Transfer Learning With Tensorflow

Practice of Artificial Intelegence

Name : 劉泰佑
Id : C100E030

1. Import all the required libraries

```
import numpy as np
import time

import PIL.Image as Image
import matplotlib.pylab as plt

import tensorflow as tf
import tensorflow_hub as hub

import datetime
```

```
▶ Launch TensorBoard Session
%load_ext tensorboard
```

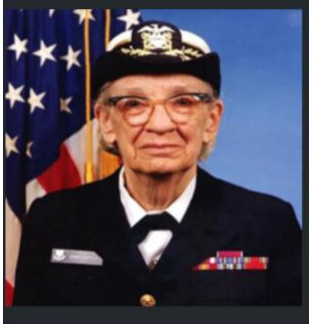
2. Download pre-trained model

```
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
```

3. Try running on single image

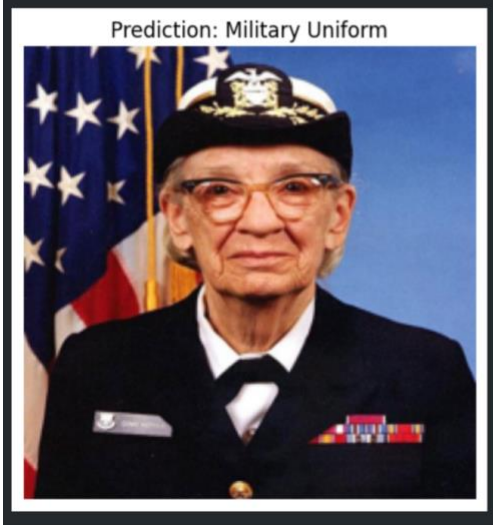
```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/grace_hopper.jpg
61306/61306 [==============================] - 0s 1us/step
```



4. Prediction

```
plt.imshow(grace_hopper)
plt.axis('off')
predicted_class_name = imagenet_labels[predicted_class]
_ = plt.title("Prediction: " + predicted_class_name.title())
```

✓ 0.1s



5. Simple transfer learning: download flower dataset

```
data_root = tf.keras.utils.get_file(
    'flower_photos',
    'https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz',
    untar=True)
```

✓ 1m 28.1s

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz
228813984/228813984 [==============================] - 87s 0us/step
```

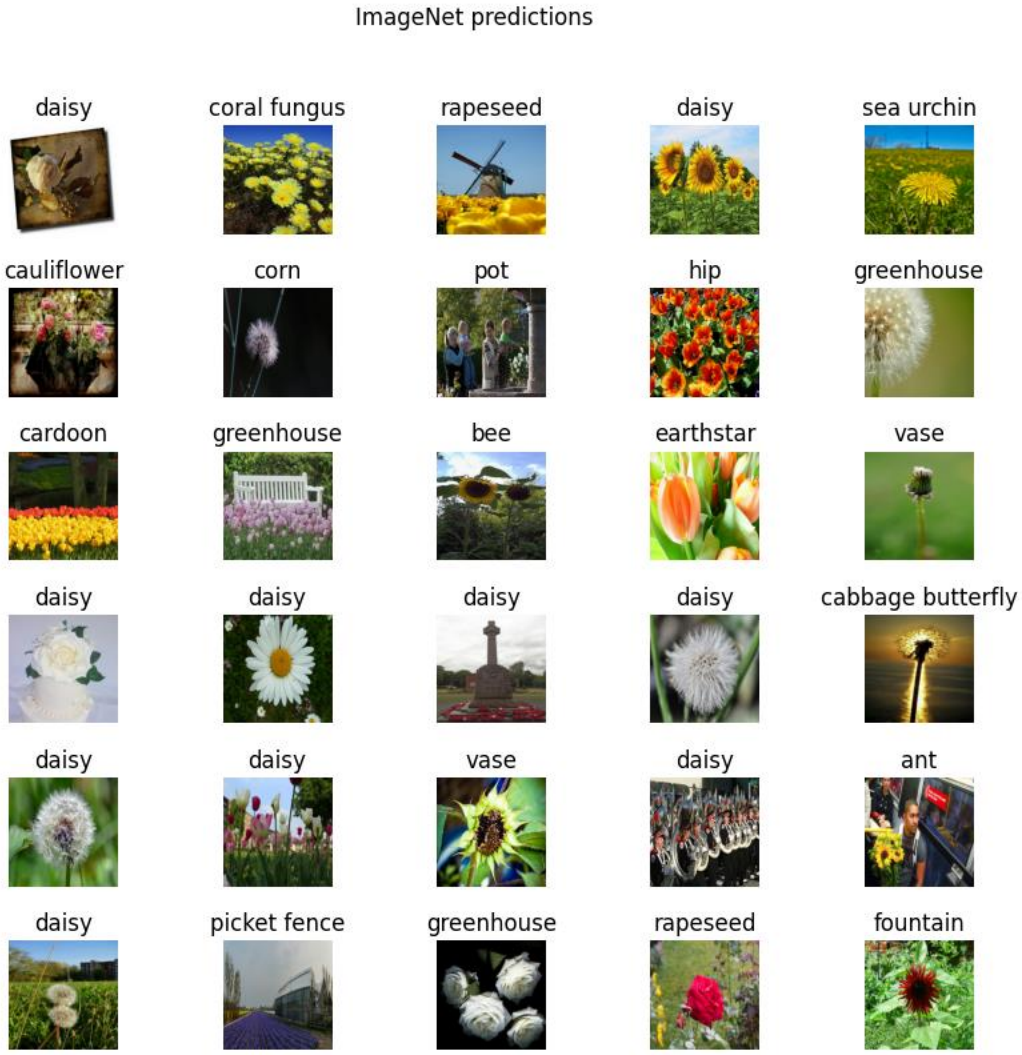
6. Classes

