HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

PROJECT REPORT

HANDWRITTEN TEXT RECOGNITION

Course Name	:	Introduction to Deep Learning
Course ID	:	IT3320E
Instructor	:	Prof. Nguyen Hung Son
Members	:	

Vu Tuan Minh (C)	20210597	minh.vt210597@sis.hust.edu.vn
Le Tuan Anh	20214874	anh.lt214874@sis.hust.edu.vn
Bui Minh Quang	20214925	quang.bm214925@sis.hust.edu.vn
Dinh Nguyen Cong Quy	20214927	quy.dnc214927@sis.hust.edu.vn
Phan Dinh Truong	20214937	truong.pd214937@sis.hust.edu.vn

Ha Noi, December 2023

Table of Contents

1. Introduction

Optical Character Recognition (OCR) is a dynamic field within machine learning, addressing the escalating needs of diverse industries like Finance (for eKYC), Education, and Government for digitalization. Traditionally, the predominant method for OCR relies on Artificial Neural Networks (ANN), often incorporating variations that include Convolutional Neural Networks (CNN). However, the introduction of the Transformer Architecture in 2017 has brought a new and compelling approach. This emerging paradigm, harnessing the power of Transformers in OCR, has garnered attention and demonstrated itself as a new promising approach in this research domain.

Within the scope of this project, our objective is to construct an OCR model utilizing a Transformer-based structure. Our primary contributions include:

- The development of a fully functional model capable of taking an image of a word as input and producing its corresponding label with a user-friendly GUI.
- Execution of diverse experiments involving two distinct models, accompanied by a comprehensive analysis based on the results.
- Implementation of grid-search experiments on these two models to refine their performance.

In dealing with the constraints of time and computational resources, we leverage publicly available models, code repositories, and free resources from platforms such as GitHub and Kaggle. This approach helps us to capitalize on well-established model theories and implementations, allowing us to focus on analysis and better deployment.

2. Dataset and Transformer Architecture

2.1. IAM Handwritting Dataset

The IAM dataset, which has been known as the standard of OCR problems and has been passed through writer identification and expert verification, is a collection of English words, sentences, and paragraphs. What's more, the dataset, which was contributed by **657 writers**, who come from British England, was a transcription. With its cohesive structures concluding a total of **1,539 handwritten pages** comprising **115,320 words** and categorized as part of modern collection, the dataset was the first choice of all those who thrive for conquering OCR problems. The database has all corresponding tags labeled.

With detailing the dataset, the IAM Handwritting Database contains unconstrained handwritten text formations, which were scanned at 300 dpi resolution and stored as PNG images with 256 gray colors. Plus, the database includes 1,539 scanned pages, 5,685 isolated and labeled sentences, 13,353 isolated and labeled text lines, 115,320 isolated and labeled words.

Based on the IAM Dataset, our team decided to choose a dataset, which was the IAM segmentation, which means all lines, paragraphs of the IAM dataset were divided into word levels, instead of sentence levels. Despites its segmentation, the quality of the database remains. The words, which were extracted from all pages and lines and and using an automatic segmentation, were tested for manual verification.

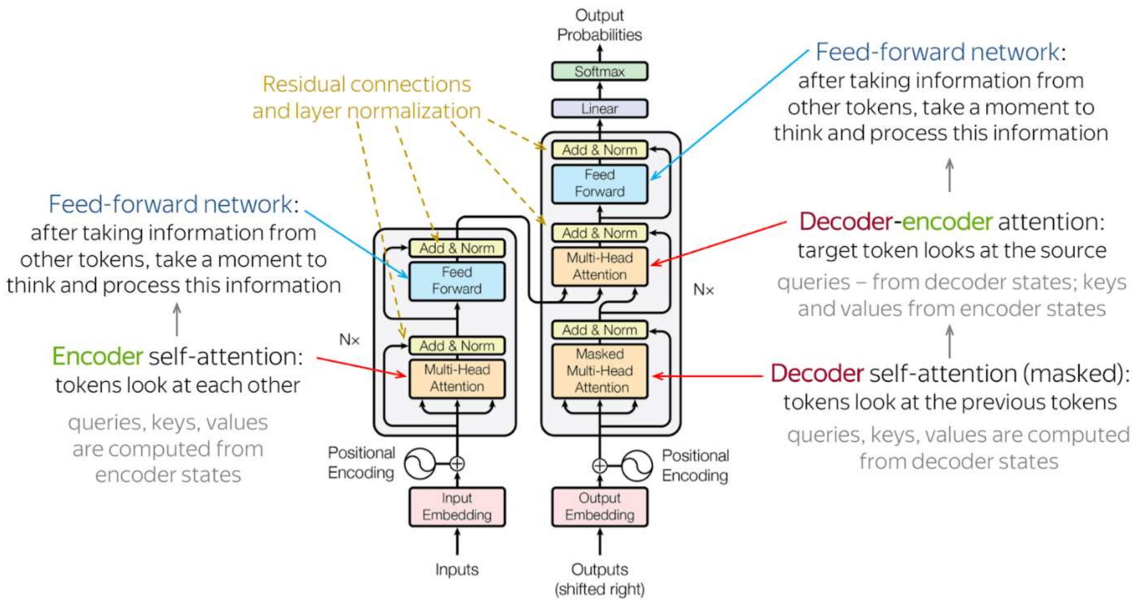
All word images are provided as PNG files and the corresponding form label files, including segmentation information and a variety of estimated parameters, are included in the image files as meta-information in XML format which is described in XML file and XML file format (DTD).

