Handwritten Equations Recognition and Conversion to LaTeX using CNNs, ViTs, and Seq2Seq Transformers

Jatin Kulkarni
Department of Computer Science
Cornell Tech
New York, NY 10044
jk2982@cornell.edu

Walter Stark
Department of Computer Science
Cornell Tech
New York, NY 10044
wzs7@cornell.edu

Abstract

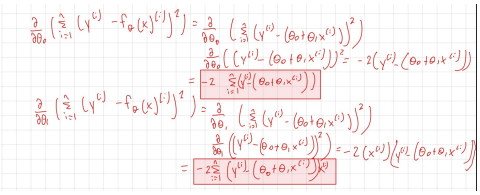
This paper addresses the difficult task of converting handwritten mathematical equations into LaTeX , a process that is traditionally tedious and error-prone. This paper proposes a novel modular multi-stage pipeline that utilizes Convolutional Neural Networks (CNNs), Vision Transformers (ViTs), and Seq2Seq Transformer networks to facilitate this conversion. By combining Canny edge detection for symbol isolation, CNNs and ViTs for symbol recognition, and a T5 transformer for LaTeX generation, the approach aims to overcome limitations of traditional optical character recognition (OCR) systems. The results demonstrate promising performance, with the ViT-Base model achieving a symbol recognition accuracy of 72.87% and the Seq2Seq transformer generating LaTeX equations with an Exact Match Rate of 66.19% and a BLEU score of 0.8428. The modular pipeline represents a step forward in automating the transformation of handwritten mathematical notation into LaTeX code.

1 Introduction

LaTeX is an indispensable tool for creating professional documents, offering uniform formatting for text, images, tables, and equations. However, its steep learning curve often discourages new users, particularly when it comes to creating complex mathematical equations. Even seasoned LaTeX users can find the process of converting handwritten equations into precise LaTeX syntax tedious and prone to error. This challenge has driven interest in automating the conversion of handwritten equations into LaTeX , promising both efficiency and accessibility.

This project introduces a novel method that combines the strengths of Convolutional Neural Networks (CNNs), pre-trained Vision Transformers (ViTs), and Seq2Seq Transformer networks. This architecture is specifically designed to excel at symbol recognition and sequence modeling, enabling precise transcription of handwritten equations into LaTeX . By leveraging the feature extraction power of CNNs and ViTs to determine specific symbols, and the sequence learning capabilities of encoder-decoder transformers, this approach aims to set a new standard for accuracy and reliability in automated LaTeX generation.

An illustrative example is shown in Figure 1, demonstrating how handwritten input can be transformed into LaTeX syntax:



(a) Example handwritten text.

```
{\color{red} \[
    \frac{\partial}{\partial \theta_0} \left( \sum_{i=1}^{n} \left( y^{(i)} - f_\theta(x)^{(i)} \right)^2 \right)
    = \frac{\partial}{\partial \theta_0} \left( \sum_{i=1}^{n} \left( y^{(i)} - \left( \theta_0 + \theta_1 x^{(i)}
    \right) \right)^2 \right)
\]
\[\]
\[
    = -2 \sum_{i=1}^{n} \left( y^{(i)} - \left( \theta_0 + \theta_1 x^{(i)} \right) \right)
\]
\[\]
\[
    \frac{\partial}{\partial \theta_1} \left( \sum_{i=1}^{n} \left( y^{(i)} - f_\theta(x)^{(i)} \right)^2 \right)
    = \frac{\partial}{\partial \theta_1} \left( \sum_{i=1}^{n} \left( y^{(i)} - \left( \theta_0 + \theta_1 x^{(i)}
    \right) \right)^2 \right)
\]
\[\]
\[
    = -2 \sum_{i=1}^{n} \left( x^{(i)} \left( y^{(i)} - \left( \theta_0 + \theta_1 x^{(i)} \right) \right) \right)
\]}
```

(b) Example LaTeX code.

Figure 1: Comparison of handwritten text and corresponding LaTeX code.

Preprint. Under review.

