Fine-Tuning TrOCR for Malayalam Handwritten Text Recognition

-Syamlal

Objective: Convert the given Malayalam handwritten documents to text.

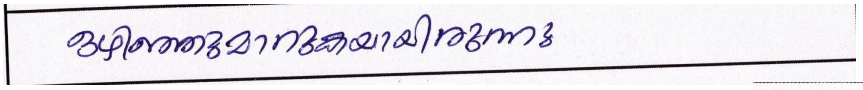
Solution: Create a robust OCR model, identify and fine-tune the model with the given dataset.

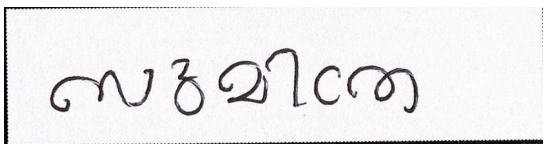
Dataset: IIIT-INDIC-HW-WORDS: A Dataset for Indic Handwritten Text Recognition [Link]

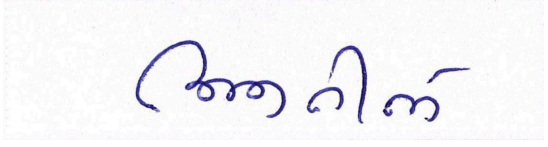
Dataset Folder Structure

```
DatasetFolder/
├── train/
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ... (total: 85,270 files)
├── test/
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ... (total: 19,635 files)
├── val/
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ... (total: 11,878 files)
├── train.txt
│   └── (entries: train/1.jpg, 1551)
├── test.txt
│   └── (entries: test/1.jpg, 11512)
├── val.txt
│   └── (entries: val/1.jpg, 8489)
└── vocab.txt
    ├── അംഗ
    ├── അംഗങ്ങളായ
    ├── അംഗങ്ങളുമായ
    ├── അംഗങ്ങളേ
    ├── അംഗത്തെ
```

Data sample







Fine-tuning approach

Model selection: TrOCR, used the *trocr-base-handwritten* pretrained model.

Trocr-base-handwritten is trained on handwritten text in English.

Since it is trained on English, we needed a tokenizer capable of handling Malayalam script efficiently. So, the custom tokenizer is created using SentencePiece on the Malayalam corpus and converted to HuggingFace format.

Malayalam corpus: Used public domain data and Wikipedia Malayalam data
Corpus info:

Total characters: 248443889
Total words: 24188460

Fine-tuning key steps:

Data Preparation:

Images were preprocessed and loaded using a Dataset class compatible with Hugging Face Trainer.

Model Setup:

Utilized TrOCRProcessor (which includes a VisionEncoderDecoder architecture).

Processor was initialized with a pretrained vision encoder and custom tokenizer.

Training:

Hugging Face Seq2SeqTrainer was used for training.

Metrics such as Character Error Rate (CER) were computed during evaluation.

Checkpoints were saved, and the final model was stored locally.

Evaluation:

The model was reloaded from disk and tested on a few sample images.

Decoded output was compared to ground truth to assess accuracy.

