Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем і технологій

Лабораторна робота №5

Проектування та реалізація програмних систем з нейронними мережами
Згорткові нейронні мережі типу Inception

Київ 2023

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 написати програму, що відкриває та аналізує зображення.

Виконання роботи

```
import os
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
```

```
path = os.path.join(os.getcwd(), '/content/drive/MyDrive/Dataset/laboratory5/')
```

```
datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.3
)
```

```
train_generator = datagen.flow_from_directory(
    path,
    target_size=(150,150),
    batch_size=32,
    shuffle=True,
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    path,
    target_size=(150,150),
    batch_size=32,
    shuffle=True,
    subset='validation'
)
```

```python
pre_trained_model = tf.keras.applications.inception_v3.InceptionV3(
    input_shape=(150, 150, 3),
    include_top=False,
    weights='imagenet'
)

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

```python
last_layer = pre_trained_model.get_layer('mixed10')
last_layer_output = last_layer.output
```

```python
x = tf.keras.layers.GlobalAveragePooling2D()(last_layer_output)
output = tf.keras.layers.Dense(2, activation='softmax')(x)
model = tf.keras.Model(pre_trained_model.input, output)
model.summary()
```

```
 conv2d_273 (Conv2D)            (None, 3, 3, 320)    655360      ['mixed9[0][0]']

 batch_normalization_275 (Batch  (None, 3, 3, 384)   1152        ['conv2d_275[0][0]']
 Normalization)

 batch_normalization_276 (Batch  (None, 3, 3, 384)   1152        ['conv2d_276[0][0]']
 Normalization)

 batch_normalization_279 (Batch  (None, 3, 3, 384)   1152        ['conv2d_279[0][0]']
 Normalization)

 batch_normalization_280 (Batch  (None, 3, 3, 384)   1152        ['conv2d_280[0][0]']
 Normalization)

 conv2d_281 (Conv2D)            (None, 3, 3, 192)    393216      ['average_pooling2d_26[0][0]']

 batch_normalization_273 (Batch  (None, 3, 3, 320)   960         ['conv2d_273[0][0]']
 Normalization)

 activation_275 (Activation)    (None, 3, 3, 384)    0           ['batch_normalization_275[0][0]']

 activation_276 (Activation)    (None, 3, 3, 384)    0           ['batch_normalization_276[0][0]']

 activation_279 (Activation)    (None, 3, 3, 384)    0           ['batch_normalization_279[0][0]']

 activation_280 (Activation)    (None, 3, 3, 384)    0           ['batch_normalization_280[0][0]']

 batch_normalization_281 (Batch  (None, 3, 3, 192)   576         ['conv2d_281[0][0]']
 Normalization)

 activation_273 (Activation)    (None, 3, 3, 320)    0           ['batch_normalization_273[0][0]']

 mixed9_1 (Concatenate)         (None, 3, 3, 768)    0           ['activation_275[0][0]',
                                                                  'activation_276[0][0]']

 concatenate_5 (Concatenate)    (None, 3, 3, 768)    0           ['activation_279[0][0]',
                                                                  'activation_280[0][0]']

 activation_281 (Activation)    (None, 3, 3, 192)    0           ['batch_normalization_281[0][0]']

 mixed10 (Concatenate)          (None, 3, 3, 2048)   0           ['activation_273[0][0]',
                                                                  'mixed9_1[0][0]',
                                                                  'concatenate_5[0][0]',
                                                                  'activation_281[0][0]']

 global_average_pooling2d_2 (Gl  (None, 2048)        0           ['mixed10[0][0]']
 obalAveragePooling2D)

 dense_2 (Dense)                (None, 2)            4098        ['global_average_pooling2d_2[0][0
                                                                 ]']

==================================================================================================
Total params: 21,806,882
Trainable params: 4,098
Non-trainable params: 21,802,784
__________________________________________________________________________________________________
```