The resulting LaTeX output for the given example is as follows:

$$\frac{\partial}{\partial\theta_0}\left(\sum_{i=1}^{n}\left(y^{(i)}-f_\theta(x)^{(i)}\right)^2\right)=\frac{\partial}{\partial\theta_0}\left(\sum_{i=1}^{n}\left(y^{(i)}-\left(\theta_0+\theta_1x^{(i)}\right)\right)^2\right)$$

$$=-2\sum_{i=1}^{n}\left(y^{(i)}-\left(\theta_0+\theta_1x^{(i)}\right)\right)$$

$$\frac{\partial}{\partial\theta_1}\left(\sum_{i=1}^{n}\left(y^{(i)}-f_\theta(x)^{(i)}\right)^2\right)=\frac{\partial}{\partial\theta_1}\left(\sum_{i=1}^{n}\left(y^{(i)}-\left(\theta_0+\theta_1x^{(i)}\right)\right)^2\right)$$

$$=-2\sum_{i=1}^{n}\left(x^{(i)}\left(y^{(i)}-\left(\theta_0+\theta_1x^{(i)}\right)\right)\right)$$

2 Background

The task of converting handwritten mathematical equations into LaTeX syntax intersects with several challenging domains, including optical character recognition (OCR). OCR systems are designed to recognize and digitize text from images, but extending this capability to mathematical symbols introduces unique difficulties [1, 2]. Mathematical notation involves a vast array of symbols, intricate structures (e.g., fractions, subscripts, and superscripts), and two-dimensional spatial relationships that are not present in standard text. These complexities make accurate recognition and contextual interpretation a significant challenge in OCR for mathematical content [3].

2.1 Key Features of CNNs, ViTs, and LSTMs

To address these challenges, this project leverages advanced machine learning architectures: Convolutional Neural Networks (CNNs), Vision Transformers (ViTs), and Encoder-Decoder Transformers networks. Each of these components brings distinct advantages:

- CNNs are adept at identifying visual patterns in images and are widely used in object detection and recognition tasks. By extracting hierarchical features such as edges, textures, and complex shapes, CNNs are critical for symbol detection [4].
- ViTs offer a complementary capability by applying self-attention mechanisms to image data. Unlike CNNs, ViTs process images as a sequence of patches, enabling them to capture long-range dependencies and contextual relationships between different regions of an image [5].
- Encoder-Decoder Networks (Seq2Seq Models) are effective for tasks that involve transforming one sequence into another. The encoder processes the input, capturing its features, while the decoder generates the output sequence step by step. In handwritten equation recognition, these networks enable converting letters and their positions in an image to LaTeX by maintaining alignment and adhering to LaTeX syntax [6, 7].

2.2 Prior Work in Handwritten Equation Recognition and LaTeX Conversion

Efforts to automate handwritten equation recognition have primarily focused on OCR-based systems enhanced by machine learning techniques. Traditional approaches relied heavily on feature engineering and rule-based systems, which struggled with the variability in handwriting styles and the inherent complexity of mathematical notation [1, 2].

More recent advancements have introduced deep learning models to improve accuracy. For example, CNNs have been applied to classify individual symbols, while recurrent neural networks (RNNs), including LSTMs, have been used to sequence these symbols into coherent representations [8]. However, these models often face limitations in handling equations with complex layouts or ambiguous handwriting. Additionally, early methods lacked the scalability to incorporate newer architectures, such as ViTs, which have shown promise in other image recognition tasks [3].

Many related works have delved into handwritten equation recognition, but focused only on individual symbol recognition not 2D positional relationships between characters and their subsequent conversion to LaTeX . One such example implemented techniques like Naive Bayes, SVM, random forests, and CNNs to achieve over 90 percent classification accuracy [3, 9]. Some researchers proposed methods of using ensemble models, de-noising input images, and active learning strategies for using OCR to convert handwritten symbols to LaTeX , but don't

offer experiments to validate these proposed methods [10]. Other works explore positional and syntactical requirements with OCR for chemistry-oriented datasets, illustrating the possibility of using these deep learning techniques for equations where positions and syntax are crucial [11].

The most relevant work to date addresses the entire pipeline from images to latex. This paper proposes a neural encoder-decoder model that uses a coarse-to-fine attention mechanism to generate LaTeX markup from images of mathematical expressions [12]. On the IM2LATEX-100K dataset, the model achieved a 77.46% exact match accuracy and 38.7% on CROHME 2014, comparable to top competition systems. The approach presented in this work aims to outperform the previously mentioned model's performance with a custom approach.