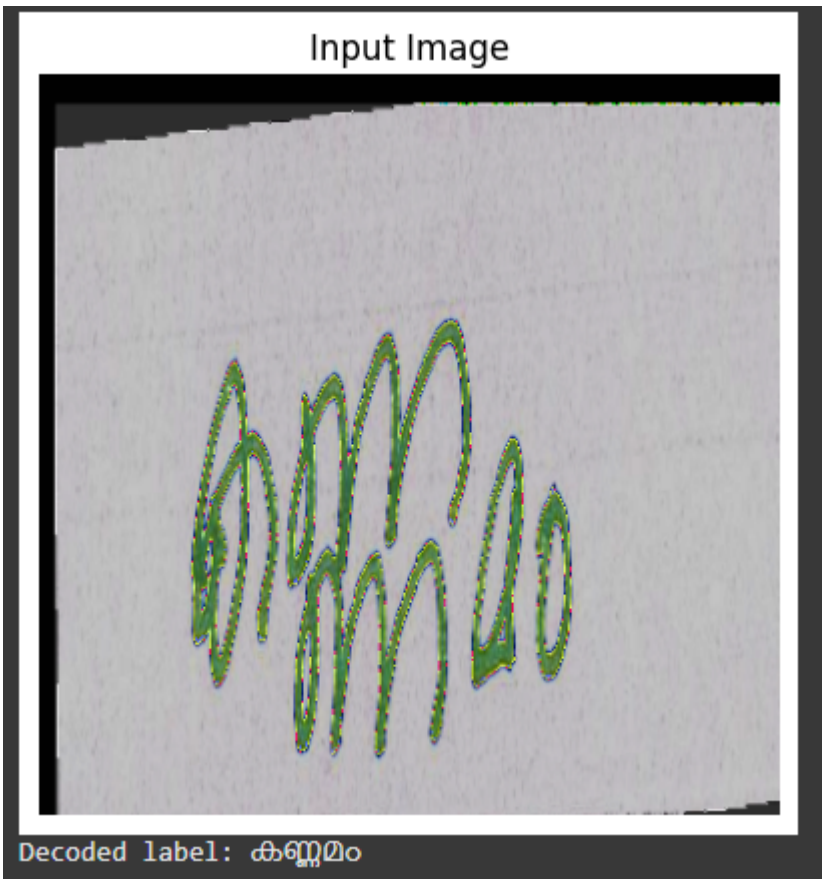
Visualization:

Training and evaluation losses were plotted.

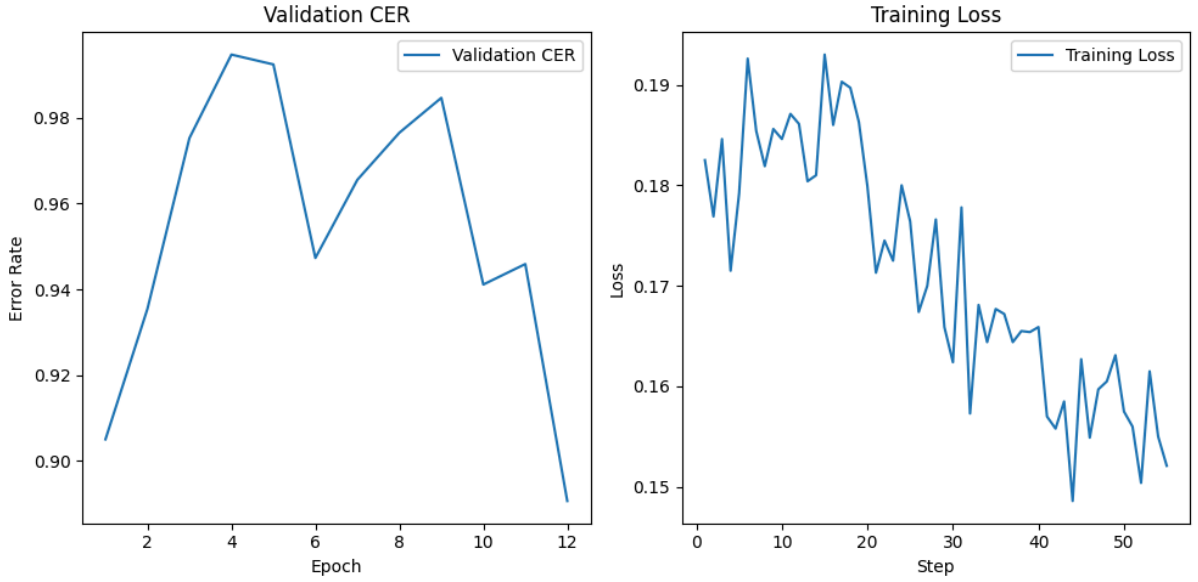
Curve plots indicate convergence and model stability across epochs.

