University of Bremen

Karanjaspreet Singh: 6351611 | Prakunj Pratap Singh: 6352018 | Shikha Mukesh Chaudhary: 6351894

Proposal for AI System to Digitize Handwritten Exams in German Language Studies

1). Problem Description

The German education system mandates that students learn the German language, including writing skills assessed through handwritten exams. These handwritten assessments test students on their grasp of grammar, spelling, and the clarity of their penmanship. However, the growing number of students per teacher creates challenges for timely evaluation, as grading these exams by hand requires significant manual effort. Teachers often resort to overtime to meet grading deadlines, affecting their work-life balance and efficiency.

The objective of this proposal is to develop an AI-driven system utilizing the **TrOCR architecture** to automate the digitization of handwritten exams into text files. This system aims to assist educators by minimizing manual grading efforts and improving overall productivity. By leveraging advanced handwriting recognition, it ensures high accuracy while preserving the nuances of spelling and grammar, thereby supporting fair and consistent evaluation of student responses.

2). Key Challenges

Variability in Handwriting Styles

- Students exhibit diverse handwriting styles, ranging from clear and structured to poorly legible. This variability presents a significant challenge for optical character recognition (OCR) systems.

German Language-Specific Challenges

- **Umlauts (ä, ö, ü) and Eszett (ß)** may not be correctly recognized by generic OCR models.
- **Long compound words** common in German could be misinterpreted or split incorrectly.
- **Grammar-sensitive evaluation**: Small errors can change meaning, requiring context-aware processing.

Noise in Exam Papers

- Exams may include smudges, annotations, or other extraneous marks that can interfere with accurate text extraction.

Scalability

- The system must efficiently process large volumes of exam papers without significant performance degradation.

Data Privacy and Security

- Handling student exam data requires strict adherence to data privacy regulations (e.g., GDPR) to protect sensitive information.

Evaluation Context

- Recognizing words accurately is not enough—the system must also capture and interpret context to ensure grammatically correct representations and support automated or semi-automated grading.

3). Related Work

Optical Character Recognition (OCR) systems like **Tesseract** and **EasyOCR** are widely used for extracting text from scanned documents. However, they often struggle with handwritten text due to challenges like variations in handwriting styles, overlapping characters, and inconsistent spacing. In contrast, deep learning models such as **CRNN** and transformers like **TrOCR** have proven highly effective for handwritten text recognition (HTR) tasks. Additionally, research into German-specific text correction and spell-check systems provides valuable insights for handling the unique grammatical and spelling complexities of the German language. While many educational platforms focus on automating grading for multiple-choice and structured text responses, few have tackled the intricate challenges of handwritten submissions.

4). Model Description

We used the TrOCR (Transformer OCR) model, a deep learning model with transformer-based architecture designed for optical character recognition tasks. The architecture of the model consists of two main components:

- **Vision Transformer (ViT) Encoder** – Extracts features from the input images. The input images are split into **fixed-size patches (16x16)**, with each patch embedded into a vector. The encoder consists of **12 ViTLayer**, each containing **multi-head self-attention and feed-forward networks**. The self-attention mechanism captures dependencies between patches, while the feed-forward network applies a fully connected layer followed by a **GELU activation** function. Layer normalization is applied both before and after attention to stabilize training. Finally, a pooling layer generates a summary representation of the image, using a dense layer followed by a **Tanh activation**.

- **GPT-2 Style Decoder** – Converts extracted features into text. The decoder receives hidden states from the encoder and consists of 12 layers with self-attention, cross-attention to the encoder, and feed-forward networks. In self-attention, the decoder attends to its previous token embeddings, while in cross-attention, it attends to the encoder's outputs to capture the image context. After attention, the output passes through a feed-forward network with a ReLU activation. Finally, the decoder generates token predictions for the text, using a linear layer to project the output to the vocabulary size of 50,265 tokens.

At first, we tried EasyOCR and TesseractOCR for recognizing handwritten text. Both tools support the German language and work well on machine printed texts. However, their performance on handwritten text was inadequate due to model limitation as these systems are optimized for clean and printed texts, so we decided not to use them.

We then chose TrOCR (Transformer-based OCR) because it is a powerful model, its architecture makes it highly adaptable and effective for various OCR tasks that can be fine-tuned on domain specific datasets for better results. According to our research, TrOCR gives better results than EasyOCR and TesseractOCR. TrOCR was originally trained on the IAM dataset, which contains mostly English and a small number of German texts. We fine-tune the model on another dataset.

5). Methodology: Fine-Tuning TrOCR for German Handwriting Recognition

Our project focuses on training and improving a handwritten text recognition system using the TrOCR (Transformer OCR) model. To achieve this, we fine-tuned the model by adjusting its weights and training it on a new dataset different from the IAM dataset.

1). Dataset Description

For training our handwritten text recognition model, we used the "German Handwriting" dataset from the Hugging Face datasets library (fhswf/german_handwriting). This dataset contains around 10,000 handwriting images and corresponding text labels, it contains images of handwritten German words and short phrases along with their correct transcriptions.

We chose this dataset as it contains handwriting from 15 different people with the help of transcripts from school and university so perfect for this task requirements.

University of Bremen

2). Data Preprocessing

- **Data Format:** The dataset consists of RGB images of handwritten text.
- **Data Cleaning:** We removed any samples with missing or empty transcription labels and ensured that all images were valid PIL images in RGB mode.
- **Data Splitting:**
  - Training Set (80%) – Used to fine-tune the model.
  - Test Set (20%) – Used to evaluate model performance.

