CSC490 Final Report

Babur Nawyan
University of Toronto
babur.nawyan@mail.utoronto.ca

Sabrina Hirani
University of Toronto
sabrina.hirani@mail.utoronto.ca

Srisaran Chandramouli
University of Toronto
srisaran.chandramouli@utoronto.ca

Abstract

Handwritten Mathematical Expression Recognition (HMER) has extensive applications in digitization of education and office automation. Recently, multi-task learning models with encoder-decoder architectures that aim to predict both the LaTeX sequence and the hierarchical relationships between symbols have achieved state-of-the-art performance [1, 2]. Nevertheless, HMER remains challenging because of the inherent variability in handwriting styles and the complex spatial and hierarchical relationships among LaTeX symbols. In this paper, we evaluate three distinct approaches based on the TAMER model [2]: (i) a multi-task learning model that jointly optimizes LaTeX sequence prediction, parent-child relationship prediction, and position identifier prediction; (ii) an approach integrating self-critical sequence training (SCST) to directly optimize the evaluation metric while tackling exposure bias; and (iii) a method introducing a gradient-based adaptive loss function alongside a novel Longest Sequence Metric (LSM). Despite their theoretical alignment with HMER's challenges, our experiments conducted on the CROHME benchmark dataset show no significant recognition accuracy gains over baseline models such as PosFormer [1] and TAMER [2]. Further analysis reveals critical limitations of each approach: multi-task learning with two different LaTeX structure prediction task does not force the model to learn complementary information, SCST sees diminishing benefits as training progresses, and adaptive loss fails to generalize due to over-prioritization of one loss term. Moreover, our newly proposed LSM correlates with other standard metrics such as ExpRate, ≤ 1, and ≤ 2 which are variations of ExpRate where we tolerate 1 and 2 symbol errors respectively. We discuss the implications of these findings to guide future work on robust HMER methods.

Code Repository - https://github.com/sabrinahirani/CSC490-TAMER

1 Introduction

Handwritten Mathematical Expression Recognition (HMER) aims to convert handwritten mathematical expressions, often available in the form of raw bitmaps, scan images, or pen trajectories, into machine-readable representations like LaTeX scripts. Its significance spans various practical scenarios, including automated grading in education, digital archiving of scientific manuscripts, etc. Because mathematical expressions generally follow a two-dimensional layout with symbols (e.g., summations, fractions, integrals, subscripts, superscripts) organized in hierarchical structures, HMER involves a more complicated pattern recognition task than typical optical character recognition (OCR) tasks.

Most existing HMER models can be categorized into two primary families: sequence-based methods and tree-structured methods. **Sequence-based** methods approach HMER as an image-to-sequence translation problem, directly predicting the LaTeX sequence in an autoregressive manner [3, 4, 5]. By contrast, **tree-structured** methods which predict relationships such as parent-child or upper-lower symbol dependencies, often guaranteeing syntactically valid structures [6, 7]. In response to the limitations of purely sequence-based or purely tree-structured approaches, several **hybrid** or **multi-task models** have recently emerged [1, 2]. These methods improve bracket matching and nested expression handling but retain challenges tied to sequence decoding, such as exposure bias and rigid loss balancing.

Despite progress in models for HMER, inherent challenges persist. Multi-task learning, which combines LaTeX sequence prediction task and LaTeX structure prediction task [1, 2], falters with diverse handwriting styles, as curated training date inadequately reflects real-world variability. Autoregressive decoding suffers from exposure bias, where training- inference mismatches propagate cascading errors. In this paper, we present three novel extensions of the TAMER model [2] to evaluate how well they tackle these challenges:

Multi-task extension. We jointly optimize an encoder-decoder for LaTeX sequence prediction, parent-child relationship prediction, and position identifier estimation. This approach is motivated by the hypothesis that additional structural cues—especially from symbolic positions—could enhance bracket matching, reduce symbol-level errors, and better handle nested expressions.

Self-critical sequence training (SCST). Self-Critical Sequence Training (SCST) [8] is a reinforcement learning-based approach that addresses limitations in training sequence generation models. It has successfully been applied to transformer-based decoder models to (1) directly optimize task-specific evaluation metrics and (2) mitigate exposure bias. Exposure bias occurs when the model is trained using previous ground truth tokens but must rely on its own previous predictions during inference, creating a mismatch that can cause cascading errors.

