WEEK-14

Aim:- To implement a pre-trained CNN model as a feature extractor using transfer learning models.

Objective:-

- To understand transfer learning and feature extraction.
- To reuse features learned from Imagenet for a new task.
- To reduce training time and improve accuracy.
- Replace and train the final classification layer on a new dataset

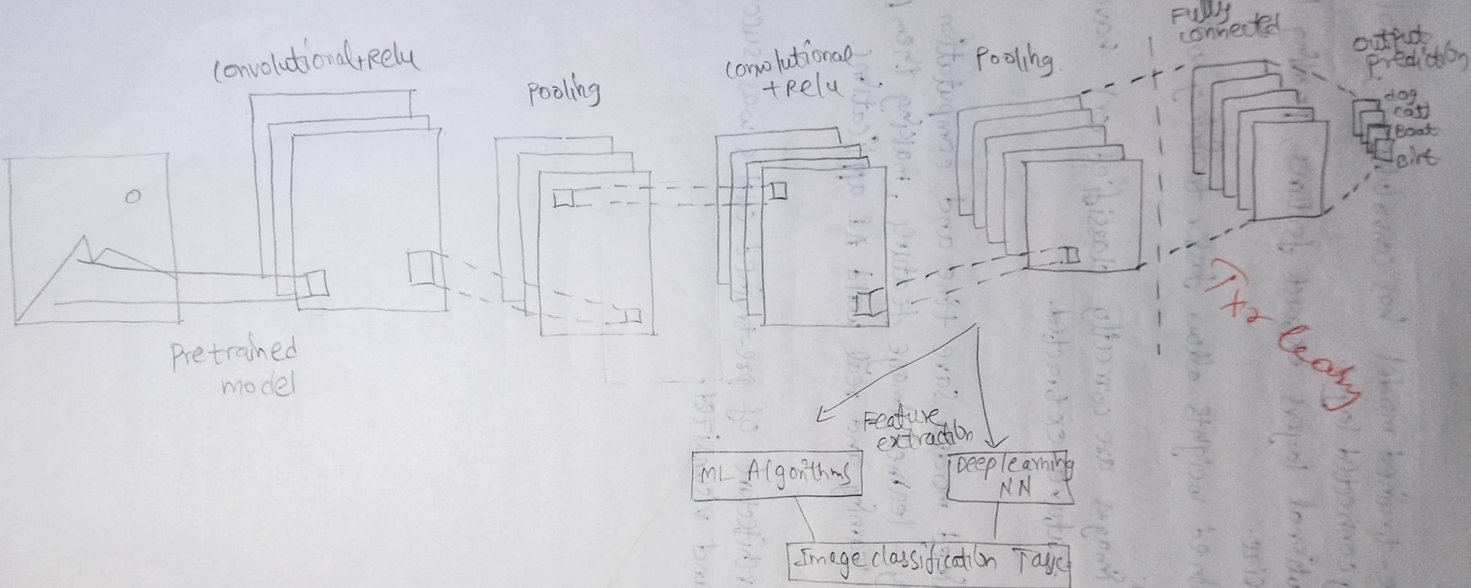
Algorithm:-

1.) Load the dataset (eg:- cats vs dogs)
2.) Load pre-trained model.
3.) Freeze convolutional base layers.
4.) Add new custom dense layers for classification.
5.) compile and train on small dataset
6.) Evaluate accuracy.

Pseudocode:-

```
    Load VGG16(weights='imagenet', include_top=false)
    Freeze all layers
    Add Flatten() -> Dense(128, relu) ->
      Dense(num_classes, softmax)
    model = pretrained_base + New_ head.
compile(optimizer='adam', loss='categorical_crossentropy')
Train on new dataset
Evaluate performance.
```

