

Image-to-LaTeX Converter for Mathematical Formulas and Text Deep Learning

Matthijs van der Lende (S4325621)
Jeremias Lino Ferrao (S4626451)
Niclas Müller-Hof (S4351495)

February 2, 2025

Abstract

In this project, we implemented and reproduced the transformer based optical character recognition model from [Gurgurov and Morshnev \(2024\)](#) for converting images of LaTeX equations or handwritten equations to the equivalent LaTeX code. The model uses an encoder-decoder architecture consisting of a Swin Transformer as the encoder and the GPT-2 model as the decoder. A base model was trained on the synthetic dataset from the authors and another model was trained by finetuning the base model on an handwritten dataset. Using equivalent (where applicable) hyperparameters we were able to reproduce the results from the authors. The model showed solid performance in terms of BLEU score on the synthetic test data. Finetuning showed significant improvements in BLEU score on handwritten data, but slightly reduced the score on the synthetic data. Additional inference testing showed that the model performed worse on out of distribution data (e.g., screenshots made by the user) than on test data, which we suspect was due to a lack of variety in the training data and lack of data augmentation.

1 Introduction

The automatic recognition of mathematical formulas and symbolic text from scanned documents, handwritten notes, or rendered images is a longstanding challenge in the field of Optical Character Recognition (OCR) ([Zanibbi & Blostein, 2012](#); [Memon, Sami, Khan, & Uddin, 2020](#)). While traditional OCR methods have achieved high accuracy on standard printed text, mathematical expressions introduce added complexity because of specialized symbols, nonlinear layouts, and stacked elements such as fractions or exponents ([Hamad & Kaya, 2016](#); [Garain & Chaudhuri, 2005](#)). Early approaches to mathematical OCR often involved handcrafted feature extraction and rule-based parsing, as seen in systems like Tesseract, which performed well on standard characters but struggled with math-specific notation and two-dimensional structures ([Smith, 2007](#); [Islam, Islam, & Noor, 2017](#)).

Over the past decade, deep learning has significantly improved OCR performance. Convolutional Neural Networks (CNNs) have proven particularly effective for feature extraction ([Wei, Sheikh, & Ab Rahman, 2018](#); [Srivastava, Priyadarshini, Gopal, Gupta, & Dayal, 2019](#)), and Recurrent Neural Networks (RNNs) such as LSTMs have been widely adopted for sequence modelling in text recognition ([Parthiban, Ezhilarasi, & Saravanan, 2020](#); [Michael, Labahn, Grüning, & Zöllner, 2019](#)). However, these architectures often require careful processing of images to handle complex arrangements of mathematical symbols. Methods that combine CNNs with attention-based decoders, such as the Image-to-Markup model proposed by [Deng, Kanervisto, Ling, and Rush \(2017\)](#), demonstrated how attention mechanisms can capture both global structure and fine local details, inspiring subsequent Transformer based-pipelines ([Zhong, ShafieiBavani, & Jimeno Yepes, 2020](#)). More recently, Transformer-based models have emerged as a powerful alternative, allowing

for holistic, end-to-end systems that unify visual encoding and text decoding (Wick, Zöllner, & Grüning, 2021). Vision Transformers, such as ViT and Swin Transformer, capture long-range dependencies within an image, while large-scale text models like GPT-2 provide robust language understanding and generation capabilities (Khan et al., 2022; Radford et al., 2019).

In mathematical OCR specifically, the TrOCR framework (Li et al., 2022) has demonstrated that pairing a transformer-based vision encoder with a text decoder can yield highly accurate results on various datasets. This end-to-end paradigm has been explored in diverse systems, including Pix2Tex MFR (Breezedeus, 2024) and Sumen (Trung, 2024), which differ in their choice of encoder architectures and training.

Building on these advances, the present project reproduces and extends the work of (Gurgurov & Morshnev, 2024), who introduces a Swin Transformer encoder and GPT-2 decoder model for Image-to-LaTeX conversion in a framework known as Im2Latex. The key novelty of their approach is that they use large pre-trained vision and language transformers in a unified encoder-decoder framework, TrOCR, which is adept at handling various formula layouts. After training on a large set of synthetically generated formulas, their model showed promising results on both printed and handwritten expressions, with minimal reliance on specialized heuristics or post-processing. In our work, we follow their methodology to replicate their main findings, and explore their fine-tuning strategy using Low-Rank Adaptation (LoRA) (Hu et al., 2021), a parameter-efficient fine-tuning technique that freezes the pre-trained model weights and injects trainable low-rank decomposition matrices into each layer of the Transformer architecture. This approach significantly reduces the number of trainable parameters compared to full fine-tuning. It also highlights how large-scale pre-trained Transformers can be flexibly tailored to more specialized or niche datasets.