2.2 Visual Transformer

As the name suggests, **Visual Transformer** (published in ICLR 2021) is a model based on Transformer Architecture but for vision tasks (classification in this paper). To understand this model there are 2 main concepts: **Attention** and **Transformer**.

2.2.1. Attention and Transformer

Attention mechanism was proposed as early as 2010 (though it is not named "Attention" at this time) and is particularly popular in the NLP field. The core idea of this mechanism is to focus more on a part of the input when making a prediction. In [6], authors have proposed an attention mechanism (combined with RNN) that can vastly improve the machine translation task and since then Attention has gained more and more attention in the NLP field. Following this trend, in 2017, [7] was published in NIPS, which introduced a new model, Transformer, to utilize this Attention Mechanism without RNN parts for translation tasks.

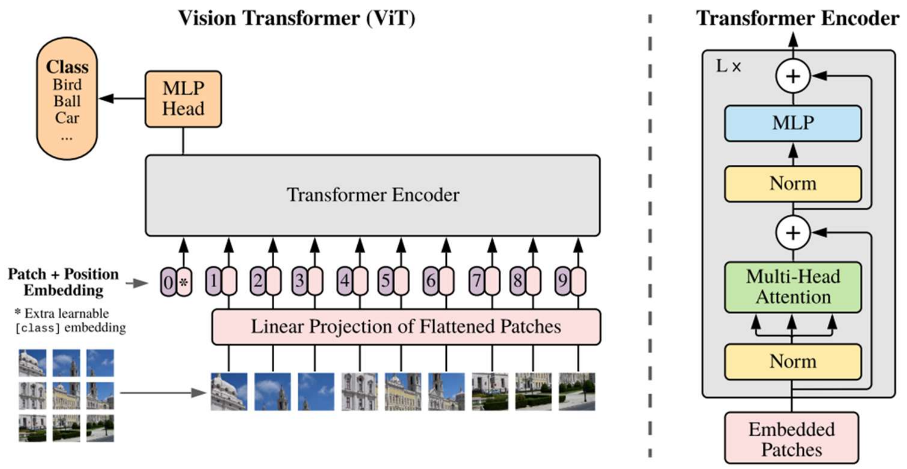

Output Probabilities
Softmax
Linear
Residual connections and layer normalization
Feed-forward network: after taking information from other tokens, take a moment to think and process this information
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Decoder-encoder attention: target token looks at the source
queries – from decoder states; keys and values from encoder states
Feed-forward network: after taking information from other tokens, take a moment to think and process this information
Nx
Add & Norm
Masked Multi-Head Attention
Decoder self-attention (masked): tokens look at the previous tokens
queries, keys, values are computed from decoder states
Encoder self-attention: tokens look at each other
queries, keys, values are computed from encoder states
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

2.2.2 Visual Transformer

Inspired by Transformer success in NLP, researchers attempted to use them in vision tasks as well. Specifically, in [8], authors have introduced a Transformer-based model, which can achieve excellent performance on image classification tasks with different datasets (88.55% on ImageNet, 90.72% on ImageNet-ReaL, 94.55% on CIFAR-100, and 77.63%).

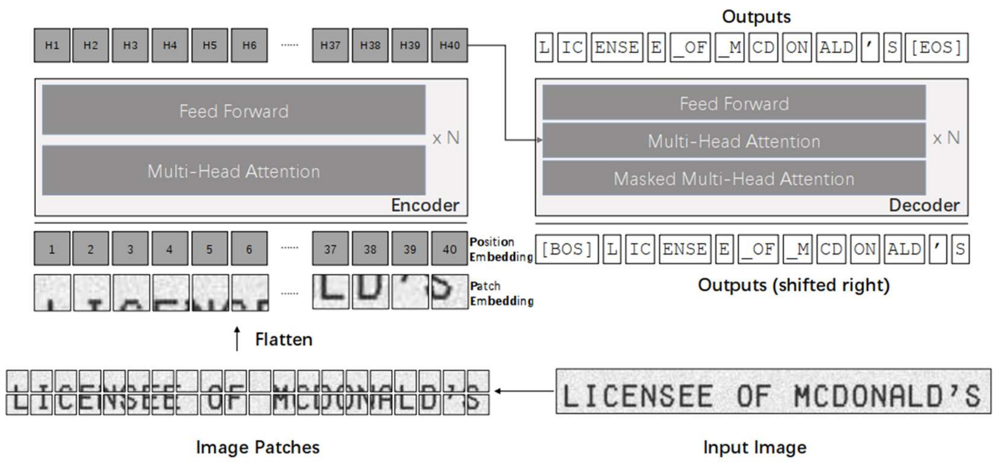
Visual Transformers is, as its name suggests, based on the principle of the Transformer with Attention mechanism for different parts of the image). An image is divided into 16x16 patches (which explains the paper name), each of which is then embedded and concatenated with its Positional Embeddings and passed through the standard Transformer model to output the result.