```
class_names = np.array(train_ds.class_names)
print(class_names)
```

✓ 0.2s

```
['daisy' 'dandelion' 'roses' 'sunflowers' 'tulips']
```

7. First imageNet prediction



8. Try using the headless model: re-download the pre-trained models

```
mobilenet_v2 = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
inception_v3 = "https://tfhub.dev/google/tf2-preview/inception_v3/feature_vector/4"

feature_extractor_model = mobilenet_v2
```

9. Attach the classification head

```
num_classes = len(class_names)

model = tf.keras.Sequential([
    feature_extractor_layer,
    tf.keras.layers.Dense(num_classes)
])

model.summary()
```

✓ 0.2s

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_1 (KerasLayer)  (None, 1280)              2257984

 dense (Dense)               (None, 5)                 6405

=================================================================
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
_________________________________________________________________
```

10. Model training

```
Epoch 1/10

2022-11-25 12:01:36.644869: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer
for device_type GPU is enabled.

92/92 [==============================] - ETA: 0s - loss: 0.7253 - acc: 0.7371

2022-11-25 12:02:21.817201: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer
for device_type GPU is enabled.

92/92 [==============================] - 60s 597ms/step - loss: 0.7253 - acc: 0.7371 - val_loss: 0.4195 - val_acc: 0.8733
Epoch 2/10
92/92 [==============================] - 51s 552ms/step - loss: 0.3697 - acc: 0.8736 - val_loss: 0.3403 - val_acc: 0.8937
Epoch 3/10
92/92 [==============================] - 76s 825ms/step - loss: 0.2901 - acc: 0.9063 - val_loss: 0.3112 - val_acc: 0.9046
Epoch 4/10
92/92 [==============================] - 53s 577ms/step - loss: 0.2415 - acc: 0.9292 - val_loss: 0.2979 - val_acc: 0.9005
Epoch 5/10
92/92 [==============================] - 50s 540ms/step - loss: 0.2070 - acc: 0.9441 - val_loss: 0.2913 - val_acc: 0.9033
Epoch 6/10
92/92 [==============================] - 52s 567ms/step - loss: 0.1805 - acc: 0.9530 - val_loss: 0.2878 - val_acc: 0.9019
Epoch 7/10
92/92 [==============================] - 50s 538ms/step - loss: 0.1592 - acc: 0.9598 - val_loss: 0.2856 - val_acc: 0.9005
Epoch 8/10
92/92 [==============================] - 49s 530ms/step - loss: 0.1417 - acc: 0.9656 - val_loss: 0.2841 - val_acc: 0.9033
Epoch 9/10
92/92 [==============================] - 55s 592ms/step - loss: 0.1269 - acc: 0.9707 - val_loss: 0.2827 - val_acc: 0.9019
Epoch 10/10
92/92 [==============================] - ETA: 0s - loss: 0.1143 - acc: 0.9751
```