SCST addresses this issue by using a self-critical reward signal. During training, the model generates two sequences: a baseline sequence generated by applying beam search and a sampled sequence generated by stochastic sampling. A reward based on an evaluation metric is computed for both sequences by comparing the generated sequence with the ground truth sequence. The model is then trained to improve the sampled reward relative to the baseline reward. This approach encourages the model to generate sequences which outperform its own baseline predictions, leading to improved generalization and performance.

Adaptive loss with a longest sequence metric (LSM). We introduce a gradient-based adaptive loss that dynamically adjusts the relative weight of sequence-level and structure-level objectives based on each loss's gradient magnitude. Additionally, we introduce the Longest Sequence Metric (LSM), which captures the length of the longest subsequence that matches ground truth, normalized by the prediction length. Our interest lies in diagnosing partial matches—particularly in expressions with partially correct substructures—and refining the training signals in real time through the adaptive loss.

Surprisingly, thorough evaluations on the CROHME datasets show no consistent improvements over baseline models such as PosFormer [1] and TAMER [2], even though each of the proposed strategies addresses a known limitation of existing HMER solutions. Our deeper investigation reveals that multi-task learning with two LaTeX structure prediction taks can create competing gradients for delicate symbols (especially those that appear in both numerator and subscript), that SCST saw diminishing benefits as training progresses, and that the adaptive loss can over-prioritize one loss term leading to suboptimal results. Lastly, the LSM metric reveals that partial sequential matches correlates with full sequential match metrics such as ExpRate and its variations.

2 Background and related work

Sequence-based decoding. In early efforts, HMER was frequently treated as an extended optical character recognition (OCR) task, wherein individual symbols were recognized by handcrafted or neural-based classification frameworks, followed by structural analysis. With the emergence of

modern sequence-to-sequence architectures, WAP ("Watch, Attend, and Parse") introduced attention-based RNN decoders that directly translate image features into LaTeX tokens [4]. Subsequent works replaced the RNN encoder with deeper CNNs like DenseNet, and introduced attention refinement strategies to better address coverage issues—examples include DWAP, ABM, and CAN [5, 9, 10].

Tree-based decoding. A different line of work posits that the hierarchical structure in mathematical expressions must be modeled explicitly during decoding. DenseWAP-TD and its successor TDv2 [6] employ two-dimensional tree decoders that generate parent-child relationships, ensuring syntactic integrity. SAN ("Syntax-Aware Network") constrains the decoding process by grammar cues, guaranteeing correctness in bracket matching and fraction structures [7]. Although these tree-based models offer strong structural consistency, they can struggle with large, diverse datasets and occasionally lack the general ease of integration that comes from purely sequence-based methods.

Transformer-based approaches soon followed. BTTR [11] utilized a bidirectional decoding strategy to mitigate the left-to-right decoding bias, and CoMER [3] refined attention by integrating a coverage mechanism. Most recently, hybrid or multi-task variants add symbolic structure awareness within an otherwise sequence-based pipeline, aiming to robustly handle brackets, nested fractions, and deeply stacked superscripts.

Hybrid and Multi-task methods. Because purely sequence- or tree-based approaches each have strengths and limitations, a new wave of methods targets the integration of structural and sequential prediction. PosFormer frames the mathematical expression as a "position forest" and jointly optimizes expression recognition and position recognition in a sequence-based framework, greatly improving bracket matching for more intricate expressions [1]. TAMER (Tree-Aware Transformer) likewise combines tree relationships into a Transformer setup, jointly optimizing the tree structure alongside the LaTeX sequence, and introduces a specialized scoring mechanism to penalize syntactically invalid decodes during beam search [2].

Studies have also extended these ideas to multi-task learning settings, incorporating auxiliary tasks such as symbolic counting or syntactic verification. While these approaches occasionally lead to improved recognition robustness, they also raise difficulties in balancing multiple objectives, especially in the presence of delicate tokens or complex symbol interactions.

In this paper, we focus on three specific TAMER-based strategies, each targeting a particular shortcoming of existing multi-task or high-level structural decoders: (1) extending TAMER to include LaTeX sequence, parent-child, and position identifier tasks; (2) reducing exposure bias via self-critical sequence training; (3) laying down a Contextual Dependency Model for graph-based symbolic reasoning. Experiments in Section (refer to your experiment section) demonstrate that, despite robust theoretical motivations, these methods struggle to outperform the strong baselines of PosFormer and TAMER on the CROHME benchmark [12, 13, 14].

