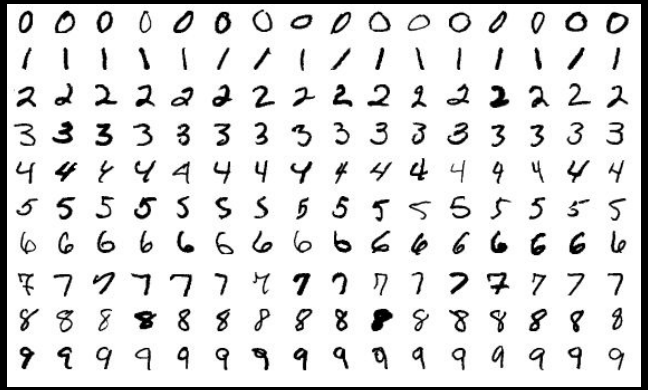
WriteThrough: Handwriting Recognition and Generation with OCR and cGANs !

Ali Zia and Nathan Demssie

Background

Handwriting recognition has been a key aspect of solving computer vision problems ever since the advent of Yann Lecun's digit recognition technology in the late 1980s. Many machine learning models exist to transform handwriting into typed text. **Yet, there exists few solutions for attempting to translate this typed text back into handwriting**, as well as providing recommendations and next-word prediction that is then outputted in a handwriting style. Our project combines a fine-tuned handwriting recognition pipeline with pre-existing word embeddings and a cGAN-based handwriting generator for generating singular letters in a word in a handwriting style.



Pipeline

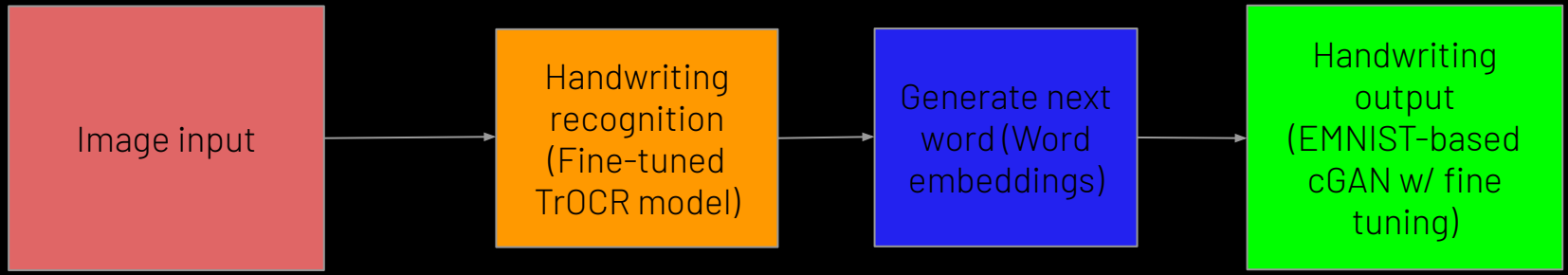
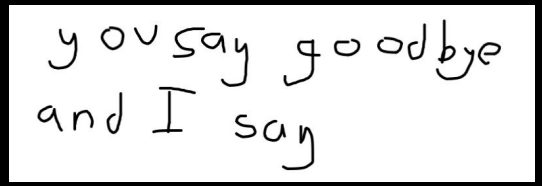
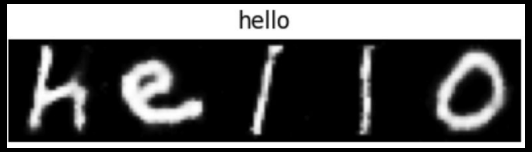
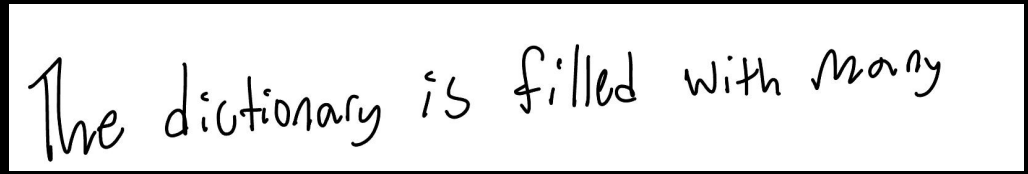