3 Method

3.1 Pipeline Overview

The methodology employs a multi-step approach to convert handwritten equations into their corresponding LaTeX representations. This pipeline consists of three primary stages: symbol detection and isolation, symbol recognition and LaTeX mapping, and equation generation. Each stage is deigned to handle specific aspects of the problem, ensuring a seamless transformation from raw handwritten input to syntactically correct LaTeX code. The following sections describe these stages in detail, including datasets used, preprocessing, models, training procedures, and evaluation metrics.

3.1.1 Symbol Detection and Isolation

In the first stage, handwritten symbols are detected and isolated from the input image for further analysis. Unlike stages that employ deep learning models, this step utilizes a preprocessing pipeline. The input image is converted to gray-scale and median filtering is applied to reduce noise. The Canny edge detection algorithm identifies edges in the image, and morphological dilation connects nearby elements, simplifying the isolation of individual symbols [13]. Contours are detected around connected components, and bounding boxes are extracted to isolate symbols. These bounding boxes are spatially sorted from left-to-right and top-to-bottom. Symbols within the bounding boxes are cropped, resized to 45x45 pixels, and normalized to a range of $[0, 1]$. This process ensures consistent input dimensions for subsequent stages. The isolated symbols, along with their metadata, such as bounding box coordinates, are stored as arrays for further processing. Intermediate processing steps can also be saved for debugging or visualization.

3.1.2 Symbol Recognition and LaTeX Mapping

The second stage involves recognizing isolated symbols and mapping them to their corresponding LaTeX representations. We predict the LaTeX representation out of 86 possible options for each image. Different models are experimented to identify the best-performing architecture. CNNs are employed for their ability to extract hierarchical features such as edges and shapes, while Vision Transformers (ViTs) such as `vit_base_patch16_224` and `vit_small_patch16_224` are tested for their capacity to capture global dependencies and contextual relationships within the symbols [5].

For training, the models are fine-tuned using CROHME and IM2LATEX-100K datasets, supplemented with synthetic data to improve generalization [14, 12]. Random rotations are applied during the symbol recognition stage to augment the training data. These rotations, ranging within $\pm 15^{\circ}$, simulate variations in handwriting styles and improve the model's robustness in recognizing symbols under different orientations. Training is performed using the Adam optimizer, which is chosen for its adaptive learning rate capabilities, to ensure effective convergence [15]. Cross-entropy loss is used for classification, supplemented by a custom mapping loss to ensure accurate alignment with LaTeX syntax. Hyperparameter experiments involve testing learning rates of $1e-5$, $1e-4$, and $1e-3$, as well as addressing class imbalances in the datasets. The best-performing model is selected based on classification accuracy, precision, and recall.

3.1.3 Equation Generation

In the final stage, recognized symbols and their LaTeX mappings are sequenced to generate syntactically correct LaTeX equations. This task is approached using a T5 transformer model, which has demonstrated strong performance in sequence-to-sequence tasks [7]. The output from the symbol recognition stage is fed into the model along with the bounding box coordinates from the symbol detection stage, and its predictions are compared with the ground truth LaTeX equations from the CROHME and IM2LATEX-100K datasets [14, 12]. Training is performed using the Adam optimizer, which is chosen for its adaptive learning rate capabilities, to ensure effective convergence [15]. Hyperparameter experiments test learning rates of $1e-5$, $1e-4$, and $1e-3$ to optimize performance. Evaluation metrics include BLEU score, Exact Match Rate (EMR), and Translation Error Rate (TER). These metrics capture both syntactic and semantic correctness, ensuring high-quality LaTeX equation generation.

4 Experimental Analysis

4.1 Experimental Setup

The experiments conducted aim to evaluate the performance of different models and configurations across various stages of the pipeline. The evaluation focuses on training and validation metrics, including accuracy, loss, and equation-level correctness. All models were implemented using PyTorch and trained on systems equipped with NVIDIA RTX 3090 GPUs and an M3 Pro MacBook Pro. The datasets used for training and evaluation include CROHME, IM2LATEX-100K, and the Transformer and CNN/ViT datasets, as described in the methodology. For the Transformer dataset, a total of 921 samples were used, divided into 70% for training, 15% for testing, and 15% for validation. For the CNN/ViT models, a larger dataset of 9600 samples was utilized, also split into 70% for training, 15% for testing, and 15% for validation.