Self-Critical Sequence Training (SCST) SCST is a variation of the reinforcement learning algorithm REINFORCE that introduces a self-critical baseline to reduce variance[8]:

Given a sequence $Y = (y_1, y_2, ..., y_T)$ sampled from the model distribution $p_\theta(Y)$ with a task-specific reward function $r(Y)$, we seek to optimize the objective:

$$L_{\text{SCST}}(\theta) = -\mathbb{E}_{Y^s \sim p_\theta}\left[r(Y^s)\right]$$

To optimize L_{SCST}, we compute the policy gradient (as seen in REINFORCE):

$$\nabla_\theta L_{\text{SCST}} = -\mathbb{E}_{Y^s \sim p_\theta}\left[r(Y^s)\nabla_\theta \log p_\theta(Y^s)\right]$$

However, a significant issue with this formulation is the high variance in gradient estimation. SCST addresses this by introducing a self-critical baseline, where the reward for a sampled sequence Y^s is compared against a baseline sequence Y^b, typically generated using beam search:

$$\nabla_\theta L_{\text{SCST}} = -\mathbb{E}_{Y^s \sim p_\theta}\left[(r(Y^s) - r(Y^b))\nabla_\theta \log p_\theta(Y^s)\right]$$

Adaptive Loss Adaptive loss is a loss function that dynamically adjusts its parameters to best reflect the characteristics of the training data. A couple of studies have explored this idea to enhance model generalization performance. A study by Google explores and proposes generalized loss function that

adapts to several loss functions like Cauchy, L1/L2, Welsch by introducing robustness as a continuous parameter which improves learning-based tasks [15]. Another study proposes linearly adaptive cross-entropy loss as an optimization over traditional cross-entropy loss by introducing an additional term that depends linearly on the predicted probability of the correct class being identified [16]. This is an optimization process used to improve the classification task.

3 Data

In this paper, we adopt the CROHME datasets which are publicly available consisting of single-line handwritten mathematical expressions. The training set consists of 8836 images and the CROHME 2014 [12], 2016 [13], and 2019 [14] test sets each comprise 986, 1147, and 1199 images respectively.

Input data. A variable size greyscale image of a single-line handwritten expression.

Figure 1: Example Input Image from CROHME 2014 Test Set

Ground truth data. A sequence of LaTeX tokens corresponding to the input image.

```
\lim \limits _ { a \rightarrow \infty } f ( a )
```

4 Model architecture

We use TAMER as the base architecture for all the three proposed models where the blue components in Figure 2 make up the architecture of the original TAMER model.

4.1 Model 1

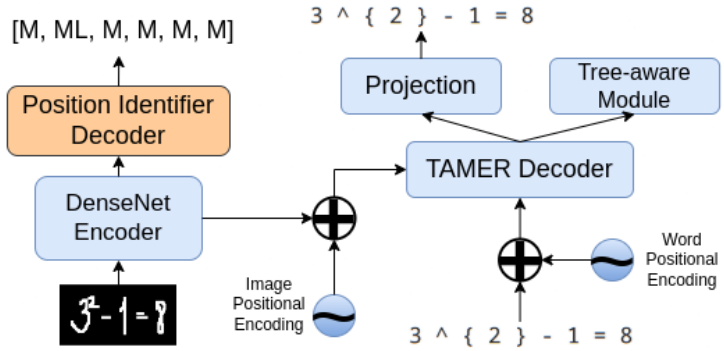


Figure 2: Proposed model architecture based on the TAMER model.

4.1.1 Motivation

We noted in section 2 how state-of-the-art models such as PosFormer and TAMER benefit from multi-task learning where they jointly optimize a LaTeX sequence prediction task and a LaTeX structure prediction task. These state-of-the-art models introduce their own LaTeX structure prediction task based on different encoding and representation of the hierarchical/structural relationship of the LaTeX symbols. Hence, we experimented with model 1 whether HMER models with two different LaTeX structure prediction tasks based on two different encoding of the hierarchical relationship of the LaTeX symbols would enable the model learn the hierarchical relationship of the symbols in different and complementary ways such that it improves the performance of the model.