2 Background

The Im2Latex model by Gurgurov and Morshnev (2024) follows TrOCR meta-architecture from Li et al. (2022) by utilizing pre-trained transformers models in an encoder-decoder fashion (see Figure 1). The encoder processes the image and produces a set of feature vectors that represent the visual information. The decoder then generates the LaTeX code, using a cross-attention mechanism to incorporate these feature vectors into the multi-head attention layer of each block. This mechanism allows each generated LaTeX token to be directly influenced by the complete set of image features from the encoder, ensuring that the output accurately reflects the visual input. Formulated mathematically, the purpose of the whole Im2Latex OCR model is to learn conditional generative models of the form $p(\mathbf{y}|\mathbf{x})$, where \mathbf{y} is a valid LaTeX string and \mathbf{x} is an image of a handwritten or compiled LaTeX equation:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^T p(y_t|y_{1:t-1}, \mathbf{x}). \quad (1)$$

As seen in Equation 1, the model is auto-regressive where $\mathbf{y} = (y_1, y_2, \dots, y_T)$ is a LaTeX string and $y_{1:t-1} = (y_1, \dots, y_{t-1})$ represents all tokens up to time t . At inference time the encoder of the model is presented with an image \mathbf{x} and the decoder autoregressively generates the LaTeX equation. The training data \mathcal{D} of size N are image-text pairs $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) \mid i = 1, \dots, N\}$ and the whole model θ with P parameters $\theta \in \mathbb{R}^P$ is trained end-to-end by minimizing the negative log-likelihood:

$$\mathcal{L}(\theta) = - \sum_{t=1}^T \log p_{\theta}(y_t|y_{1:t-1}, \mathbf{x}). \quad (2)$$

2.1 Encoder

The encoder block is a Vision Transformer (ViT). The original ViT (Dosovitskiy et al., 2021) applied transformer layers to image data by dividing an image \mathbf{x} into 16×16 patches. Each patch was then mapped to an input embedding through a learned linear transformation. A sequence of embedding

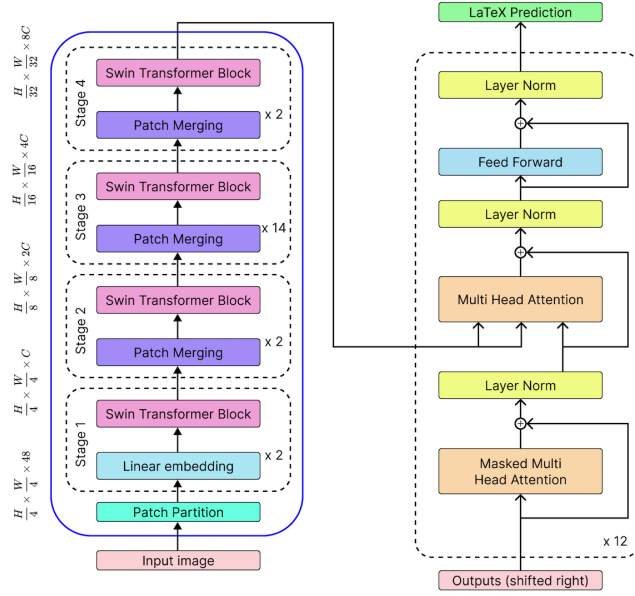


Figure 1: Visualization of the Im2Latex model architecture from Gurgurov and Morshnev (2024). The Swin Transformer encoder (left) provides a hierarchical feature representation of the input image. These features are then used by the GPT-2 decoder (right) to generate the LaTeX prediction through a series of attention and normalization layers.

vectors is then obtained which can be passed through a block of transformer layers, producing a sequence of feature vectors ($\mathbf{z}_1, \dots, \mathbf{z}_m$). Although this approach works, breaking up the image does not scale well as the image resolution increases and the global receptive field of the image results in challenges in attending to fine detail (for example when doing semantic segmentation).

The Swin Transformer (Liu et al., 2021) overcomes these limitations by constructing hierarchical feature maps, mimicking the multi-scale representations in CNNs. This is achieved by progressively merging patches in deeper layers. Self-attention is computed locally within non-overlapping windows that partition the image to further reduce computational cost. This results in a linear computational complexity with respect to image size, as opposed to the quadratic complexity of standard ViTs.