4.2 Symbol Recognition Experiments

4.2.1 Model Comparisons

To determine the best-performing model for symbol recognition, experiments were conducted using CNNs and Vision Transformers (ViTs), including ViT-Large and ViT-Small. The models were evaluated based on training loss, validation loss, training accuracy, validation accuracy, precision, recall, and F1 score. Figures 1 and 2 show the comparative performance of these models.

The results indicate that ViT-Base achieved the highest validation accuracy at 72.87% with a precision of 73.49%, recall of 72.87%, and an F1 score of 73.04%. ViT-Small (with a learning rate of 1e-5) followed with an accuracy of 69.99%, precision of 71.08%, recall of 69.99%, and F1 score of 70.06%. CNNs showed consistent, though lower, performance with an accuracy of 49.43%, precision of 48.66%, recall of 49.43%, and an F1 score of 46.75%. While CNNs demonstrated simplicity and effectiveness for numeric characters, they struggled with complex or less common symbols. Among the ViT models, ViT-Base emerged as the top-performing model at a higher computational cost, whereas ViT-Small provided a balance of efficiency and accuracy.

4.2.2 Learning Rate Experiments

Subsequent experiments focused on the ViT-Small model to determine the impact of different learning rates. Models were trained with learning rates of 1e-5, 1e-4, and 1e-3. The results show that a learning rate of 1e-5 provided the most stable and optimal performance, achieving a validation accuracy of 69.99%, precision of 71.08%, recall of 69.99%, and F1 score of 70.06%. A learning rate of 1e-4 resulted in slightly lower performance with an accuracy of 62.59%, precision of 71.38%, recall of 62.59%, and F1 score of 66.12%. A learning rate of 1e-3 performed poorly, achieving only 5.45% accuracy, 13.29% precision, 5.45% recall, and an F1 score of 7.51%. These findings highlight the critical role of selecting an appropriate learning rate for optimal model convergence and performance.

4.2.3 Data Balancing

Further experiments were conducted to evaluate the impact of class balancing on the symbol recognition task. Training with balanced datasets slightly decreased the validation accuracy of the ViT-Small model, from 69.99% without class balancing to 67.11% with class balancing. This decrease highlights potential trade-offs between addressing dataset imbalances and overall model performance. Figure 3 illustrates the training and validation metrics with and without class balancing for the ViT-Small model trained with a learning rate of $1e-5$ (refer to Appendix A.5).

4.3 Seq2Seq Transformer Performance

Experiments for the equation generation stage evaluated the performance of a Seq2Seq transformer model. The evaluation focused on metrics such as Exact Match Rate (EMR), BLEU score, and Translation Error Rate (TER) to assess the syntactic and semantic correctness of the generated LaTeX equations. Table 1 summarizes the results of different learning rates tested:

Learning Rate	EMR ↑	BLEU ↑	TER ↓
5×10^{-4}	0.6619	0.8428	2.7914
1×10^{-3}	0.6547	0.8346	3.0576
2×10^{-3}	0.5540	0.7858	3.8273

Table 1: Comparison of Different Learning Rates for Seq2Seq Transformer on Test Set

The results indicate that a learning rate of 5×10^{-4} yielded the best overall performance, achieving the highest EMR and BLEU score while minimizing TER. This highlights the importance of tuning the learning rate to balance model convergence and output quality. (refer to Appendix B.1)

5 Discussion

The models tested reveal trade-offs between accuracy, complexity, and computational efficiency. The CNN model, while simple and effective for frequently occurring symbols, struggled with limited accuracy and complex patterns. The ViT Large model achieved the highest accuracy and highest F1-score but was computationally expensive and prone to overconfidence in rare symbols. The ViT Small model offered a balanced solution, providing faster training and inference times while maintaining solid accuracy, albeit with some limitations in handling complex or underrepresented symbols.

ViT-Small was selected for its close accuracy to ViT-Large and significantly faster training and inference, addressing computational constraints. This efficiency allowed for extended experiments with data augmentation and hyperparameter tuning, establishing it as a practical choice.

In equation generation, the Seq2Seq transformer model achieved the best results with a learning rate of 5×10^{-4}, balancing high Exact Match Rate (EMR), BLEU score, and low Translation Error Rate (TER). The T5-Small configuration proved optimal, combining computational efficiency with superior syntactic and semantic accuracy. The BLEU score indicates the model translated the text relatively well at 0.8428 which is typically considered as a medium to good translation [16]. These results highlight the importance of tailored hyperparameter tuning.

