Identifying Indian Language Fonts

Key Steps in Developing a Font Recognition Model for Indian Languages

Problem Definition

The core problem addressed in this project is the development of a model capable of recognising and understanding various Indian language fonts. Current libraries and tools are predominantly tailored to Western languages such as English and French, leaving a significant gap in support for Indian languages.

The primary aim is to enable the model to adapt to Indian languages, thereby providing a solution that facilitates better integration and accessibility for millions of people who use these languages. This enhancement will help individuals who currently face difficulties in engaging with global digital platforms, allowing them to find solutions more effectively and integrate more seamlessly into the digital world.

Data Collection

It is the crucial step in making/training AI models. Small amount of data is not sufficient for training the model, as a result we need to gather a vast amount of data in-order to except the best result.

As there are numerous of languages in India we need tons of data to make model much clear about the language and catch up with the different types of strokes as each language has unique structure and weird strokes.

This step involves gathering a diverse and representative dataset that covers a wide range of fonts and scripts. Here's how to approach it. The data can be hand written or images that are readily available over internet.

While collecting data it is preferred to collect both images and hand written text as the way of writing changes from one person to another and it is the challenging part of it.

Types of Data to be collected:-

- **Text Images**: Collect images of text written in various fonts and scripts. Ensure that these images cover different sizes, styles, and contexts (e.g., headlines, body text, captions).

- **Font Files**: Gather font files (e.g., TTF, OTF) for the targeted scripts to generate synthetic datasets.

Data Labeling:-

Data indexing is crucial as it enables efficient data retrieval, improves data management, enhances search ability, ensures data integrity, and facilitates efficient model training and testing.

- **Manual Labelling**: Manually label the collected text images with the correct font names. This step ensures the accuracy of the dataset.
- **Automated Tools**: Utilise automated tools for preliminary labelling, followed by manual verification to ensure high-quality labels.

Data Annotation:-

Verify the process of labelling the data is accurate to avoid the misleading of the model.

Data Preparation

This step ensures your collected data is clean, consistent, and ready for training your model. This preparation step involves eliminate the noisy data or any irrelevant data that leads model in wrong direction, fix any errors in the data, such as incorrect labels or corrupted images. Ensure all images have a consistent background to avoid distractions during training.

As it is regarding the fonts it is compulsory to cross verified by human, so the data that is being provided to the model. By doing this chances of raising errors decreases.

Data Normalisation:-

Resizing: Resize all images to a uniform size to ensure consistency.
Scaling: Normalise pixel values to a range of 0 to 1 by dividing by 255. This helps in faster convergence during training.

Data Augmentation:-

This is a technique which issued to process the image in machine learning to artificially increase the size and diversity of a training dataset. This is achieved by creating modified versions of images in the dataset through various transformations. It helps in improving the generalisation ability of a model by exposing it to a wider variety of data.

It is mainly useful to increase the efficiency of the model when the size of data is less and makes model robust, vibrant and prevent the model to get overfitted. By using this technique we can make model to perform well on unseen data.

Label Encoding:-

As machine cannot understand images of raw data it is converted to numeric value using pre existing modules in python. Later the data is spliced into two different parts

1. Training Data
2. Testing Data

From the given data most of the part is taken for training the model and few part is used for testing the model and much data is required to ensure good understanding of machine.

