Automated Data Labeling Pipeline for OCR

This designed is automatically extract text from a folder of images, detect and crop the text regions using EasyOCR, clean and resize, it saves both the cropped images and their corresponding text

The pipeline consists of three main components integrated into a cohesive workflow:

1. Data Extraction Module

Automatically detect and extract text regions from input images. Utilizes EasyOCR library for initial text detection

Output is Text regions with bounding boxes and confidence scores

2. Data Preprocessing Module

Process extracted text regions and prepare them for training. Image enhancement, normalization, cropping, and quality filtering.

Output is standardized image-text pairs suitable for model training

3. Data Validation Module

Ensure data quality and consistency across the dataset, quality checks, statistical analysis, and error detection.

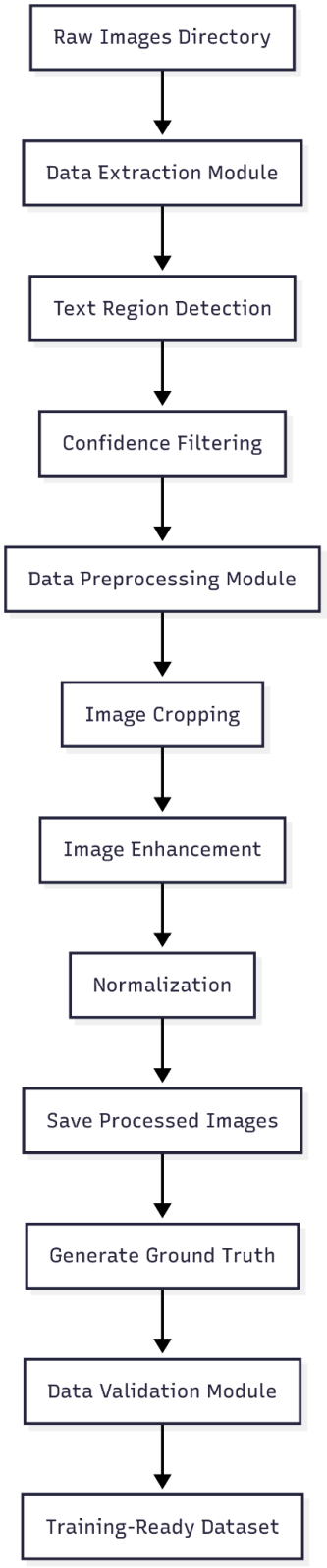
Output is validation reports and dataset statistics

Output Dataset Structure

data/processed/ocr_dataset/image
├── image_001.jpg
├── image_002.jpg
├── image_003.jpg

data/processed/ocr_dataset/
├── gt.txt

Workflow for labelling



Choosing FineTuned model TrOCR

I selected **TrOCR** (Transformer-based Optical Character Recognition) as the model for fine-tuning on a multilingual dataset. There are many newer models, I believe TrOCR is still one of the best choices because of its strong performance, open-source availability, and solid documentation

- **Proven Architecture:**
  TrOCR uses an encoder-decoder structure. It combines a Vision Transformer to understand the image and a language model to generate the text output. This setup works well for capturing both visual and language features.

- **Power of Pretraining:**
  The model is already pre-trained on a large set of printed and handwritten text. So it learns general patterns of how text looks in images. This makes it more powerful when fine-tuning on smaller, specific datasets like exam papers.

- **Multilingual Support:**
  Although pre-trained checkpoints mostly focus on English, TrOCR can still be adapted to multiple languages. With a good multilingual dataset, we can fine-tune it to recognize other scripts and languages.

- **Open Source & Accessible:**
  TrOCR is available on Hugging Face, and we can easily access model weights, tokenizer, and training configurations. This makes it very practical for development and research.

Reference Paper & GitHub Link

- **Paper Title:**
  TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models
  Authors: Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, Furu Wei

- **Hugging Face Model Hub:**
  https://huggingface.co/microsoft/trocr-base-printed

Fine-Tuning Strategy

1. Data Preparation

- **Image Quality:** I made sure the images are clean, without too much noise or distortion, and that the labels are accurate and resize.
- **Tokenizer:** TrOCR uses a fixed tokenizer. For multilingual datasets, some characters might not exist in the original vocabulary.

2. Key Training Parameters (Hugging Face)

When using TrainingArguments for fine-tuning, I applied the following:

- **learning_rate**: 1e-5 to 5e-5 Small to avoid forgetting pretrained knowledge
- **batch_size**: Start with 4 or 8 depending on GPU RAM
- **num_train_epochs** : 3 to 5 epochs is usually enough
- **weight_decay**: 0.01 – helps prevent overfitting
- **evaluation_strategy:** "steps"
- **eval_steps**: choose based on dataset size
- **save_steps**: Save checkpoints regularly

Evaluation Metrics

To measure how well the model performs, I used two common OCR metrics:

- **Character Error Rate (CER):**
  Measures how many characters are wrong. Based on edit distance between predicted and ground-truth text. The lower, the better.
- **Word Error Rate (WER):**
  Similar to CER, but based on whole words. Also better when it's low.

