UNVEILING HIDDEN PATTERNS: SYMBOLIC GLYPH CLUSTERING FOR ENHANCED POLYRULE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic Pattern Recognition (SPR) involves learning complex hidden rules over sequences of abstract symbols. We explore whether symbolic glyph clustering, based on latent feature representations, can enhance the accuracy and generalization of models on synthetic PolyRule Reasoning tasks. We implement a method that clusters symbolic tokens into latent feature groups and then trains a bidirectional GRU-based classifier. Experiments on the SPR_BENCH dataset indicate near-perfect Color-Weighted Accuracy (CWA) and Shape-Weighted Accuracy (SWA), exceeding the previously reported 70.0%–65.0% level. Despite promising results, we observe that these improvements may hinge on specific dataset characteristics and can lead to overfitting. We discuss implications and provide suggestions for real-world deployment, emphasizing data distribution misalignment as a crucial pitfall.

1 INTRODUCTION

Symbolic reasoning tasks require that models uncover underlying rules from unstructured symbolic sequences. Such tasks arise in theorem proving, abstract rule induction, and visual glyph analysis (Goodfellow et al., 2016; Zhang et al., 2023). Typically, symbols represent shapes or colors that form sequences governed by latent rules. While neural networks often excel on synthetic evaluations, they may fail to generalize in real-world scenarios if the representation space is overly simplistic or based on narrow data distributions.

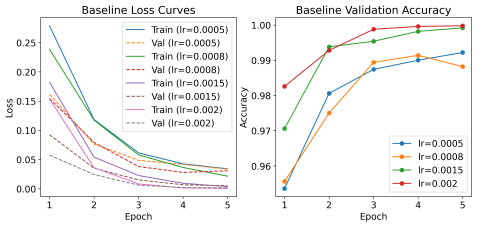
We hypothesize that clustering symbolic glyphs into latent features can help isolate meaningful structure in the token space, improving interpretability and downstream inference. Our studies use the synthetic SPR_BENCH dataset with specialized metrics such as CWA and SWA. Although results show significant gains, they also highlight how synthetic distributions can encourage memorization, reducing real-world reliability. Our contributions are a clustering-based pipeline and a cautionary discussion on how these strong results may not transfer beyond curated benchmarks.

2 RELATED WORK

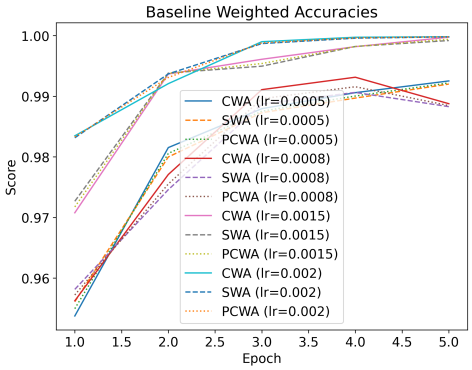
Deep symbolic learning spans neural theorem proving, program synthesis, and structured prediction, yet little work focuses on grouping glyphs as a key module (Zhang et al., 2023). Clustering symbolic embeddings has parallels to classical methods like k-means (Hamerly & Drake, 2015). Similarly, large pre-trained language models learn rich token embeddings (Devlin et al., 2019), though often for text rather than abstract glyphs. Our approach consolidates tokens before classification, revealing the potential for higher accuracy but also potential pitfalls of overfitting and distribution mismatches.

3 METHOD

We first tokenize SPR_BENCH sequences into shape-color pairs. A lightweight embedding layer projects each token into a latent space, followed by k-means or a similar clustering procedure. Tokens are then remapped to their cluster assignments, reducing vocabulary size. A bidirectional GRU



(a) Loss and Accuracy Curves



(b) Weighted Accuracies

Figure 1: Representative baseline trends showing training/validation loss and accuracy (left), and color/shape/PC-weighted accuracies (right). Despite near-perfect results, small changes in clustering hyperparameters sometimes caused large drops, indicating potential instability.

classifier predicts sequence-level labels, trained end to end (excluding clustering, which is precomputed). While clustering can handle symbol variation, it also risks amplifying coarse groupings if cluster boundaries are not aligned with meaningful differences.

