MAT-170 Spring 2025 Final Project

Authors: Aditya Joshi (918859330), Noe Velasquez (919936126),
Ohad Cortez (913416202), Sterling Beauchamp (916237750)

Instuctor: Dr.Yunpeng Shi

Abstract

Digitizing handwritten mathematical expressions is an essential yet challenging task in education, research, and accessibility. Handwritten notes, whiteboard derivations, and legacy teaching materials often remain in analog form, making them difficult to search, edit, or share. Automating the conversion of such expressions into LaTeX can dramatically enhance their usability. This work explores robust machine learning approaches for recognizing and transcribing handwritten mathematical content into structured, digital formats. We will compare PAL-v2 model against two recent state-of-the-art approaches: TAMER (Tree-Aware Transformer) and a Vision Transformer-based model. At the end of this work we have also shown an implementation of the PAL-v2 model.

1 Introduction

Handwritten Mathematical Expression Recognition (HMER) presents unique challenges due to the spatial and syntactic complexity of mathematical notation. Unlike standard text, mathematical expressions involve two-dimensional arrangements—fractions, superscripts, subscripts, roots, and nested structures—that require precise interpretation. Moreover, diverse handwriting styles introduce further ambiguity, making recognition systems difficult to generalize.

To address these issues, we examine HMER via Paired Adversarial Learning (PAL-v2), a model designed to learn semantic-invariant features from handwritten expressions by leveraging paired data with printed equivalents [2]. Through adversarial training and a convolutional decoder with attention, PAL-v2 aims to enhance recognition robustness across varying handwriting styles. We compare PAL-v2 against two recent state-of-the-art models. The first, TAMER (Tree-Aware Transformer), integrates structural supervision by jointly learning LaTeX sequence prediction and mathematical tree structures [3]. This dual objective improves the syntactic correctness of generated LaTeX, especially for complex expressions. The second, a Vision Transformer-based approach, replaces traditional CNN encoders with self-attention mechanisms to better capture long-range spatial relationships in handwritten images [1]. Furthermore, in the supplementary material section after the references, we have implemented the PAL-v2 model and included a short analysis.

2 Method Description

The major idea of the PAL-v2 paper relates to modeling how humans use what they've already read to interpret what comes next (e.g., reading left-to-right). To simulate this, the authors use a multi-block convolutional neural network decoder with a modified attention mechanism. This method introduces a *Pre-aware Unit* that incorporates previous attention patterns.

The attention weight at location (i, j) and layer l is computed as:

$$\alpha^l_{t,(i,j)} = \frac{\exp\left(P(h^l_t) \cdot f_{i,j}\right)}{\sum_{m=1}^{H} \sum_{n=1}^{W} \exp\left(P(h^l_t) \cdot f_{m,n}\right)}$$

The function $P(h^l_t)$ introduces prior attention knowledge:

$$P(h^l_t) = \left(V^l_1 W_{pa,t} h^l + V^l_2 h^l_t\right) + h^l_t$$

The first term, $V^l_1 W_{pa,t} h^l$, aggregates the past $t-1$ attention states (kind of like a summary of what we've already looked at). The second term, $V^l_2 h^l_t$, represents weighted current-step info, and h^l_t is added as a residual connection to aid the gradient's back propogation. Most other attention mechanisms ignore this historical context and only use unweighted current features(i.e. $P(h^l_t) = h^t_l$ in that case).

The training objective is also nonstandard. At the end of the computational workflow, a *discriminator* is introduced to distinguish whether features come from a handwritten or printed input. The goal is to make the recognizer (decoder) learn features that are *semantic-invariant*—so that handwritten and printed expressions with the same meaning look similar in feature space. The discriminator:

$$\begin{aligned}\mathcal{L}_D =& \mathbb{E}_{(X,Y)} \mathbb{E}_t \Big[\log D(a_t(x_p, y_{0:t-1}, \theta_R), \theta_D) \\ &+ \log(1 - D(a_t(x_h, y_{0:t-1}, \theta_R), \theta_D)) \Big]\end{aligned}$$

maximizes the probability of correctly classifying printed vs handwritten features.