The specific Swin Transformer model used in this work is the Swin-B (Base) variant, which was pre-trained on the ImageNet-22k dataset. This corpus consists of 14 million images across 21,841 classes, providing a diverse and large-scale pre-training corpus. Notably, the Swin Transformer achieved state-of-the-art performance on various benchmarks, including image classification, object detection, and semantic segmentation, at the time of its release, demonstrating its effectiveness as a general-purpose vision backbone.

2.2 Decoder

The decoder in our model is based on the GPT-2 small architecture (Radford et al., 2019), a large-scale self-supervised language model built upon the Transformer (Vaswani et al., 2023). Unlike traditional approaches, GPT-2 demonstrates that a single language model can achieve strong performance on a variety of language tasks in a zero-shot setting, without any task-specific training.

GPT-2 small is a Transformer decoder-only model that utilizes a stack of Transformer blocks. Each block comprises a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The multi-head self-attention mechanism allows the model to attend to different parts of the input sequence when generating each token by computing multiple attention outputs in parallel (multiple “heads”), each attending to a different part of the sequence. Layer normalization (Ba, Kiros, & Hinton, 2016) and residual connections (He, Zhang, Ren, & Sun, 2015)

are employed around each sub-layer to facilitate training of deep networks.

The training data for GPT-2 was WebText, a large dataset of over 8 million documents collected by scraping web pages linked from Reddit that received at least 3 karma, ensuring a diverse and high-quality dataset of natural language. GPT-2 achieved state-of-the-art results on a wide range of language tasks without any task-specific fine-tuning. In our model, the GPT-2 decoder was used to generate LaTeX code autoregressively, conditioned on the visual features extracted by the Swin Transformer encoder.

2.3 Cross-Attention

The cross-attention mechanism in the TrOCR meta-architecture is a crucial component that bridges the encoder and decoder, enabling the decoder to leverage the visual features extracted from the input image. In the standard Transformer decoder, each layer contains a self-attention sub-layer and a feed-forward sub-layer. In the decoder, a third sub-layer, the cross-attention module, is introduced between these two. This module performs multi-head attention over the output of the encoder stack, allowing the decoder to attend to the relevant parts of the input image when generating each token.

In the cross-attention module, the queries are derived from the decoder’s input (the previously generated tokens), while the keys and values are derived from the output of the encoder (the image features). Specifically, the decoder input is projected into a query matrix, and the encoder output is projected into key and value matrices. Mathematically, the cross-attention mechanism can be expressed as:

$$\text{Attention}(Q_{dec}, K_{enc}, V_{enc}) = \text{softmax}\left(\frac{Q_{dec}K_{enc}^\top}{\sqrt{d_k}}\right)V_{enc}, \quad (3)$$

where Q_{dec} is the query matrix derived from the decoder input, K_{enc} and V_{enc} are the key and value matrices derived from the encoder output (\top denotes transposition), and d_k is the dimension of the keys. This mechanism allows each position in the decoder to attend to all positions in the input image, enabling the model to effectively integrate visual and textual information during the generation process.

3 Methodology

In reproducing the results from [Gurgurov and Morshnev \(2024\)](#) we closely followed their methodology, this includes training and fine-tuning on the same data, using the same initial pre-trained Transformer models and GPU optimizations.

Training and evaluation of the Vision Encoder Decoder model for LaTeX images, “Im2Latex”, was done as follows. First, on the implementation side, the code itself uses the same tools (e.g., the `Transformer` library developed by [Huggingface](#)) as the codebase from the authors openly available on [GitHub](#), but was essentially written from scratch using object oriented patterns, which significantly reduces code duplication and improves modularity compared to the original codebase. As an exception, we used the same data processing classes for reproducibility purposes. Additionally, we were working in a single GPU setting and dropped the distributed GPU support as we could not test its implementation easily.

The base Swin Transformer and GPT2 model were obtained from HuggingFace: [microsoft/swin-base](#) and [openai-community/gpt2](#). The full encoder-decoder model is fairly massive with $P = 243, 433, 656$ total parameters. As described in [Gurgurov and Morshnev \(2024\)](#), we combine the base Swin Transformer and GPT-2 model into an encoder-decoder model and training was performed in two steps. We used the same hyperparameters as the authors which are summarized in [Table 1](#), with batch size being the exception due to VRAM limitations (24GBs). The batch size was limited to 12 as opposed to 32.

In the first step, we trained on a cleaned dataset of 550k formula-image pairs ([0leehey0/latex-formulas](#)) that the authors used. The dataset was split in an 80:10:10 ratio, randomly, but using the same seed (42) as the authors. This resulted in 441,872 samples for training, 55,234 for validation, and 55,234 for testing. All images were (resized to) 224×468 .

