandwidth: 104.43GiB/s
2022-11-07 20:00:18.452172: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not fo
2022-11-07 20:00:18.454003: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cublas64_11.dll'; dlerror: cublas64_11.dll not four
2022-11-07 20:00:18.455898: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cublasLt64_11.dll'; dlerror: cublasLt64_11.dll not
2022-11-07 20:00:18.457783: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cufft64_10.dll'; dlerror: cufft64_10.dll not found
2022-11-07 20:00:18.459630: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'curand64_10.dll'; dlerror: curand64_10.dll not four
2022-11-07 20:00:18.461548: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cusolver64_10.dll'; dlerror: cusolver64_10.dll not
2022-11-07 20:00:18.463498: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cuspars64_11.dll'; dlerror: cuspars64_11.dll not
2022-11-07 20:00:18.465645: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cudnn64_8.dll'; dlerror: cudnn64_8.dll not found
2022-11-07 20:00:18.465938: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1757] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above a
Skipping registering GPU devices...
2022-11-07 20:00:18.467456: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use th
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-11-07 20:00:18.469284: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1261] Device interconnect StreamExecutor with strength 1 edge matrix:
2022-11-07 20:00:18.469524: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1267]
2022-11-07 20:00:18.469798: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set
Processing Frame : 1
2022-11-07 20:00:21.475974: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
Processing Frame : 2
Processing Frame : 3
```

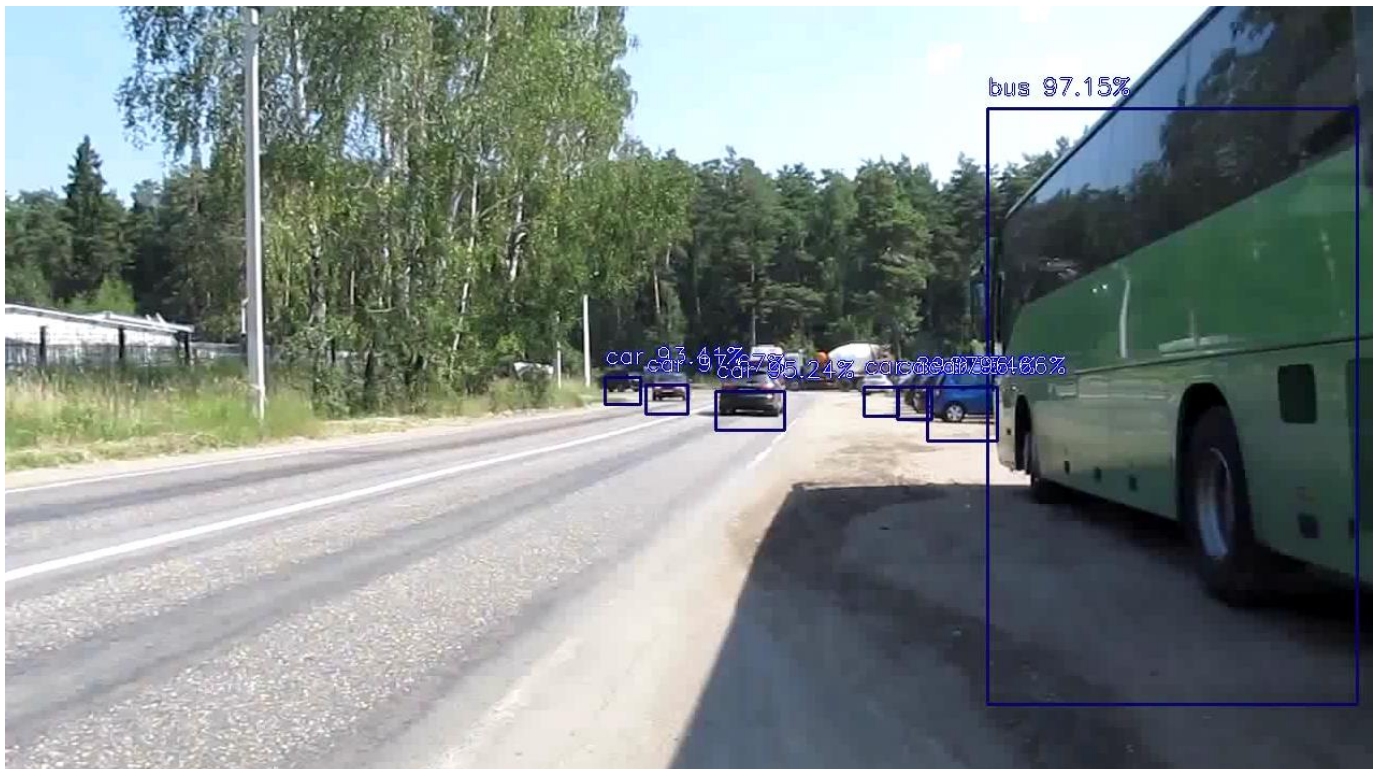
```
pythonProject1 - main.py
pythonProject1 C:\Users\Asus\
main.py
traffic.mp4
traffic_detected.avi
yolo.h5
External Libraries
Scratches and Consoles

