Model Development Report

Project Overview:

The goal of this project is to develop a font recognition model powered by machine learning. The model should be capable of analyzing images, detecting text regions, and identifying the fonts used within these regions. It aims to accurately recognize different fonts from an image input, even when the image contains text in multiple fonts. The desired output of the model is to provide the detected text alongside the identified font for each text box within the image.

Project Objective:

1. Text Detection:

- Utilize pytesseract to perform optical character recognition (OCR) on the preprocessed image.
- Extract text regions or bounding boxes containing detected text using pytesseract's capabilities.

2. Font Recognition:

- Develop a machine learning or deep learning model capable of identifying the font used in each detected text box.
- Consider approaches like CNN-based feature extraction followed by classification, or utilizing pre-trained models for transfer learning.

3. Model Training and Evaluation:

- Train the model on a diverse dataset containing various fonts and text orientations.

- Evaluate the model's accuracy and efficiency in font recognition using appropriate evaluation metrics.

4. User Interface for Model Testing:

- Create a simple user interface or streamlit app allowing users to upload images for font recognition.
- Display the detected text and corresponding font(s) for each text box identified within the image.

Research and Analysis:

- Explore existing techniques and models for text detection and font recognition.
- Analyze the characteristics of the font recognition dataset, including the distribution of fonts, text orientations, and image sizes.
- Investigate methods for data preprocessing, augmentation, and balancing to improve model performance.

Conceptual Framework Development:

- Identify key concepts such as text detection, font recognition, and model evaluation.
- Define variables such as image features, font labels, and evaluation metrics.
- Develop visual representations like flow charts or diagrams to illustrate the framework.
- Establish clear boundaries and parameters within the conceptual model to guide analysis.

Strategy:

Image Data Collection:

- Identify data sources and collect a diverse dataset of images containing text in various fonts.
- Perform data scraping if necessary and augment the dataset to increase its size and diversity.
- Label the images with text boxes and font information for model training and evaluation.

Data Collection:

Gather a diverse dataset of images containing text in various fonts. Annotated images with labeled text boxes and font information will be beneficial for model training and evaluation.

Link to the Dataset -

https://www.kaggle.com/datasets/muhammadardiputra/font-recognition-data/data

This dataset is basically taken from this Kaggle link (https://www.kaggle.com/dhruvmak/font-recognition-data).

The basic thing that you can do with this dataset is to create a machine learning or deep learning model which is able to recognize different font styles. There are 48 different classes, where each of those consists of 5000 pictures of texts. Keep in mind that the size of those images are varying. So you'll probably need to do a little bit of preprocessing towards their size. In addition, there are two versions of this dataset: first the one that is colored in grayscale (folder "Font Dataset Large"), and second, the one that is colored in RGB (folder "Font Dataset Large Color").

I have used Font Dataset Large folder due to computational resource limitation to train the deep learning CNN model to detect the Font from Textual Image,the selected four classes for trainings are-

1:Times New Roman

2: Franklin Gothic

3: Arial

4: Bodoni

Image Data Preprocessing:

- Resize, normalize, crop, denoise, and enhance the images to improve model performance.
- Apply techniques like histogram equalization to adjust image contrast and brightness.

Image Data Scaling:

- Scale the image data using techniques like min-max scaling or z-score normalization to ensure uniform feature scaling.

Feature Engineering:

- Extract features using Convolutional Neural Networks (CNNs) or transfer learning models like VGG or ResNet.
- Fine-tune model parameters and perform feature selection to enhance model interpretability and performance.

Model Selection and Training:

- Select a CNN architecture suitable for font recognition tasks.
- Tune hyperparameters such as learning rate, batch size, and dropout regularization.
- Split the dataset into training and testing sets and evaluate the model using cross-validation techniques.