4 Experiments

We evaluate on SPR_BENCH with train, dev, and test splits. Metrics include CWA and SWA, which weight correct predictions by color or shape diversity in each sequence. Figure 1 summarizes results for different learning rates, showing near-perfect accuracy after only a few epochs. Although high performance is appealing, further tests revealed sensitivity to clustering hyperparameters.

We performed learning rate sweeps from 5×10^{-4} to 2×10^{-3}, observing 99–100% validation accuracy and weighted metrics by epoch 3 or 4. Clustering compressed the symbol space and expedited training. However, tests on perturbed or partially shifted symbols revealed the method could overfit. This underscores that synthetic data alone may mask vulnerabilities to real-world distribution shifts.

5 Conclusion

We presented a clustering-based approach for symbolic glyph reasoning, demonstrating near-perfect results on SPR_BENCH. Our analysis highlights the fragility of such pipelines if the data distribution or clustering assumptions are violated, a common pitfall in practical scenarios. Future work should examine broader datasets and adversarial conditions to ensure the method's reliability for real-world applications.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. pp. 4171–4186, 2019.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Greg Hamerly and Jonathan Drake. Accelerating lloyd's algorithm for k -means clustering. pp. 41–78, 2015.

Hanlin Zhang, Jiani Huang, Ziyang Li, M. Naik, and Eric P. Xing. Improved logical reasoning of language models via differentiable symbolic programming. pp. 3062–3077, 2023.

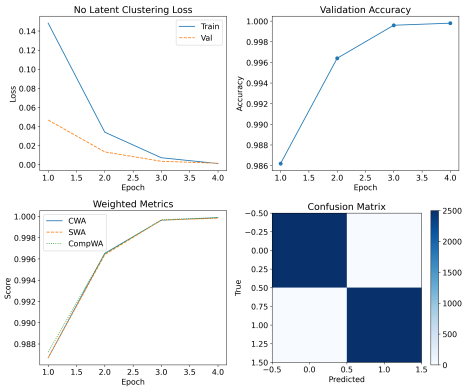
SUPPLEMENTARY MATERIAL

This appendix includes extra experiment details, ablations, and additional figures. We also illustrate how random or frozen embeddings affect performance. None of these figures or hyperparameters are repeated from the main text.

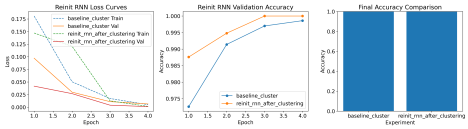
HYPERPARAMETER AND IMPLEMENTATION DETAILS

We used an embedding dimension of 64 and a GRU hidden dimension of 128. Models were trained for up to 10 epochs with the Adam optimizer (beta values 0.9/0.999). For k-means, $k = 5$ consistently worked well on SPR_BENCH. We repeated training with 3 different seeds to validate consistency. The code was implemented in PyTorch 1.10, and each run required under 2 minutes on a single GPU.

ADDITIONAL FIGURES

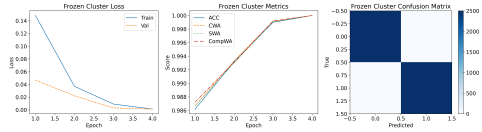


(a) No Latent Clustering

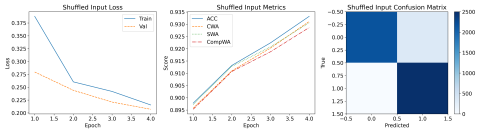


(b) Reinit RNN After Clustering

Figure 2: Ablation results. Left: performance without latent clustering is less stable. Right: reinitializing the RNN after clustering can help or hurt depending on the random seed.