The second step involved fine-tuning on 1338 handwritten formulas ([linxy/LaTeX_OCR](#)) consisting of 1338 samples (1200 for train, 68 for validation and 70 for test). Like the authors, we employed a LoRA (Low-Rank Adaptation of Large Language Models) with reduction factor $r = 16$, scaling factor $\alpha = 16$ and dropout rate of 0.2¹.

When we obtained our models from the first step ([Matthijs0/im2latex_base](#)) and second step ([Matthijs0/im2latex_finetuned](#)), we evaluated and compared with the model from [Gurgurov and Morshnev \(2024\)](#) on the test sets.

Table 1: Hyperparameters for Training on Printed and Handwritten Formulas.

Hyperparameter	Step 1: Printed Formulas	Step 2: Handwritten Formulas
Batch Size	12	12
Learning Rate	1×10^{-4}	2×10^{-4}
Gradient Clipping	Maximum norm of 1.0	
Epochs	10	40
Evaluation Frequency (Steps)	200	40
Optimizer	AdamW with Linear Scheduler and Warmup Steps	

3.1 Metrics

To evaluate the performance of our Im2Latex model, we followed the same evaluation procedure as [Gurgurov and Morshnev \(2024\)](#). We report the negative log-likelihood (NLL) and the Google BLEU score on the test set.

The NLL, as defined in Equation 2, quantifies the average negative log-probability assigned by the model to the ground truth LaTeX sequence for each image in the test set. The NLL is a non-positive value. Lower NLL values signify superior model performance, indicating higher confidence in predicting the correct sequence.

The Google BLEU score, in contrast to the standard BLEU metric ([Papineni, Roukos, Ward, & Zhu, 2002](#)), is designed to be more suitable for evaluating individual sentences. It is computed by first recording all sub-sequences of 1, 2, 3, or 4 tokens (n-grams) in both the generated and target sequences. Then, it calculates the recall, which is the ratio of the number of matching n-grams to the total number of n-grams in the target (ground truth) sequence. It also computes the precision, defined as the ratio of the number of matching n-grams to the total number of n-grams in the generated sequence. The Google BLEU score is then the minimum of the precision and recall values.

Formally, the Google BLEU score can be expressed as:

$$\text{Google BLEU} = \min(\text{Precision}, \text{Recall}), \quad (4)$$

where:

$$\text{Precision} = \frac{\text{Number of matching n-grams}}{\text{Total number of n-grams in generated sequence}}, \quad (5)$$

$$\text{Recall} = \frac{\text{Number of matching n-grams}}{\text{Total number of n-grams in target sequence}}. \quad (6)$$

¹See [Gurgurov and Morshnev \(2024\)](#) for more details on which modules the LoRA adaptation was applied to.

The Google BLEU score ranges from 0 to 1, where 0 indicates no matching n-grams and 1 indicates a perfect match between the generated and target sequences. It is symmetrical, meaning that the score is the same regardless of whether a sequence is considered the prediction or the reference.

4 Results

The training process for the combined model on the printed formulas dataset required approximately 72 hours using our computational setup. In contrast, the LoRA fine-tuning on the handwritten dataset was completed in roughly 10 minutes, owing to the significantly smaller size of the latter dataset. Figure 2 displays the metrics recorded during the initial training phase, while Figure 3 illustrates the metrics captured during the fine-tuning process on the handwritten dataset.

Table 2 presents the test set results, comparing the performance of our trained models with that of the model developed by Gurgurov and Morshnev (2024). Our base model (Matthijs0/im2latex_base) achieved a Google BLEU score of 0.69 and a loss of 0.09 on the OleeHy0/latex-formulas test set, demonstrating comparable performance to the original authors’ model (DGurgurov/im2latex), which scored 0.67 in Google BLEU and 0.10 in loss on the same dataset. This indicates that our model is generally effective at predicting LaTeX equations from printed images within the distribution of the test set. On the handwritten dataset (linxy/LaTeX_OCR), both models achieved a Google BLEU of 0.03 and a loss of 3.4, indicating poor generalization to handwritten formulas without finetuning.

Our fine-tuned LoRA model (Matthijs0/im2latex_finetuned) achieved a Google BLEU score of 0.63 and a loss of 0.15 on the handwritten test set, demonstrating its effectiveness in recognizing handwritten formulas. However, its performance on the printed test set decreased to a Google BLEU score of 0.55 and a loss of 0.19, suggesting some degree of trade-off between performance on printed and handwritten formulas. As the original authors did not share their fine-tuned model or results, a direct comparison on the handwritten dataset is not possible.