The recognizer is trained with three objectives:

1. An adversarial term that tries to *fool* the discriminator:

$$\mathcal{L}_{D_{\text{adv}}} = -\mathbb{E}_{(X,Y)} \mathbb{E}_t \left[\log D(a_t(x_h, y_{0:t-1}, \theta_R, \theta_D))\right]$$

2. A supervised term on handwritten samples:

$$\mathcal{L}_{C_h} = -\mathbb{E}_{(X_h,Y)} \sum_{t=1}^{T} \log p\left(y_t \mid x_h;\, y_{0:t-1};\, \theta_R\right)$$

3. A supervised term on printed samples:

$$\mathcal{L}_{C_p} = -\mathbb{E}_{(X_p,Y)} \sum_{t=1}^{T} \log p\left(y_t \mid x_p;\, y_{0:t-1};\, \theta_R\right)$$

Together, the total loss for the recognizer is:

$$\mathcal{L}_R = \mathcal{L}_{C_h} + \mathcal{L}_{C_p} + \lambda \mathcal{L}_{D_{\text{adv}}}$$

Here, λ balances the trade-off between recognizing expressions correctly and learning features that generalize across handwriting styles. When $\lambda = 0$, the recognizer is just trained normally on printed and handwritten data. As λ increases, more weight is given to learning semantic-invariant features and less to fine-grained symbol discrimination.

3 Comparative Analysis

3.1 TAMER Model

This model integrates a tree module into a Transformer architecture. By doing so, the model recognizes inherent hierarchical structure of mathematical expressions. Elements such as fractions, square roots, and exponents form a tree like organization structure to where they are easily modeled. Unlike the ViT based on a sequence approach, TAMER ensures syntactic correctness. The model consists of a CNN encoder to extract visual features followed by a decoder and an improved with the Tree-Aware Module (TAM). To ensure correctness is maintained this system uses a scoring mechanism to evaluate the validity of each expression. The tradeoff is a more complex training process and slower inference speed compared to some other models. The sequence loss function $L_{seq} = -\sum_{t=1}^{T} \log\left(P(y_t)\right)$ calculates the probability of distribution of possible outcomes. The incorrect solutions get penalized, and the correct ones get valued higher. The tree structure has a similar equation used for structure prediction. This finds the correct parent node and indicates the structure of the equation. Finally, the loss function $L = L_{seq} + \lambda L_{Tree}$, L_{seq} is the loss for the sequence, L_{Tree} is the loss for the structure and λ is the scaling factor between the two. A higher λ leads to predicting the correct structure, while a lower λ favors a correct sequence.

3.2 Automated LaTex Code Generation

This paper discusses the application of primarily Vision Transformers (ViT), to convert handwritten or digital mathematical expressions into LaTex. It starts with a CNN-LSTM baseline. The study replaces the CNN encoder with a pretrained ResNet50 and further advances to a ViT encoder. Other systems such as TAMER explicitly model the hierarchical structure of mathematical expressions. The symbols are organized into trees representing the relationships. Instead, this model treats tasks as pure sequence generation problems, reading the entire image and predicting the LaTex output. The system uses global self-attention in ViT to capture the pretrained images and local connections to improve feature extraction. This combination allows the model to effectively and accurately transcribe handwritten math expressions into Latex. It is most effective when using large scale data sets, with errors of handling complex expressions. It lacks handling expressions where structure is essential.

Model Type	TAMER	Vision Transformer (ViT-based)	PAL-v2
Focus	Explicit modeling of hierarchical structure (tree) for structure correctness.	Sequence generation using global self-attention	Robust handling of handwriting style variation via paired adversarial learning.

Architecture	CNN encoder (DenseNet) + Transformer decoder + Tree-Aware Module (TAM) + Tree Scoring.	ViT encoder + Transformer decoder.	DenseNet encoder + DenseMD (multi-directional RNN) + convolutional decoder with Pre-aware Coverage Attention (PCA)
Sequence Loss Function	$L_{\text{seq}} = -\sum_{t=1}^{T} \log P(y_t)$	No explanation	$\mathcal{L}_R = \mathcal{L}_{C_h} + \mathcal{L}_{C_p} + \lambda \mathcal{L}_{D_{\text{adv}}}$
Structure Loss Function	$L_{\text{seq}} = -\sum_{t=1}^{T} \log P(y_t)$	None	None
Total Loss Function	$L = L_{\text{seq}} + L_{\text{Tree}}$	No explanation	Recongizer:$\mathcal{L}_R = \mathcal{L}_{C_h} + \mathcal{L}_{C_p} + \lambda \mathcal{L}_{D_{\text{adv}}}$ Discrimnator: $\mathcal{L}_D$
Strengths	Structure correctness; high structural accuracy	High accuracy; efficient sequence generation; scalable	Robust to handwriting variability; effective feature learning; strong attention with PCA.
Weaknesses	Complex training; slower inference.	No structure correctness checks; relies on large datasets	Requires paired data(written and transcribed); complex adversarial training
Best Used For	Automated grading of complex math with nested structures.	Efficient LaTeX transcription for large datasets.	Robust transcription from handwritten math with high variability.

