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 #5

kulubecioglu mehmet
IM-14 FIOT

Implementing an InceptionV3 Model for Deep Learning Using TensorFlow

Kyiv
IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE
2024

Step 1: Importing Required Libraries

```
In [4]: import tensorflow as tf
```

- In this step, we import the TensorFlow library, which is necessary for building the deep learning model.

Step 2: Defining the InceptionV3Model Class

```
# Define Inception V3 model (modify as needed)
class InceptionV3Model(tf.keras.Model):
    def __init__(self, num_classes):
        super(InceptionV3Model, self).__init__()
```

- We define a class named InceptionV3Model. This class represents a model using the Inception architecture. The __init__ function is used for initializing the model.

Defining the Inception Module

```
# Define Inception modules with varied filter
def inception_module(x, num_filters):
```

- The inception_module function contains the layers that make up an Inception module, including filters of different sizes and bottleneck layers.

Step 3: Defining the Model Architecture

```
        # Example model architecture (modify as needed)
        input_layer = tf.keras.layers.Input(shape=(224, 224, 3))
        x = input_layer

        # Add Inception modules
        for _ in range(3):  # Add three Inception modules (adjust as needed)
            x = inception_module(x, 256)

        # Global average pooling
        x = tf.keras.layers.GlobalAveragePooling2D()(x)

        # Fully connected layers
        x = tf.keras.layers.Dense(512, activation='relu')(x)
        x = tf.keras.layers.Dropout(0.5)(x)

        # Output layer
        output_layer = tf.keras.layers.Dense(num_classes, activation='softmax')(x)

        # Create the model
        self.model = tf.keras.Model(inputs=input_layer, outputs=output_layer)
        # Build the model
        self.build((None, 224, 224, 3))
```

- In this section, we define the model architecture. We specify the input layer, add three Inception modules, include global average pooling, fully connected layers, and an output layer.

Step 4: Defining the call Function

```
    def call(self, inputs, training=None, mask=None):
        return self.model(inputs)
```

- This function specifies what should happen when the model is called. Here, we call the inner model.

Step 5: Creating the Model and Displaying its Summary

```
# Create an instance of the InceptionV3Model class
num_classes = 10  # Replace with the actual number of classes
inception_model = InceptionV3Model(num_classes)

# Display the model summary
inception_model.summary()
```

Output:

```
Model: "inception_v3_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 model (Functional)          (None, 10)                1245706

=================================================================
Total params: 1245706 (4.75 MB)
Trainable params: 1245706 (4.75 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```