USN NUMBER:1RVU22CSE018
NAME:Akshay B

Ex No: 5 Date: 04-09-2024	Transfer Learning

Objective:

The objective of this lab is to build a flower classification model using the MobileNet V2 pre-trained model from TensorFlow Hub. The model will classify different types of flowers (daisy, dandelion, roses, sunflowers, and tulips) using transfer learning. We will preprocess the dataset, train the model, and evaluate its performance.

Descriptions:

The code demonstrates how to use the MobileNet V2 model, pre-trained on ImageNet, for flower classification. The model is first loaded using TensorFlow Hub and frozen so that its weights are not updated during training. The dataset is prepared by downloading a set of flower images, resizing them to the required dimensions, and normalizing the pixel values. A new model is then created using the pre-trained model as a feature extractor, followed by a dense layer to classify the images into the five flower categories. The model is trained, evaluated, and then used to make predictions on unseen images.

```
[]
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_2 (KerasLayer)  (None, 1280)              2257984

 dense (Dense)               (None, 5)                 6405

=================================================================
Total params: 2264389 (8.64 MB)
Trainable params: 6405 (25.02 KB)
Non-trainable params: 2257984 (8.61 MB)
_________________________________________________________________
```

Steps to Build the Model:

Install Necessary Libraries: Install TensorFlow, TensorFlow Hub, and OpenCV.

Import Libraries: Import all required libraries, such as TensorFlow, TensorFlow Hub,

USN NUMBER:1RVU22CSE018
NAME:Akshay B

OpenCV, PIL, NumPy, etc.

Define Image Shape: Specify the shape of the images to be used.

Load Pre-Trained Model: Use a pre-trained MobileNet V2 model from TensorFlow Hub for feature extraction.

Load and Preprocess Images: Load images from the dataset, resize, normalize, and prepare them for input to the model.

Create a Model: Define a Sequential model using the pre-trained MobileNet V2 as a feature extractor and add a dense output layer for classification.

Compile the Model: Compile the model with an appropriate loss function and optimizer.

Train the Model: Fit the model using the training dataset.

Evaluate the Model: Evaluate the model's performance on the test dataset.

Make Predictions: Use the trained model to predict the class of new images.

```
mobilenet_v2
="https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4"
#inception_v3                                                    =
"https://tfhub.dev/google/imagenet/inception_v3/classification/5"

classifier_model = mobilenet_v2
```

Selecting a MobileNetV2 pre-trained model from TensorFlow Hub and wrapping it as a Keras layer

```
IMAGE_SHAPE = (224, 224)


# Load the pre-trained MobileNet V2 model from TensorFlow Hub
mobilenet_model = hub.KerasLayer(

"https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4",
    input_shape=IMAGE_SHAPE+(3,),  # Specify input shape with color
channels
    trainable=False  # Freeze the pre-trained weights
)
```

USN NUMBER:1RVU22CSE018
NAME:Akshay B

```
# Create a Sequential model

classifier = tf.keras.Sequential([

    tf.keras.layers.Lambda(lambda x: mobilenet_model.call(x))

])
```

The input shape is defined, and the model is set to non-trainable to prevent updates to its weights during training.

Creates a Sequential model using the pre-trained MobileNet model as a feature extractor.

```
# gold_fish =
tf.keras.utils.get_file('image.jpg','https://storage.googleapis.com/do
wnload.tensorflow.org/example_images/gold_fish.jpg')



gold_fish =
Image.open("/kaggle/input/4seprvu/goldfish.jpg").resize(IMAGE_SHAPE)

gold_fish = np.array(gold_fish)/255.0

gold_fish[np.newaxis, …]

gold_fish.shape

gold_fish
```

Opens an example image, resizes it, and normalizes the pixel values to be between 0 and 1.

```
result = classifier.predict(gold_fish[np.newaxis, ...])
result.shape
```

Predicts the class of the example image using the pre-trained model and identifies the index of the predicted label.

```
The result is a 1001-element vector of logits, rating the probability
of each class for the image.
```

```
#
tf.keras.utils.get_file('ImageNetLabels.txt','https://storage.googleap
```