Each model's strengths and weaknesses suggest avenues for improvement. The CNN model's limitations point to a need for augmentation and additional training data. The ViT Large model's misclassification patterns indicate overfitting to common classes. The ViT Small model represents a promising balance but may benefit from targeted data augmentation and further tuning. Future work could involve hybrid or ensemble approaches to leverage the strengths of all models, enhancing pipeline robustness for handwritten equation recognition and LaTeX conversion (refer to Appendix A.6).

6 Error Analysis

A per-symbol accuracy and confidence analysis was conducted for both models. Commonly misidentified symbols included operators such as $\leq$, parentheses, and some lowercase Latin characters. The CNN model performed better with numeric characters, whereas the ViT model exhibited a high variance in confidence across different classes, likely indicating overfitting to a subset of the symbols. Additionally, some classes had no samples or very low confidence, impacting the model's overall generalization capability. In the equation generation stage, token limit restrictions were observed as a key issue. The input, which included box coordinates of the predicted LaTeX characters from the previous pipeline stage, sometimes exceeded the model's capacity, causing the generated LaTeX expressions to be truncated, particularly in complex equations. This limitation highlights the need for optimizing input representations or exploring models with higher token limits.

7 Conclusion

This study advances the field of handwritten equation recognition by addressing the limitations of traditional OCR-based systems and early deep learning approaches, which struggled with the handwriting variability, positional relations, and complex layouts. The proposed pipeline integrates Vision Transformers (ViTs) and Seq2Seq Transformer models to deliver improvements in scalability and accuracy, while its modular design allows for the testing of divers combinations of classifier and Seq2Seq models, enhancing flexibility and adaptability.

Unlike prior work that often focused on isolated tasks, this study employs a holistic, pipeline-based approach that improves symbol recognition, resolves 2D positional relationships, and ensure syntactic coherence in LaTeX generation. While results are not directly comparable to earlier efforts due to the absence of a fully completed pipeline, the modularity of the approach represents a key advantage, enabling experimentation with a broader range of architectures and configurations.

Future work will focus on addressing key issues that contributed to errors, such as output length limitations, which accounted for approximately 50% of observed inaccuracies, as well as improving rare symbol recognition to enhance model robustness.

References

[1] Jamshed Memon, Maira Sami, and Rizwan Ahmed Khan. Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr), 2020.

[2] Thi Tuyet Hai Nguyen, Adam Jatowt, Mickael Coustaty, and Antoine Doucet. Survey of post-ocr processing approaches. *ACM Comput. Surv.*, 54(6), July 2021.

[3] Parth Hasmukh Jain, Vivek Kumar, Jim Samuel, Sushmita Singh, Abhinay Mannepalli, and Richard Anderson. Artificially intelligent readers: An adaptive framework for original handwritten numerical digits recognition with ocr methods. *Information*, 14(6), 2023.

[4] Sakshi Indolia, Anil Kumar Goswami, S.P. Mishra, and Pooja Asopa. Conceptual understanding of convolutional neural network- a deep learning approach. *Procedia Computer Science*, 132:679–688, 2018. International Conference on Computational Intelligence and Data Science.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[7] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[8] Asma Naseer and Kashif Zafar. Comparative analysis of raw images and meta feature based urdu ocr using cnn and lstm. *International Journal of Advanced Computer Science and Applications*, 9, 2018.

[9] Saravanakumar R Nandhinishri S.N Navaneetha Krishnan M, Swetha G and Arthi R. Comparative analysis of convolutional neural network and character recognition techniques for handwritten mathematical equation solver. *Journal of Survey in Fisheries Sciences*, 2023.

[10] Everistus Zeluwa Orji, Ali Haydar, İbrahim Erşan, and Othmar Othmar Mwambe. Advancing ocr accuracy in image-to-latex conversion—a critical and creative exploration. *Applied Sciences*, 13(22), 2023.

[11] Nan Zhang, Connor Heaton, Sean Timothy Okonsky, Prasenjit Mitra, and Hilal Ezgi Toraman. Peace: A chemistry-oriented dataset for optical character recognition on scientific documents, 2024.

[12] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. Image-to-markup generation with coarse-to-fine attention, 2017.

[13] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[14] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, and U. Garain. Icfhr2016 crohme: Competition on recognition of online handwritten mathematical expressions. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 607–612, 2016.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[16] Ehud Reiter. A structured review of the validity of bleu. *Computational Linguistics*, 44(3):393–401, 09 2018.