4.1.2 Methodology

The state-of-the-art model PosFormer introduced the Position Forest Coding algorithm for its LaTeX structure prediction task. It is based on the observation that LaTeX expressions can be divided into multiple independent or nested substructures. Additionally, it takes advantage of the fact that each LaTeX substructure consists of a main symbol/keyword, and optionally an upper part and/or a lower part (e.g., subscripts, the curly bracket part of `\sqrt{}`, etc) relative to the main symbol. Thus, for each structure in a LaTeX expression, the algorithm then assigns a position identifier which denotes the relative position of the LaTeX symbols in the substructure. We introduce this algorithm for a new (second) LaTeX structure prediction task. Hence, we add the Position Identifier Decoder (the orange decoder in Figure 2) to additionally predict position identifiers of LaTeX symbols. Additionally, as attempt to improve the proposed model's prediction accuracy on deeply nested LaTeX symbols, we introduce depth-weighted cross entropy loss for the newly introduced position identifier prediction task (the second LaTeX structure prediction task) where each log term for a symbol s is multiplied by the following weight.

$$\text{weight}(s) = 1 + 0.5(\text{len}\left(\text{PosIdentifier}(s)\right) - 1)$$

4.1.3 Novelty

Model 1 have two novel aspects. First, it optimizes one LaTeX sequence prediction task and two LaTeX structure prediction tasks while existing models generally optimize LaTeX sequence prediction task and one LaTeX sequence prediction task. Through this model, we experiment whether multiple LaTeX structure prediction tasks with different encoding of the hierarchical relationship of the LaTeX symbol leads to improved performance. Additionally, Model 1 introduces depth-weighted CE loss for the position identifier prediction task where existing models employ only the base CE loss. Through the depth-weighted CE loss, we experiment whether punishing prediction errors on deeply nested symbols more harshly leads to improved performance.

4.2 Model 2

4.2.1 Methodology

We implemented SCST within the existing TAMER repository for HMER. SCST was primarily implemented within the training loop, where additional training logic was introduced to augment the existing loss with a SCST loss (computed as described above using `RapidFuzz Levenshtein Distance` as the reward metric). At each SCST training step, the model generates two sequences: a sampled sequence Y^s generated through stochastic sampling (temperature- scaled multinomial sampling was implemented directly into the decoder) and a baseline sequence Y^b generated through beam search. The reward function $r(Y)$ is applied to both sequences and loss is computed by taking the difference between the rewards, encouraging the model to generate sampled sequences with higher rewards than the baseline.

Now, to analyze the impact of SCST on model performance, we conducted an experiment by training the model (for only 200 epochs) at varying SCST update frequencies: every 250 batches, every 500 batches, every 1000 batches. We compared these results to assess the impact of SCST on model performance.

4.2.2 Novelty

This is a novel approach as the existing literature does not use a reinforcement learning-based approach or implement SCST for HMER.

4.3 Model 3

We use the same model architecture as the base TAMER model (Figure 2). Novel extensions in this model were made by incorporating gradient based adaptive loss instead of cross-entropy loss as discussed section 5.

5 Implementation Details

5.1 Model 1

The total loss for all jointly optimized tasks is as follows where L_{seq} is the CE loss for the LaTeX sequence prediction task, L_{struct} is the CE loss for the LaTeX structure prediction task, and L_{pos} is the depth-weighted CE loss for the position identifier prediction task.

$$L = \frac{L_{seq} + 0.5 \cdot L_{struct} + 0.5 \cdot L_{pos}}{2}$$

5.2 Model 3

We explored in section 2 that replacing traditional loss functions with adaptive loss by adding a dynamically adjusting parameter can improve learning-based tasks and generalization of the model. As a result, in this model extension, we experimented by adding dynamically adjusting weights that depend on the gradients of parameters of the model.