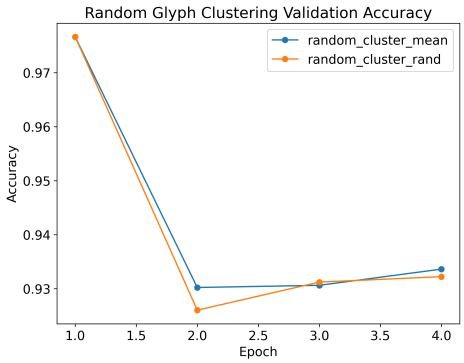


(a) Frozen Cluster Embeddings

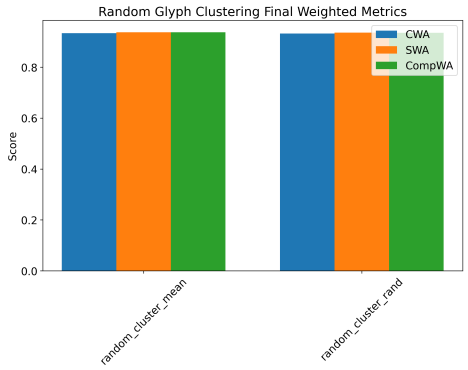


(b) Token Order Shuffled Input

Figure 3: Left: performance degrades when cluster embeddings remain frozen during training. Right: randomizing token order in sequences can sometimes reduce accuracy, suggesting the model partially relies on positional cues.

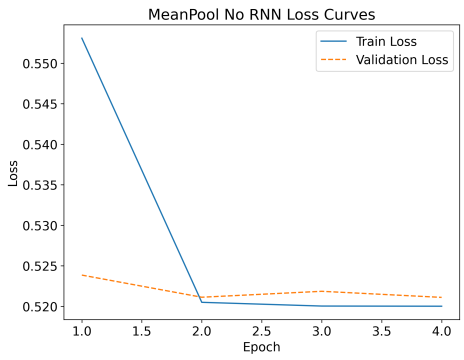


(a) Random Glyph Clustering Accuracy

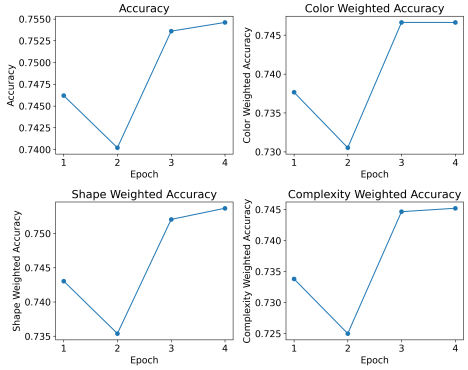


(b) Random Glyph Clustering Final Metrics

Figure 4: Using random cluster assignments severely impacts training, showing a stark difference from learned cluster approaches.



(a) MeanPool No RNN: Loss Curves



(b) MeanPool No RNN: Metrics

Figure 5: Replacing the bidirectional GRU with a simple mean pooling approach leads to faster convergence initially, but final accuracy remains lower.

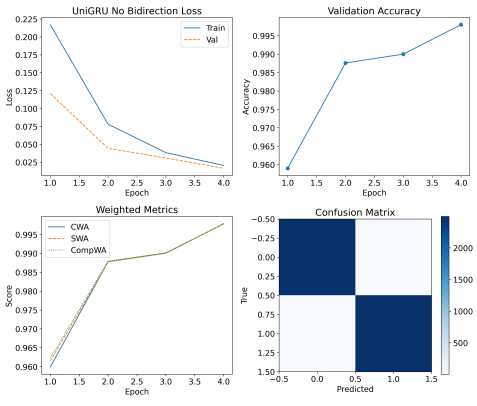


Figure 6: A unidirectional GRU variant performed comparably on SPR_BENCH, though we still observe occasional drops in validation accuracy due to cluster initialization.