Week - 14

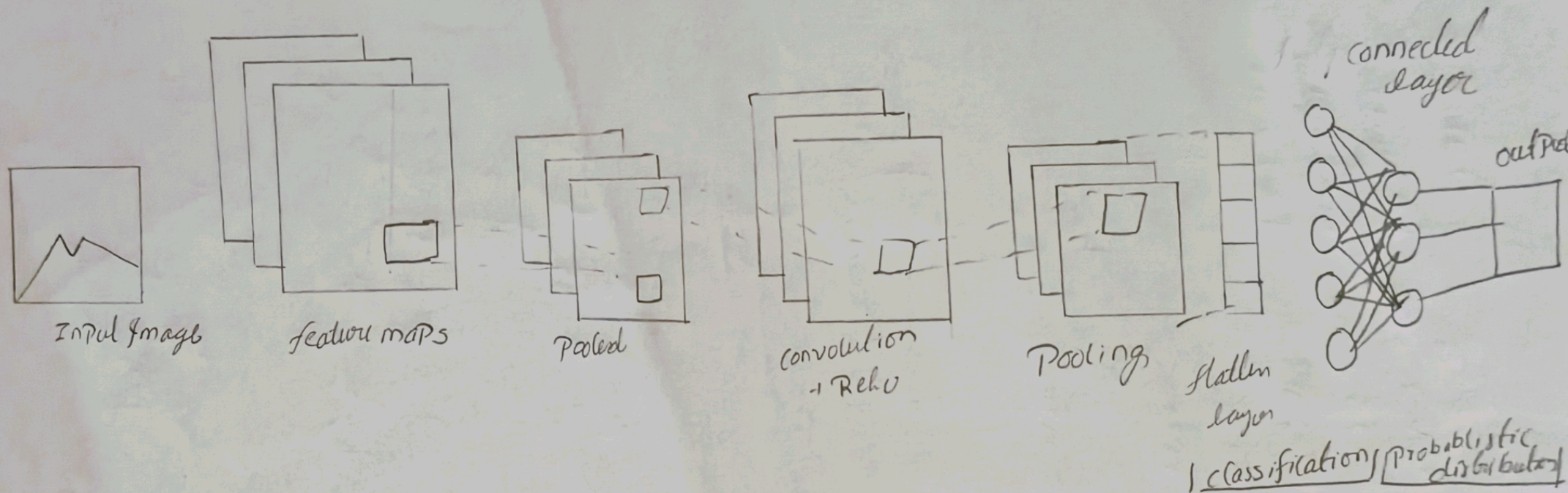
Aim: To implement a PreTrained Convolutional Neural Network such as ResNet50

Algorithm:

* Import Libraries - PyTorch, Torchvision
* Load a Pretrained CNN
* Freeze all convolutional layers so that learned weights are not updated.
* Replace the final classification layer with a new fully connected layer
* Load and Preprocess dataset
* Train only the new classifier using features extracted by frozen CNN base
* Evaluate model Performace on test data

Observation

* Faster Training, since the pretrained CNN layers are already trained on large Dataset.
* Improved Accuracy, Using a Pre Trained model typically improves accuracy over a shallow model.
* The lower layers of CNNs capture universal image features, which are useful across different image



CNN Pretrained Architecture model

Pseudocode:

```
BEGIN
  IMPORT torch, torchvision and required modules
  LOAD PreTrained CNN model
  For each Parameter in CNN base:
      SET require-grad = False
  REPLACE final layer with new Linear layer
  DEFINE loss function (CrossEntropyLoss) and optimizer
  LOAD Dataset and apply transformations (resize)
  For each epoch:
      For each batch (image, labels):
          EXTRACT features using CNN base
          PREDICT using classifier layer
          Compute loss and backpropogate
          UPDATE weights of classifier layer only
      Evaluate model accuracy on test set
  END
```

Observation:

* Training time greatly reduced compared to training from scratch
* The model achieves high Accuracy

Result:

Pre trained model of CNN successfully Implemented

Output:

Epoch [1/5] : Loss : 0.8481

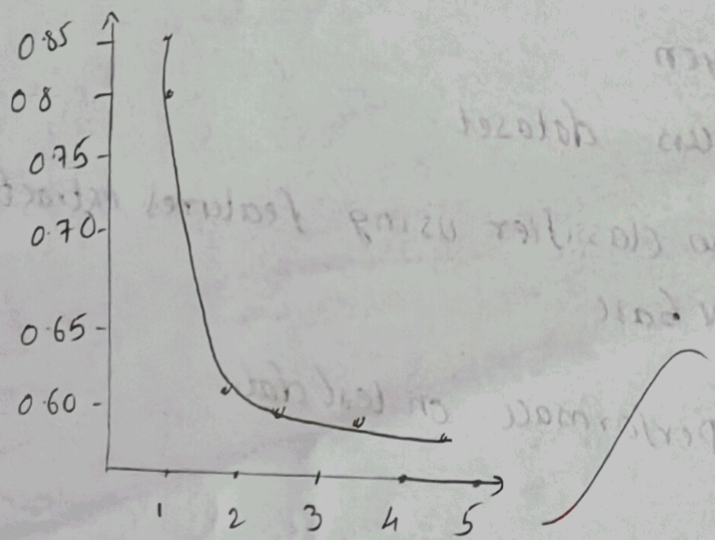
Epoch [2/5] : Loss : 0.6266

Epoch [3/5] : Loss : 0.5937

Epoch [4/5] : Loss : 0.5799

Epoch [5/5] : Los : 0.5699

Training loss curve + Transfer learning Parameter



```
# Experiment 14: Transfer Learning with MobileNetV2 (Fixed Version using CIFAR-10)

import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model

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

# Normalize pixel values
x_train, x_test = x_train / 255.0, x_test / 255.0

# Convert labels to one-hot encoding
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)

# Load pre-trained MobileNetV2 (no top layer)
base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(32,32,3))
base_model.trainable = False  # freeze base

# Add new classifier
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)
```

```
model = Model(inputs=base_model.input, outputs=predictions)


# Compile model

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


# Train the model

history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5,
batch_size=64)


# Evaluate

loss, acc = model.evaluate(x_test, y_test)

print(f"\n Validation Accuracy: {acc:.2f}")
```

```python
# Experiment 14: Transfer Learning with MobileNetV2 (Fixed Version using CIFAR-10)

import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model

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

# Normalize pixel values
x_train, x_test = x_train / 255.0, x_test / 255.0

# Convert labels to one-hot encoding
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)

# Load pre-trained MobileNetV2 (no top layer)
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(32,32,3))
base_model.trainable = False  # freeze base

# Add new classifier
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5, batch_size=64)

# Evaluate
loss, acc = model.evaluate(x_test, y_test)
print(f"\n Validation Accuracy: {acc:.2f}")
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
/tmp/ipython-input-208358661.py:19: UserWarning: `input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224].
  base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(32,32,3))
Epoch 1/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 41s 32ms/step - accuracy: 0.2567 - loss: 2.0496 - val_accuracy: 0.3212 - val_loss: 1.8752
Epoch 2/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.3289 - loss: 1.8464 - val_accuracy: 0.3321 - val_loss: 1.8412
Epoch 3/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.3401 - loss: 1.8144 - val_accuracy: 0.3412 - val_loss: 1.8223
Epoch 4/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.3506 - loss: 1.7849 - val_accuracy: 0.3411 - val_loss: 1.8132
Epoch 5/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 7s 8ms/step - accuracy: 0.3614 - loss: 1.7643 - val_accuracy: 0.3510 - val_loss: 1.8017
313/313 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.3510 - loss: 1.7954

 Validation Accuracy: 0.35
```