The adaptive loss function is implemented by first computing the loss terms, sequential loss and structural loss as computed in TAMER [2]. Then, we compute a vector of gradients of parameters of the model with respect to each loss term and compute its magnitude. The sum of these magnitudes are used to normalize each vector magnitude which are then used as weights in the adaptive loss.

$$w_{seq} = \frac{||\boldsymbol{v_{seq}}||}{||\boldsymbol{v_{seq}}|| + ||\boldsymbol{v_{struct}}||}, \quad w_{struct} = \frac{||\boldsymbol{v_{struct}}||}{||\boldsymbol{v_{seq}}|| + ||\boldsymbol{v_{struct}}||}$$

where $\boldsymbol{v_{seq}}$ and $\boldsymbol{v_{struct}}$ are the vector of gradients with respect to L_{seq} and L_{struct}. Hence, the adaptive loss used to train the model is:

$$L = w_{seq} \cdot L_{seq} + w_{struct} \cdot L_{struct}$$

5.3 Evaluation Metrics

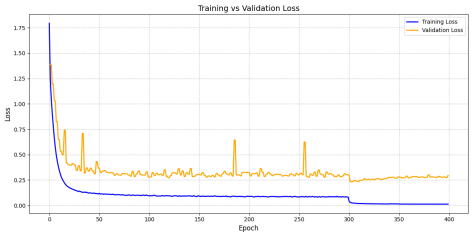
ExpRate is a commonly employed evaluation metric that measures the proportion of samples predicted that match the ground truth exactly. Additionally, $\leq \mathbf{1}$ and $\leq \mathbf{2}$ are variations of ExpRate that tolerates one and two symbol errors respectively.

Longest Subsequence Metric (LSM) is an evaluation metric introduced in this paper. The LSM score for a prediction y with ground truth t is computed by:

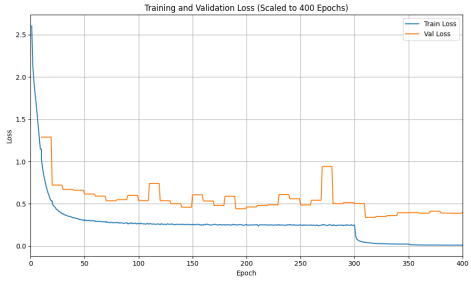
$$LSM(y, t) = \frac{\text{len of longest subsequence of } t \text{ in } y}{\text{len of y}}$$

LSM computes the mean percentage of longest contiguous characters from the ground truth that match exactly in each prediction. This metric is used to measure partial matches between prediction and ground truth. It was introduced to address limitations of ExpRate and its variations ($\leq 1, \leq 2$), which only accounts for partial matches up to 2 symbol tolerance.

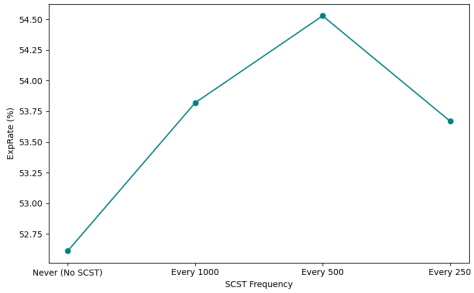
6 Results



(a) Training and Validation Loss for Model 1



(b) Training and Validation Loss for Model 3



(c) SCST Results for Model 2

Figure 3: Overall caption for the grouped figures.

6.1 Training and Validation Loss of Model 1

Figure 3a is a plot of the training and validation loss for model 1. We can observe that the validation loss is always slightly larger than the training loss which is as expected. Additionally, the training loss demonstrates a sharp initial decline followed by gradual stabilization, suggesting the model has effectively learned patterns in the training set. On the other hand, the validation loss shows an overall downward trend but there is also more fluctuations compared to the training loss. This can attributed to the difficulty of the validation set and the fact that CROHME training set is quite small which makes it difficult for the model to generalize well in a stable manner. Note that model 1 was trained with a batch size of 16 while evaluated with a batch size of 4. Thus, the fluctuations in the validation loss is also to be expected. Lastly, since the model was trained with a milestone of 300 and 350 epochs set, the learning rate scheduler reduced the learning rate as it was configured which lead to small drop in both training and validation loss at epoch 300.

6.2 Model 2 Performance at Different SCST Update Frequencies

As shown in Table 1, SCST consistently improves performance at all update frequencies across all datasets. We observe that applying SCST every 500 batches yields the best performance, suggesting that overly frequent updates (i.e. every 250 batches) may introduce instability due to high variance in gradient updates. Conversely, infrequent updates (i.e. every 1000 batches) may limit the effectiveness of reinforcement learning by reducing exposure to the reward signal.