A Loss Graphs

A.1 CNN Loss Graph

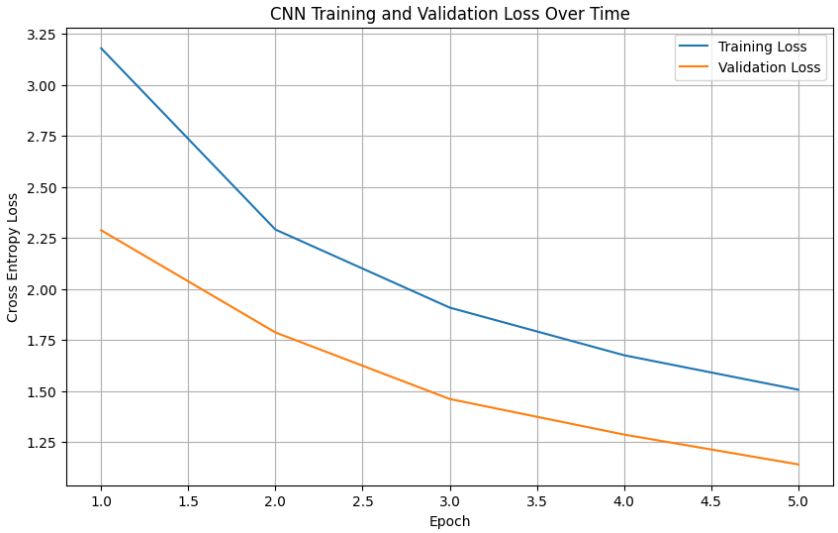


Figure 2: Loss Graph for the CNN Model

A.2 ViT Base Loss Graph

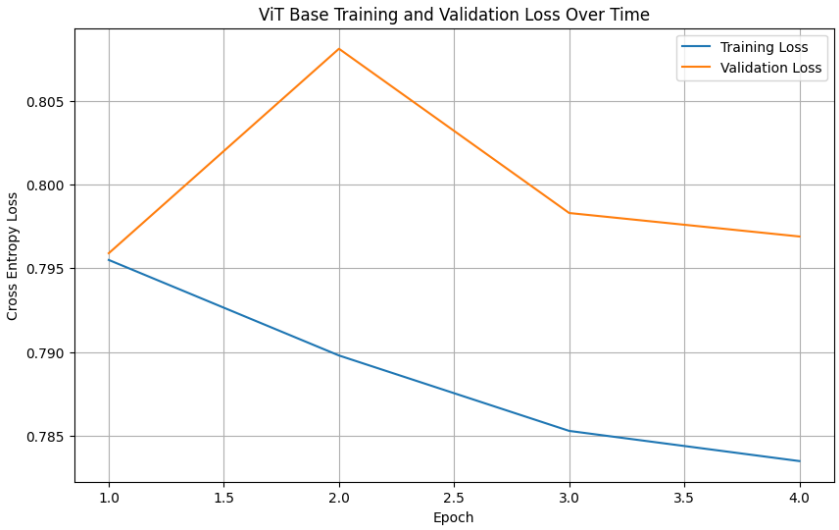


Figure 3: Loss Graph for the ViT Model (`vit_base_patch16_224`)

A.3 ViT Small Loss Graph

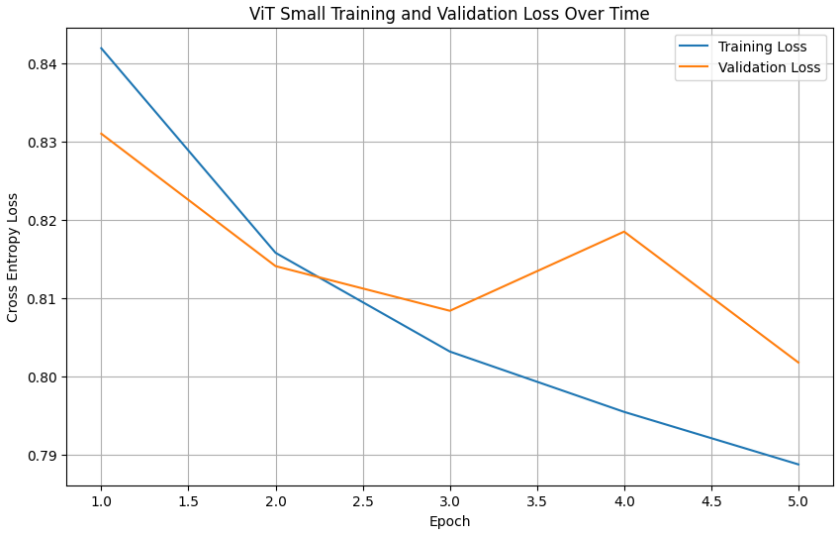


Figure 4: Loss Graph for the Small ViT Model (`vit_small_patch16_224`)