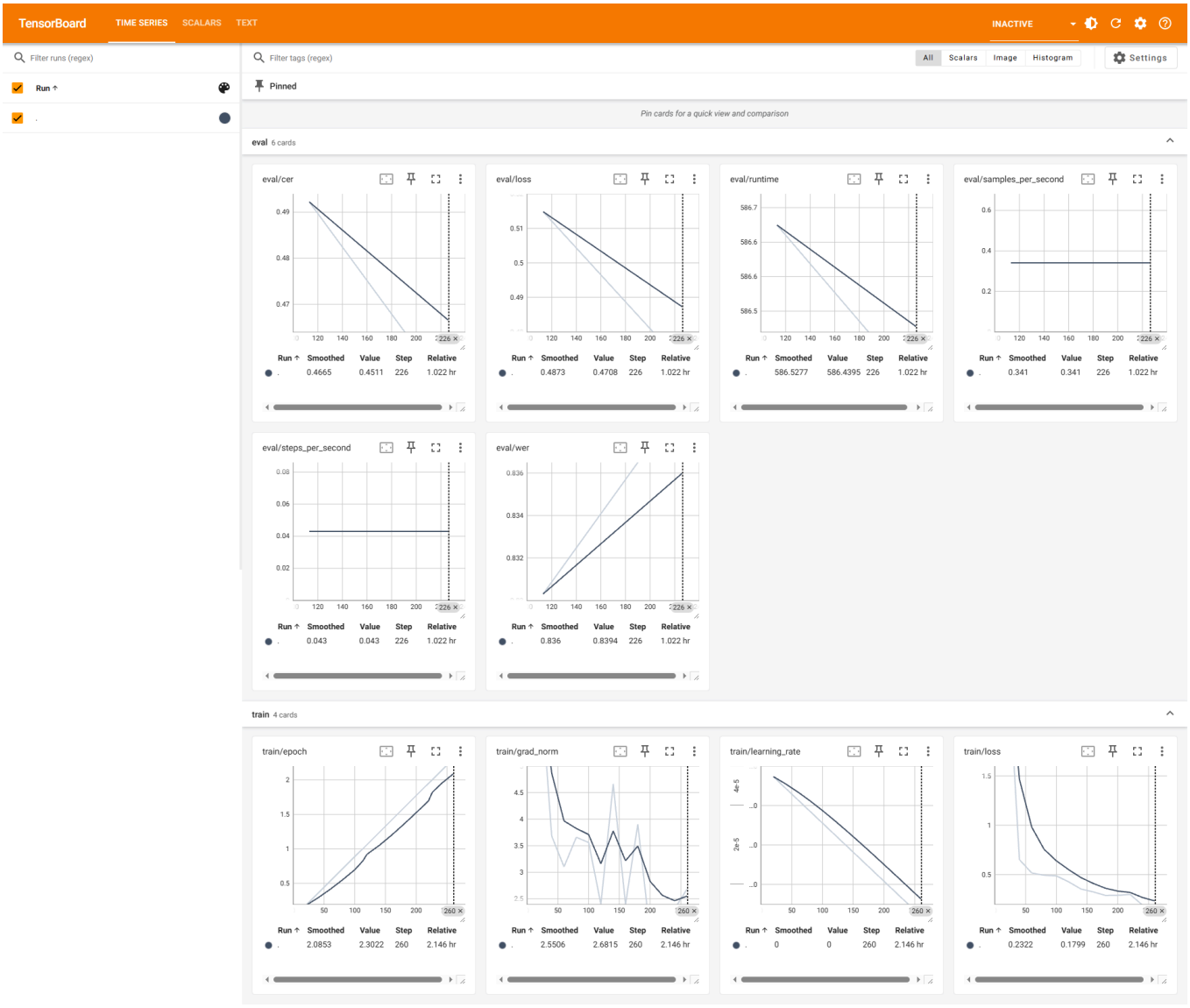
When to Stop Fine-Tuning

- **Early Stopping:** If validation loss doesn't improve after a few steps, I stop training early to save resources and prevent overfitting.
- **Overfitting Warning:** If training loss keeps going down but validation loss goes up

Fitted Model

- A model is fitted when it has learned the patterns in the training data well enough to make good predictions on unseen data.
- When validation loss reaches its minimum and doesn't improve anymore.
- If training stops too early, the model is underfit it hasn't learned enough, and performance will be worse.

Result from Tensorboard after stop training



- **Training loss** decreased steadily from ~1.6 to ~0.17, indicating good learning behavior.
- **Evaluation loss** dropped from ~0.7 to ~0.48, showing improving generalization.
- **Character Error Rate (CER)** reduced from ~0.72 to ~0.47 — strong improvement in recognizing individual characters.
- **Word Error Rate (WER)** however, increased from ~0.83 to ~0.89, possibly due to decoding issues or incomplete training.