The most noticeable limit of Transformer-based models compared to normal CNN-based ones for vision tasks is the lack of inductive bias which causes Transformer-based models to perform worse on small to medium size datasets. One possible explanation for this is that the CNN models often possess some degree of translation equivariance, locality, etc. (which makes CNN a well-established type of model for vision-related tasks). However, as the size of the datasets increases, this inductive bias effect is overshadowed, thus, Transformed-based models can produce very competitive results.



3. Proposed Model

3.1 TrOCR Architecture



Our initial model is primarily inspired by the architecture presented in reference [9]. The model structure closely resembles the **Vision Transformer** (ViT) architecture. Initially, input images are resized to a fixed dimension of (H, W) and then divided into distinct patches, each accompanied by Positional Embedding. The decoder employs the key (K) and value (V) from the encoder, while the query (Q) is derived from the decoder's own output. This process is repeated N times (with N heads) to capture various aspects of the input, allowing the model to attend to different features.

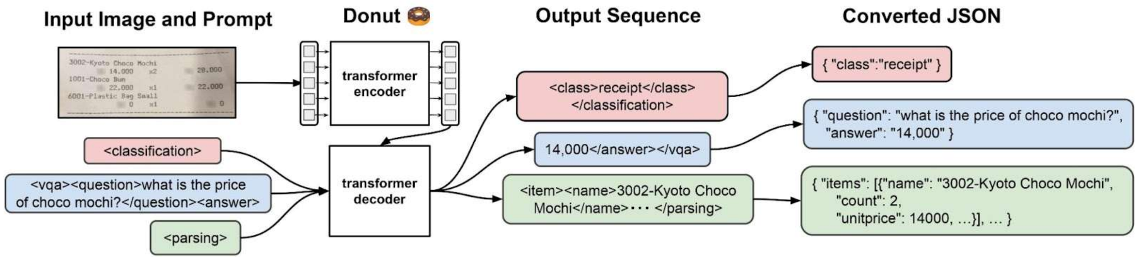
To ensure that the model utilizes only information from its preceding predictions, **Masked Multi-Head Attention** is implemented. This implementation mirrors the standard Transformer architecture commonly used for natural language processing tasks, with a particular focus on translation.

In our model implementation, we leave several hyperparameters as default:

- The number of heads: 12 (both Encoder and Decoder)
- Encoder Embedding: 3 channels to 768 channels with 16x16 kernel size
- Encoder Query, Key, Value: linear transformation of size 768 (size of each embedding of patches)
- Decoder: Sinusoidal Position Embedding, linear projection layer from 50265 vocab size to 1024-dimension vector.

3.2 Donut

The **Donut model**, known for its proficiency in multimodal tasks involving both text and image processing, forms the cornerstone of this proposed method. We follow a multimodal approach by processing both text and images. The images are first resized to a consistent dimension, ensuring uniformity across image inputs. Then, we initialize a tokenizer based on the **XLM-RoBERTa** architecture, to enable the conversion of raw textual information into tokenized sequences, enabling compatibility with the Donut model's input requirements. We integrate the configured text tokenizer and image processor into the **DonutProcessor**. This combined processor manages both text and image data, facilitating their transformation into formats compatible with the Donut model's input requirements.


Input Image and Prompt
Donut
Output Sequence
Converted JSON
transformer encoder
transformer decoder
<classification>
<vqa><question>what is the price of choco mochi?</question><answer>
<parsing>
<class>receipt</class></classification>
14,000</answer></vqa>
<item><name>3002-Kyoto Choco Mochi</name>··· </parsing>
{ "class":"receipt" }
{ "question": "what is the price of choco mochi?", "answer": "14,000" }
{ "items": [{"name": "3002-Kyoto Choco Mochi", "count": 2, "unitprice": 14000, …}], … }

Our methodology mainly revolves around the pre-trained variant of the VisionEncoderDecoderModel, which is fine-tuned for tasks involving text detection and recognition within images. The **DonutSwinModel** encoder and the **MbartForCausalLM** decoder are used in the model implementation.

In this project, most of the hyperparameters are left as default, apart from the vocabulary size of the embedding layer, where we change from 57525 to 250002.

3.3 Adam Optimizer

In this project, we employed the **Adaptive Moment Estimation** (Adam) method, an extension of the renowned Stochastic Gradient Descent technique. Adam calculates adaptive learning rates for individual parameters based on first and second-moment gradient estimates. The name “Adam” is derived from adaptive moment estimation.

This method is strategically designed to harness the strengths of two widely used approaches: AdaGrad, effective for sparse gradients, and RMSProp, known for its performance in online and non-stationary settings.

The Adam Optimizer dynamically adjusts learning rates for each parameter by estimating both the first and second moments of the gradients. It proves highly effective in training deep neural networks and has gained popularity across diverse applications, including computer vision tasks like facial expression recognition.

For each paremeter w^j:

$$w_t = \beta \times v_{t-1} - (1 - \beta_1) \times g_t$$
$$s_t = \beta_2 \times s_{t-1} - (1 - \beta_2) \times g_t^2$$
$$\Delta w_t = -\eta \frac{v_t}{\sqrt{s_t} + \epsilon} \times g_t$$
$$w_{t+1} = w_t + \Delta w_t$$

Where:

η*: Initial learning rate*

g_t*: Gradient at time* t *along* w^t

v_t: *Exponential Average of gradients along* w_j

s_t*: Exponential Average of squares of gradients along* w_j

β_1, β_2*: Hyperparemeters*

3.4 Charater Error Rate

The Character Error Rate (CER) is a metric commonly employed in the evaluation of word-related tasks, particularly in the field of natural language processing and optical character recognition.