As illustrated in Figures 2 and 3, our training and validation loss curves reveal a slight degree of overfitting in the base model. The training loss reached 0.08, while the validation loss was slightly higher at 0.1. In contrast, our LoRA fine-tuned model did not exhibit this trend, with a training loss of 0.05 and a validation loss of 0.02. This difference may be attributed to the smaller size and potentially simpler nature of the handwritten validation set compared to its training set.

See Appendix A for example LaTeX outputs generated by the models.

Table 2: Test Set Results

Model	OleeHy0/latex-formulas		linxy/LaTeX_OCR	
	Google BLEU	Loss	Google BLEU	Loss
Matthijs0/im2latex_base (ours)	0.69	0.09	0.03	3.46
DGurgurov/im2latex	0.67	0.10	0.03	3.43
Matthijs0/im2latex_finetuned (ours)	0.55	0.19	0.63	0.15

5 Discussion & Conclusion

Our study successfully reproduced the Im2Latex model proposed by Gurgurov and Morshnev (2024), demonstrating comparable performance on a large dataset of printed LaTeX formulas. As shown in Table 2, our base model achieved a Google BLEU score of 0.69 on the printed test set, similar to the original authors’ results (Google BLEU: 0.67). This validates our implementation and confirms the effectiveness of the Im2Latex architecture, combining a Swin Transformer encoder and GPT-2 decoder for the task of image-to-LaTeX conversion on data similar to the training distribution.

As part of our reproduction, we followed the original authors’ methodology by employing LoRA to fine-tune our model on the dataset of handwritten formulas. This resulted in a substantial

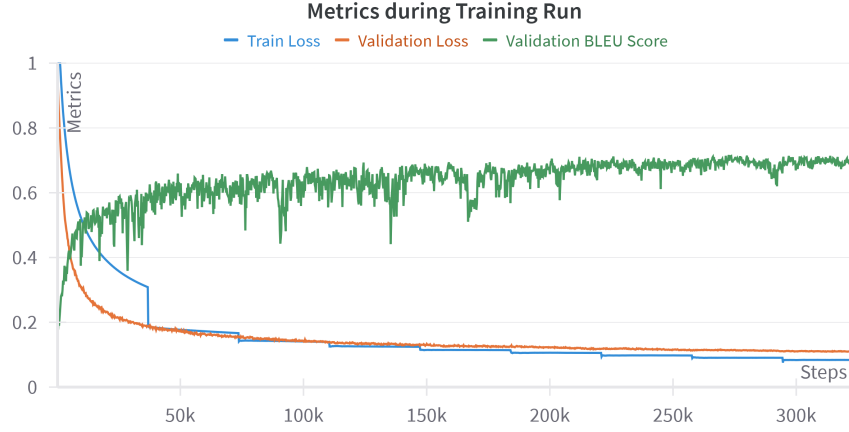


Figure 2: Training and validation loss along with validation BLEU score during training where steps is equal to batch iterations. Our models were trained using a sequential data loader which causes the step-like train loss curve whenever a new pass through the dataset begins. A low degree of overfitting is observed as the validation loss is slightly higher than the training loss.

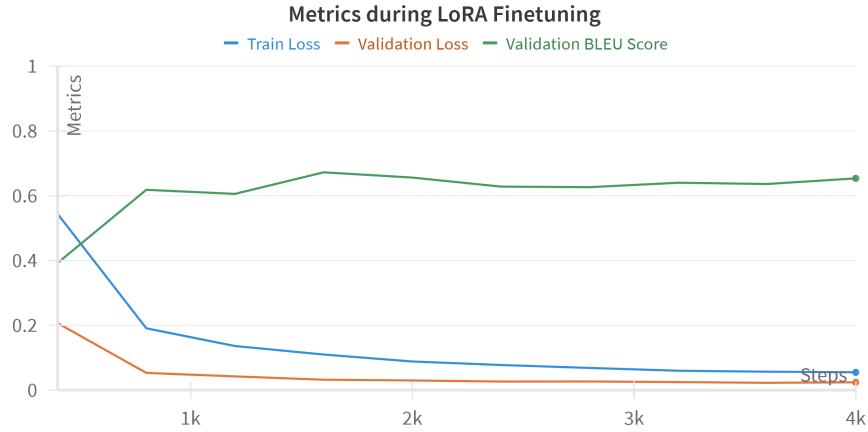


Figure 3: Training and validation loss along with BLEU score during finetuning where steps is equal to batch iterations. In contrast to the regular model, the fine-tuning is complete in a few thousand iterations.

improvement in performance on handwritten formulas (Google BLEU: 0.63), demonstrating the effectiveness of LoRA for adapting the model to a new domain with limited data. However, this fine-tuning led to a slight decrease in performance on the printed formula dataset (Google BLEU: 0.55), suggesting a trade-off between specialization and generalization. While we were unable to compare our fine-tuned model to the original authors' due to the unavailability of their model and results, our findings indicate that LoRA fine-tuning can be a valuable technique for adapting large pre-trained models to specific domains.