A.4 Learning Rate Hyperparameter Tuning Graph

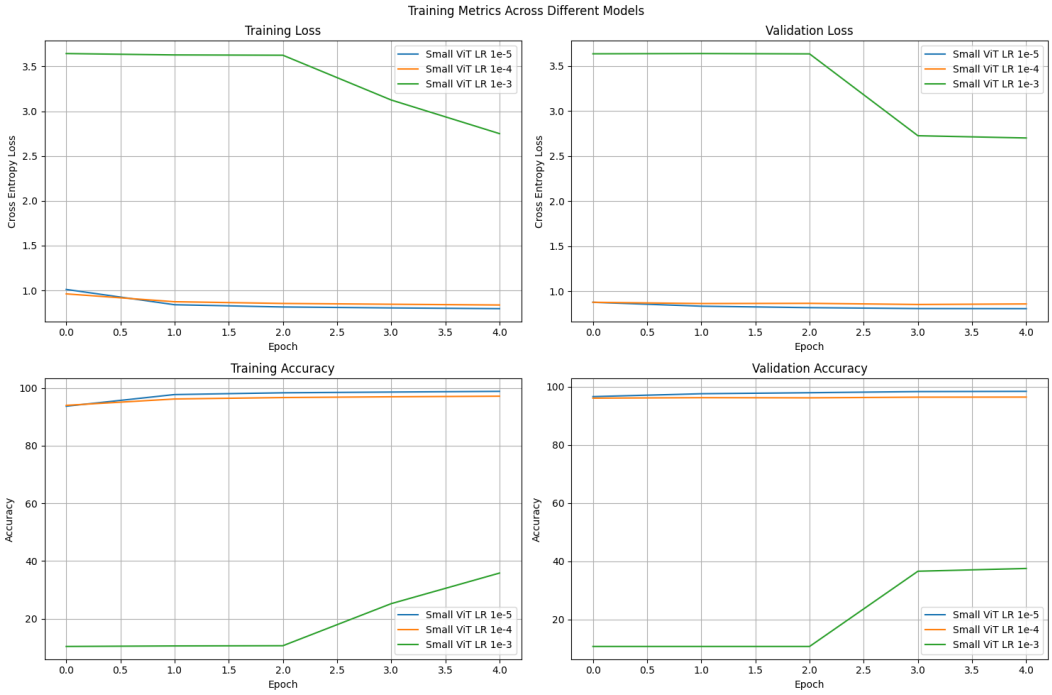


Figure 5: Loss Graph for finetuning learning rate for Small ViT Model (`vit_small_patch16_224`)