4 Critical Evaluation

4.1 Strengths

- **Semantic-Invariant Feature Learning** - Paired Adversarial learning is used to train the model on handwritten and printed LaTeX expressions.This process wherein a discriminator and a attentional recognizer compete with each other encourages the model to learn representations that are consistent across different writing styles while learning to ignore the differences that exist in the way different people write different symbols such as drawing a “t” with a curve at the bottom compared to just a straight line without a curve amongst numerous other such differences.

- **Pre-aware Coverage Attention (PCA)** - Convolutional decoders predict all of the symbols at the same time without placing inherent value on left to right reading as humans would. This creates a problem wherein such systems may focus on particular areas of symbols many times and on other areas rarely resulting in attentional errors in the system. The Pal-v2 framework helps solve this problem by using PCA to give the system memory of what has already been done.

4.2 Limitations

- **Accuracy** While PAL-v2 is an improvement over other systems the state of the art is still very inaccurate only being able to correctly identify expressions roughly 50% of the time depending on various factors. This is obviously not yet at the stage where it could be useful as a consumer level tool.

- **Sensitivity to Adversarial Loss Hyperparameter (λ)** - The performance of PAL-v2 is highly sensitive to the hyperparameter λ, which balances semantic invariant learning and discriminative learning. Too low a value results in poor generalization to different writing styles and too high a value degrades classification accuracy by suppressing discriminative features between symbols.

4.3 Failure Modes

- When symbols overlap or are visually similar such as is the case with “9”, “q”, and “g” or uppercase “C” versus lowercase “c” the system will often fail outright to correctly classify the characters. Additionally when symbols are slanted or roasted that will also negatively impact the ability to correctly classify them.

5 Contributions & Future Directions

Our primary contributions include implementing a simplified version of PAL-v2 and providing an in-depth critical analysis of its strengths and weaknesses, particularly focusing on its adversarial framework and PCA. Unlike other models, PAL-v2 learns to recognize the meaning of mathematical expressions regardless of writing style, making it especially suited for diverse, real-world handwriting. The addition of Pre-aware Coverage Attention further strengthens decoding accuracy by mimicking how humans remember previously attended symbols during reading.

However, our evaluation highlighted critical limitations, notably PAL-v2’s lack of syntactic enforcement, limited scalability due to paired data reliance, and sensitivity to adversarial hyperparameter tuning. Moreover, its reliance on paired data (handwritten and printed) limits scalability, and its performance is highly sensitive to the adversarial loss weighting.

Based on our analysis propose the following future directions:

- **Integrate Structure-Aware Decoding:** Incorporate a lightweight tree decoder or structural consistency loss, inspired by TAMER's tree-based supervision, to enhance PAL-v2's handling of syntactically complex expressions such as fractions and matrices.

- **Replace CNN with Transformer-based Encoders**: Replace the current DenseNet encoder with a Vision Transformer (ViT) to better capture global layout and spatial relationships. This modification could significantly reduce misclassification errors, particularly among visually similar symbols.

- **Style Simulation through Data Augmentation**: Expand data augmentation strategies by introducing realistic handwriting variations—such as zoom, rotations, shears, and shifts—to improve robustness and generalization of the PAL-v2 model to unseen handwriting styles.

In summary, PAL-v2 represents a significant advancement toward robust handwritten expression recognition, emphasizing semantic consistency. Nevertheless, structural inaccuracies and scalability issues remain substantial barriers. Our analysis indicates that integrating structural awareness (via TAMER's tree-based decoding), adopting Transformer encoders for better spatial modeling, and leveraging enhanced data augmentation techniques collectively provide a promising route toward creating highly accurate, resilient, and practical models suited for education and accessibility.

