Machine Learning- Mini Project

Team Number- 3
Team Members : Subhiksha S - 185001172
Sukeerthi M - 185001175
Vaishnavi M R - 185001187
Varshinee M M - 185001190

Problem Title- Handwritten Character Recognition

Motivation - In a world that is computationally exceeding the limits of human outreach, there is an immense necessity for the digitalisation and documentation of information in a raw transcript format written by various people.

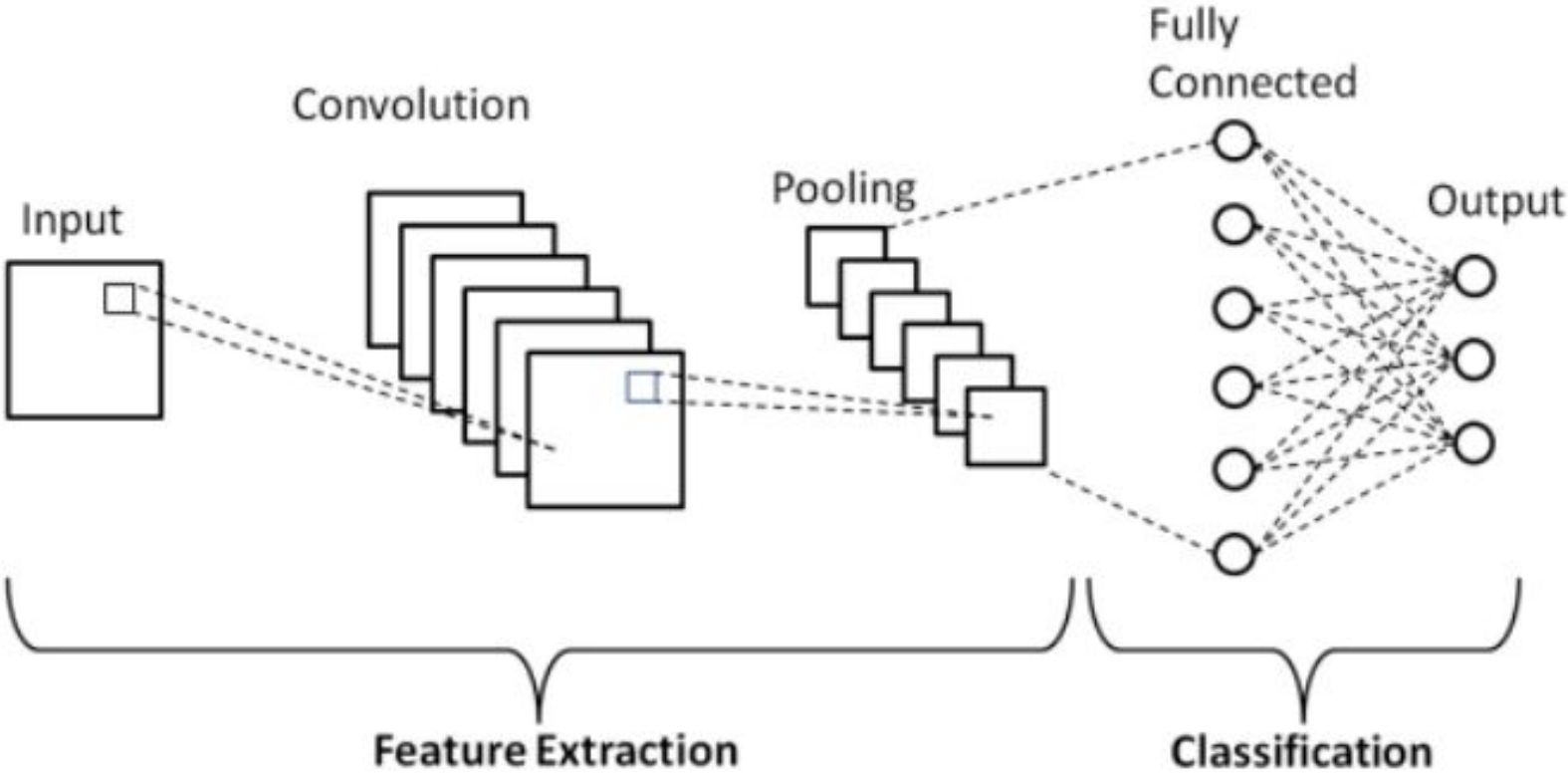
In this machine learning project, we will recognize handwritten characters, i.e, English alphabets from A-Z. This we are going to achieve by modeling a neural network that will have to be trained over a dataset containing images of alphabets.