CER measures the difference between a predicted sequence and a target sequence by quantifying the number of insertions, deletions, and substitutions needed to align the two sequences. CER is calculated as the ratio of the total edit operations (including substitution, insertion, and deletion) to the length of the target sequence, expressed as a percentage.

This metric finds widespread application in assessing the performance of algorithms and models dealing with character-level predictions, such as in transcription tasks or the recognition of handwritten or scanned text. For this reason, we use this metric to calculate the loss function during the training and validation.

$$CER = \frac{S + D + I}{N}$$

Where:

S*: Subtitution*

D*: Deletion*

I*: Insertion*

4. Experimental Results

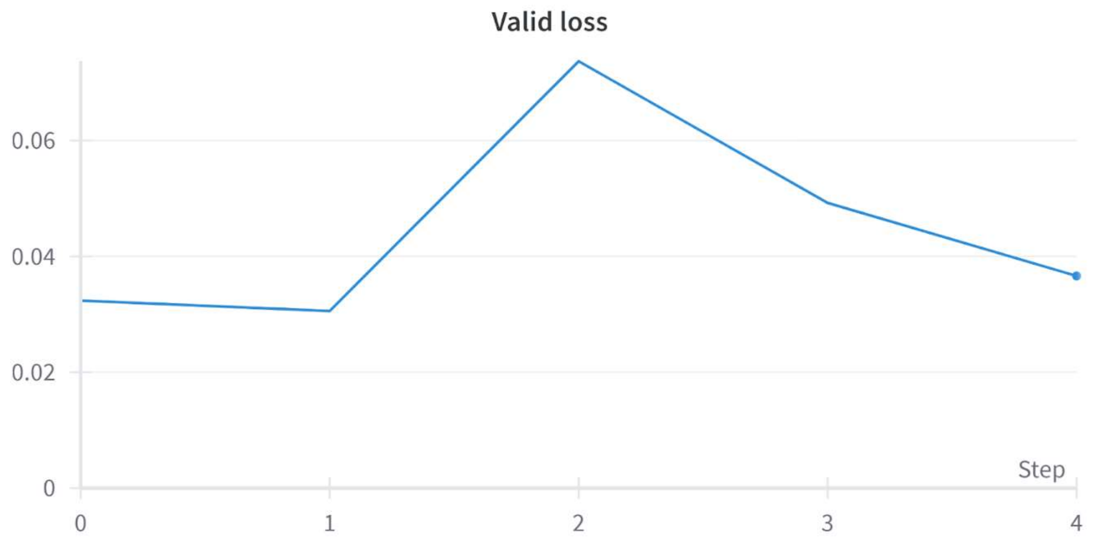
All the value of loss function is the value of cross entropy.

After having a comprehensive explanation of the architecture of the models, we will take insight into how some changes could enhance the overall performance on the

OCR tasks. Therefore, we made some modifications to figure out what learning rate could be the optimal solution for the task. Note that because of the extreme computational cost, grid-search can only be performed with a limited number of epochs (We used 5).

4.1. Overall Accuracy

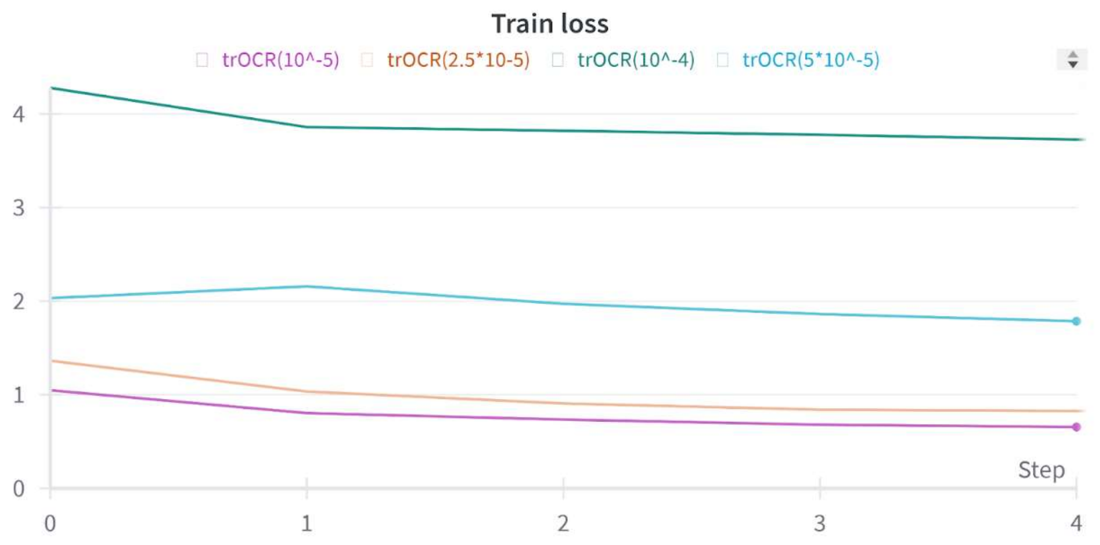
The model train loss for TrOCR model varies on different runs, however, the valid loss is around 0.033.