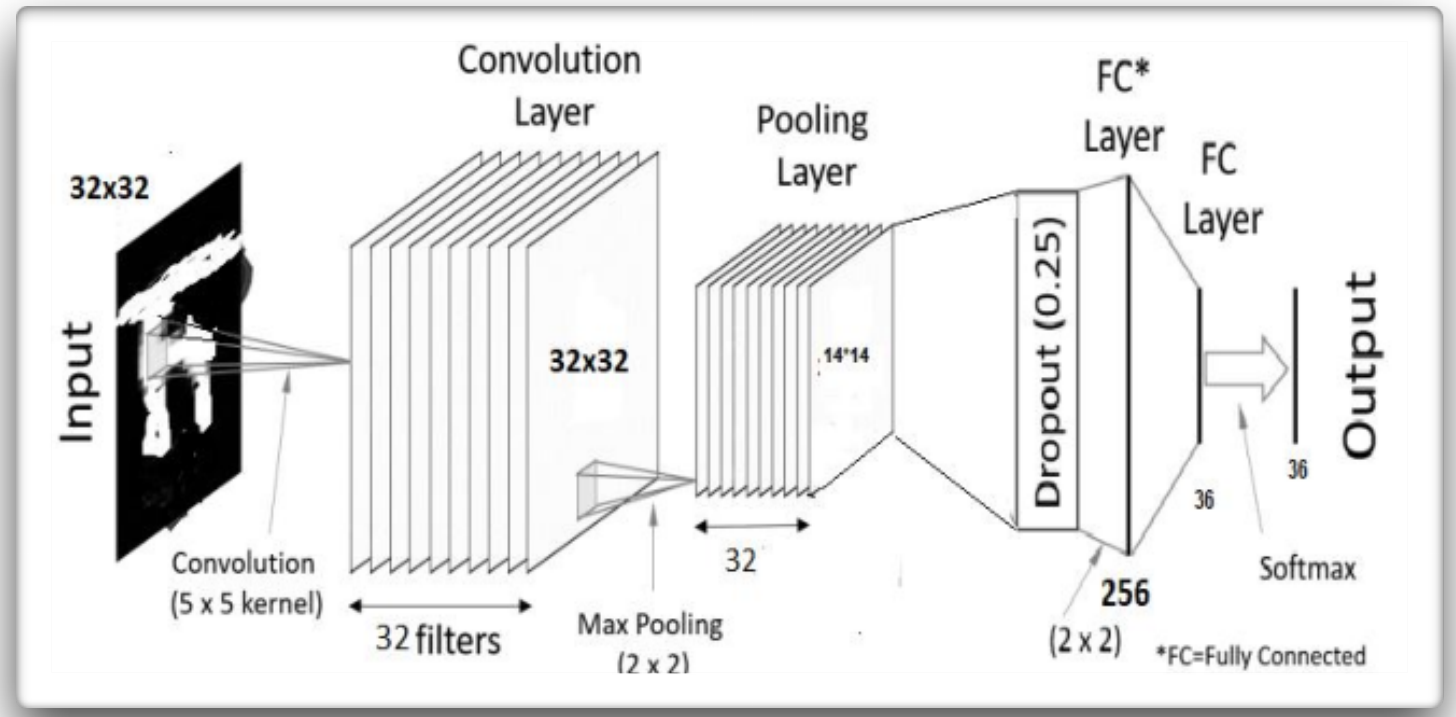
Feature Extraction

Feature Extraction is mainly used for dimensionality reduction and focuses on most important and relevant data. This makes model super fast to learn with high accuracy by removing noisy and irrelevant data from dataset

---

Model Selection

Custom deep learning models, particularly Convolutional Neural Networks (CNNs), are highly effective for this task due to their ability to extract and learn complex features from images. The process begins with data collection, ensuring a comprehensive dataset that includes diverse fonts and handwriting styles across multiple Indian languages. Preprocessing this data, including normalisation and noise reduction, is crucial for improving model accuracy. Data augmentation techniques such as rotation, translation, and brightness adjustment further enhance the model's robustness. A well-designed CNN architecture, combined with techniques like dropout and batch normalisation, can prevent overfitting and improve generalisation. Hyper-parameter tuning and regular evaluation using metrics like accuracy, precision, and recall ensure the model performs optimally.



CNN Model for recognising the fonts

Modules

TensorFlow/Keras: Building and training deep learning models.
PyTorch: Building and training deep learning models, research and development.
OpenCV: Image processing and computer vision tasks.
PIL (Pillow): Image processing, manipulation, and augmentation.
Tesseract OCR: Optical character recognition for extracting text from images.
EasyOCR: Simple and ready-to-use OCR for multiple languages.
NumPy: Numerical operations and array manipulations.
scikit-learn: Machine learning utilities for evaluation and preprocessing.
Matplotlib/Seaborn: Data visualization.
Flask/Django: Model deployment as a web service.
TensorFlow Serving: Deploying TensorFlow models for production.

Model Training

Model training is a critical phase in developing a text recognition system. It involves feeding labeled data into the model, adjusting its internal parameters through optimization algorithms, and iteratively refining its performance.
Here is the place where the machine learns by getting trained by giving data.