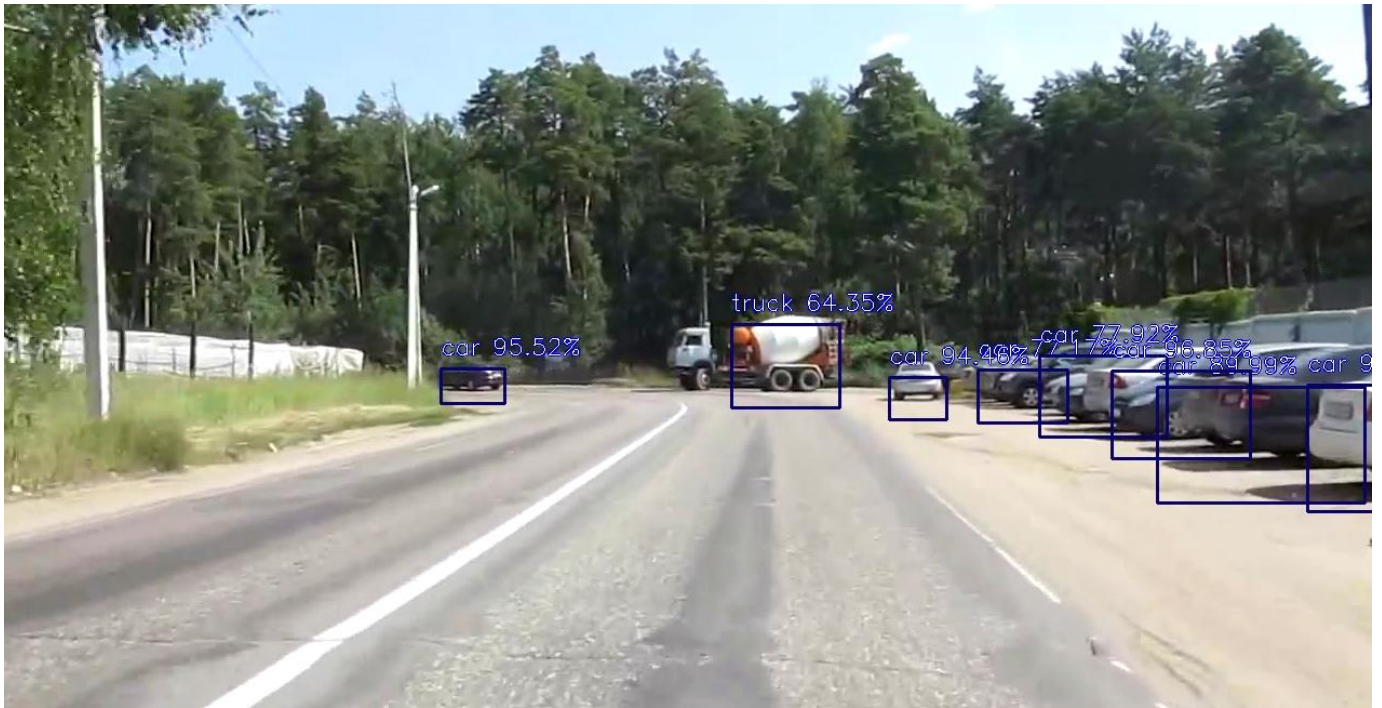
1 from imageai.Detection import VideoObjectDetection
2 import os
3
4 execution_path = os.getcwd()
5
6 detector = VideoObjectDetection()
7 detector.setModelTypeAsYOLOv3()
8 detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
9 detector.loadModel()
10
11 video_path = detector.detectObjectsFromVideo(
12     input_file_path=os.path.join(execution_path, "traffic.mp4"),
13     output_file_path=os.path.join(execution_path, "traffic_detected"),
14     frames_per_second=20,
15     log_progress=True
16 )
17
18 print(video_path)
```

Run: main

Processing Frame : 107
Processing Frame : 108
Processing Frame : 109
Processing Frame : 110
Processing Frame : 111
Processing Frame : 112
Processing Frame : 113
Processing Frame : 114
Processing Frame : 115

Результат виконання програми:





Далі наведений додаток який представляє голосового помічника, він слухає та відповідає на питання.

Код програми:

```
# Підключення всіх необхідних бібліотек
# Нам потрібно: speech_recognition, os, sys, webbrowser
# Для першої бібліотеки прописуємо також псевдонім

import speech_recognition as sr
import sys
import webbrowser
import pyttsx3

# Функція, яка дозволяє промовляти слова
# Приймає параметр "Слова" і прогріває їх

def talk(words):
    engine = pyttsx3.init()
    engine.say(words)
    engine.runAndWait()

# Виклик функції та передача рядка
# саме цей рядок буде проговорено комп'ютером
talk("Привет!")

# Функція command() використовується для відстеження мікрофона.
# Викликаючи функцію ми будемо слухати що скаже користувач, при цьому для
# прослуховування буде використано мікрофон. Отримання даних буде конвертовано
# в рядок і далі відбуватиметься їх перевірка.

def command():
    # Створюємо об'єкт на основі бібліотеки
    # speech_recognition і викликаємо метод визначення даних
    r = sr.Recognizer()

    # Починаємо прослуховувати мікрофон і записуємо дані в source
    with sr.Microphone() as source:
        # Просто висновок, щоб ми знали коли говорити
        print("Слушаю вас?")
    # Встановлюємо паузу, щоб прослуховування
```

```

# почалося лише після 1 секунди
    r.pause_threshold = 1
# використовуємо adjust_for_ambient_noise для видалення сторонніх шумів з
аудіо доріжки
    r.adjust_for_ambient_noise(source, duration=1)
# Отримані дані записуємо у змінну audio
# поки ми отримали лише mp3 звук
    audio = r.listen(source)

    try:
# Розпізнаємо дані із mp3 доріжки. Вказуємо що мова російська, що відстежується.
Завдяки lower() наводимо все в нижній регістр. Тепер ми отримали дані у форматі
рядка, які спокійно можемо перевірити в умовах.
        zadanie = r.recognize_google(audio, language="ru-RU").lower()
        print("Вы сказали: " + zadanie)
    except sr.UnknownValueError:
        talk("Я вас не поняла, повторите")
        zadanie = command()

