as a feature Extractor Using Transfer learning

EXP NO: 14 :- Implement a pre-Trained CNN model

Aim :-

To implement a pre trained CNN model as a feature extractor for image classification using transfer learning

Objectives :-

1. To use a pre-trained model trained on Imagenet as a fixed feature extractor.
2. To extract deep visual features from images for classification tasks.
3. To attach and train a new classifier on top of the extracted features.
4. To reduce training time & improve accuracy using transfer learning

Algorithm :-

1. Import dataset (CIFAR-10 or Custom dataset).
2. Load a pre-trained model (VGG16 or ResNet50) without top (fully connected) layers.
3. Freeze pre-trained model weights to prevent retraining
4. Add custom layers (Dense + Softmax) for classification
5. Compile & train the new model using extracted features.
6. Evaluate accuracy on test images.

Pseudo Code:-

Load dataset (train, test)

preprocess images to target size

Load pre-trained CNN (include_top= False, weights = 'imagenet')

Freeze pre-trained layers.

Add custom layers:

Flatten → Dense (256, ReLU) → Dense(num_classes, softmax)

Compile model (adam, categorical_crossentropy)

Train model on dataset

Evaluate accuracy

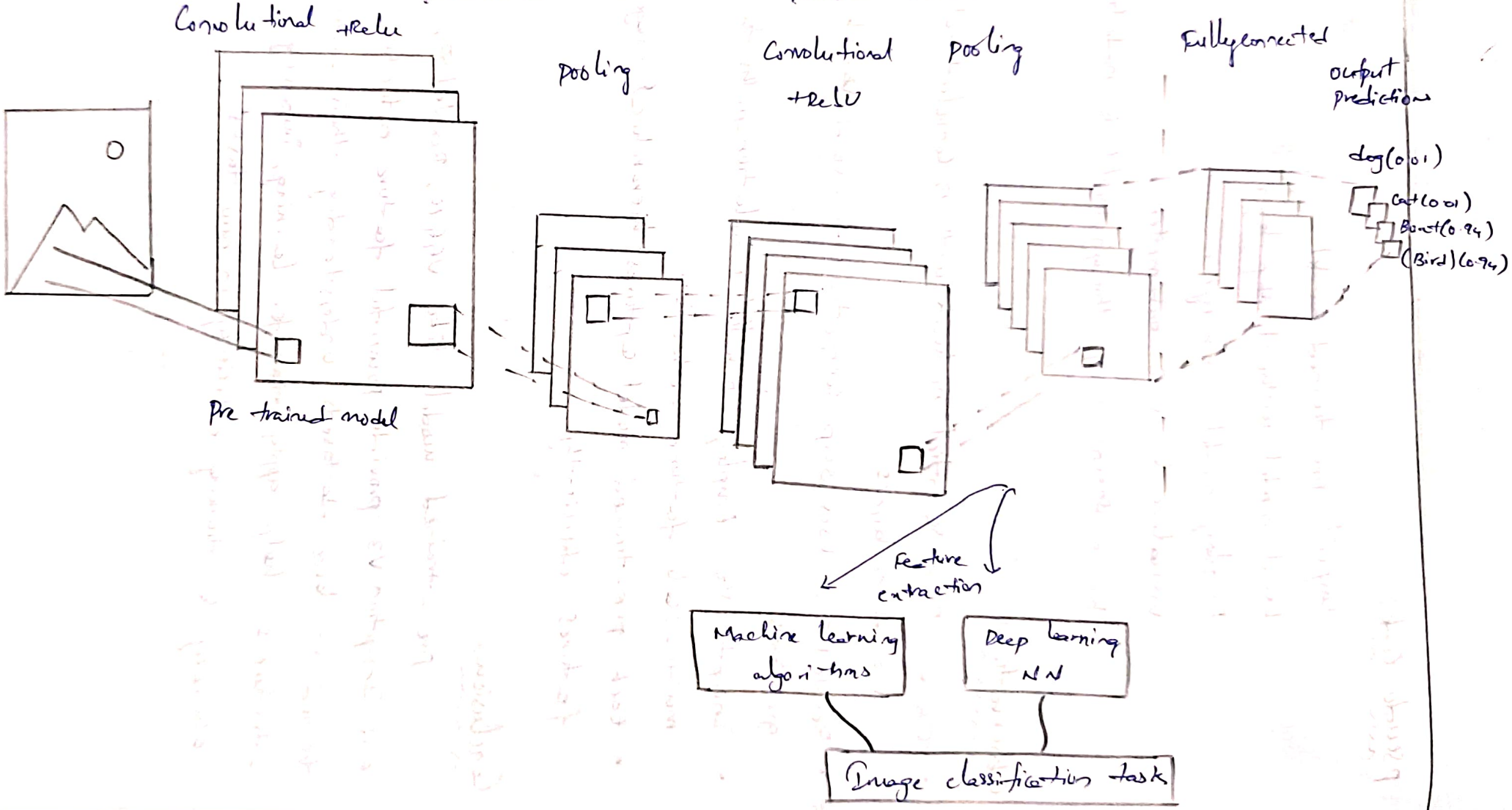
Observations

- The pre-trained VGG16 network extracts robust visual features from images.
- Training time is greatly reduced compared to training from scratch
- The model achieves high accuracy with limited epochs.

Conclusion:-

Using a pre-trained CNN as a feature extractor provides efficient & accurate image classification. Transfer learning leverages previously learned features, significantly improving performance & saving computation time

TRANSFER LEARNING.



Output 1 -

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 : 0.5697

Training Complete using pre-trained ResNet 18 as feature Extractor.

Training loss curve - Transfer learning with ResNet 18