3). Creating a Custom Dataset for Training

Before feeding the data into the model, we created a custom PyTorch Dataset class to manage image-text pairs. Each sample image text pairs are processed using the TrOCRProcessor, which performed:

- **Image Processing:** Converting images into tensor format (multi - dimensional array) using the processor that can represent data that the model can understand.
- **Text Tokenization:** Converting transcriptions into numerical token IDs, ensuring all text samples fit within a fixed size. And padding tokens are replaced with -100 for compatibility of loss function that ignores padding

It is returned as a dictionary for Training and Evaluation.

Created two Pytorch data loaders objects that handles batching and shuffling for training and evaluation

- **train_dataloader:** Loads batches of data from the `train_dataset`, with a batch size of 8. The data is shuffled to randomize the training order.
- **eval_dataloader:** Loads batches of data from the `eval_dataset`, also with a batch size of 8, but without shuffling, since the evaluation data should not be randomized.

Both data loaders utilize the **`collate_fn`** function to manage the batching of data.:

- **`pixel_values`**: The images in the batch are stacked together into a single tensor using `torch.stack()`, allowing them to be passed through the model in one go.
- **`labels`**: The text labels for each image are padded using `pad_sequence` so that all the text sequences in the batch have the same length. The padding value is set to **`-100`**, which ensures that these padding tokens are ignored during training by loss functions.
- The function returns a dictionary with `pixel_values` (the stacked image tensors) and `labels` (the padded text sequences).

University of Bremen

4). Model Setup & Fine-Tuning

- Why fine-tune? Since TrOCR was originally trained on the IAM dataset, which includes mostly English handwriting, we used TrOCR (Microsoft/trocr-base-stage1) as the base model and fine-tuned the model on our German handwriting dataset to improve accuracy for German text.
- The model's internal weights were adjusted to better capture German handwriting patterns, and newly initialized weights, such as encoder.pooler.dense.bias and encoder.pooler.dense.weight, were fine-tuned.

To speed up training, we modified several parameters in both the Encoder and Decoder configurations of the TrOCR model.

Encoder Modifications:

- Reduced **hidden size**, **number of layers**, and **attention heads**.
- Increased **dropout rates**.

Decoder Modifications:

- Decreased **model size**, **decoder layers**, **attention heads**, **feed-forward network size**, and **maximum position embeddings**. Increased **dropout**.

Effects of Changes:

- **Training speed increased significantly**.
- **Accuracy dropped**, and **training loss increased** due to reduced model capacity.

5). Optimizer

We used the Adam optimizer with a learning rate of 5×10^{-5} to adjust the model's weights during training. Although we did not use a learning rate scheduler, it can be added to improve training efficiency.

6). Training Process

We fine-tuned the model for 10 training cycles (epochs) using the following steps:

Forward Pass:

- The model took an image as input and generated a predicted text transcription.

- It compared the prediction with the actual transcription to calculate the cross-entropy loss (a measure of error).

Backward Pass & Optimization:

- Performs backpropagation to compute gradients for the model's parameters based on the loss.
- The model updated its weights using the Adam optimizer to reduce errors in future predictions.

Loss Tracking:

- We logged the loss value at each epoch to monitor progress.

7). Model Evaluation

To check the model's accuracy, we evaluated it using the Character Error Rate (CER) metric:

- The model generated predictions for the test set.
- Predictions and actual transcriptions were converted back into text.

- We calculated the CER, which measures the percentage of incorrect characters by comparing the predicted texts with the correct label texts.
- A lower CER means better performance.

System Workflow

- **Uploading Exams:** Teachers will scan and upload handwritten exams through the web interface (not yet implemented).
- **Image Preprocessing:** The system will use OpenCV to clean images, remove noise, and adjust contrast for better text extraction.
- **Text Recognition:** The fine-tuned TrOCR model will process the images and extract handwritten text.
- **Post-Processing (Not Implemented Yet):** Planned features include AI-based spell-checking and grammar correction to refine OCR results.
- **Review & Download:** Teachers will review the extracted text, make manual corrections if need be, and download the final transcription.
- **Automated Grading (Not Implemented Yet):** A potential future feature where AI-based grading algorithms analyze content and suggest grades.

University of Bremen

Limitations

Even though our model has improved handwriting recognition, there are still some challenges:

- **Messy or Hard-to-Read Handwriting:** If the handwriting is very unclear or inconsistent, the model might struggle to recognize the text correctly.
- **Grammar Mistakes:** The model focuses on recognizing words, but it doesn't fully understand complex German grammar, which might lead to incorrect transcriptions.
- **Limited Training Data:** Our dataset contains individual words and short phrases, but the model performs better when trained on full sentences or paragraphs. This could affect its accuracy when recognizing longer handwritten text.
- **Data Security Risks:** Since exams contain sensitive information, strong encryption and secure storage methods are needed to protect student data.
- **Model Training Challenges:** Fine-tuning the model requires high-quality, diverse handwriting samples, but our dataset may not cover all handwriting styles. This could

Future Improvements

- **Better Handwriting Recognition:** Improve the model by adding more diverse German handwriting samples for training.
- **Enhanced Post-Processing:** Implement AI-based grammar correction and spell-checking to refine OCR results.
- **Full Document OCR:** Develop a word segmentation method to extract entire handwritten sentences for better recognition.
- **Deployment & Accessibility:** Build a Flask-based web app for easy use and future integration with other platforms.