References

[1] Jayaprakash Sundararaj, Akhil Vyas, and Benjamin Gonzalez-Maldonado. *Automated LaTeX Code Generation from Handwritten Mathematical Expressions*. arXiv preprint arXiv:2412.03853. 2024. URL: `https://arxiv.org/abs/2412.03853`.

[2] Jin-Wen Wu et al. "Handwritten Mathematical Expression Recognition via Paired Adversarial Learning". In: *International Journal of Computer Vision* 128.9 (2020), pp. 2386–2401. DOI: `10.1007/s11263-020-01291-5`.

[3] Jianhua Zhu et al. "TAMER: Tree-Aware Transformer for Handwritten Mathematical Expression Recognition". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. To appear. Association for the Advancement of Artificial Intelligence, 2025. URL: `https://github.com/qingzhenduyu/TAMER`.

6 PAL-v2 Implementation and Supplementary Material

6.1 Implementation Details

Our implementation closely followed the architecture described in the PAL-v2 paper, but with some key differences and steps we were unable to complete due to time and hardware constraints. Our implementation can be found on our GitHub.

6.1.1 Completed Steps

- **Model Architecture**: We successfully implemented the core components of the PAL-v2 model, including the DenseNet-based encoder, the convolutional decoder, the Pre-aware Coverage Attention (PCA) mechanism, and the token-level discriminator.

- **Data Preprocessing**: We replicated the data pipeline by parsing raw InkML files from the CROHME dataset, converting them into images, and storing them in an efficient LMDB format. We also implemented the BPE tokenizer to convert LaTeX strings into token sequences.

- **Adversarial Training**: The paired adversarial training loop, which updates the generator (recognizer) and discriminator alternately, was implemented as described. The combined loss function $\mathcal{L}_R = \mathcal{L}_{C_h} + \mathcal{L}_{C_p} + \lambda \mathcal{L}_{D_{adv}}$ was used to train the recognizer.

- **Validation and Analysis**: We created a validation script to measure model performance, including perplexity and error-tolerant expression rates (ExpRate), and to visualize prediction examples.

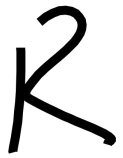
6.2 Experimental Findings

After training our simplified model for two epochs, we evaluated it on the CROHME 2013 dataset. Due to time constrains, we did not use the separate test dataset the paper used, instead this evluation is done on a 10% validation split of the train dataset. The model achieved a final validation perplexity of **6.01**. The expression recognition rates are detailed below:

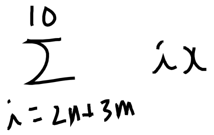
Our Model Performance	
Metric	**ExpRate(%)**
Exact Match (0 errors)	0.23%
ExpRate ($\leq$1 error)	6.80%
ExpRate ($\leq$2 errors)	10.87%
ExpRate ($\leq$3 errors)	17.89%
Paper's Model Performance	
Exact Match (0 errors)	54.87%
ExpRate ($\leq$1 error)	70.69%
ExpRate ($\leq$2 errors)	75.76%
ExpRate ($\leq$3 errors)	79.01%

The low exact match rate of 0.23% is expected at this early stage of training. However, the error-tolerant metrics show a more promising trend, with nearly 18% of expressions recognized with three or fewer symbol-level errors. This indicates that the model is learning meaningful representations, even if it cannot yet generate perfect sequences.

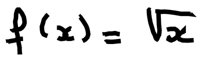
6.3 Recognition Examples



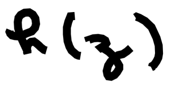
truth: `\mbox { R }`
reco: `\mbox { R }`



truth:`\sum_{i = 2 n + 3 m}^{1 0} i x`
reco:`\sum_{m = 2 n + 3 m}^{1 0} i x`



truth: `f(x) = \sqrt{x}`
reco: `y(x) = asqrt{x}`



truth: `$h(z)$`
reco: `$f (x)$`

6.4 Limitations and Deviations

Our implementation did not fully replicate the performance described in the original paper, primarily due to hardware constraints, limited training time, and architectural simplifications:

- **Hardware Constraints**: Training on a MacBook Pro rather than the two Titan X GPUs used in the original study resulted in frequent memory errors. This limitation forced us to use smaller batch sizes, slowing convergence and affecting stability.