Image input
Handwriting recognition (Fine-tuned TrOCR model)
Generate next word (Word embeddings)
Handwriting output (EMNIST-based cGAN w/ fine tuning)

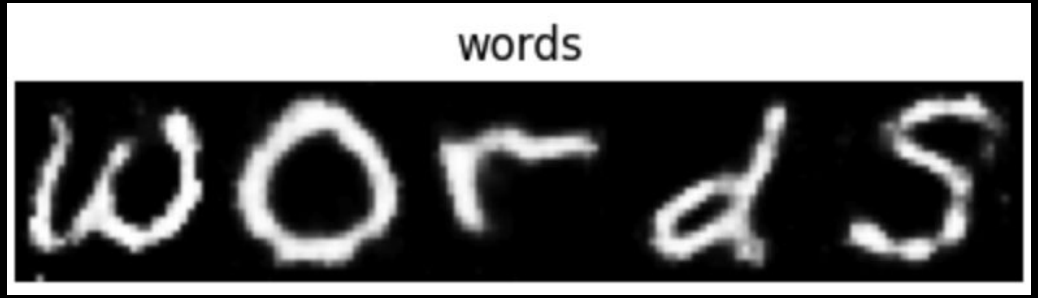

you say goodbye
and I say


hello




```
Some weights of VisionEncoderDecoderModel were not initialized from the model checkpoint at ndemssie/trocr-finetuned_DeepLRNGCV and are newly initialized: ['encoder.pooler.dense.bias', 'encoder.pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Generated Text: The dictionary is filled with many
```

```
[nltk_data]   Package reuters is already up-to-date!
Predicted next word for 'The dictionary is filled with many': words
```



What are TrOCRs?

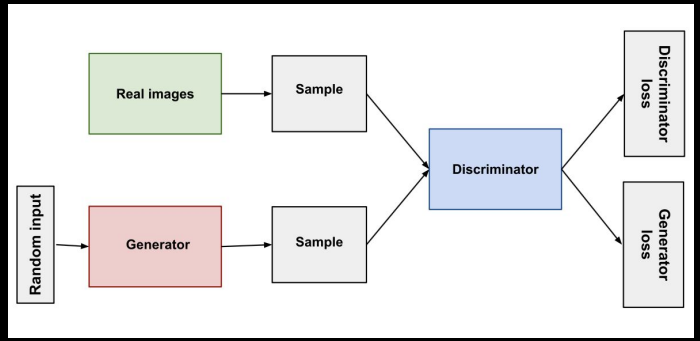
Optical Character Recognition (OCR) converts images of text into machine-readable formats by analyzing visual patterns to identify characters and text structures. OCRs are typically powered by deep learning models like CNNs or transformers, achieving high accuracy by recognizing complex patterns in fonts and handwriting. The very many applications for OCRs include digitizing documents, automating data entry, enabling text search, and supporting assistive technologies, with advanced tools handling multi-language text and noisy images.

The one downside with OCRs is that they perform pretty poorly with handwritten textual images. This is where TrOCRs come in. TrOCR (Transformer-based Optical Character Recognition) is a specialized deep learning model designed to perform OCR tasks, converting images of handwritten or printed text into machine-readable formats. It combines the strengths of Vision Transformers (ViT) for image encoding and autoregressive decoders, like GPT-style transformers, for generating text sequences. This integration enables TrOCR to excel at both visual feature extraction and contextual language modeling.

What is a GAN?

A Generative Adversarial Network (GAN) consists of two neural networks: a **generator** and a **discriminator**. The generator creates fake data (e.g., images) from random noise and labels, while the discriminator evaluates whether the data is real (from a training dataset) or fake (from the generator). The generator aims to produce increasingly realistic data to "fool" the discriminator, while the discriminator aims to accurately distinguish between real and fake data.

The two networks are trained simultaneously in a process known as **adversarial training**, where the generator improves by receiving feedback from the discriminator, and the discriminator improves by learning to better detect fake data. In a cGAN, the generation process is conditioned on additional information. In this case, that additional information is our character labeling per image.



