MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

Volodymyr Shymkovych

Design and implementation of software systems with neural networks

LABORATORY WORK #6

kulubecioglu mehmet
IM-14 FIOT

Logo Recognition Using a Pre-trained Xception Model in TensorFlow

Kyiv
IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE
2024

Logo Recognition with Xception Model

Step 1: Importing Required Libraries

```
In [*]: import os
        import cv2
        import numpy as np
        from tensorflow.keras.applications import Xception
        from tensorflow.keras.models import Sequential, load_model
        from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
        from tensorflow.keras.optimizers import Adam
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Step 2: Define Xception Model

- We create a base model using the pre-trained Xception model.

```
# Define a pre-trained network as Xception model
base_model = Xception(weights='imagenet', include_top=False, input_shape=(299, 299, 3))
```

Step 3: Freeze Pre-trained Layers

- We freeze all layers in the Xception model because these layers are pre-trained and we do not want to change their learning.

```
# Freeze the layers of the pre-trained model
for layer in base_model.layers:
    layer.trainable = False
```

Step 4: Create Custom Model

- We create a higher model by adding special layers to the basic model we have created.

```
# Create a custom model on top of Xception
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(1024, activation='relu'))
model.add(Dense(1, activation='sigmoid'))  # Binary classification (logo or not)
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

Step 5: Create Sample Dataset

- We create a random data set for illustration purposes. In a real project, a larger and more representative data set should be used.

```
X_train = np.random.rand(100, 299, 299, 3)
y_train = np.random.randint(2, size=(100,))
```

Step 6: Train the Model

- We train the model we created.

```
# Train the model
model.fit(
    X_train,
    y_train,
    batch_size=BATCH_SIZE,
    epochs=EPOCHS
)
```

Step 7: Save Trained Model

- After training, we save the model.

```
# Save the trained model
model.save('logo_detection_model.h5')
```

Step 8: Evaluate the Model

- We evaluate the trained model on the test dataset.

```python
# Example code for model evaluation
loss, accuracy = model.evaluate(X_train, y_train)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')
```

Step 9: Load Trained Model

- We can then load the trained model again.

```python
# Example of loading the model
loaded_model = load_model('logo_detection_model.h5')
```

Step 10: Function for Logo Detection in Video

- We define a function that detects the logo in the video.

```python
# Function for logo detection in a video
def logo_detection_in_video(video_path, model):
    cap = cv2.VideoCapture(video_path)

    frame_count = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Preprocess the frame for prediction
        frame = cv2.resize(frame, IMAGE_SIZE)
        frame = frame / 255.0  # Normalization
        frame = np.expand_dims(frame, axis=0)

        # Make a prediction using the trained model
        prediction = model.predict(frame)

        # Check if a logo is detected
        if prediction[0][0] > 0.5:  # Threshold value for logo detection
            print(f"Time when the logo is seen: {frame_count / cap.get(cv2.CAP_PROP_FPS)} seconds")

        frame_count += 1

    cap.release()
```

Step 11: Example Usage

- We use the function to detect logo on a specific video file.

```
video_path = 'C:/Users/mehme/video.mp4'
logo_detection_in_video(video_path, loaded_model)
```

```
Epoch 1/10
```

Output:

```
Epoch 1/10
4/4 [==============================] - 11s 2s/step - loss: 0.7394 - accuracy: 0.6100
Epoch 2/10
4/4 [==============================] - 8s 2s/step - loss: 1.0202 - accuracy: 0.4500
Epoch 3/10
4/4 [==============================] - 7s 2s/step - loss: 0.7800 - accuracy: 0.5500
Epoch 4/10
4/4 [==============================] - 8s 2s/step - loss: 0.8346 - accuracy: 0.5500
Epoch 5/10
4/4 [==============================] - 8s 2s/step - loss: 0.6850 - accuracy: 0.5200
Epoch 6/10
4/4 [==============================] - 9s 2s/step - loss: 0.7305 - accuracy: 0.4600
Epoch 7/10
4/4 [==============================] - 9s 2s/step - loss: 0.6552 - accuracy: 0.6700
Epoch 8/10
4/4 [==============================] - 9s 2s/step - loss: 0.6525 - accuracy: 0.6000
Epoch 9/10
4/4 [==============================] - 9s 2s/step - loss: 0.6653 - accuracy: 0.5900
Epoch 10/10
4/4 [==============================] - 9s 2s/step - loss: 0.6489 - accuracy: 0.6400
4/4 [==============================] - 9s 2s/step - loss: 0.6599 - accuracy: 0.5500
Test Loss: 0.6599
Test Accuracy: 0.5500
```

```
In [ ]:
```

- **Epoch 1:** Training begins and training loss appears to be low and accuracy appears to be high. However, the results in the first epoch are generally optimistic because the model is better than random guesses.

- **Epoch 2-5:** Training loss increases, accuracy decreases. This may indicate that the model is experiencing problems in the learning process or has started to overfit. The model may be over-adapted to the training data set and have poor generalization ability.

- **Epoch 6-7:** Training loss and accuracy values fluctuate. This may indicate that the model is trying to perform better than the previous stage. However, the problem of overlearning may still persist.

- **Epoch 8-10:** Training loss continues to decline but accuracy fluctuates. In this case, training the model over more epochs may increase generalization ability or increase the overlearning problem.

- **Test Loss and Accuracy:** The performance of the model on the test data set is low. Test accuracy value is 0.55, i.e. 55%. This may indicate that the model's ability to generalize what it has learned is weak and cannot adapt to new data.