Interestingly, our base model exhibited a small degree of overfitting, as evidenced by the slight difference between training and validation loss (0.08 vs. 0.1). This was not observed in the LoRA fine-tuned model, potentially due to the smaller size and simpler nature of the handwritten validation set.

While our results demonstrate the promise of the Im2Latex architecture for image-to-Latex conversion, they also highlight certain limitations, particularly when confronted with out-of-

distribution (OOD) data. When evaluating both our base model and the original authors’ on the handwritten dataset without fine-tuning, performance dropped significantly (Google BLEU: 0.03). Further investigation using screenshots of LaTeX equations (Appendix A) revealed that the base model struggles with inputs that deviate from the characteristics of the training data, such as variations in image quality, font, or background.

However, it is important to note that increasing the scale of both data and model parameters, along with techniques like quantization, could be potential solutions to enhance the model’s robustness and ability to handle OOD data. Larger models trained on more diverse datasets may learn more generalizable features, while quantization, despite potentially introducing minor inaccuracies, allows for the deployment of these larger models with reasonable computational resources. The Im2Latex architecture, leveraging a Swin Transformer encoder and a GPT-2 decoder, already represents a significant step in this direction. The Swin Transformer, with its hierarchical design, is well-suited to capture both global and local features in the input images. Additionally, the GPT-2 decoder has demonstrated strong capabilities in generating coherent and contextually appropriate text.

In conclusion, our study confirms the potential of Transformer-based models for image-to-LaTeX conversion, while also highlighting the need for further research into scaling and generalization. Future work should focus on exploring data augmentation strategies, investigating more robust encoder architectures, and incorporating mechanisms to enforce structural constraints during decoding. Additionally, exploring the benefits of larger models, trained on more diverse data, and optimized with techniques like quantization, will be crucial for advancing the state-of-the-art in this field and handling the complexities of the real-world and out-of-distribution data.

References

- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). *Layer normalization*. Retrieved from <https://arxiv.org/abs/1607.06450>
- Breezedeus. (2024). *Pix2text mfr*. <https://huggingface.co/breezedeus/pix2text-mfr>.
- Deng, Y., Kanervisto, A., Ling, J., & Rush, A. M. (2017, 06–11 Aug). Image-to-markup generation with coarse-to-fine attention. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 980–989). PMLR. Retrieved from <https://proceedings.mlr.press/v70/deng17a.html>
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2021). *An image is worth 16x16 words: Transformers for image recognition at scale*. Retrieved from <https://arxiv.org/abs/2010.11929>
- Garain, U., & Chaudhuri, B. B. (2005). A corpus for ocr research on mathematical expressions. *International Journal of Document Analysis and Recognition (IJDAR)*, 7, 241–259.
- Gurgurov, D., & Morshnev, A. (2024). *Image-to-latex converter for mathematical formulas and text*. Retrieved from <https://arxiv.org/abs/2408.04015>
- Hamad, K., & Kaya, M. (2016). A detailed analysis of optical character recognition technology. *International Journal of Applied Mathematics Electronics and Computers*(Special Issue-1), 244–249.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition*. Retrieved from <https://arxiv.org/abs/1512.03385>
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... Chen, W. (2021). *Lora: Low-rank adaptation of large language models*. Retrieved from <https://arxiv.org/abs/2106.09685>
- Islam, N., Islam, Z., & Noor, N. (2017). A survey on optical character recognition system. *arXiv preprint arXiv:1710.05703*.
- Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s), 1–41.
- Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., ... Wei, F. (2022). *Trocr: Transformer-based optical character recognition with pre-trained models*. Retrieved from <https://arxiv.org/abs/2109.10282>

- Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... Guo, B. (2021). *Swin transformer: Hierarchical vision transformer using shifted windows*. Retrieved from <https://arxiv.org/abs/2103.14030>
- Memon, J., Sami, M., Khan, R. A., & Uddin, M. (2020). Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr). *IEEE access*, 8, 142642–142668.
- Michael, J., Labahn, R., Grüning, T., & Zöllner, J. (2019). Evaluating sequence-to-sequence models for handwritten text recognition. In *2019 international conference on document analysis and recognition (icdar)* (pp. 1286–1293).
- Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics* (p. 311–318). USA: Association for Computational Linguistics. Retrieved from <https://doi.org/10.3115/1073083.1073135> doi: 10.3115/1073083.1073135
- Parthiban, R., Ezhilarasi, R., & Saravanan, D. (2020). Optical character recognition for english handwritten text using recurrent neural network. In *2020 international conference on system, computation, automation and networking (icscan)* (pp. 1–5).
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 9.
- Smith, R. (2007). An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (icdar 2007)* (Vol. 2, pp. 629–633).
- Srivastava, S., Priyadarshini, J., Gopal, S., Gupta, S., & Dayal, H. S. (2019). Optical character recognition on bank cheques using 2d convolution neural network. In *Applications of artificial intelligence techniques in engineering: Sigma 2018, volume 2* (pp. 589–596).
- Trung, H. (2024). *sumen-base*. <https://huggingface.co/hoang-quoc-trung/sumen-base>.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2023). *Attention is all you need*. Retrieved from <https://arxiv.org/abs/1706.03762>
- Wei, T. C., Sheikh, U., & Ab Rahman, A. A.-H. (2018). Improved optical character recognition with deep neural network. In *2018 IEEE 14th international colloquium on signal processing & its applications (cspa)* (pp. 245–249).
- Wick, C., Zöllner, J., & Grüning, T. (2021). Transformer for handwritten text recognition using bidirectional post-decoding. In *International conference on document analysis and recognition* (pp. 112–126).
- Zanibbi, R., & Blostein, D. (2012). Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, 15, 331–357.
- Zhong, X., ShafieiBavani, E., & Jimeno Yepes, A. (2020). Image-based table recognition: data, model, and evaluation. In *European conference on computer vision* (pp. 564–580).

A Test Set Generation Examples

Figure 4 and 5 show the generated LaTeX code of two images from the test split of the synthetic dataset and similarly Figure 6 shows the generated LaTeX for a test sample from the handwritten dataset. To make the comparison easier the LaTeX code was inserted into the report code and included in compilation rather than showing the exact generated tokens. To get a bit more insight, Grad-CAM was applied by extracting feature maps from the final layer of the Swin Transformer encoder, computing gradients of the predicted output with respect to these activations, and generating a heatmap highlighting the most important image regions for the model’s decision. The heatmap was then overlaid on the original image and logged for visualization.

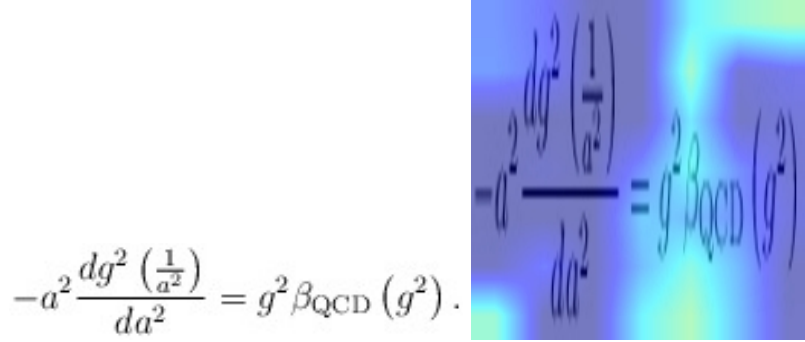


Figure 4: Left: LaTeX image, Right: Grad-CAM output for the Swin Transformer. Generated output of `im2latex_base`: $-a^2 \frac{dg^2(\frac{1}{a^2})}{da^2} = g^2 \beta_{\text{QCD}}(g^2)$. It is worth noting however it is inside a `beginalign*` and a `endgather*` which is not syntactically correct.

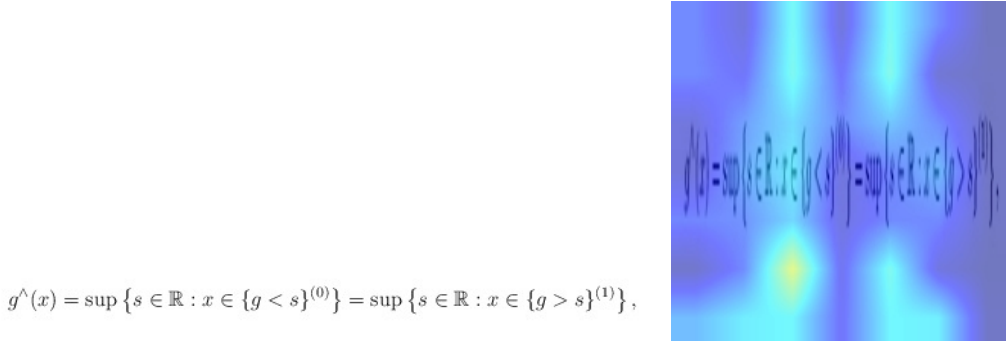


Figure 5: Left: LaTeX image, Right: Grad-CAM output for the Swin Transformer. Generated output of `im2latex_base`: $g^\wedge(x) = \sup \{s \in \mathbb{R} : x \in \{g < s\}^{(0)}\} = \sup \{s \in \mathbb{R} : x \in \{g > s\}^{(1)}\}$., It is worth noting however it is prepended with a `beginalign*` and a `endalign*`, the ‘,’ at the end being syntactically incorrect.