Source: Google for Developers

Transcribing Handwritten Text

To transcribe written text, our team decided to fine-tune a pre-trained TrOCR model on the IAM Handwriting Database using the VisionEncoderDecoderModel class, which combines a BEiT-based encoder and a RoBERTa-based decoder. Specifically we decided to leverage a model that had not yet been trained on this dataset. When dealing with the IAM dataset we chose to prepare it by creating a PyTorch dataset, where images were resized and normalized, and text annotations were tokenized into labels using the TrOCRProcessor. The Seq2SeqTrainer was used for training, with custom configurations for evaluation using beam search and metrics such as Character Error Rate (CER). The fine-tuning process optimized the model for handwritten text, resulting in a lower CER and significantly improved recognition accuracy with respect to the base model.

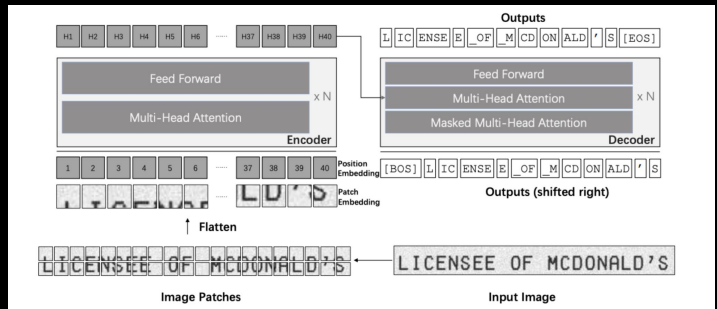


Figure 1: Model Architecture of TrOCR, where an encoder-decoder model is designed with a pre-trained image Transformer as the encoder and a pre-trained text Transformer as the decoder.

Generating Next Word

To tackle word generation, our team decided to opt for a standard word embedding system, first using a trigram model to improve accuracy. The model relies on sequences of three consecutive words from a large text corpus trained on a Reuters corpus. When a sentence is provided, it is first tokenized into individual words, and the last two words of the sentence are identified. The model then looks for all trigrams in the training data where the first two words match the last two words of the input sentence.

From these trigrams, it **selects the word that most frequently follows the last two words in the corpus**. This word is considered the most likely candidate to follow the input sentence and is predicted as the next word. If the input sentence does not contain enough words for a trigram (e.g., fewer than two words), the model defaults to using bigrams, where it matches the last word of the sentence with all bigrams in the training data and selects the most frequent word that follows. The combined approach outputted recommendations with a 70% accuracy on the test set.

Generating Handwriting

Multiple approaches were considered for attempting to generate handwriting from a text input. Popular research papers on the topic often make use of diffusion models to artificially generate handwriting based on a test dataset of tagged written words. **However, these solutions are often computationally expensive**, with fine-tunes on models such as Stable Diffusion creating a significant technical cost.

To avoid these computational issues, our team opted instead to build and evaluate a cGAN from scratch. A conditional generative adversarial network (cGAN) uses labels to generate outputs with characteristics similar to its training data set. This is different from GAN because it uses **labeled** data to provide context, which yields more targeted results from the generator.

This processed involved writing a generator and discriminator from scratch. For our data, we opted to use the **EMNIST dataset**–an extension of the popular MNIST dataset to include handwritten letters (124,000 samples). The input of the text from the embedding was passed in and then parsed letter by letter by the model. The model was evaluated through a loss analysis and an accuracy test on the discriminator.

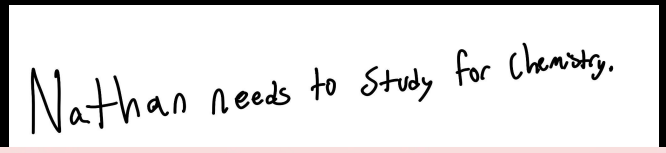
Results - Handwriting Recognition

After training for several epochs, the model achieved progressively lower CER scores, indicating improved transcription accuracy for handwritten text recognition. While fine-tuning the model we kept checkpoints at every 200 iterations and found that the training loss continually decreased (signifying the models improving ability to fit to the training data), the validation loss also continued to decrease (signaling that the model is not just memorizing the training data but is actually able to make accurate predictions on new and previously unseen examples). When comparing this to the paper's results we see that the TrOCR small had a CER of 4.22 percent, the TrOCR Base had a CER of 3.42 percent while the TrOCR Large had a CER of 2.89 percent. Our fine tuned model achieved a CER of 5.1 percent.