SCST Update Frequency	2014	2016	2019	Average
Every 1000 batches	54.77%	52.66%	54.13%	53.82%
Every 500 batches	**54.46%**	**55.62%**	**53.71%**	**54.53%**
Every 250 batches	54.67%	53.36%	52.97%	53.67%
No SCST	53.85%	51.70%	52.38%	52.61%

Table 1: Model Performance at Different SCST Update Frequencies

6.3 Model 3 Performance

Figure 3b is a visualization of the training and validation loss for Model 3. Notice that the training loss declines smoothly after which there is a sharp drop in training loss after convergence at 0.24 at around 300^{th} epoch which happens for the same reason mentioned in section 6.1. Observe that the validation loss is noisier as it converges and is consistently higher than training loss. As mentioned in Section 6.1, this can be because of the small evaluation set of CROHME dataset. Note that Model 3 is trained with a batch size of 8 and evaluated with a size of 4.

6.4 CROHME Test Set Performance

The best performance for Model 1, Model 2, and Model 3 is summarized in Table 2.

Dataset	Scale Augmentation	Model	ExpRate↑	$\leq 1 \uparrow$	$\leq 2 \uparrow$	LSM ↑
CROHME 2014	×	PosFormer	60.45	77.28	**87.83**	-
		TAMER	61.23	76.77	83.25	82.80
		Model 1	**61.36**	76.27	83.16	82.71
		Model 2	60.85	**77.79**	83.97	**82.86**
		Model 3	60.14	76.17	83.98	82.33
CROHME 2016	×	PosFormer	60.94	76.72	83.87	-
		TAMER	60.26	76.91	84.05	82.29
		Model 1	59.37	76.55	83.78	82.35
		Model 2	**61.90**	**78.29**	**85.53**	**83.46**
		Model 3	59.37	76.98	84.57	82.67
CROHME 2019	×	PosFormer	**62.22**	**79.40**	**86.57**	-
		TAMER	61.97	78.97	85.80	**83.81**
		Model 1	60.80	78.48	85.58	83.52
		Model 2	60.96	79.15	85.82	83.38
		Model 3	58.30	77.40	85.15	82.57

Table 2: Model Performance for Three CROHME Test Sets

7 Discussion and Limitations

First, it can be observed from Table 2 that all three proposed models based on TAMER has very similar performance as state-of-the-art models such as PosFormer and TAMER. We analyze the reason for the model performance not improving from the base models below.

7.1 Analysis of the Performance of Model 1

Model 1 which introduced a new (second) LaTeX structure prediction task paired with the depth-weighted CE loss did not improve the model performance. We believe that the original LaTeX structure prediction task in the TAMER model based on a good enough encoding of the hierarchical relationship of LaTeX symbols. Thus, even though the two LaTeX structure prediction tasks are based on different algorithms to encode the hierarchical relation of LaTeX symbols, the model learns the same information implicitly through the joint optimization of the one LaTeX sequence prediction task and two LaTeX structure prediction task. Hence, we believe that unless the existing LaTeX structure prediction task is based on a flawed algorithm that decodes the hierarchical relationship of symbols, introducing an additional LaTeX structure prediction task with a different algorithm does not lead to an improvement in performance. Therefore, future research should focus on experimenting with unexplored modifications of the encoder, decoder, positional encoding, and the LaTeX structure prediction task with different algorithms to represent the hierarchical relationship of LaTeX symbols.

7.2 Analysis of the Performance of Model 2

Motivated by findings from Table 1, we proceeded to fully train the model at an SCST update frequency of 500 batches. However, as shown in Table 2, we did not observe a significant improvement

in performance over the original model. A possible explanation for this is that the benefits of SCST diminish as training progresses since its primary advantage lies in improving generalization, which may be more impactful in earlier stages of training.

Future work could explore a dynamic approach to SCST, adjusting the update frequency during training to better leverage its advantages at different stages of training. Additionally, investigating alternative strategies to address the variance and instability at higher update frequencies could further enhance the effectiveness of SCST.