Following steps are taken to train the model:

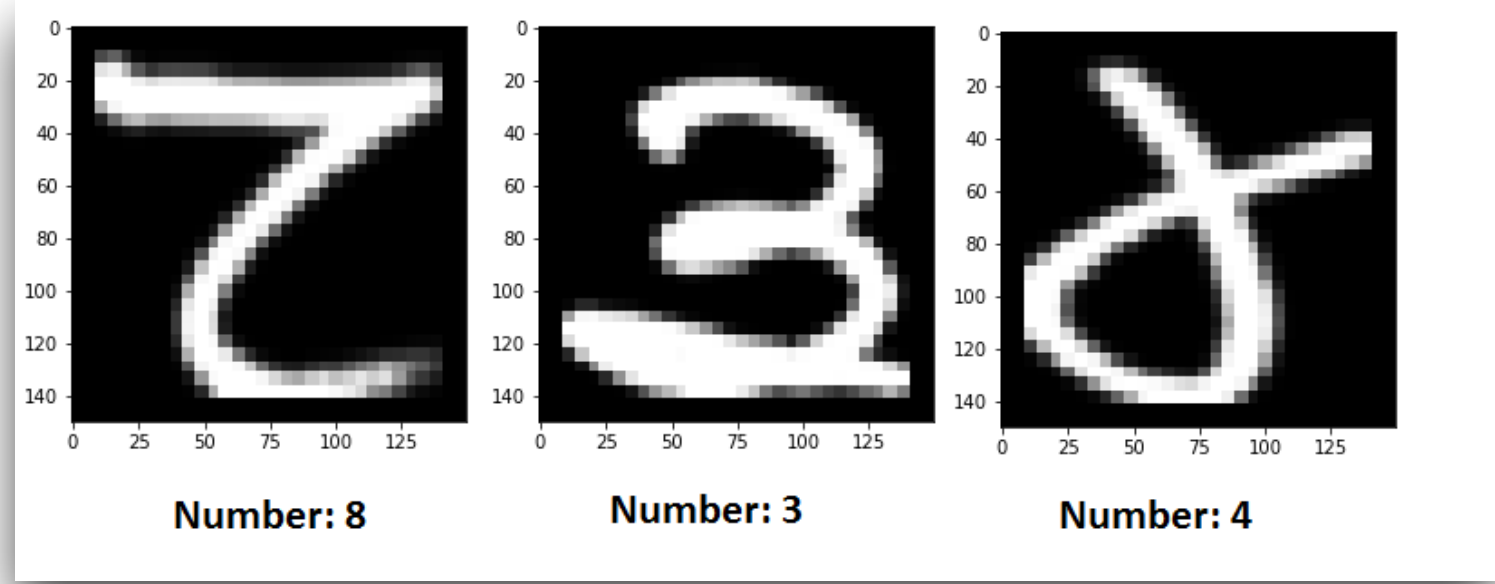
- **Data Preparation**
- **Model Architecture**
- **Training Process**
- **Hyperparameter Tuning**
- **Model Evaluation**
- **FIne Tuning**
- **Documentation**

Model Evaluation and Fine-Tuning

Fine-tuning involves iteratively refining the model's architecture and parameters to enhance its performance. This process includes adjusting hyper-parameters like learning rates, batch sizes, and regularisation techniques to strike a balance between model complexity and generalisation. Additionally, leveraging techniques such as transfer learning allows for the adaptation of pre-trained models to the specific nuances of text recognition tasks, accelerating convergence and improving overall accuracy.

Testing

After finish of training of the model, it has to be tested by using clumsy and blur images to get its score. By the score we will get to know whether to improve it or it is ready to deploy.
To make testing easy, the dataset is divided into training and test data. So that it makes more easy to work and make model robust and versatile.



Pre-Existing Models

- **Tesseract OCR**
- **EasyOCR**
- **Google Cloud Vision OCR**
- **Microsoft Azure Computer Vision**
- **Deep Learning Models for Handwritten Text Recognition (e.g., CRNN, CNN-based models)**

Above shown models are pre-existing model that are capable of recognising the Indian language fonts and work with.