МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

з лабораторної роботи №6

з навчальної дисципліни «Проектування та реалізація програмних систем з нейронними мережами»

Виконав:
студент групи ІП-15
Мєшков Андрій Ігорович

Перевірив:
Шимкович В.М.

Київ 2024

ЛАБОРАТОРНА РОБОТА №6

Тема: Згорткові нейронні мережі типу Xception

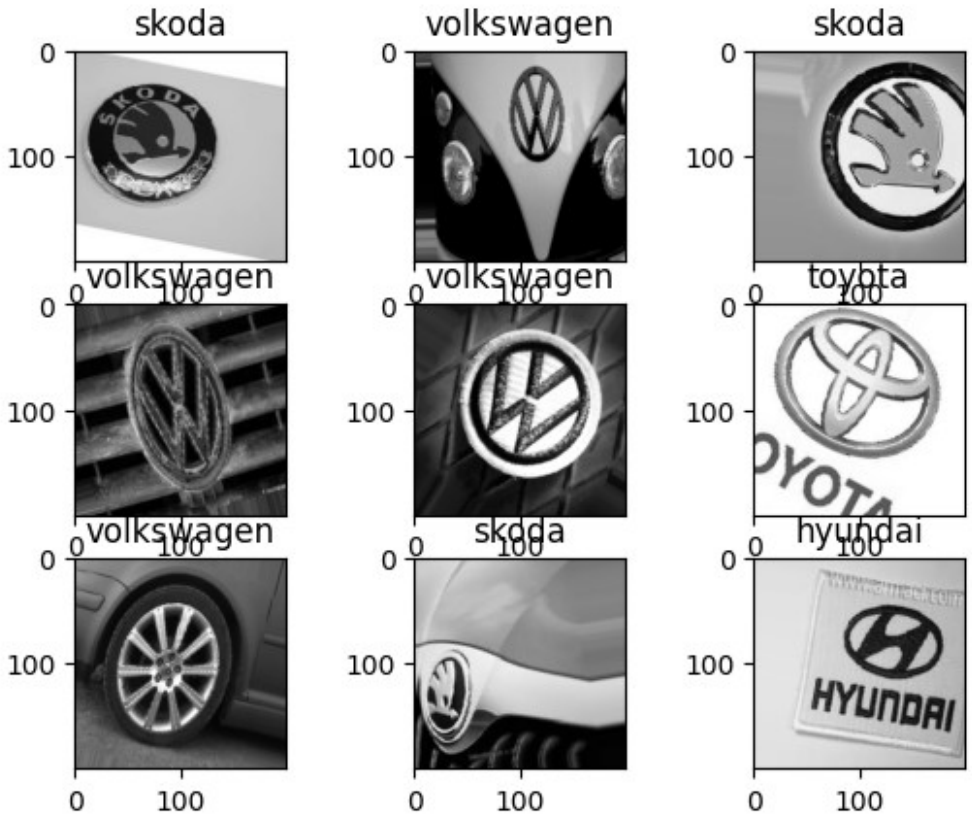
Завдання – Написати програму що реалізує згорткову нейронну мережу Xception для розпізнавання об’єктів на відео. Створити власний дата сет з папки на диску, навчити нейронну мережу на цьому датасеті розпізнавати логотип вашого улюбленого бренду, скажімо Apple чи BMW. Навчену нейронну мережу зберегти на комп’ютер написати програму, що відкриває та аналізує відео, результат – час на якому з’являвся логотип.

```
import cv2
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Conv2D, Input, SeparableConv2D,
Add, Dense, BatchNormalization, ReLU, MaxPool2D, GlobalAvgPool2D
from tensorflow.keras import Model
from tensorflow.keras.utils import plot_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

IMAGE_RESOLUTION = (200, 200)
INPUT_SHAPE = (200, 200, 1)

#Загруження даних
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    height_shift_range=0.2)

test_datagen = ImageDataGenerator(
    rescale=1./255)

train_ds = train_datagen.flow_from_directory(
    '../src/Car_Brand_Logos/train',
    target_size=IMAGE_RESOLUTION,
    batch_size=16,
    class_mode='categorical',
    color_mode='grayscale'
)

test_ds = test_datagen.flow_from_directory(
    '../src/Car_Brand_Logos/test',
    target_size = IMAGE_RESOLUTION,
    batch_size = 16,
    class_mode='categorical',
    color_mode='grayscale'
)
```

```
Found 2513 images belonging to 8 classes.
Found 400 images belonging to 8 classes.
```

```
#Демонстрація
NUMBER_OF_CLASSES = 8

def get_label_name(label_index):
    class_names
=["hyundai","lexus","mazda","mercedes","opel","skoda","toyota","volksw
agen"]
```

```
    return class_names[label_index]

for i in range(0, 9):