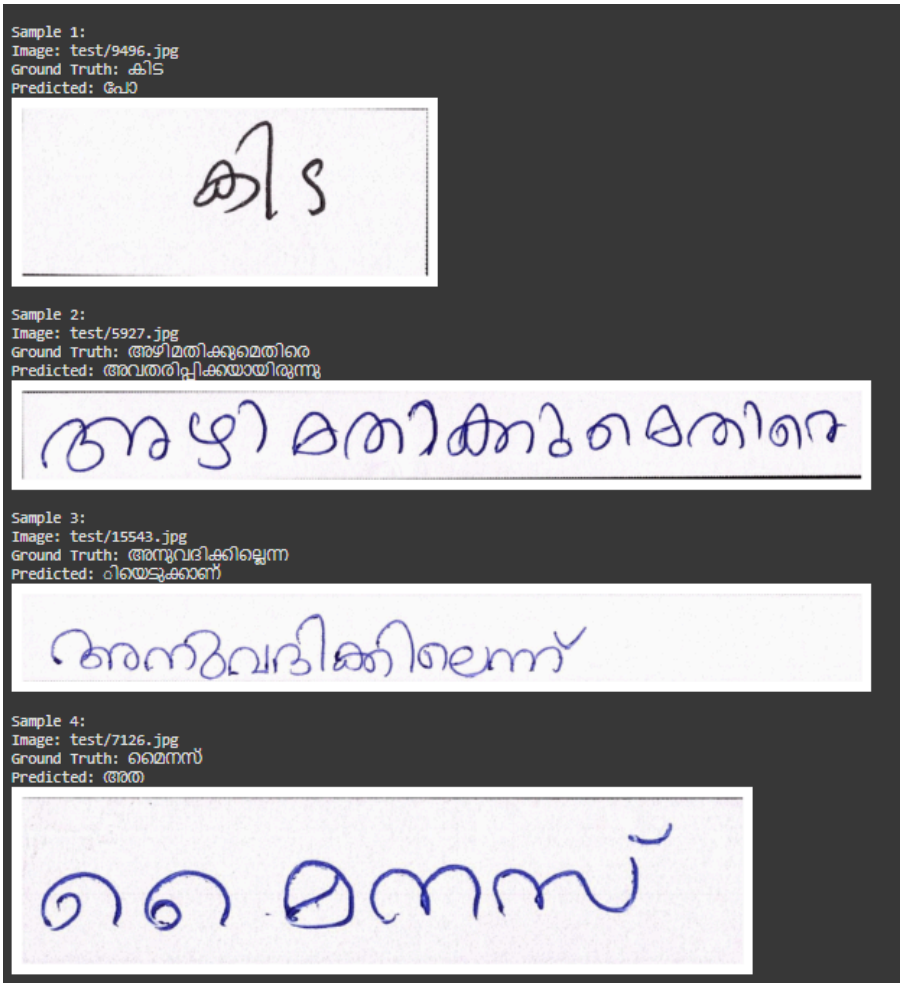
Github Links: Custom Tokenizer

TrOCR Fine-tuning



TrOCRProcessor will automatically process the image to the expected image input size, 384×384 pixels.





Full Document Processing Step:

The current model accuracy is relatively low; we need to fine-tune the model with a larger dataset that includes more handwritten samples

For full document reading, the following steps are recommended:

Text Line Detection:
Process the document image using a line segmentation algorithm to identify and extract the coordinates of each text line.

Line Cropping:
Crop the image based on detected line coordinates. Each cropped line becomes an input to the recognition model.

Text Recognition:
Feed each cropped line into the fine-tuned TrOCR model for line-level recognition. The TrOCRProcessor will automatically handle resizing and normalization.

Text Reconstruction:
Concatenate the recognized lines in sequence to reconstruct the full document text.