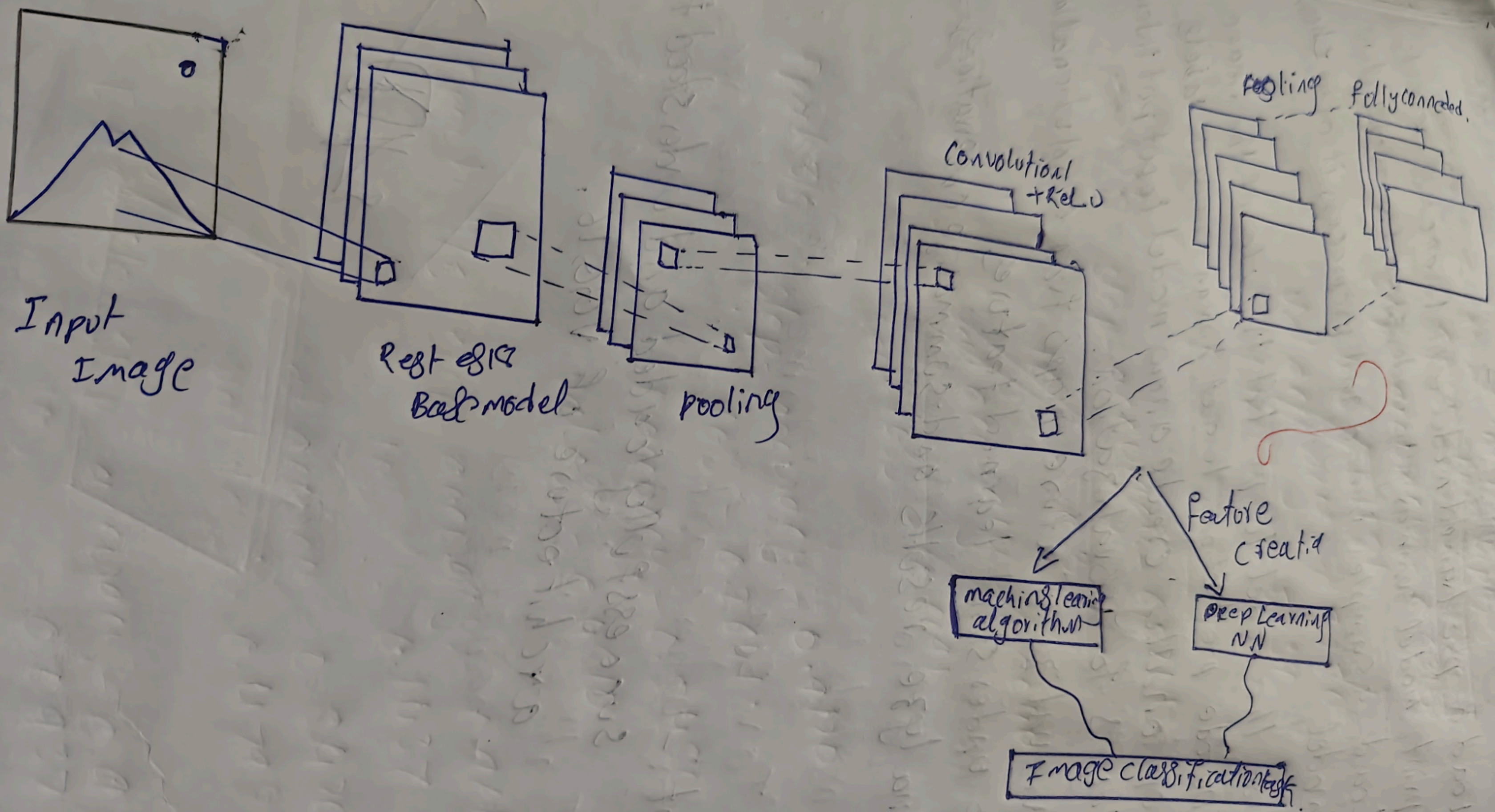
ex:-14:- Implement a pre-trained CNN model as Feature Extractor using Transfer learning models.

Aim:- To implement a memory-efficient pre trained CNN as a feature extractor for image classification on CIFAR-10.

Objectives:-

1. Load a pre trained MobileNetV2 model trained on image Net
2. Freeze all pre-trained layers to prevent weight updates
3. Resize and pre process CIFAR-10 images to match MobileNetV2 input Requirements.
4. Build a classifier on top of mobile net V2 using Global Average Pooling 2D and Dense layers.
5. Train the classifier directly on batches of image avoiding precomputing and storing features
6. ~~Evaluate the models~~).