    image, label = next(iter(train_ds))
    plt.subplot(330 + 1 + i)

    plt.imshow(cv2.cvtColor(image[0], cv2.COLOR_BGR2RGB))
    plt.title(get_label_name(np.argmax(label)))

plt.show()
```




```
# MODEL
def conv_bn(x, filters, kernel_size, strides=1):
    x = Conv2D(filters=filters,
        kernel_size=kernel_size,
        strides=strides,
        padding="same",
        use_bias=False)(x)
    x = BatchNormalization()(x)
    return x

def sepr_bn(x, filters, kernel_size, strides=1):
    x = SeparableConv2D(filters=filters,
```

```
        kernel_size=kernel_size,
        strides=strides,
        padding="same",
        use_bias=False)(x)
    x = BatchNormalization()(x)
    return x


def entry_flow(x):
    x = conv_bn(x, filters=32, kernel_size=3, strides=2)
    x = ReLU()(x)
    x = conv_bn(x, filters=64, kernel_size=3)
    tensor = ReLU()(x)

    x = sepr_bn(tensor, filters=128, kernel_size=3)
    x = ReLU()(x)
    x = sepr_bn(x, filters=128, kernel_size=3)
    x = MaxPool2D(pool_size=3, strides=2, padding="same")(x)

    tensor = conv_bn(tensor, filters=128, kernel_size=1, strides=2)
    x = Add()([tensor,x])

    x = ReLU()(x)
    x = sepr_bn(x, filters=256, kernel_size=3)
    x = ReLU()(x)
    x = sepr_bn(x, filters=256, kernel_size=3)
    x = MaxPool2D(pool_size=3, strides=2, padding="same")(x)
    tensor = conv_bn(tensor, filters=256, kernel_size=1, strides=2)

    x = Add()([tensor,x])
    x = ReLU()(x)
    x = sepr_bn(x, filters=728, kernel_size=3)
    x = ReLU()(x)
    x = sepr_bn(x, filters=728, kernel_size=3)
    x = MaxPool2D(pool_size=3, strides=2, padding="same")(x)
    tensor = conv_bn(tensor, filters=728, kernel_size=1, strides=2)
    x = Add()([tensor,x])
    return x


def middle_flow(tensor):
    for _ in range(8):
        x = ReLU()(tensor)
        x = sepr_bn(x, filters=728, kernel_size=3)

        x = ReLU()(x)
        x = sepr_bn(x, filters=728, kernel_size=3)

        x = ReLU()(x)
        x = sepr_bn(x, filters=728, kernel_size=3)
```

```
        tensor = Add()([tensor,x])