USN NUMBER:1RVU22CSE018
NAME:Akshay B

```
is.com/download.tensorflow.org/data/ImageNetLabels.txt')
image_labels = [] # a list

# load image lables from a text file
# This code assumes that the file "ImageNetLabels.txt" contains one
label per line.
with open("/kaggle/input/4seprvu/imagenet_labels.txt", "r") as f: #
with is used to close the file automatically
    image_labels = f.read().splitlines()

image_labels[:5] # print the first five labels
```

Store the image_labels into a list from a txt file

```
image_labels[predicted_label_index]
```

Printing the predicted label index

```
dataset_url                                                          =
"https://storage.googleapis.com/download.tensorflow.org/example_images
/flower_photos.tgz"

data_dir          =          tf.keras.utils.get_file('flower_photos',
origin=dataset_url,  cache_dir='.', untar=True)
```

Loading the flower dataset

```
flowers_images_dict = {
    'roses': list(data_dir.glob('roses/*')),
    'daisy': list(data_dir.glob('daisy/*')),
    'dandelion': list(data_dir.glob('dandelion/*')),
    'sunflowers': list(data_dir.glob('sunflowers/*')),
    'tulips': list(data_dir.glob('tulips/*')),
}
flowers_labels_dict = {
    'roses': 0,
    'daisy': 1,
    'dandelion': 2,
    'sunflowers': 3,
    'tulips': 4,
```

USN NUMBER:1RVU22CSE018
NAME:Akshay B

```
}
```

Reading the flower images as a numpy array and assigning it an index value

```
X, y = [], []

for flower_name, images in flowers_images_dict.items():
    for image in images:
        img = cv2.imread( str(image) )
        if img is not None:
            resized_img = cv2.resize(img, (224, 224))
            X.append(resized_img)
            y.append(flowers_labels_dict[flower_name])
        else:
            print(f"Error reading image: {image}")
            continue
```

Adding image and corresponding labels as X and y

```
X = np.array(X)
y = np.array(y)
```

Storing X and y as a np array

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
X_train_scaled = X_train / 255
X_test_scaled = X_test / 255
```

Splitting the dataset into train, test batches

```
feature_extractor_model =
"https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"

pretrained_model_without_top_layer = hub.KerasLayer(
            feature_extractor_model, input_shape=(224, 224, 3),
trainable=False)
feature_extractor_model =
"https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"

pretrained_model_without_top_layer = hub.KerasLayer(
            feature_extractor_model, input_shape=(224, 224, 3),
```

USN NUMBER:1RVU22CSE018
NAME:Akshay B

```
trainable=False)

num_of_flowers = 5

model = tf_keras.Sequential([
  pretrained_model_without_top_layer,
  tf_keras.layers.Dense(num_of_flowers)
])
pretrained_model_params                                                =
pretrained_model_without_top_layer.trainable_weights
print(pretrained_model_params)
```

Defines and compiles a new Sequential model with the pre-trained MobileNet V2 as the feature extractor and an additional dense layer for classification.

```
model.compile(
  optimizer="adam",

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['acc'])

model.fit(X_train_scaled, y_train, epochs=5)
model.evaluate(X_test_scaled,y_test)
model.evaluate(X_test_scaled,y_test)
```

compiling and running the new model

```
from      tensorflow.keras.preprocessing.image      import      load_img,
img_to_array
def preprocess_image(image_path):
    img = load_img(image_path, target_size=(224, 224))
    img_array = img_to_array(img)
        img_array  =  np.expand_dims(img_array,  axis=0)    #  Add  batch
dimension
    img_array /= 255.0  # Normalize to [0, 1] range
    return img_array
test_image_path = '/kaggle/input/4seprvu/sun.jpg'

test_image = preprocess_image(test_image_path)
```

USN NUMBER:1RVU22CSE018
NAME:Akshay B

```
predictions = model.predict(test_image)
predicted_class = np.argmax(predictions, axis=1)

print(f"Predicted class: {predicted_class}")

# Example class names (should match your dataset structure)
class_names = ['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
print(f"Predicted flower: {class_names[predicted_class[0]]}")
```

Predicting on a test image after preprocessing it and normalising it

Updates: No updates

GitHubLink:
https://github.com/Akb-25/Foundations-of-Deep-Learning/blob/main/transfer-learning.ipynb