Hyperparameter Tuning:

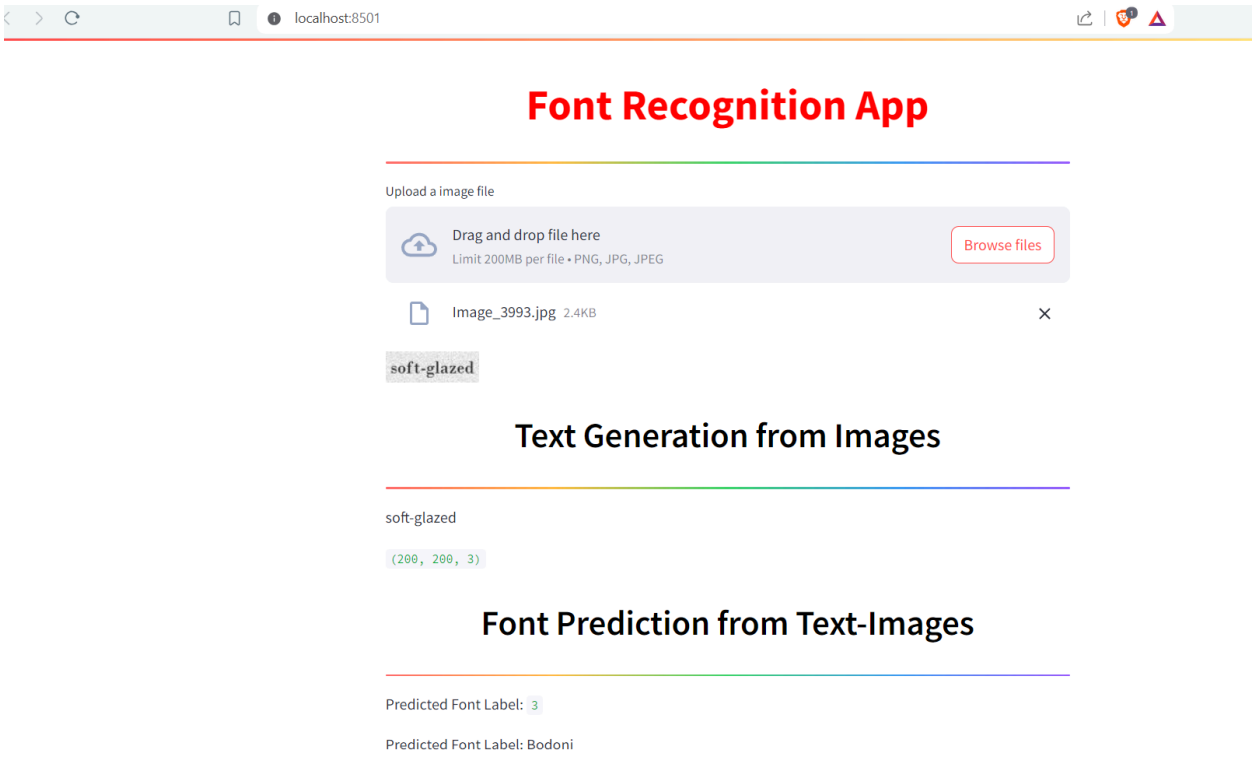
- Adjust learning rates, batch sizes, dropout rates, and other hyperparameters to optimize model performance.
- Implement early stopping criteria and weight decay to prevent overfitting.

Model Evaluation and Validation:

- Assess model accuracy, precision, recall, F1-score, and AUC-ROC using appropriate evaluation metrics.
- Visualize evaluation results using techniques like confusion matrices, precision-recall curves, and ROC curves.
- Evaluate the trained model on a separate validation dataset to assess its performance in font recognition. Use evaluation metrics such as accuracy, precision, recall, and F1-score to measure model performance.

Streamlit App Creation:

- Develop a user-friendly interface using Streamlit for users to upload images and view font recognition results.
- Design UI layout, create widgets, and handle user interactions to ensure smooth functionality.



Model Loading:

- Load the trained model and initialize model parameters for font recognition.
- Serialize and deserialize the model for easy deployment and validation.

Input Processing:

- Validate input data, preprocess images, and perform data transformation for model input.
- Normalize, encode, and extract features from input images to prepare them for font recognition.

Prediction:

- Pass input data to the model, perform predictions, and post-process the results.
- Apply confidence thresholding and format the output to display detected text and corresponding font(s).

Error Handling:

- Implement exception handling, error logging, and error message display to handle input validation errors and model prediction errors.
- Integrate debugging tools for efficient error detection and resolution.

Conclusion:

In conclusion, the development of a font recognition model involves various stages, including data collection, text detection, model development, training, evaluation, and user interface development. By following a systematic approach and leveraging machine learning techniques, we aim to create an accurate and efficient font recognition system that meets the project objectives.