    return tensor


def exit_flow(tensor):
    x = ReLU()(tensor)
    x = sepr_bn(x, filters=728, kernel_size=3)

    x = ReLU()(x)
    x = sepr_bn(x, filters=1024, kernel_size=3)

    x = MaxPool2D(pool_size=3, strides=2, padding="same")(x)

    tensor = conv_bn(tensor, filters=1024, kernel_size=1, strides=2)
    x = Add()([tensor, x])

    x = sepr_bn(x, filters=1536, kernel_size=3)
    x = ReLU()(x)

    x = sepr_bn(x, filters=2048, kernel_size=3)
    x = ReLU()(x)

    x = GlobalAvgPool2D()(x)

    x = Dense(units=NUMBER_OF_CLASSES, activation="softmax")(x)
    return x

input_= Input(shape=INPUT_SHAPE)
x = entry_flow(input_)
x = middle_flow(x)
output_ = exit_flow(x)
model = Model(inputs=input_, outputs=output_)
model.compile(optimizer='adam',
              loss=keras.losses.categorical_crossentropy,
              metrics=['accuracy'])


history = model.fit(train_ds, epochs=10, validation_data=test_ds)
```

```
Epoch 1/10

/Users/andrey/Documents/DOcument_study/Year 3.2/ПЗПтаРНС/Lab 6/src/.venv/lib/python3.9/site-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:120: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
```

```
158/158 ━━━━━━━━━━━━━━━━━━━━ 740s 5s/step - accuracy: 0.1537 - loss:
2.2972 - val_accuracy: 0.1250 - val_loss: 2.0795
Epoch 2/10
158/158 ━━━━━━━━━━━━━━━━━━━━ 686s 4s/step - accuracy: 0.2041 - loss:
2.0520 - val_accuracy: 0.1250 - val_loss: 2.0794
Epoch 3/10
158/158 ━━━━━━━━━━━━━━━━━━━━ 682s 4s/step - accuracy: 0.2762 - loss:
1.9143 - val_accuracy: 0.1250 - val_loss: 2.1212
Epoch 4/10
158/158 ━━━━━━━━━━━━━━━━━━━━ 685s 4s/step - accuracy: 0.3493 - loss:
1.7792 - val_accuracy: 0.2325 - val_loss: 2.5208
Epoch 5/10
158/158 ━━━━━━━━━━━━━━━━━━━━ 680s 4s/step - accuracy: 0.4058 - loss:
1.6339 - val_accuracy: 0.4075 - val_loss: 2.0070
Epoch 6/10
158/158 ━━━━━━━━━━━━━━━━━━━━ 672s 4s/step - accuracy: 0.4617 - loss:
1.5248 - val_accuracy: 0.2850 - val_loss: 4.1649
Epoch 7/10
158/158 ━━━━━━━━━━━━━━━━━━━━ 674s 4s/step - accuracy: 0.5320 - loss:
1.3124 - val_accuracy: 0.4775 - val_loss: 3.1745