```
model.compile(
    optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

```
history = model.fit(train_generator, epochs=30, validation_data=validation_generator)
```

```
Epoch 8/30
2/2 [==============================] - 5s 3s/step - loss: 0.7598 - accuracy: 0.7174 - val_loss: 0.9034 - val_accuracy: 0.6111
Epoch 9/30
2/2 [==============================] - 3s 3s/step - loss: 0.7217 - accuracy: 0.7609 - val_loss: 0.8959 - val_accuracy: 0.6111
Epoch 10/30
2/2 [==============================] - 3s 2s/step - loss: 0.6801 - accuracy: 0.7391 - val_loss: 0.8456 - val_accuracy: 0.6111
Epoch 11/30
2/2 [==============================] - 5s 3s/step - loss: 0.6457 - accuracy: 0.7826 - val_loss: 0.8564 - val_accuracy: 0.6667
Epoch 12/30
2/2 [==============================] - 3s 3s/step - loss: 0.6116 - accuracy: 0.7609 - val_loss: 0.8071 - val_accuracy: 0.6667
Epoch 13/30
2/2 [==============================] - 3s 2s/step - loss: 0.6066 - accuracy: 0.7609 - val_loss: 0.8532 - val_accuracy: 0.6667
Epoch 14/30
2/2 [==============================] - 5s 4s/step - loss: 0.5672 - accuracy: 0.7609 - val_loss: 0.7781 - val_accuracy: 0.6667
Epoch 15/30
2/2 [==============================] - 3s 1s/step - loss: 0.5367 - accuracy: 0.7826 - val_loss: 0.7168 - val_accuracy: 0.6667
Epoch 16/30
2/2 [==============================] - 3s 2s/step - loss: 0.5457 - accuracy: 0.7391 - val_loss: 0.7096 - val_accuracy: 0.6667
Epoch 17/30
2/2 [==============================] - 4s 3s/step - loss: 0.4990 - accuracy: 0.7826 - val_loss: 0.6939 - val_accuracy: 0.6667
Epoch 18/30
2/2 [==============================] - 3s 3s/step - loss: 0.4676 - accuracy: 0.8261 - val_loss: 0.6685 - val_accuracy: 0.6667
Epoch 19/30
2/2 [==============================] - 3s 1s/step - loss: 0.4478 - accuracy: 0.8261 - val_loss: 0.6748 - val_accuracy: 0.7222
Epoch 20/30
2/2 [==============================] - 5s 3s/step - loss: 0.4287 - accuracy: 0.8478 - val_loss: 0.5973 - val_accuracy: 0.6667
Epoch 21/30
2/2 [==============================] - 4s 2s/step - loss: 0.4075 - accuracy: 0.8478 - val_loss: 0.5742 - val_accuracy: 0.7778
Epoch 22/30
2/2 [==============================] - 3s 3s/step - loss: 0.3824 - accuracy: 0.8478 - val_loss: 0.5853 - val_accuracy: 0.6667
Epoch 23/30
2/2 [==============================] - 3s 2s/step - loss: 0.3660 - accuracy: 0.8478 - val_loss: 0.5814 - val_accuracy: 0.7222
Epoch 24/30
2/2 [==============================] - 5s 4s/step - loss: 0.3805 - accuracy: 0.8696 - val_loss: 0.5704 - val_accuracy: 0.7222
Epoch 25/30
2/2 [==============================] - 3s 2s/step - loss: 0.3350 - accuracy: 0.8478 - val_loss: 0.5561 - val_accuracy: 0.8333
Epoch 26/30
2/2 [==============================] - 3s 2s/step - loss: 0.3050 - accuracy: 0.8696 - val_loss: 0.5736 - val_accuracy: 0.7222
Epoch 27/30
2/2 [==============================] - 3s 2s/step - loss: 0.2943 - accuracy: 0.8478 - val_loss: 0.5924 - val_accuracy: 0.7222
Epoch 28/30
2/2 [==============================] - 5s 4s/step - loss: 0.2791 - accuracy: 0.8696 - val_loss: 0.5415 - val_accuracy: 0.8333
Epoch 29/30
2/2 [==============================] - 3s 2s/step - loss: 0.2533 - accuracy: 0.8913 - val_loss: 0.5177 - val_accuracy: 0.8333
Epoch 30/30
2/2 [==============================] - 3s 2s/step - loss: 0.2843 - accuracy: 0.8478 - val_loss: 0.4911 - val_accuracy: 0.8333
```

```
model.save('lab5_model')
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compi
```

```
!zip -r /content/lab5-model.zip /content/lab5_model
```

```
  adding: content/lab5_model/ (stored 0%)
  adding: content/lab5_model/keras_metadata.pb (deflated 96%)
  adding: content/lab5_model/assets/ (stored 0%)
  adding: content/lab5_model/fingerprint.pb (stored 0%)
  adding: content/lab5_model/variables/ (stored 0%)
  adding: content/lab5_model/variables/variables.index (deflated 79%)
  adding: content/lab5_model/variables/variables.data-00000-of-00001 (deflated 7%)
  adding: content/lab5_model/saved_model.pb (deflated 92%)
```