A.5 Learning Rate Class Balancing Tuning Graph

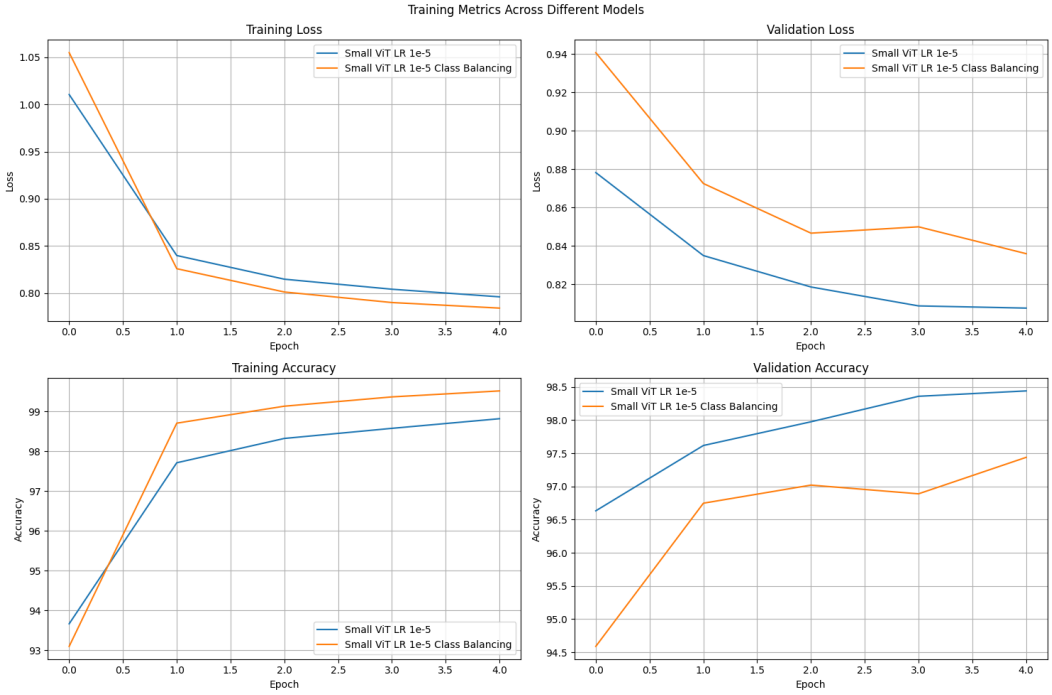


Figure 6: Loss Graph for using Class Balancing for Small ViT Model (`vit_small_patch16_224`)

A.6 Precision, Recall, and F1 Scores for Models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
CNN Model	49.43	48.66	49.43	46.75
ViT Base Model	72.87	73.49	72.87	73.04
ViT Small Model LR 1e-5	69.99	71.08	69.99	70.06
ViT Small Model LR 1e-4	62.59	71.38	62.59	66.12
ViT Small Model LR 1e-3	5.45	13.29	5.45	7.51
ViT Small Model Class Balancing	67.11	72.52	67.11	69.07

Table 2: Performance metrics for various symbol classification models.

B Transformer Training

B.1 Learning Rate Transformer Tuning Graph

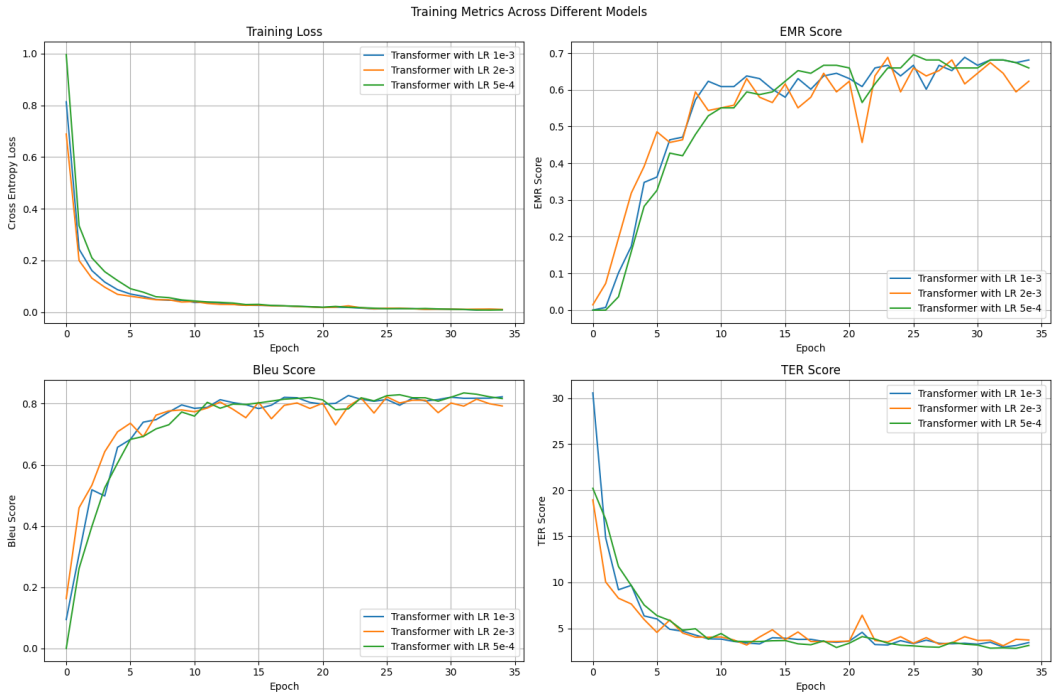


Figure 7: Training Metrics for using Learning Rate for Transformer Model