TRANSFER LEARNING



Output :-

Epoch [1/5], Loss : 0.8481

Epoch [2/5], Loss : 0.6266

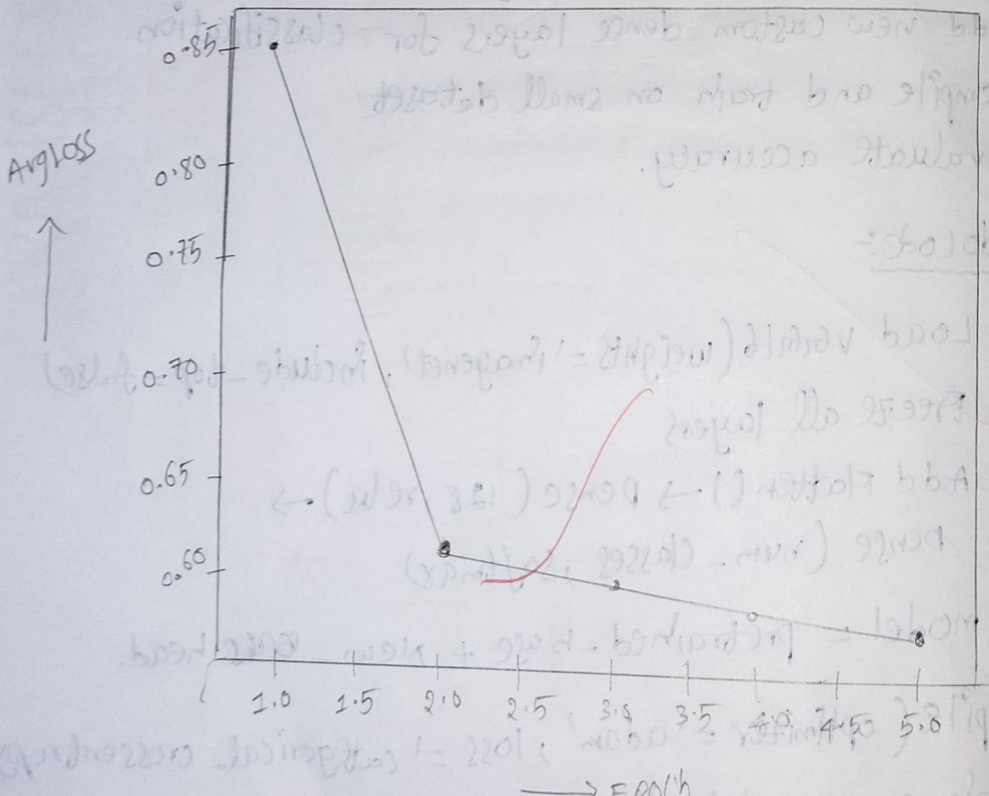
Epoch [3/5], Loss : 0.5937

Epoch [4/5], Loss : 0.5797

Epoch [5/5], Loss : 0.5699

Training complete using pretrained ResNet 18 as Feature extraction.

Training Loss curve - Transfer learning with ResNet 18.



observation :-

→ Ther pre trained ResNet 18 achieved good accuracy even with few epochs.

→ Training loss decreased steadily, confirming effective feature use.

→ Test accuracy remained stable, showing generalization from pre-learned features.

→ model performed well even with small dataset.

conclusion :-

Transfer learning allows reuse of learned visual features from large datasets, improving model accuracy and reducing training cost for new applications.

Result :-

The pre trained CNN successfully extracted useful features and provided high classification accuracy on a new dataset

```python
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()

x_train, y_train = x_train[:10000], y_train[:10000]
x_test, y_test = x_test[:2000], y_test[:2000]
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

base_model = tf.keras.applications.MobileNetV2(
    input_shape=(64, 64, 3),
    include_top=False,
    weights='imagenet'
)
base_model.trainable = False  # freeze layers
model = tf.keras.Sequential([
    layers.Resizing(64, 64),
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

history = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=2,
    batch_size=64,
    verbose=1
)
loss, acc = model.evaluate(x_test, y_test, verbose=0)
print(f"\n✅ Test Accuracy: {acc:.4f}")
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.legend()
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-1693943737.py:18: UserWarning: `input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
  base_model = tf.keras.applications.MobileNetV2(
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/2
157/157 ━━━━━━━━━━━━━━━━━━━━ 43s 239ms/step - accuracy: 0.3827 - loss: 1.8126 - val_accuracy: 0.6035 - val_loss: 1.1824
Epoch 2/2
157/157 ━━━━━━━━━━━━━━━━━━━━ 40s 257ms/step - accuracy: 0.6609 - loss: 1.0089 - val_accuracy: 0.6180 - val_loss: 1.1267

✅ Test Accuracy: 0.6180
```