Transfer learning.



Pseudocode:-

1. Import tensorflow, keras, numpy.
2. Load CIFAR-10 dataset
3. Resize images to mobile net V2 input size (96x96).
4. preprocess images using preprocess input
5. Load pre trained layers
6. Freeze pre trained layers
7. Build model
   a. mobile net V_2 base
   b. Global Average pooling 2D.
   c. Dense (10, activation = 'Softmax')
   d: Dense (128, activation = 'relu')
8. compile model (optimizer = adam, loss = Sparse_categorical_crossentropy).
9. Train model directly on batches of images.
10. Evaluate model on test data.

Result:- Successfully implemented mobile Net V2 as Featore Extractor using Transfer learning models

output :-

1. memory - efficient :- No precomputed feature arrays
:- model runs on low - ram laptops.

Epoch 1/5 · step - accuracy :- 0.6315
loss :- 1.4181
val accuracy :- 0.8250
val_loss :- 0.5289

Epoch 2/5 step - accuracy - 0.8711
loss :- 0.3891
val accuracy - 0.8040
val-loss - 0.6091

Epoch 3/5 step - accuracy - 0.9120
loss - 0.2641
val accuracy - 0.8480
valloss - 0.4810

Epoch 4/5 . step - accuracy :- 0.9610
loss :- 0.1437
val accuracy :- 0.8470
val_loss :- 0.4798

Epoch 5/5 step_accuracy :- 0.9757
loss :- 0.0937
val accuracy :- 0.8320
val_loss :- 0.5536.

Pseudoc
1. Imp
2. Loa
3. Resi
4. prep
5. Loa
6. Fre
7. Bu
8.
9
R

Welcome to Colab - Colab | Compression on MNIST dataset

colab.research.google.com/#scrollTo=1a5MOUjTid8e

Gmail | Maps | News | Translate | Disney+ Hotstar -... | YouTube | All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

Table of contents

+ Section

[3] 36s

```python
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Use small subset for low RAM
x_train, y_train = x_train[:5000], y_train[:5000]
x_test, y_test = x_test[:1000], y_test[:1000]

# Resize images to 96x96 and preprocess
x_train_resized = tf.image.resize(x_train, (96,96))
x_test_resized = tf.image.resize(x_test, (96,96))
x_train_preprocessed = preprocess_input(x_train_resized)
x_test_preprocessed = preprocess_input(x_test_resized)

# Load MobileNetV2 and freeze layers
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(96,96,3))
base_model.trainable = False

# Build memory-efficient model
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

Variables | Terminal

03:31 | T4 (Python 3)

Welcome to Colab - Colab | Compression on MNIST dataset

colab.research.google.com/#scrollTo=1a5MOUjTid8e

Gmail | Maps | News | Translate | Disney+ Hotstar -... | YouTube | All Bookmarks

Welcome to Colab | Cannot save changes

File Edit View Insert Runtime Tools Help

Share

Commands | + Code | + Text | Run all | Copy to Drive

RAM Disk

Table of contents

+ Section

[3] 36s

```
])

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

# Train model directly
history = model.fit(x_train_preprocessed, y_train, epochs=5, batch_size=32,
                    validation_data=(x_test_preprocessed, y_test))

# Evaluate
loss, accuracy = model.evaluate(x_test_preprocessed, y_test)
print("Test Accuracy:", accuracy)
```

```
Epoch 1/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 22s 76ms/step - accuracy: 0.6315 - loss: 1.1181 - val_accuracy: 0.8250 - val_loss: 0.5289
Epoch 2/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8711 - loss: 0.3891 - val_accuracy: 0.8040 - val_loss: 0.6091
Epoch 3/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - accuracy: 0.9120 - loss: 0.2641 - val_accuracy: 0.8480 - val_loss: 0.4810
Epoch 4/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - accuracy: 0.9610 - loss: 0.1437 - val_accuracy: 0.8470 - val_loss: 0.4798
Epoch 5/5
157/157 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - accuracy: 0.9757 - loss: 0.0937 - val_accuracy: 0.8320 - val_loss: 0.5536
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8478 - loss: 0.5003
Test Accuracy: 0.8320000171661377
```

Variables | Terminal

03:31 | T4 (Python 3)