7.3 Analysis of the Performance of Model 3

Model 3 which introduced gradient based adaptive loss instead of the original sequential and structural loss did not improve the model's performance. Observe in Table 2 that Model 3's performance is consistently lower than TAMER by $\approx 1\%$ in ExpRate metric, but the difference is lower (sometimes better) in ≤ 1, ≤ 2, and LSM metrics. This tells us that the predictions made by Model 3 has a comparable partial match rate with ground truth compared to TAMER. Notice that Model 3 has similar difference with both Model 1 and Model 3 as it does with TAMER.

We believe that Model 3 fails to generalize due to overfitting. We believe that the model, in an attempt to optimize the loss term with larger gradient norm, compromises the other loss term enough to decrease the model's performance. Furthermore, the larger gradient norm may not continuously reflect that the loss term contributing to it is more important or that the loss term with lower gradient norm is less important. It could be that the batch size is smaller, which leads to noisier convergence which can falsely reflect the importance of the loss term. Additionally, this could also lead to overfitting by lowering the importance of the term with lower gradient norm.

We can improve upon this technique by modifying the weights to be linear with respect to the loss to better reflect the performance gains mentioned by J.W. Shim [16]

8 Conclusion

In this study, we explored three distinct models built upon the TAMER framework [2] with the aim of enhancing performance on the CROHME test sets. Each proposed model incorporated novel aspects designed to deepen our understanding of state-of-the-art handwritten mathematical expression recognition (HMER) models. Despite the innovative modifications, experimental results revealed that none of the models significantly improved the ExpRate metric, its variations or LSM.

These findings highlight important challenges in the domain of HMER and underscore the need for continued research. The insights gained from these experiments suggest that future work may benefit from one LaTeX sequence prediction task and one LaTeX structure prediction task with explored modifications made to the encoder, decoder, positional encoding, and the algorithm used to represent the hierarchical representation of the LaTeX symbols. Furthermore, conducting more granular error and performance analysis could provide additional insights into the inner-workings of existing HMER models which can be used to design and experiment with future models.

In conclusion, while the proposed approaches did not yield the expected improvements, the lessons learned are instrumental in shaping future research directions. The negative results reported here contribute to a more comprehensive understanding of the limitations of current HMER models and serve as a foundation for subsequent investigations in this rapidly evolving field.

9 Contributions

Babur introduced and implemented the position identifier prediction task (including the Position Identifier Decoder) and the depth-weighted CE loss in Model 1 based on the official GitHub repository of TAMER [2]. The actual code is implemented in the `multi-task-learning-position-identifier` branch of the project repository.

Sabrina implemented Self-Critical Sequence Training. The actual code is implemented in the `self-critical-sequence-training` branch of the project repository.

Srisaran incorporated gradient based adaptive loss in Model 3 and also implemented LSM metric based on the official GitHub repository of TAMER [2]. The modifications made to Model 3 is in `weighted_loss` branch of the project repository and the LSM implementation is in all branches.

Lastly, note that all three members contributed equally to the literature review of the background and related materials on HMER and to the creation of this report.

References

[1] Tongkun Guan, Chengyu Lin, Wei Shen, and Xiaokang Yang. Posformer: Recognizing complex handwritten mathematical expression with position forest transformer. *arXiv preprint arXiv:2407.07764*, 2024. Accessed: Month, Year.

[2] Jianhua Zhu, Wenqi Zhao, Yu Li, Xingjian Hu, and Liangcai Gao. Tamer: Tree-aware transformer for handwritten mathematical expression recognition. *arXiv preprint arXiv:2408.08578*, 2024. Code: https://github.com/qingzhenduyu/TAMER/.

[3] Wenqi Zhao and Liangcai Gao. Comer: Modeling coverage for transformer-based handwritten mathematical expression recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3343–3352, 2022.

[4] Xing Zhang, Jun Du, Lirong Dai, and Chin-Hui Lee. Watch, attend and parse: An end-to-end neural network based approach to mathematical expression recognition. *Pattern Recognition*, 71:196–206, 2017.

[5] Chen Li, Xiaoyu Zhou, Qun Li, et al. Improving handwritten mathematical expression recognition using counting and balancing strategies. In *International Conference on Pattern Recognition (ICPR)*, pages 123–130, 2022.

[6] Jade Wu, Connor Smith, Ethan Brown, et al. Tdv2: Enhanced two-dimensional decoder for large-scale handwritten math expression recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2022.

[7] Jin Yuan, Wei Li, Jianshu Chen, et al. Syntax-aware network for offline handwritten mathematical expression recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12345–12352, 2022.

