Experiment No. 5
Explore the working of any pre-trained model towards outcome generation
Date of Performance:
Date of Submission:

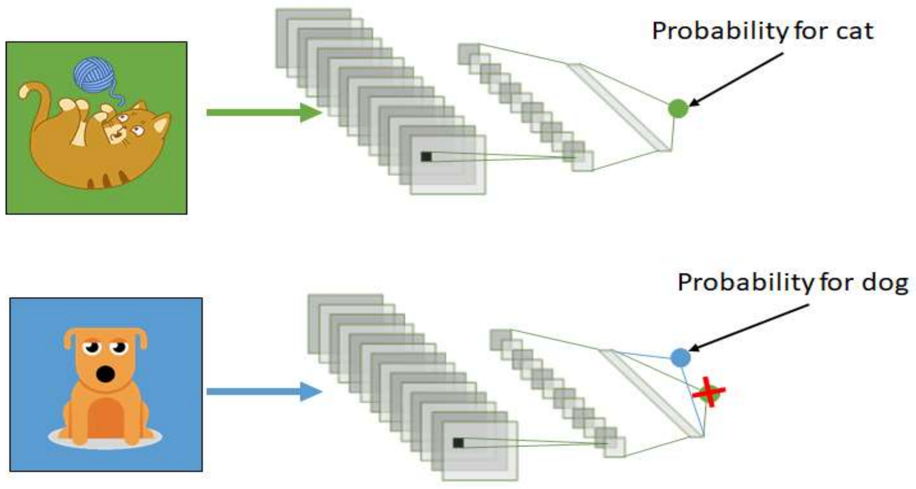
Aim: Explore the working of any pre-trained model towards outcome generation.

Objective: Ability to make use of the pretrained models for a different purpose using the concepts of transfer learning.

Theory:

The reuse of a pre-trained model on a new problem is known as transfer learning in machine learning. A machine uses the knowledge learned from a prior assignment to increase prediction about a new task in transfer learning. You could, for example, use the information gained during training to distinguish beverages when training a classifier to predict whether an image contains cuisine.

The knowledge of an already trained machine learning model is transferred to a different but closely linked problem throughout transfer learning. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the model's training knowledge to identify other objects such as sunglasses.



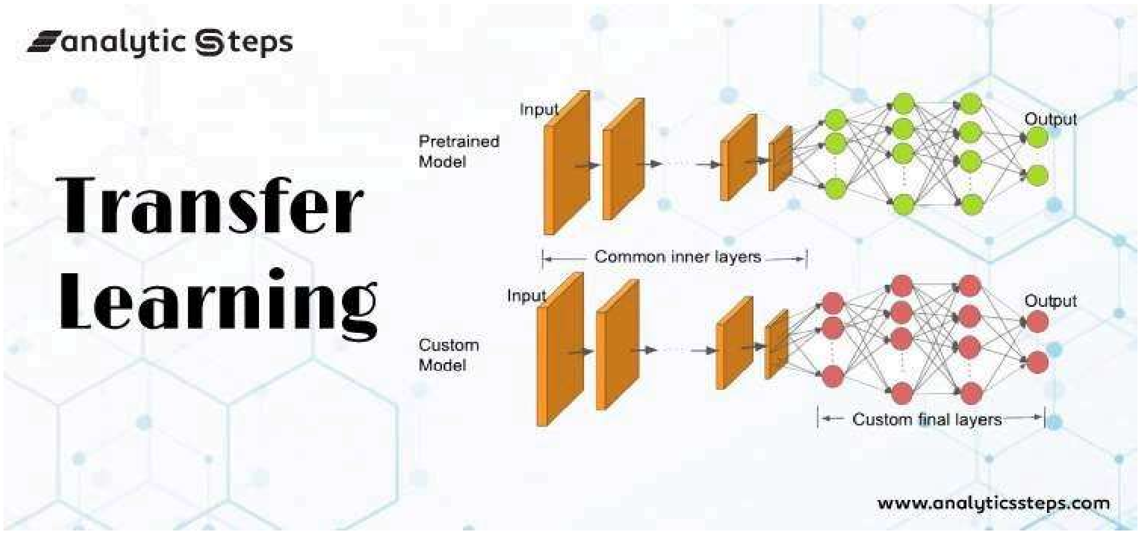
With transfer learning, we basically try to use what we've learned in one task to better understand the concepts in another. weights are being automatically being shifted to a network performing "task A" from a network that performed "task B."

Because of the massive amount of CPU power required, transfer learning is typically applied in computer vision and natural language processing tasks like sentiment analysis.

How Transfer Learning Works?

In computer vision, neural networks typically aim to detect edges in the first layer, forms in the middle layer, and task-specific features in the latter layers.

The early and central layers are employed in transfer learning, and the latter layers are only retrained. It makes use of the labelled data from the task it was trained on.



Let's return to the example of a model that has been intended to identify a backpack in an image and will now be used to detect sunglasses. Because the model has trained to recognise objects in the earlier levels, we will simply retrain the subsequent layers to understand what distinguishes sunglasses from other objects.

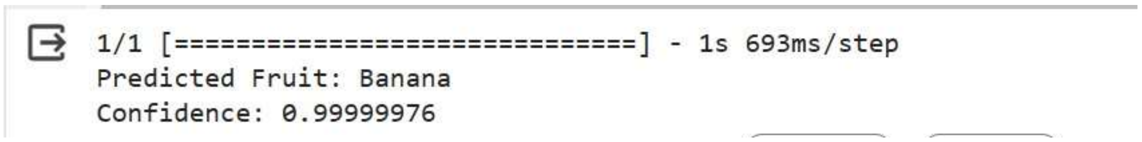
Code:

```
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
import numpy as np

# Load the pre-trained VGG16 model without the top layer (fully connected layers)
```

```
base_model      =      VGG16(weights='imagenet',      include_top=False,
input_shape=(224, 224, 3))

#  Freeze  the  convolutional  base  to  prevent  it  from  being  trained
further

base_model.trainable = False

# Add a new fully connected layer for our classification task

x = tf.keras.layers.Flatten()(base_model.output)
x = tf.keras.layers.Dense(512, activation='relu')(x)
output = tf.keras.layers.Dense(10, activation='softmax')(x)
# 10 classes for fruits

# Create the new model

model = tf.keras.models.Model(inputs=base_model.input, outputs=output)

# Compile the model

model.compile(optimizer='adam',  loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Placeholder image for demonstration

image_path = '/content/pexels-brian-van-den-heuvel-1313267.jpg'

# Define class labels for fruits

class_labels  =  ["Apple",  "Banana",  "Orange",  "Strawberry",  "Grape",
"Pineapple", "Watermelon", "Mango", "Peach", "Kiwi"]

# Once trained, you can use this model to classify images of fruits

def predict_fruit(image_path):
    img = image.load_img(image_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)
    prediction = model.predict(img_array)
    predicted_class = np.argmax(prediction)
    predicted_label = class_labels[predicted_class]
    confidence = prediction[0][predicted_class]
    return predicted_label, confidence

# Example usage:
predicted_label, confidence = predict_fruit(image_path)
print("Predicted Fruit:", predicted_label)
print("Confidence:", confidence)
```

Input:

Output:



```
1/1 [==============================] - 1s 693ms/step
Predicted Fruit: Banana
Confidence: 0.99999976
```

Conclusion:

Overall, the use of a pre-trained VGG16 model for transfer learning in the provided code is an effective approach for accurate fruit classification. The model leverages its pre-existing knowledge from ImageNet to achieve good performance on the fruit dataset, demonstrating the effectiveness of transfer learning for improving accuracy in classification tasks.