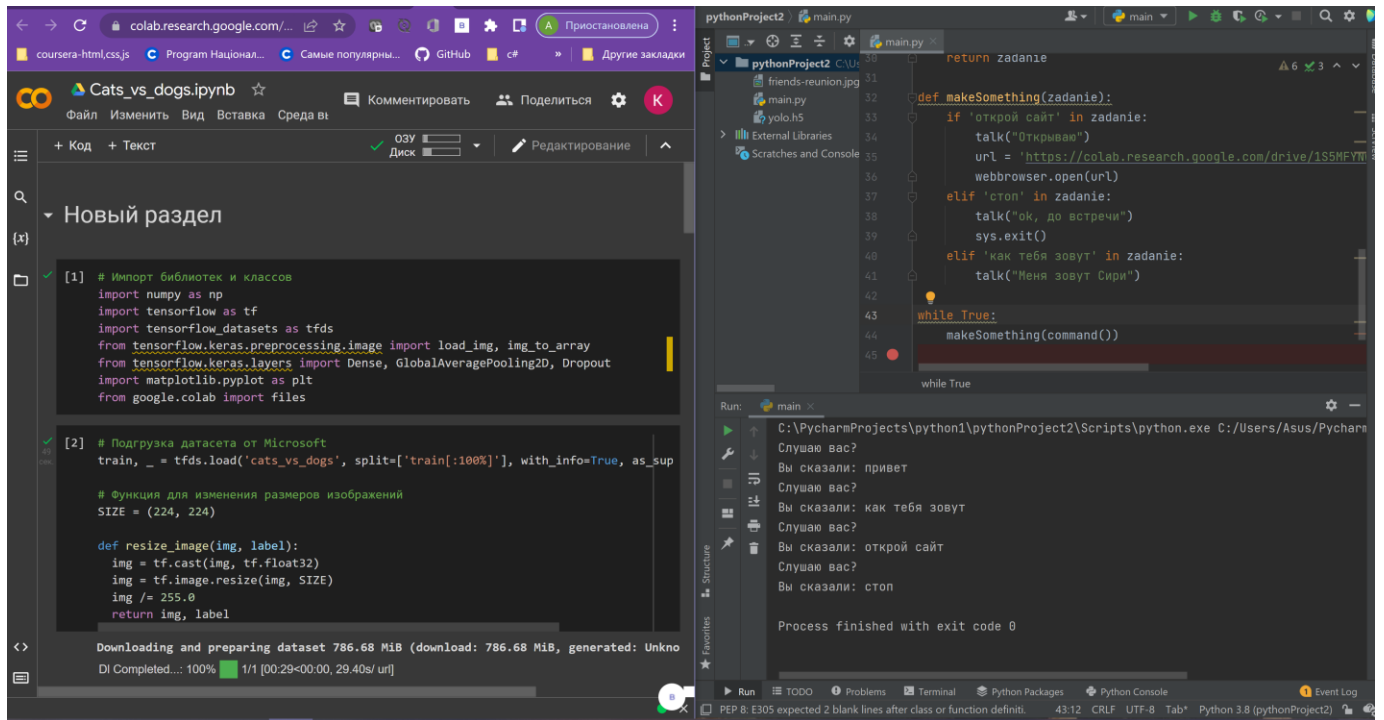
    return zadanie

# Ця функція служить для перевірки тексту,
# що сказав користувач (zadanie - текст від користувача)
def makeSomething(zadanie):
    if 'открой сайт' in zadanie:
        talk("Открываю")
        url =
'https://colab.research.google.com/drive/1S5MFYNwfwQRYHjUwvhFRFMjZA-
L32rT7#scrollTo=KYHzwocwT_7Q'
        webbrowser.open(url)
    elif 'стоп' in zadanie:
        talk("ok, до встречи")
        sys.exit()
    elif 'как тебя зовут' in zadanie:
        talk("Меня зовут Сири")

# Виклик функції для перевірки тексту буде здійснюватися постійно, тому тут
# прописаний нескінченний цикл while
while True:
    makeSomething(command())

```

Результат виконання програми, відкритий сайт за командою голосом, та прописані команди у консолі, або за посиланням https://colab.research.google.com/drive/1phrg_YdRdF2r9IBCjHat7triz3dkg-7e?usp=sharing



У цьому додатку наш штучний інтелект не розпізнаватиме всіх об'єктів, на кшталт: машин, інших тварин, людей тощо. Не робитиме він це з однієї причини. Ми в якості датасету або, іншими словами, набору даних для тренування будемо використовувати датасет від компанії Microsoft. У датасеті у них зібрано понад 25 000 фотографій котів та собачок, що дасть нам можливість натренувати правильні ваги для розпізнавання наших власних фото.

Ми використовуватимемо бібліотеку Tensorflow. Вона створена компанією Google і служить для вирішення задач побудови та тренування нейронної мережі. За рахунок неї процес навчання нейронки трохи простіше, ніж при написанні з використанням тільки numpy.

Також використовуємо бібліотека Matplotlib. Вона служить для візуалізації даних двовимірною графікою. На її основі можна побудувати графіки, зображення та інші візуальні дані, які людиною сприймаються набагато простіше і краще, ніж нулі та одиниці.

Як середовище розробки використовуватимемо спеціальний сервіс від Google - Colab. Colab дозволяє будь-кому писати та виконувати довільний код Python через браузер і особливо добре підходить для машинного навчання, аналізу даних та навчання. Colab дозволяє виконувати код блоками. Наприклад, ми можемо виконати блок коду, де в нас йде навчання нейронки, а далі ми можемо завжди виконувати не

всю програму з початку і до кінця, а лише та ділянка коду, де ми вказуємо нові дані для тестування вже навченої нейронки. Такий принцип суттєво заощаджує час.

Код програми:

```
import numpy as np
import tensorflow as tf
import tensorflow_datasets as tfds
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D,
Dropout
import matplotlib.pyplot as plt
from google.colab import files

# Завантаження датасету від Microsoft
train, _ = tfds.load('cats_vs_dogs', split=['train[:100%]'],
with_info=True, as_supervised=True)

# зміна розмірів зображення
SIZE = (224, 224)

def resize_image(img, label):
    img = tf.cast(img, tf.float32)
    img = tf.image.resize(img, SIZE)
    img /= 255.0
    return img, label

# зміна розмірів усіх зображень отриманих з датасету
train_resized = train[0].map(resize_image)
train_batches = train_resized.shuffle(1000).batch(16)

# створення основного слою для створення моделі
base_layers =
tf.keras.applications.MobileNetV2(input_shape=(SIZE[0], SIZE[1],
3), include_top=False)

# Створення моделі нейромережі
model = tf.keras.Sequential([
    base_layers,
    GlobalAveragePooling2D(),
    Dropout(0.2),
    Dense(1)
```

```

])
model.compile(optimizer='adam',
loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
metrics=['accuracy'])

# навчання неймережі (одна ітерація)
model.fit(train_batches, epochs=1)

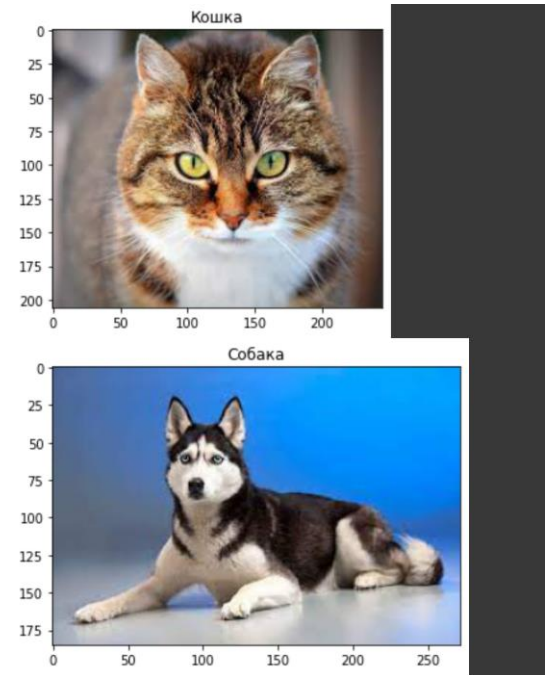
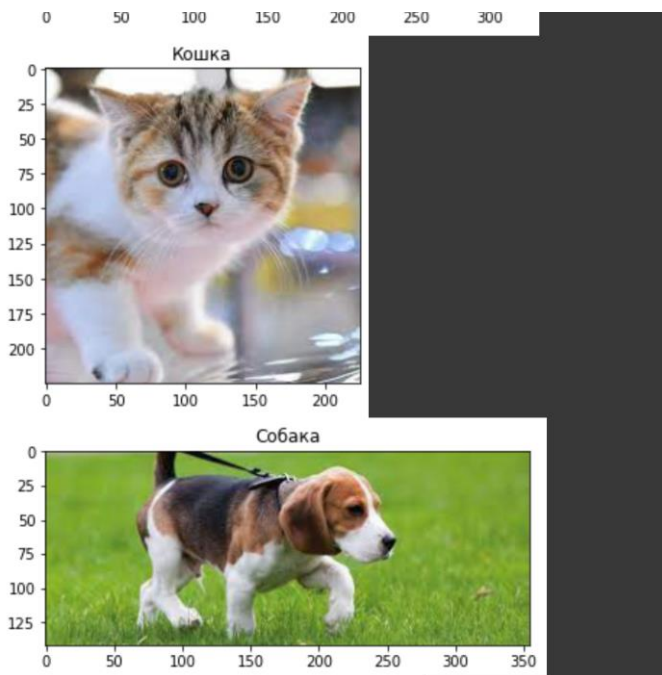
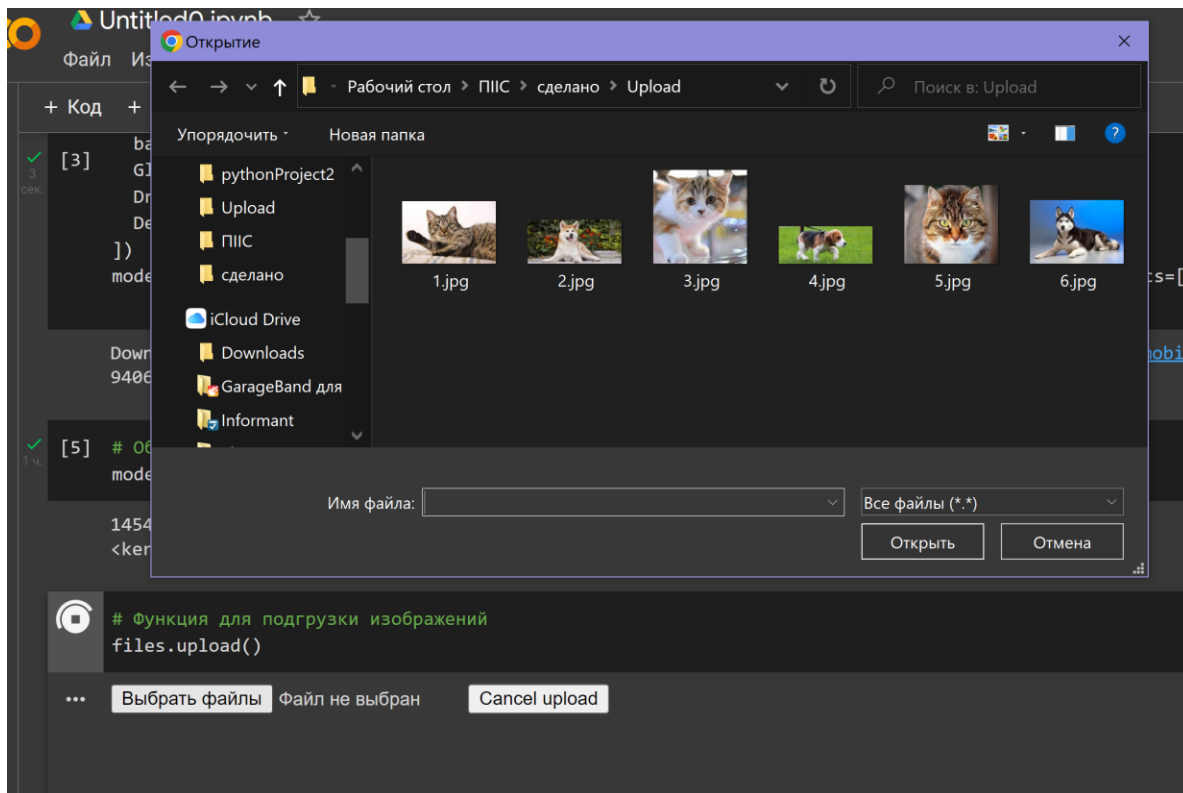
# завантаження зображень
files.upload()

# назви завантажених зображень
images = []

# перебираємо всі зображення та даємо можливість неймережі
розпізнати що на фото
for i in images:
    img = load_img(i)
    img_array = img_to_array(img)
    img_resized, _ = resize_image(img_array, _)
    img_expended = np.expand_dims(img_resized, axis=0)
    prediction = model.predict(img_expended)
    plt.figure()
    plt.imshow(img)
    label = 'Собачка' if prediction > 0 else 'Кошка'
    plt.title('{}'.format(label))

```

Результати виконання програми:



Висновок: Python є однією з найперспективніших мов, що дозволяє втілювати штучний інтелект у життя.