[8] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195, 2017.

[9] Xing Zhang, Jun Du, and Lirong Dai. Dwap: Improved densenet for watch, attend and parse based handwritten mathematical expression recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 158–163, 2019.

[10] Zhaoxiang Bian, Cheng Li, Ruosi Yan, et al. Abm: Attention-balanced network for handwritten math expression recognition. *Pattern Recognition*, 128:108722, 2022.

[11] Wenqi Zhao, Qian Li, Liangcai Gao, et al. Bttr: Bidirectional transformer for handwritten mathematical expression recognition. In *International Conference on Pattern Recognition (ICPR)*, pages 3457–3465, 2021.

[12] Harold Mouchère, Christian Viard-Gaudin, and Richard Zanibbi. Competition on recognition of online handwritten mathematical expressions (crohme) 2014. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 779–784, 2014.

[13] Harold Mouchère, Richard Zanibbi, and Christian Viard-Gaudin. Competition on recognition of online handwritten mathematical expressions (crohme) 2016. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 578–583, 2016.

[14] Khurram A. Mahdavi, Richard Zanibbi, Harold Mouchère, Christian Viard-Gaudin, Tanveer Ahmad, Tomas Toufar, Jan Hajic, G. Andreu Sànchez, et al. Icdar 2019 crohme + tfd: Competition on recognition of handwritten mathematical expressions and typeset formula detection. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1533–1537, 2019.

[15] Jonathan T. Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4331–4339, 2019.

[16] Jae Wan Shim. Enhancing cross entropy with a linearly adaptive loss function for optimized classification performance. *Scientific Reports*, 14(1):4964, 2024.

10 Appendix

10.1 Position Forest Coding Algorithm

Figure 4 is a visualization of the Position Forest Coding algorithm. For the latex expression `2\sqrt{x} + \frac{2^x}{5}`, the algorithm first divides it into four independent substructures; `2`, `\sqrt{x}`, `+`, and `\frac{2^x}{5}`. Then, for each substructure, the main symbol gets assigned the position identifier M. Other symbols within the substructure gets assigned position identifiers relative to the main symbol M.

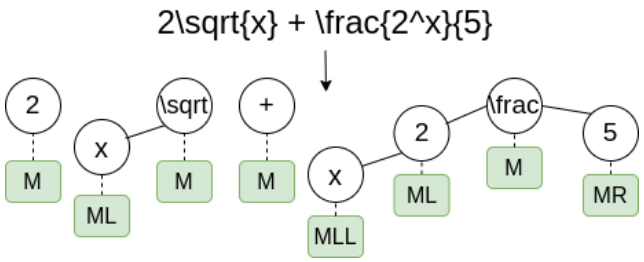


Figure 4: Position Forest Coding Algorithm Visualization

10.2 Hyperparameter settings.

10.2.1 Model 1

The hyperparameters for Model 1 are as follows. All the other hyperparameters are exactly the same as the base model TAMER.

- Epochs: 400
- Train Batch Size: 16
- Eval Batch Size: 4

10.2.2 Model 2

The hyperparameters for Model 2 are as follows. All the other hyperparameters are exactly the same as the base model TAMER.

- Epochs: 321
- Train Batch Size: 8
- Eval Batch Size: 4

10.2.3 Model 3

The hyperparameters for Model 3 are as follows. All the other hyperparameters are exactly the same as the base model TAMER.

- Epochs: 400
- Train Batch Size: 8
- Eval Batch Size: 4

10.3 Model Complexity, Training and Inference Speed

The base TAMER model was trained and evaluated on 4 NVIDIA 2080Ti GPUs using ddp. Model 1 and 3 were trained and evaluated on a single Nvidia GeForce RTX 4080 GPU. Model 2 was trained and evaluated on a single Nvidia A100 GPU. The CROHME dataset was used both for training and evaluation. The inference speed in Table 3 is amount of time it took the model to performance inference on the all three CROHME test sets (2014, 2016, 2019).

Model	Parameters (Million)	Number of Epochs	Training Time	Inference Time
TAMER	8.23	400	9h	2m
Model 1	11.5	400	32h	3m
Model 2	8.23	321	24h	3m
Model 3	8.23	400	52h	3m

Table 3: Model Characteristics