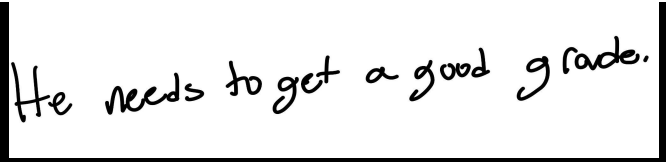
Model	Architecture	Training Data	External LM	CER
(Bluche and Messina 2017)	GCRNN / CTC	Synthetic + IAM	Yes	3.2
(Michael et al. 2019)	LSTM/LSTM w/Attn	IAM	No	4.87
(Wang et al. 2020a)	FCN / GRU	IAM	No	6.4
(Kang et al. 2020)	Transformer w/ CNN	Synthetic + IAM	No	4.67
(Diaz et al. 2021)	S-Attn / CTC	Internal + IAM	No	3.53
(Diaz et al. 2021)	S-Attn / CTC	Internal + IAM	Yes	2.75
(Diaz et al. 2021)	Transformer w/ CNN	Internal + IAM	No	2.96
$\text{TrOCR}_{\text{SMALL}}$	Transformer	Synthetic + IAM	No	4.22
$\text{TrOCR}_{\text{BASE}}$	Transformer	Synthetic + IAM	No	3.42
$\text{TrOCR}_{\text{LARGE}}$	Transformer	Synthetic + IAM	No	2.89

Step	Training Loss	Validation Loss	Cer
200	0.630900	0.568761	0.213607
400	0.582600	0.510521	0.159267
600	0.199300	0.552562	0.223106
800	0.029700	0.309079	0.092335
1000	0.071600	0.245185	0.072012
1200	0.015400	0.223296	0.061851
1400	0.000200	0.188567	0.051027

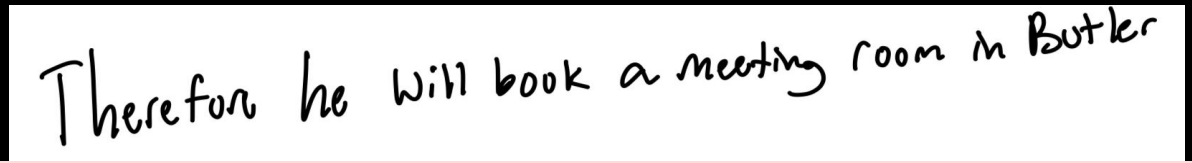
Results - Handwriting Recognition




```
Some weights of VisionEncoderDecoderModel were not initialized from the model checkpoint at ndemssie/trocr-finetuned_DeepLRNGCV and are newly
initialized: ['encoder.pooler.dense.bias', 'encoder.pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Generated Text: Nathan needs to study for chemistry .
```




```
Some weights of VisionEncoderDecoderModel were not initialized from the model checkpoint at ndemssie/trocr-finetuned_DeepLRNGCV and are newly
initialized: ['encoder.pooler.dense.bias', 'encoder.pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Generated Text: He needs to get a good grade .
```



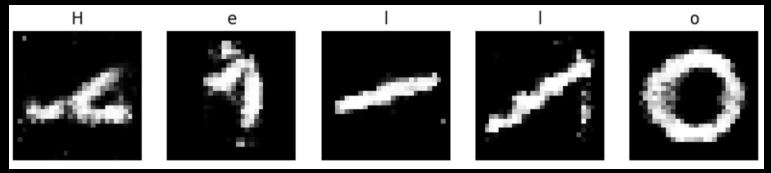

```
Some weights of VisionEncoderDecoderModel were not initialized from the model checkpoint at ndemssie/trocr-finetuned_DeepLRNGCV and are newly
initialized: ['encoder.pooler.dense.bias', 'encoder.pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Generated Text: Therefore he will book a meeting room in Butler
```

Results - Custom cGAN

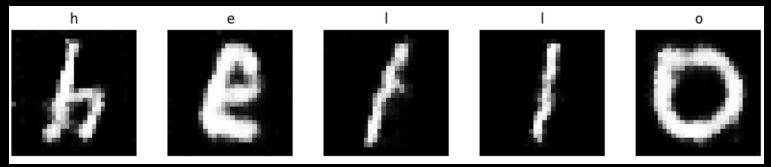
- After building a custom cGAN using the 27 classes of letters provided by EMNIST (extra class for N/A), our team noticed that the cGAN struggled with outputting the handwriting equivalent for letters with complex strokes, but performed relatively well on letters that were simpler to write, such as “l” and “o”
- A conditional GAN differs in its results compared to a standard image classifier, since it is outputting a drawn image, not a classification. This led us to the discriminator as the main source of accuracy calculations.
- We evaluated our cGAN by testing how well the discriminator performed when distinguishing real images from fake images, first feeding it a set of fake images and then real images to test its accuracy.