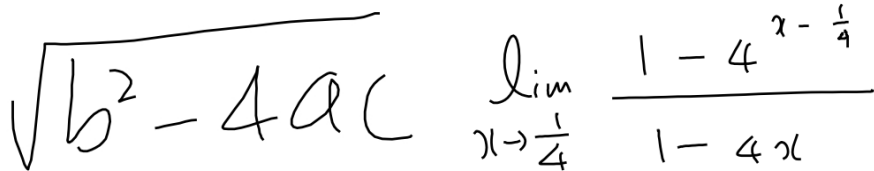
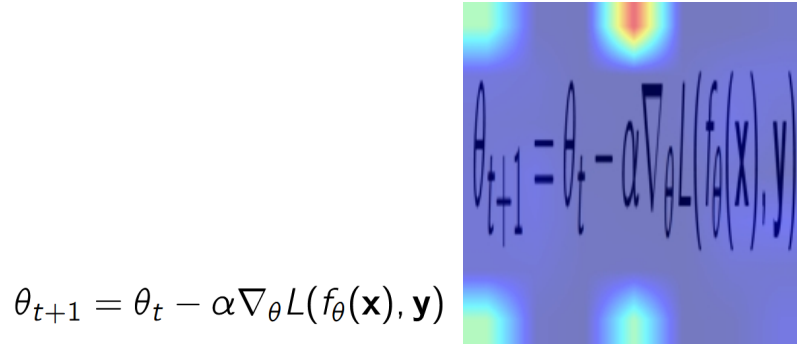


Figure 6: Generated output of `im2latex_finetuned` on left image: $\sqrt{b^2 - 4ac}$. Right image: $\lim_{x \rightarrow \frac{1}{4}} \frac{1}{1 - 4x + 1}$. This dataset does not contain LaTeX equations inside an environment of the form `begin... end...` so they are not generated by the model.

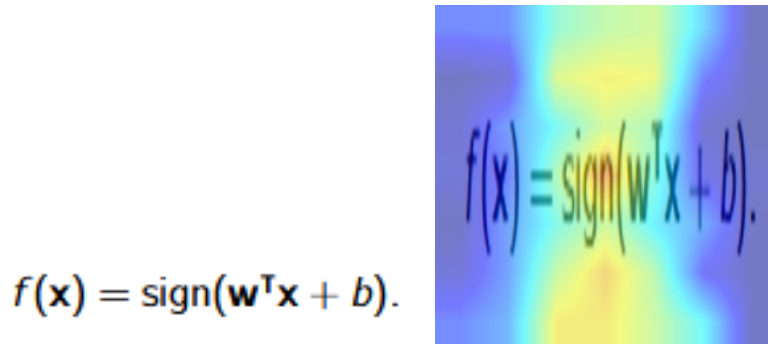
B Screenshot Generation Examples

Figure 7 and 8 show screenshots of LaTeX equations taken by us as opposed to the test set. The screenshots are from equations from the lecture slides of the Deep Learning course.



$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta} L(f_{\theta}(\mathbf{x}), \mathbf{y})$$

Figure 7: Left: LaTeX image, Right: Grad-CAM output for the Swin Transformer. Generated output of `im2latex_base` on left image: $\mathbf{f}_{1\perp 1} = \emptyset_1 - \mathcal{O}\nabla_j \left(\mathbf{f}[\{\mathbf{N}\}] \mid \mathbf{j}(\{\mathbf{N}\}) \right) \mathbf{N} \rangle$. The equation is inside a `beginalign*` and `end align*` block. The latter end statement not being syntactically correct due to the space.



$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b).$$

Figure 8: Left: LaTeX image, Right: Grad-CAM output for the Swin Transformer. Generated output of `im2latex_base` on left image: $\mathbf{f}\{\mathbf{p}\} \neq \text{sign}\{\mathbf{w}\{\mathbf{W}_{\tau\mathbf{x}} + I\}$

The equation is inside a `beginalign*` and a `endgather*` which is not syntactically correct.

Note that the input image has lower dpi than the one in Figure 7.