4.2. Grid Search

Learning rate, which is useful for optimization algorithms, is a tuning parameter, which has the responsibility for step size at each iteration, exploring the optimal solution for the loss function as well. Based on the definition, we spark the idea to conduct the learning rate experiment, so that we could find the optimal solution for OCR tasks.

- **trOCR:**

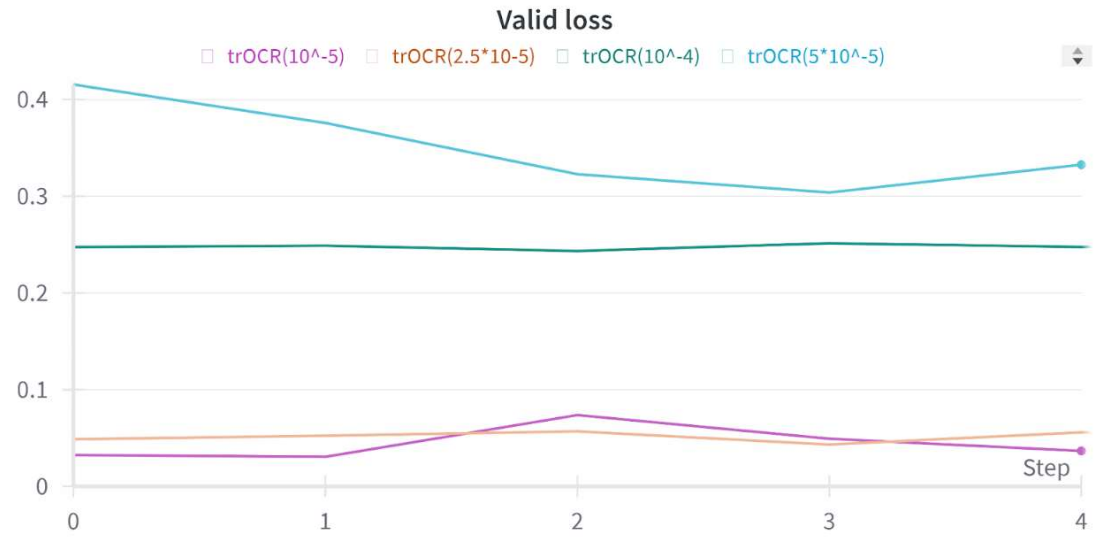


The train loss graph shows the performance of trOCR model with 4 different learning rates: 10^{-5}, 2.5×10^{-5}, 10^{-4} and 5×10^{-5}.

Overall, the general trend of the train loss of 4 categories was a decrease over 5 epochs. In addition, the trOCR model with learning rate at 10^{-5} was the lowest, while the opposite was true for that at 5×10^{-5}.

The disparity of the train loss of trOCR with learning rate 5×10^{-5} and10^{-4} was significant, which the former, whose train loss was over 4 and gradually decreased to 3.8 after 5 epochs, double the the latter, whose train loss was at 2 then gradually fell to under 2.

Differing from the gap between the 2 first categories, the **trOCR model** with learning rate at 10^{-5} and 2.5×10^{-5} share an increase trend. In detail, those both started at around 1 in the first epoch and decreased to around 0.7 in the last epoch. However, the gap between these learning rates and the trOCR with 10^{-4} learning rate is more pronounced. Theformersweres just about a fourth of the latter.



The valid loss shares similar trends as the train loss. The 5×10^{-5} learning rate witnessed a decrease until the epochs 3 before minimally bouncing back to the last epoch. In contrast, the valid loss of the second-place learning rate remains nearly unchange over the period. Moreover, the 5×10^{-5} learning rate slightly decreased, while the trOCR 10^{-5}, which was the lowest in 4 learning rates examined,reachedd its peak at epoch 3 and reduced to around 0,7.

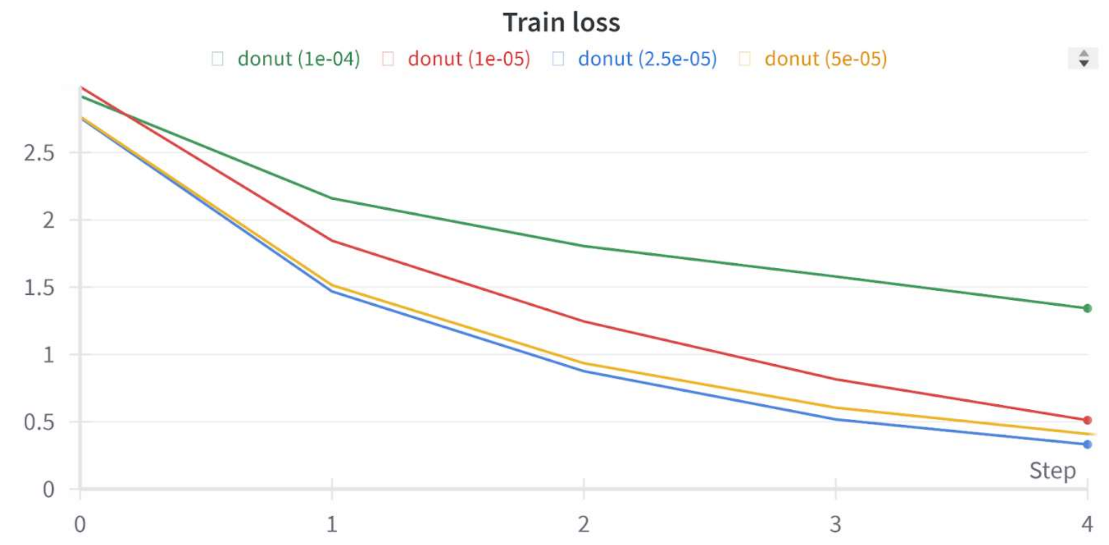
The train loss graph shows the performance of trOCR model with 4 different learning rates, 10^{-5}, 2.5×10^{-5}, 10^{-4}, and 5×10^{-5}.