- Accuracy on real images: **74.76%**
- Accuracy on fake images: **70.02%**
- Total discriminator accuracy: **71.80%**

Batch size = 64 and no rotations applied



Batch size = 32 and rotations applied

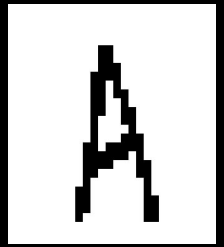


Some variations in readability are potentially due to the fact that EMNIST data included both lowercase and uppercase letters per class. Letters such as e and h, shown here with poor output, have a significantly different shape in lowercase versus uppercase.

Results - Custom cGAN (Fine-Tune)

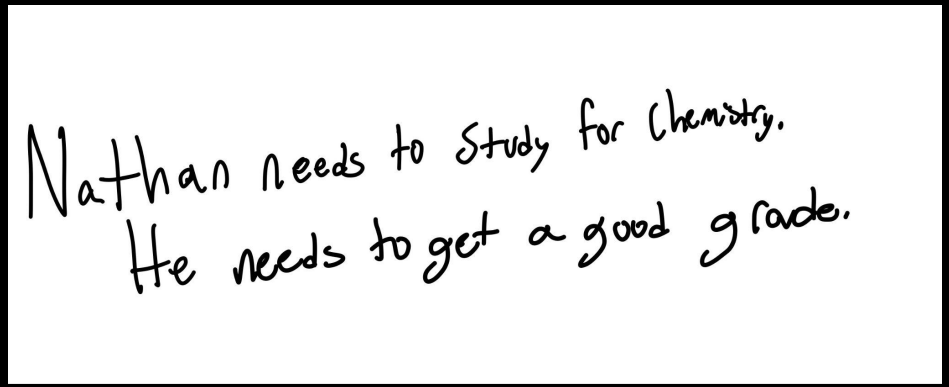
- To test the impact of fine-tuning on the cGAN, we used a smaller dataset of ~3000 handwritten images, which closely matched the handwriting style in our OCR model. We froze the embedding layers in the generator and discriminator to preserve general features learned during initial training. Despite this, the fine-tune did not significantly improve results, maintaining a 68% accuracy. Future adjustments to the learning rate and separating uppercase and lowercase data may help improve performance.

Example fine-tune data



Drawbacks/Limitations

- Limited Transcribing
  - For the most part when trying to test multiline textual images we were unable to garner any legible responses and this is something that we want to look further into.
  - Solution: Possibly refactoring images in order to partition lines and then from their feed them one by one within the model in order to formulate several sentences.




```
Some weights of VisionEncoderDecoderModel were not initialized from the model checkpoint at ndemssie/trocr-finetuned_DeepLRNGCV and are newly initialized: ['encoder.pooler.dense.bias', 'encoder.pooler.dense.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Generated Text: Northern market to get the front longitude .
```

Drawbacks/Limitations

- Limited word embedding functionality:
  - The data being trained on the Reuters corpus, a set of formal/academic words, limited the applicability of the recommendations to the more informal language of the handwritten inputs
  - Moving forward, a more advanced embedding step would likely improve the applicability of the pipeline
- Limited cGAN performance:
  - Our team found that the letter-by-letter approach of generating a handwritten word based on text **did not** always lead to the most readable text for both standard training and fine tuning. Our paper explores **a similar implementation** that provides clearer lettering and stitches outputs of the cGAN into words with line breaks. This could prove to be better suited for the use case of recommending longer phrases and actually outputting results to the end user.
  - The cGAN, while built from scratch, was not fine tuned on our own handwriting data. Moving forward, a fine tune specific to our handwriting would help personalize the data to the handwriting of the user to provide more useful results.

Applications

- Handwriting generation for patients with dyspraxia - a disorder that affects movement and coordination, especially handwriting.
- Sentence completion recommendations similar to Gmail but for handwriting apps such as Goodnotes
- Generating human handwriting for documents/signatures
- Automatic font generation based on a user's handwriting

Thank you!