Epoch 8/10
158/158 ━━━━━━━━━━━━━━━━━━━━ 672s 4s/step - accuracy: 0.6238 - loss:
1.1060 - val_accuracy: 0.5075 - val_loss: 2.5456
Epoch 9/10
158/158 ━━━━━━━━━━━━━━━━━━━━ 666s 4s/step - accuracy: 0.6564 - loss:
1.0096 - val_accuracy: 0.6250 - val_loss: 1.4120
Epoch 10/10
158/158 ━━━━━━━━━━━━━━━━━━━━ 669s 4s/step - accuracy: 0.6995 - loss:
0.8989 - val_accuracy: 0.5575 - val_loss: 1.9698
```

```
model.save('xception_my_dataset_categorical_200px.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

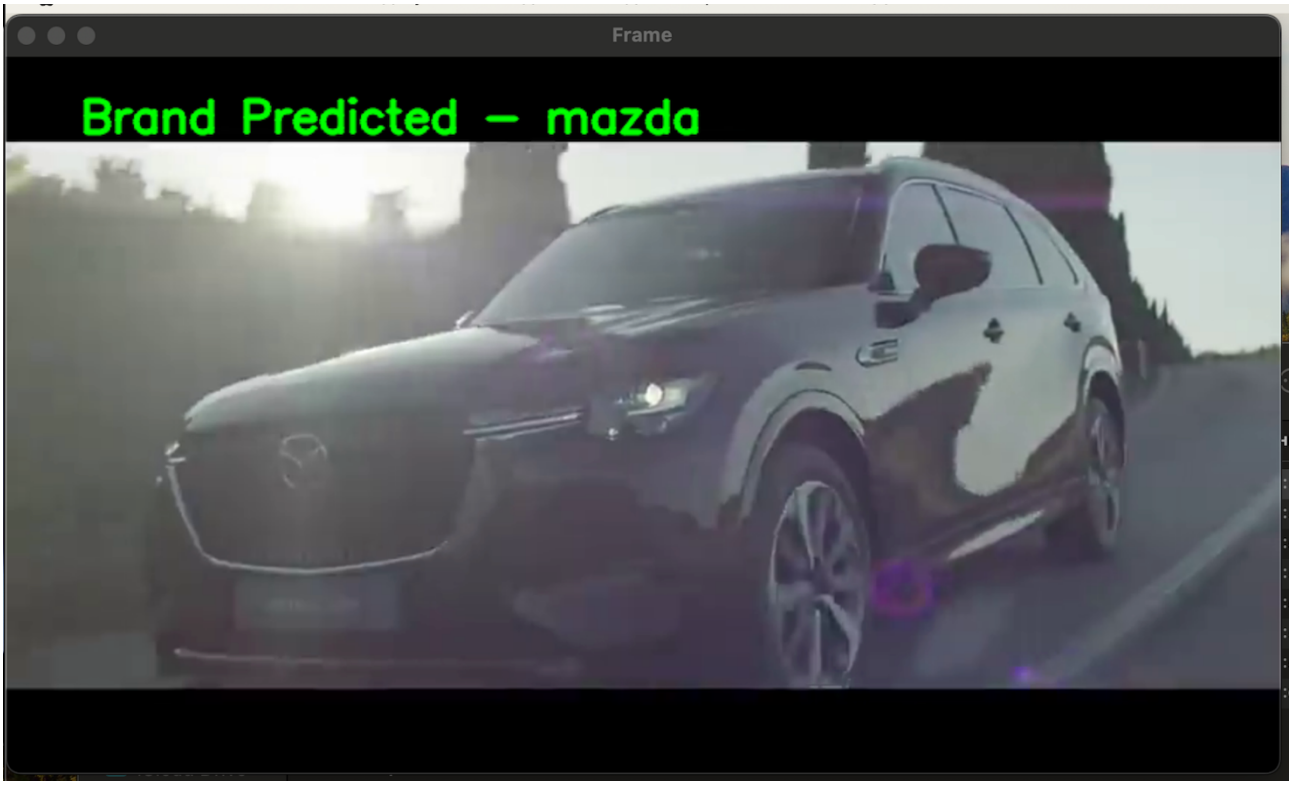
```
model.evaluate(test_ds)
```

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 24s 955ms/step - accuracy: 0.5471 - loss:
1.9902
```

```
[1.9697500467300415, 0.5575000047683716]
```

```
#Демонстрація роботи на фреймі відео
video_path = 'video1.mp4'
class_names =
["hyundai","lexus","mazda","mercedes","opel","skoda","toyota","volkswa
gen"]
loaded_model =
```

```
keras.models.load_model('xception_my_dataset_categorical_200px.h5')
cap = cv2.VideoCapture(video_path)
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    resized_frame = cv2.resize(gray_frame, (200, 200))
    normalized_frame = resized_frame / 255.0
    input_frame = np.expand_dims(normalized_frame, axis=0)
    prediction = loaded_model.predict(input_frame)
    predicted_class = class_names[np.argmax(prediction)]
    print(prediction, predicted_class)
    for _ in range(4):
        cv2.putText(frame, f"Brand Predicted - {predicted_class}",
(50, 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2,
cv2.LINE_AA)
        cv2.imshow('Frame', frame)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```


Frame
Brand Predicted – mazda

Висновок:

Під час виконання лабораторної роботи було реалізовано програму що реалізує згорткову нейронну мережу Xception для розпізнавання логотипів автомобілів на відео. Використано готовий датасет, навчено нейронну мережу на цьому датасеті розпізнавати логотип автомобілів. Навчену нейронну мережу збережено та протестовано на аналізі відео.