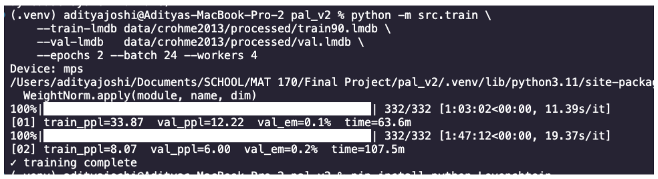
- **Limited Training**: Due to time constraints (approximately 2.5 hours per epoch including validation), we trained our model for only two epochs, significantly fewer than the extensive training used by the authors. Nevertheless, the notable reduction in perplexity from epoch 1 (12.22) to epoch 2 (6.01) indicates further improvements with additional epochs.

- **Architectural Simplifications**: Implementing a true multi-directional LSTM (MDLSTM) and integrating the external 4-gram statistical language model proved complex and beyond our project's scope. Instead, we substituted a standard convolutional layer inspired by the TAMER model, significantly deviating from the original architecture. These omissions hindered capturing the full 2D structure and correcting syntactic errors effectively.

- **Dataset Differences**: The original paper utilized the CROHME 2014 and 2016 datasets, which progressively incorporate more data from prior years. To reduce training time and resource requirements, our implementation only used the smallest and earliest dataset, CROHME 2013.