Methodology and Frameworks Used

Convolutional Neural Networks- A **convolutional neural network** (**CNN**) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. It is used to extract the features of the images using several layers of filters.

Deep learning techniques like Convolution Neural Network always show better accuracy than other conventional techniques.Its performance is highly efficient as the scale of data increases. Deep learning leverages the large amount of data set to understand and extract its features with better accuracy, giving it an advantage over traditional machine learning algorithms.

Processing Layers of CNN



Working principle of CNN

STEP 1- The first building block in our plan of attack is convolution operation where we use feature detectors, which basically serve as the neural network's filters.

STEP 2- Pooling layers are used to reduce the dimensions of the feature maps and the amount of computation performed in the network.

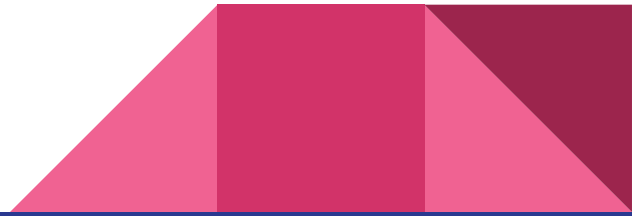
STEP 3- Flattening of the pooling layer results in a long vector of input data that can pass through the neural network for further processing.

STEP 4 - Fully connected layer is where all the inputs from one layer are connected to every activation unit of the next layer. It compiles the data extracted by previous layers to form the final output i.e classification.

Dataset

The dataset for this project is a csv file that contains 372450 images of all the alphabets of 28x28 (pixels).

Training data : 297960
Testing data : 74490
20% of the data has been assigned for testing purpose and the rest for training.



Role of python libraries in the project workflow

- **Numpy** - **Since the dataset consists of images in the form of pixels, we need the numpy library for large multi-dimensional array, matrix processing and its high-level mathematical functions**
- **Scikit-learn** - **The ML library makes the computation easier with inbuilt functions to split training and testing datasets, and shuffle data to visualize the outputs randomly to check for accuracy.**
- **TensorFlow**- **This framework involves training and running the CNNs involving tensors.**

- **Keras** - It is a high-level neural networks API that runs on top of TensorFlow. It allows us to build models, provides the necessary layers involved in feature extraction and classification, optimization algorithms and categorization functions.
- **Pandas** - It enables efficient extraction of the pixelated data from the csv dataset file.
- **Matplotlib**- It is a 2D plotting library for creating graphs and bars to visualize and represent data.
- **OpenCV**- It is an image processing library to manipulate images in the required format to feed it into the neural network.

Overall Flow of the Project

- Import the necessary machine learning libraries
- Extract the data and split into testing and training datasets 20 and 80% respectively.
- Convert the dataset into images of 28X28 pixels.
- Plot a bar graph to indicate the alphabets and the frequency of their occurences.
- Reshape the datasets to fit them into the model and add the bias unit.
- Create a convolutional kernel that is wound with input layers which helps produce a tensor of outputs that can pass through maxpool layer for reducing the spatial dimensions of the output volume and thereafter flattened to be classified into alphabets.
- Train the model over multiple epochs for better accuracy.
- Feed in a random real-world image and test the output.

Performance Metrics

- Categorical Accuracy
- Logarithmic Loss
- Precision
- Recall
- Confusion Matrix
- True Positive Rate
- True Negative Rate
- False Positive Rate
- F1 Score