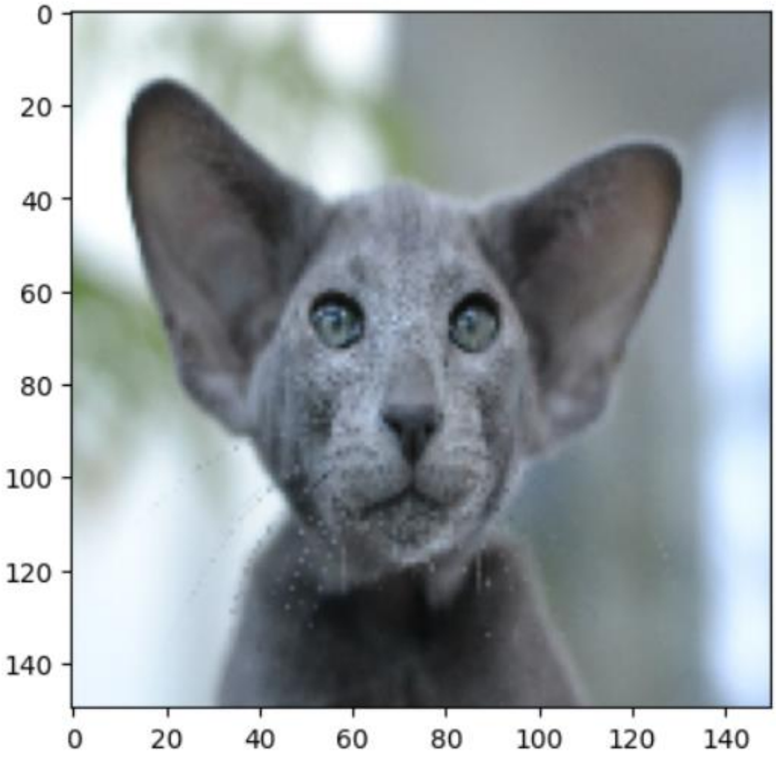
```
files.download("/content/lab5-model.zip")
```

Перевіримо:

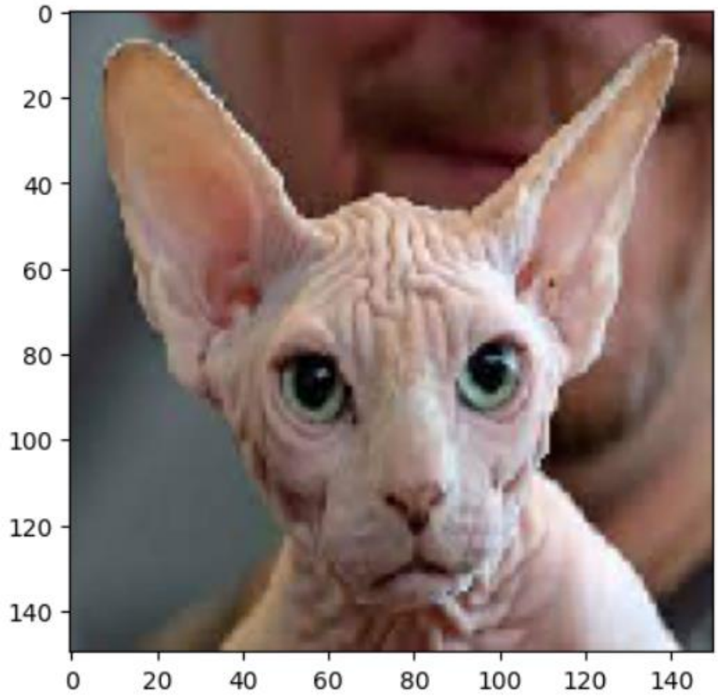
```
files_dir = os.path.join(os.getcwd(), 'drive/MyDrive/Dataset/Check')
files_names = os.listdir(files_dir)
names = ['Other','Sphynx ']
classes = []

i = 1
for item in files_names:
  img_path = os.path.join(files_dir, item)
  img = tf.keras.utils.load_img(img_path, target_size=(150, 150))
  img = tf.keras.utils.img_to_array(img) / 255
  img = np.expand_dims(img, axis=0)
  img = np.vstack([img])
  pred = model.predict(img, batch_size=10)

  num = np.argmax(pred)
  print(f'image {i} ({files_names[i-1]}) is a {names[num]}')
  plt.imshow(img[0])
  plt.show()
  i += 1
```

Результат:

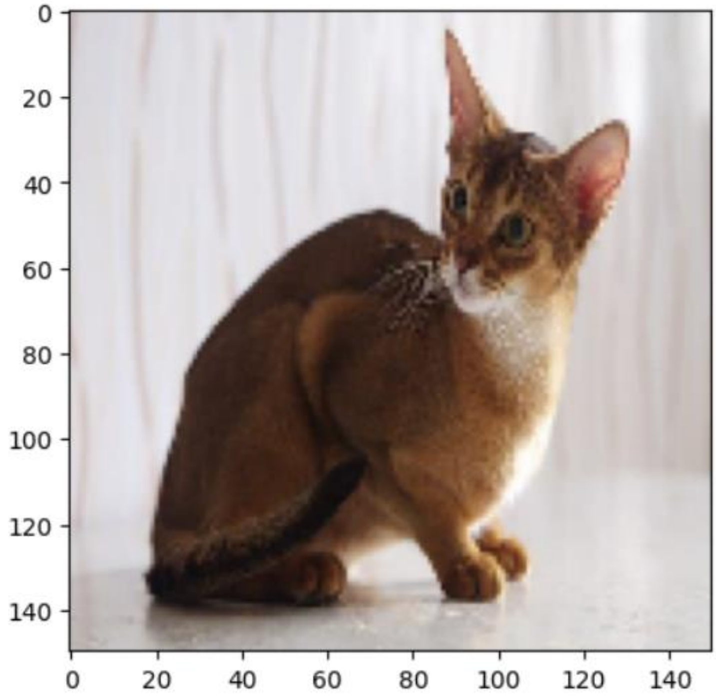
```
1/1 [==============================] - 0s 72ms/step
image 1 (oriental.jpg) is a Other
```



```
1/1 [==============================] - 0s 73ms/step
image 2 (Sphynx.jpg) is a Sphynx
```




```
1/1 [==============================] - 0s 73ms/step
image 3 (oriental1.jpg) is a Other
```



Висновок: Виконуючи дану лабораторну роботу, я реалізував згорткову нейронну мережу типу Inception для розпізнавання котів породи Сфінкса на зображеннях. Використовувався власний маленький датасет.