Overall, the train loss value of 4 LR decreases over 5 epochs and the larger the learning rate the larger the loss value tends to be. The model with 10^{-5}learning rate has the lowest training loss and valid loss while the valid loss for 10^{-4} is much higher

The model with the highest learning rate 10^{-4} though only trained with the same small number of epochs, cannot achieve decent loss values of other models. This may suggest that due to such a high learning rate, the model may fluctuate between different points and cannot achieve better results.

trOCR models with learning rates at 10^{-5} and 2.5×10^{-5} seems to fit the model better as they offer better training and loss results. As the learning rate decreases, the model may even get better results, however at the cost of potentially much slower training so we stop at the learning of 10^{-5}.

- **Donut**

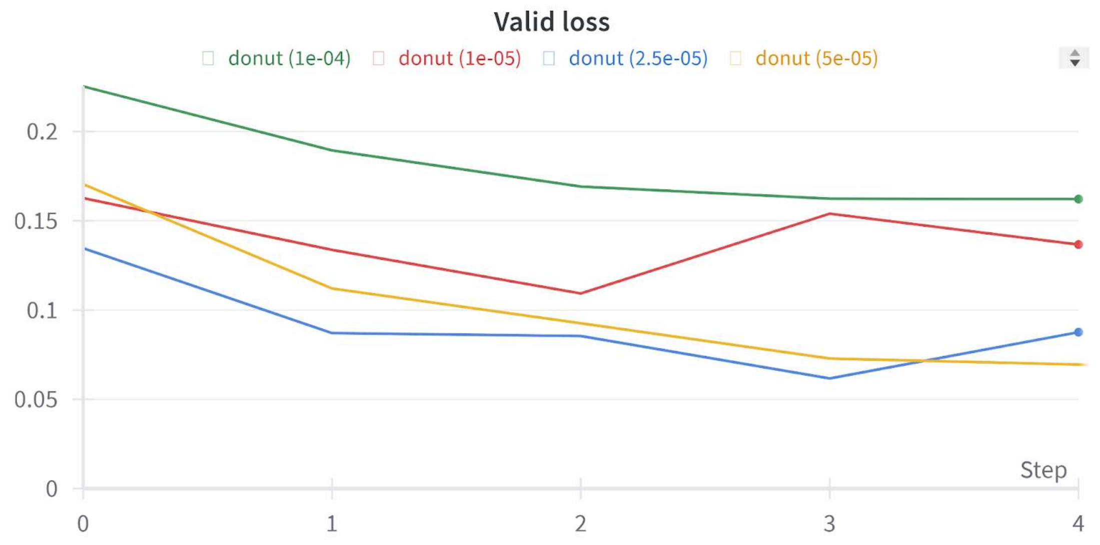


The graph compares the loss when training Donut model corresponding 4 different learning rates, 10^{-4}, 10^{-5}, 2.5×10^{-5}, and 5×10^{-5}.

Overall, the four-learning rate followed a decrease. Additionally, the learning rate 10^{-4} was the highest over 4 categories.

The learning rate 10^{-5} was around 3, closely followed by the 10^{-4} donut model. However, the gap between them widened after 4 epochs. Both figures for 10^{-4} and 10^{-5} roughly decreased, and then switched places after the first epoch.

The others shared the same trends, and at the same pace. They were both at around 2.7 and decreased by 2.2 after 5 epochs.



