DL Exp: 6

Aim: Object detection using Transfer learning CNN architectures.

a. load pre trained CNN model
b. freeze parameters
c. Add custom classifier.
d. Train classifier layers on training data available
e. fine tune hyper parameters and unfreeze more layers as needed.

Objective: To load pretrained model & improve performance by Transfer learning.

Infrastructure: Computer / Laptop / VM

Software used: Jupyter Notebook / Google colab, Tensorflow, keras.

Theory: Refers to process where a model trained on one problem is used in some way on a second related problem. Transfer learning has the benefit of decreasing the training time for neural network and results in low generalisation error.

The weights is reused layers maybe used as starting point for training process and adapted. This maybe useful when the first related problem has a lot more labelled data than problem of interest and both contexts.

How to use pre trained models:

- classifier = directly classify new images.
- standalone feature extractor = preprocess images.
- integrated feature extractor = but layers of pretrained model are trained in concert with new model.
- weight initialization: some portion of model is integrated into new model and layers are trained with new model.

It may not be clear to which usage of pre trained model may yield the best results on your new computer vision task.

Ways to fine tune model

1) Feature extraction: we can use pre trained model as feature extraction mechanism. we can remove the output layer and then use entire network as fixed feature extractor for new data set.
2) Use architecture of pre trained model: what we can do is we keep the weights randomly and train the model frozen while we retrain only higher layers.

→ Building A Deep learning Based Object Detection Model:

Training a performing deep learning model for obj detection takes a lot data and Computing power. To facilitate the dev, we can use transfer learning by fining tuning models pre trained model is to train partially. We can try and test as to how many layers to be frozen and how many to be trained.

→ Building A deep learning Based obj Detection model:

Training a performing deep learning model for object detection takes a lot of data and computing power. To facilitate dev, we can use transfer learning based on other rel datasets. Since there are multiple backend logistics such as paths and hyper parameters to take care of when training a full scale deep learning model, we can create a central dictionary to store these configuration parameters, including setting up diff paths, installing relevant libraries, and downloading pre trained models.

→ Conclusion: we conclude from exp, how to dev a model for specific application with help of transfer learning architecture in DL.

DL-6

65

```
import tensorflow as tf
import numpy as np
import cv2
import PIL.Image as Image
import os
import matplotlib.pyplot as plt
import tensorflow_hub as hub
import pathlib
```

```
Image_Shape = (224,224)
```

```
URL_dataset = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_p
```

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

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_im
228813984/228813984 [==============================] - 1s 0us/step
```

```
data_dir = pathlib.Path(data_dir)
```

```
image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
```

```
3670
```

```
flowers_images_dict = {
"daisy" : list(data_dir.glob('daisy/*')),
"dandelion" : list(data_dir.glob('dandelion/*')),
"roses" : list(data_dir.glob('roses/*')),
"sunflowers" : list(data_dir.glob('sunflowers/*')),
"tulips" : list(data_dir.glob('tulips/*'))
}
```

```
flowers_labels_dict= {
"daisy" : 0,
"dandelion" : 1,
"roses" : 2,
"sunflowers" : 3,
"tulips" : 4
```

67

```
}

X, Y = [],[]


for flower_name, images in flowers_images_dict.items():
  for image in images:
    img = cv2.imread(str(image))
    resized_img = cv2.resize(img, Image_Shape)
    X.append(resized_img)
    Y.append(flowers_labels_dict[flower_name])
X = np.array(X)
y = np.array(Y)


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


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


classifier = tf.keras.Sequential([
hub.KerasLayer(tf_model,input_shape=(224,224,3), trainable=False),
tf.keras.layers.Dense(len(flowers_labels_dict), activation="softmax")
])
classifier.summary()
classifier.compile(
optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=["accuracy"]
)
classifier.fit(X_train_scaled, y_train,epochs=5)
classifier.evaluate(X_test_scaled, y_test)
```

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

 dense (Dense)               (None, 5)                 6405

=================================================================
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
_________________________________________________________________
Epoch 1/5
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWar
```

```
    return dispatch_target(*args, **kwargs)
86/86 [==============================] - 82s 901ms/step - loss: 0.8665 - accuracy: 0.66
Epoch 2/5
86/86 [==============================] - 78s 908ms/step - loss: 0.4247 - accuracy: 0.84
Epoch 3/5
86/86 [==============================] - 80s 926ms/step - loss: 0.3304 - accuracy: 0.88
Epoch 4/5
86/86 [==============================] - 78s 904ms/step - loss: 0.2794 - accuracy: 0.91
Epoch 5/5
86/86 [==============================] - 77s 901ms/step - loss: 0.2366 - accuracy: 0.92
29/29 [==============================] - 27s 901ms/step - loss: 0.3671 - accuracy: 0.87
[0.367097318172454833, 0.8779956698417664]
```

```
from IPython import display
display.Image('/content/drive/MyDrive/rose.jpg',width=200,height=200)
```

```
from PIL import Image

img = Image.open("/content/drive/MyDrive/rose.jpg")
img = tf.keras.preprocessing.image.img_to_array(img.resize(Image_Shape))
img = np.array([img])
res = classifier.predict(img)
print("The prediction is : {}".format(list(flowers_labels_dict.keys())[np.argmax(res)]))
```

```
1/1 [==============================] - 0s 56ms/step
The prediction is : roses
```