11. Check the predictions

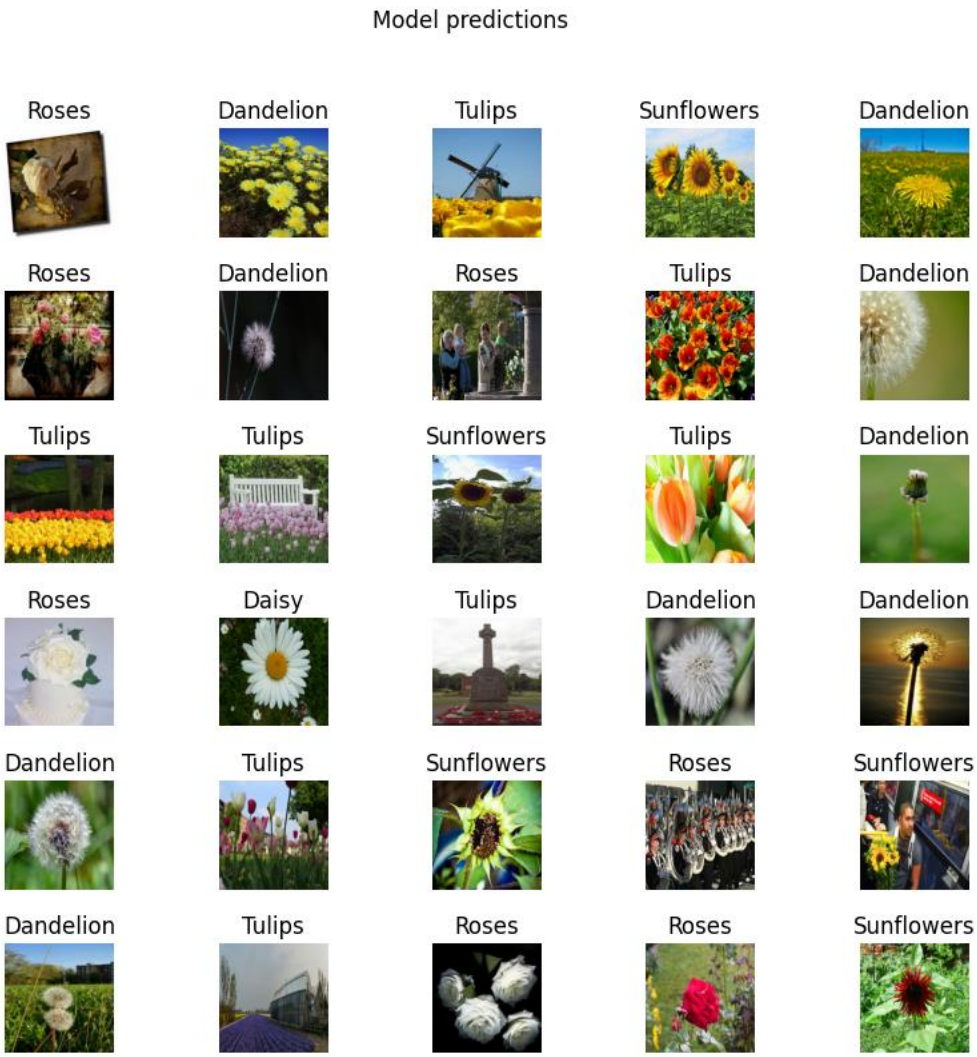
```
predicted_batch = model.predict(image_batch)
predicted_id = tf.math.argmax(predicted_batch, axis=-1)
predicted_label_batch = class_names[predicted_id]
print(predicted_label_batch)
```

✓ 2.9s Python

```
2022-11-25 12:10:48.268691: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer
for device_type GPU is enabled.

1/1 [==============================] - 2s 2s/step
['roses' 'dandelion' 'tulips' 'sunflowers' 'dandelion' 'roses' 'dandelion'
 'roses' 'tulips' 'dandelion' 'tulips' 'tulips' 'sunflowers' 'tulips'
 'dandelion' 'roses' 'daisy' 'tulips' 'dandelion' 'dandelion' 'dandelion'
 'tulips' 'sunflowers' 'roses' 'sunflowers' 'dandelion' 'tulips' 'roses'
 'roses' 'sunflowers' 'tulips' 'sunflowers']
```

12. Plot the model prediction



13. Export and reload model

```
t = time.time()

export_path = "Model/saved_models/{}".format(int(t))
model.save(export_path)

export_path
```

✓ 6.5s Pyth

```
INFO:tensorflow:Assets written to: Model/saved_models/1669350143/assets

INFO:tensorflow:Assets written to: Model/saved_models/1669350143/assets

'Model/saved_models/1669350143'
```

```
reloaded = tf.keras.models.load_model(export_path)
```

✓ 6.2s Pyth

+ Code + Markdown

```
result_batch = model.predict(image_batch)
reloaded_result_batch = reloaded.predict(image_batch)
```

✓ 4.7s Pyth

```
1/1 [==============================] - 3s 3s/step

2022-11-25 12:22:39.611500: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin optimizer
for device_type GPU is enabled.

1/1 [==============================] - 1s 1s/step
```

14. Re-run the predictions

```
reloaded_predicted_id = tf.math.argmax(reloaded_result_batch, axis=-1)
reloaded_predicted_label_batch = class_names[reloaded_predicted_id]
print(reloaded_predicted_label_batch)
```

✓ 0.1s

```
['roses' 'dandelion' 'tulips' 'sunflowers' 'dandelion' 'roses' 'dandelion'
 'roses' 'tulips' 'dandelion' 'tulips' 'tulips' 'sunflowers' 'tulips'
 'dandelion' 'roses' 'daisy' 'tulips' 'dandelion' 'dandelion' 'dandelion'
 'tulips' 'sunflowers' 'roses' 'sunflowers' 'dandelion' 'tulips' 'roses'
 'roses' 'sunflowers' 'tulips' 'sunflowers']
```

```
plt.figure(figsize=(10,9))
plt.subplots_adjust(hspace=0.5)
for n in range(30):
    plt.subplot(6,5,n+1)
    plt.imshow(image_batch[n])
    plt.title(reloaded_predicted_label_batch[n].title())
    plt.axis('off')
_ = plt.suptitle("Model predictions")
```

✓ 0.6s

15. Final results