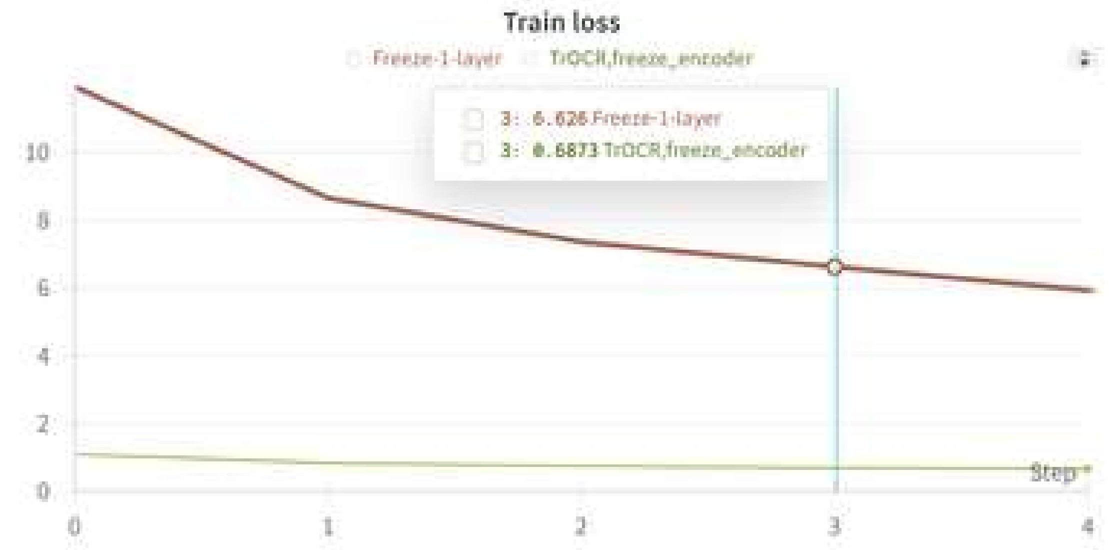
The valid loss graph compares 4 different learning rates. The donut with 10^{-4} learning rate was at around 0.3, minimally decreased by a half over 5 epochs examined. The 5×10^{-5} donut was at 0.16, followed closely by donut model with 10^{-5} learning rate, then falled back to around 1.0, which was switched place with the latter, at the second epoch before reached its own peaks at under 0.16 and negligible fell, at which time was became the lowest over 5 epochs.

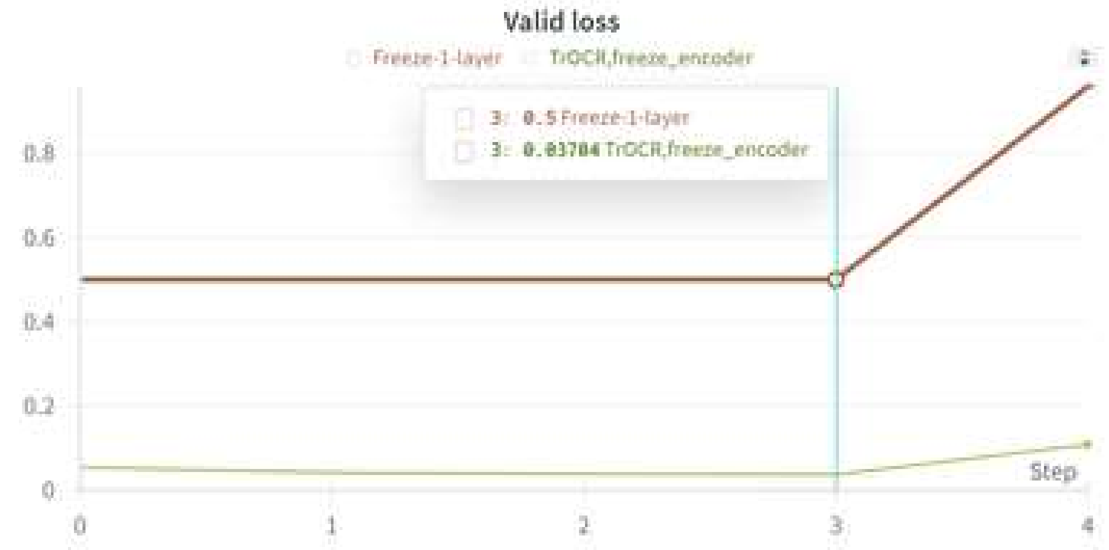
The graph compares the loss when training the Donut model corresponding to 4 different learning rates, 10^{-4}, 10^{-5}, 2.5×10^{-5}, and 5×10^{-5}.

Overall, as for a small number of training epochs, the loss value for all 4 models tends to decrease. Unlike the TrOCR models, Donut experiments might indicate the best learning rate result is around 2.5×10^{-5}.

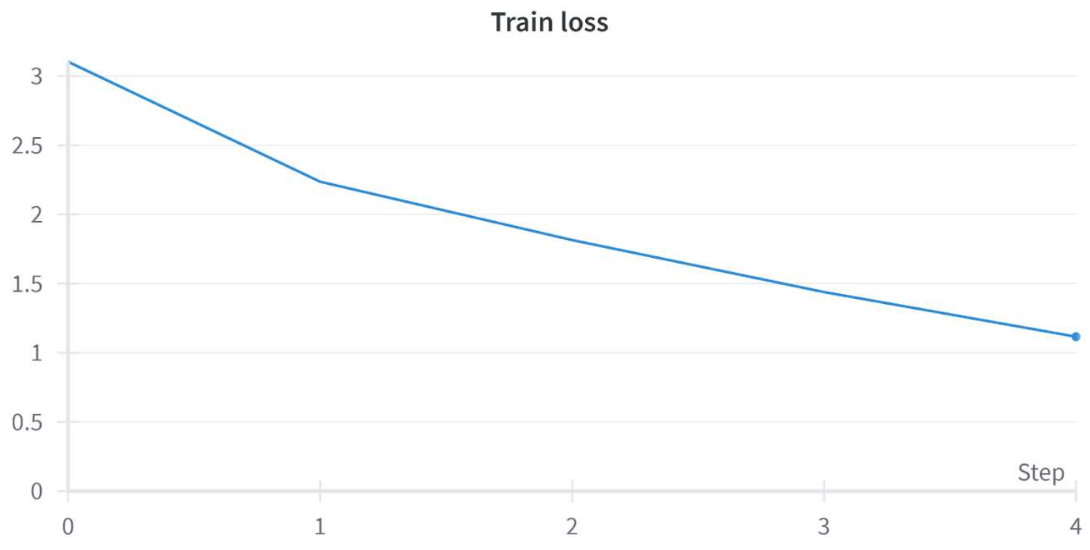
With the limited availability of computational resources, we cannot perform experiments with more than 5 epochs for each learning rate/model. From the training graph, all the models might benefit from more training as the training loss value tends to decrease rapidly.