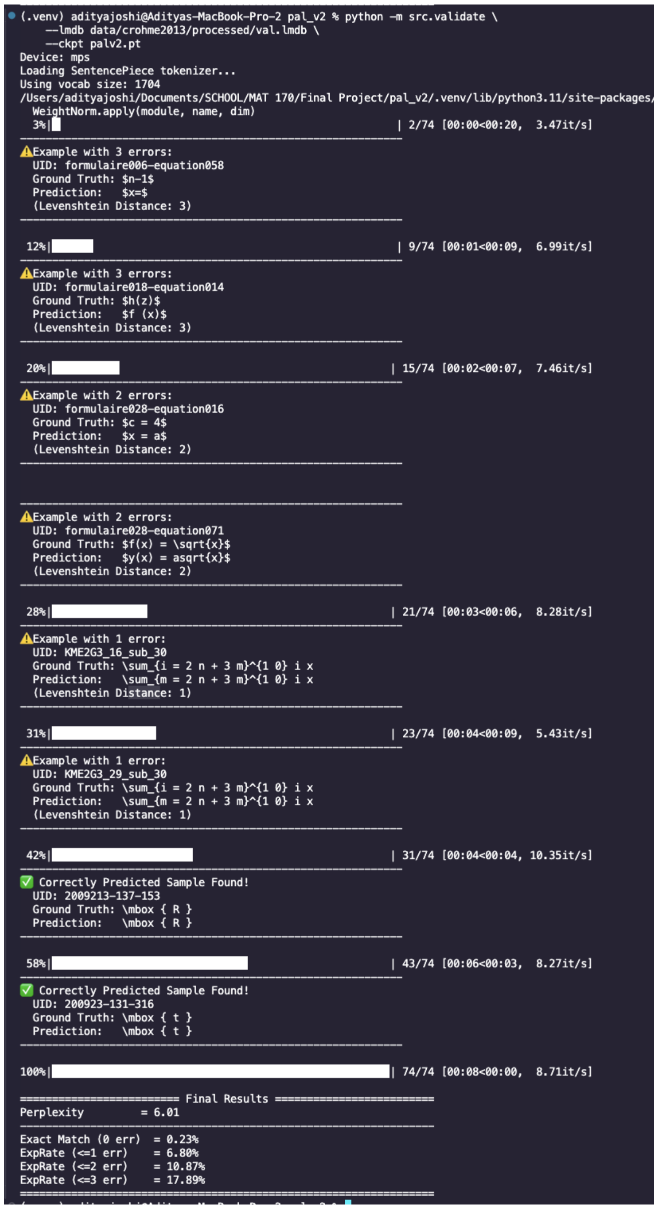
6.5 Evidence of Implementation

Below are screenshots demonstrating the successful implementation of our model.



```
(.venv) adityajoshi@Adityas-MacBook-Pro-2 pal_v2 % python -m src.train \
    --train-lmdb data/crohme2013/processed/train90.lmdb \
    --val-lmdb   data/crohme2013/processed/val.lmdb \
    --epochs 2 --batch 24 --workers 4
Device: mps
/Users/adityajoshi/Documents/SCHOOL/MAT 170/Final Project/pal_v2/.venv/lib/python3.11/site-packa
  WeightNorm.apply(module, name, dim)
100%|████████████████████████████████████████| 332/332 [1:03:02<00:00, 11.39s/it]
[01] train_ppl=33.87  val_ppl=12.22  val_em=0.1%  time=63.6m
100%|████████████████████████████████████████| 332/332 [1:47:12<00:00, 19.37s/it]
[02] train_ppl=8.07  val_ppl=6.00  val_em=0.2%  time=107.5m
✓ training complete
```

Evidence of Validation



```
(.venv) adityajoshi@Adityas-MacBook-Pro-2 pal_v2 % python -m src.validate \
    --lmdb data/crohme2013/processed/val.lmdb \
    --ckpt palv2.pt
Device: mps
Loading SentencePiece tokenizer...
Using vocab size: 1704
/Users/adityajoshi/Documents/SCHOOL/MAT 170/Final Project/pal_v2/.venv/lib/python3.11/site-packages/
  WeightNorm.apply(module, name, dim)
  3%|█                                                    | 2/74 [00:00<00:20,  3.47it/s]
------------------------------------------------------------
⚠️Example with 3 errors:
  UID: formulaire006-equation058
  Ground Truth: $n-1$
  Prediction:   $x=$
  (Levenshtein Distance: 3)
------------------------------------------------------------

 12%|██████                                               | 9/74 [00:01<00:09,  6.99it/s]
------------------------------------------------------------
⚠️Example with 3 errors:
  UID: formulaire018-equation014
  Ground Truth: $h(z)$
  Prediction:   $f (x)$
  (Levenshtein Distance: 3)
------------------------------------------------------------

 20%|██████████                                          | 15/74 [00:02<00:07,  7.46it/s]
------------------------------------------------------------
⚠️Example with 2 errors:
  UID: formulaire028-equation016
  Ground Truth: $c = 4$
  Prediction:   $x = a$
  (Levenshtein Distance: 2)
------------------------------------------------------------


------------------------------------------------------------
⚠️Example with 2 errors:
  UID: formulaire028-equation071
  Ground Truth: $f(x) = \sqrt{x}$
  Prediction:   $y(x) = asqrt{x}$
  (Levenshtein Distance: 2)
------------------------------------------------------------

 28%|██████████████                                      | 21/74 [00:03<00:06,  8.28it/s]
------------------------------------------------------------
⚠️Example with 1 error:
  UID: KME2G3_16_sub_30
  Ground Truth: \sum_{i = 2 n + 3 m}^{1 0} i x
  Prediction:   \sum_{m = 2 n + 3 m}^{1 0} i x
  (Levenshtein Distance: 1)
------------------------------------------------------------

 31%|███████████████                                     | 23/74 [00:04<00:09,  5.43it/s]
------------------------------------------------------------
⚠️Example with 1 error:
  UID: KME2G3_29_sub_30
  Ground Truth: \sum_{i = 2 n + 3 m}^{1 0} i x
  Prediction:   \sum_{m = 2 n + 3 m}^{1 0} i x
  (Levenshtein Distance: 1)
------------------------------------------------------------

 42%|█████████████████████                               | 31/74 [00:04<00:04, 10.35it/s]
------------------------------------------------------------
✅ Correctly Predicted Sample Found!
  UID: 2009213-137-153
  Ground Truth: \mbox { R }
  Prediction:   \mbox { R }
------------------------------------------------------------

 58%|█████████████████████████████                       | 43/74 [00:06<00:03,  8.27it/s]
------------------------------------------------------------
✅ Correctly Predicted Sample Found!
  UID: 200923-131-316
  Ground Truth: \mbox { t }
  Prediction:   \mbox { t }
------------------------------------------------------------

100%|██████████████████████████████████████████████████| 74/74 [00:08<00:00,  8.71it/s]

======================== Final Results ========================
Perplexity         = 6.01
---------------------------------------------------------------
Exact Match (0 err)  = 0.23%
ExpRate (<=1 err)    = 6.80%
ExpRate (<=2 err)    = 10.87%
ExpRate (<=3 err)    = 17.89%
===============================================================
```

Evidence of Training