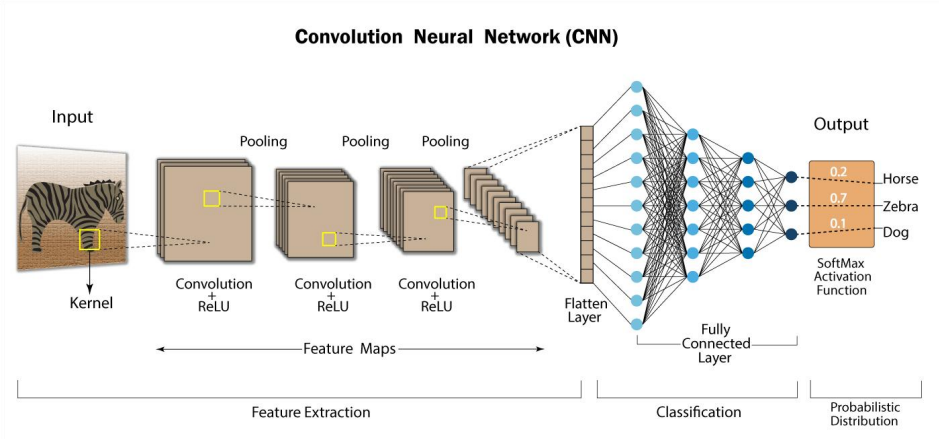
VR MINI PROJECT REPORT

Q3b. CNN as feature extractor

Ans:

What is CNN: CNN stands for Convolutional Neural Network, which is a type of neural network commonly used in computer vision tasks such as image and video recognition, object detection, and segmentation.

The key difference between a CNN and a traditional neural network is the way in which it processes data. A CNN applies a series of convolutional filters to the input data, which helps to extract features and patterns from the input image.These filters typically start by detecting simple features such as edges, lines, and corners, and then gradually progress to more complex features such as textures and shapes.After the convolutional layers, a CNN usually includes pooling layers, which downsample the feature maps produced by the convolutional layers and help to reduce the dimensionality of the data and get the global perspective of data.Finally, the output from the convolutional and pooling layers is fed into one or more fully connected layers, which perform classification or regression tasks based on the extracted features.



CNN As A Feature Extractor:To use a pre-trained CNN as a feature extractor, we can remove the fully connected layers from the network and take the output of the last pooling layer and flatten it and use it as features. These features typically represent high-level representations of the input image or data. These features can be thought of as a compressed representation of the most important information in the input that is relevant to the specific task being performed.These features can then be fed into a separate classifier or regression model to perform a specific task like image classification , object detection etc.

Using a pre-trained CNN as a feature extractor can be especially useful in scenarios where you have limited labeled data or limited computational resources to train a CNN from scratch. By leveraging a pre-trained CNN's learned features, you can save time and resources while still achieving good performance on your task.

Some popular pre-trained CNNs that are commonly used as feature extractors include AlexNet,VGG, ResNet.

Implementation Details:

Feature Extraction Module:

In our project we used **alexnet()** model from **torchvision** module of **PyTorch** library,which is previously trained on **1000 class imagenet** dataset. Then we removed the fully connected layers or classification module in front of it and only took the feature extraction module using **alexnet().features** function and added a **flatten layer** in front of it. Finally we got our feature extraction sequential model using **model = torch.nn.Sequential(features, flatten)** which had total **2,469,696** trainable parameters & **0** non trainable parameters.

Dataset Description & Preprocessing:

Then we used our feature extraction module to extract features from 5 different datasets namely

1. **Cats vs Dogs Classification Dataset**:
We got this dataset from kaggle. It contains two folders dogs and cats , dog folder contains 12500 dog images inside it and cat folder contains 12500 cat images inside it .We first merged those two folders and assigned corresponding labels with respect to the images.Then suffled the data and spiltted the data in train and test data while train data contains 20000 images and test data contains 5000 images.
2. **Birds Species Classification Dataset:** This dataset we also got from kaggle. It contains images of 500 different bird species.It contains total 80085 training images and 2500 test images of 500 different species of birds.
3. **Horse vs Bike Classification**: This dataset was provided by our course instructor.It contains total 80 bike images and 81 horse images. We mixed them ,suffled them and divided into train test.
4. **CINIC-10 Dataset:**CINIC-10 is an augmented extension of CIFAR-10. It contains the images from CIFAR-10 (60,000 images, 32x32 RGB pixels) and a selection of ImageNet database images (210,000 images downsampled to 32x32). It was compiled as a 'bridge' between CIFAR-10 and ImageNet, for benchmarking machine learning applications. It is split into three equal subsets - train, validation, and test - each of which contain 90,000 images. For our project we used 90000 images from train set for training and 1000 images from test set for testing.

Training:

In training phase we resized our each image into 224*224 size and passed them through our feature extractor module and got a feature vector of 9216 imension. For whole training data we got a 2d numpy array whose one row corresponds to one datapoint and one column corresponds to one feature. Then we trained **svm** & **xgboost** classifier from **sklearn** on training data.

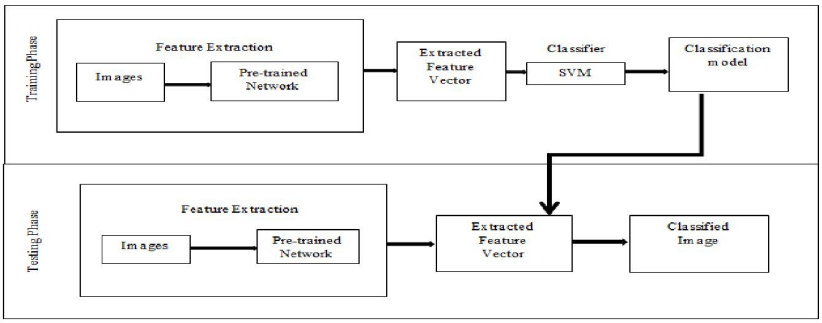


Fig. 2. Classification using Pre-trained Neural Network + SVM

Testing: In testing phase we extracted features of 9216 dimension from each test image using the same process as the training phase.Then used our trained classifier to predict on test features.Then we measured the accuracy.

Initially we tried our approach on **CIFAR-10** data as well.

Results:

Dataset	Classifier	Accuracy
CIFAR-10	SVM	61.20%
	XGBoost	56.66%
Cats Vs Dogs	SVM	73.96%
Birds Species	XGBoost	46.07%
	SVM	.16%
Horse Vs Bike	SVM	100%
CINIC-10	SVM	48.80%

Observations:

1. We saw that our method performs fairly well in case of Cats vs Dogs and Horse vs Bike dataset as they are only two class classification problem and those classes were included in imagenet dataset on which our feature extractor was pretrained. Moreover in case of Cats Vs Dogs we had a huge amount of data to train our classification model as compared to our no. of classes.
2. In case of Birds species classification we had a huge no. of classes. Moreover this classification task is domain specific. So our imagenet pretrained alexnet feature extractor could not extract the right features for this task. So our xgboost classifier gave a low accuracy of 46.07% and SVM classifier did not learn anything at all and gave a accuracy of 0.16%.