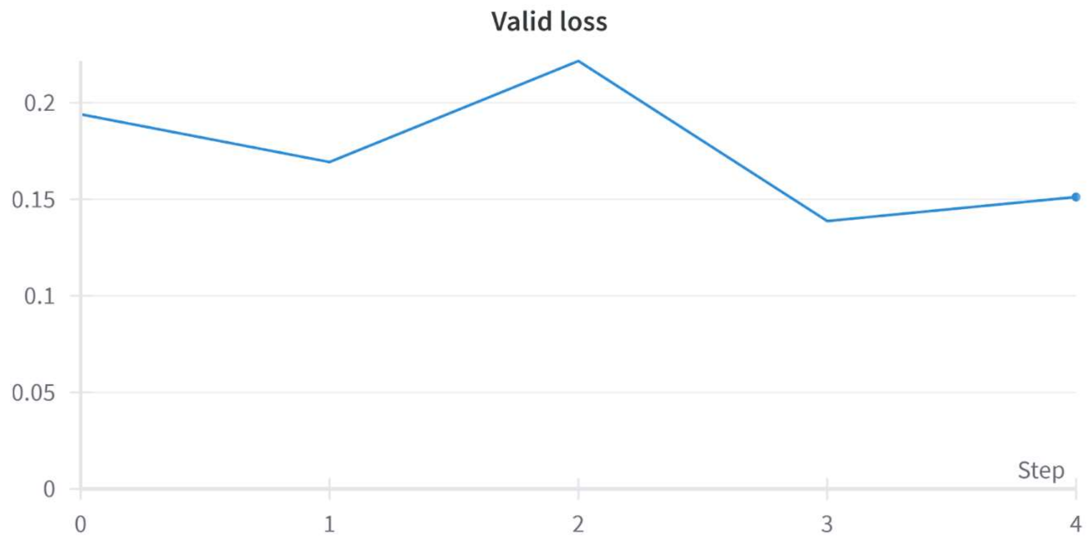
4.2. Freezing Layers





Freezing one head of the encoder results in a slightly faster computation time. However, it results in model inconsistency, as a result for the best result, as in several sources suggests, we should either freeze all encoders or train all of them.





The same case goes with the Donut model, as the computation time shrinks from about 1.5hr per epoch on T4x2 to less than 1hr, but the valid loss ramps up due to the problem of model inconsistency.

5. FutureWork

- **Training**: More work should be done on training and testing. As we have already mentioned throughout the report, we are limited in computational resources, one epoch with unfrozen TrOCR takes up to 2hr on T4x2 and around 1.5hr on P100. Also, Kaggle only provides GPU at 12 hrs per session.
- **Fine-Tuning**: More work needs to be done on freezing different parts of the TrOCR model. It is suggested that the embedding layer of the decoder can be replaced with some other embeddings

- **Performance:** Our models perform well in the case of single word but become worse and worse as the length of the sentence increases. Thus, training the models with more long-sentence data would be among our planned work in the future. The problem of word format inconsistency would be another challenge for our models, as they output the correct result for samples that have nearly similar handwriting format to the training data, but perform poorly with some specifically hard characters, such as ‘r’ or ‘f’.

References

[1] U. Marti and H. Bunke. A full English sentence database for off-line handwriting recognition. In Proc. of the 5th Int. Conf. on Document Analysis and Recognition, pages 705 - 708, 1999.

[2] U. Marti and H. Bunke. Handwritten Sentence Recognition. In Proc. of the 15th Int. Conf. on Pattern Recognition, Volume 3, pages 467 - 470, 2000.

[3] M. Zimmermann and H. Bunke. Automatic Segmentation of the IAM Off-line Database for Handwritten English Text. In Proc. of the 16th Int. Conf. on Pattern Recognition, Volume 4, pages 35 - 39, 2000.

[4] U. Marti and H. Bunke. The IAM-database: An English Sentence Database for Off-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition, Volume 5, pages 39 - 46, 2002.

[5] S. Johansson, G.N. Leech and H. Goodluck. Manual of Information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital Computers. Department of English, University of Oslo, Norway, 1978.

Transformers Models:

https://blogs.nvidia.com/blog/what-is-a-transformer-model/

https://arxiv.org/abs/2109.10282

Hugging Face Frame:

https://www.linkedin.com/learning/applied-ai-getting-started-with-hugging-face-transformers/introduction-to-hugging-face#:~:text=Hugging%20face%20is%20an%20online,Co.

https://wandb.ai/int_pb/huggingface/reports/An-Introduction-To-HuggingFace-Transformers-for-NLP--VmlldzoyOTgzMjI5