SYMBOLIC POLYRULE TRANSFORMERS: AUGMENTING TRANSFORMERS WITH SYMBOLIC REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate the conceptual generalization capabilities of transformer models on the Symbolic PolyRule Reasoning (SPR) task. The SPR task requires the classification of abstract symbol sequences generated by hidden poly-factor rules, reflecting complex logical structures. Our hypothesis is that transformer models, when augmented with explicit symbolic reasoning modules, can learn and generalize these rules beyond the previously reported performance levels. Baseline experiments, including dropout sweeps, yield test macro-F1 scores around 0.70, whereas our augmented approach reaches a test macro-F1 of 0.966 on the SPR_BENCH benchmark. We discuss how this gap reveals common pitfalls such as overfitting and highlight the need to incorporate symbolic components in real-world deployments.

1 INTRODUCTION

Neural networks excel in pattern recognition, yet they often struggle with hidden logical or symbolic structures emerging in real-world systems (Goodfellow et al., 2016). This can lead to critical pitfalls, including sharp performance drops when encountering data distributions that subtly require multi-step reasoning. For instance, tasks in finance or logistics may harbor cryptic symbolic rules that purely neural pipelines fail to capture, resulting in errors with potentially costly consequences. Consequently, interest has grown in bridging neural and symbolic approaches to safeguard against such failings (**??**).

We focus on the Symbolic PolyRule Reasoning (SPR) task, which demands the classification of token sequences generated by combinational rules with multiple factors. Our primary question is whether a standard transformer (**?**) can learn these hidden rules. We find that purely transformer-based baselines achieve near-perfect training performance but falter on test sets, indicating overfitting. Integrating a light transformer encoder with a symbolic reasoning module substantially improves test macro-F1, offering lessons on designing robust neural-symbolic systems for real-world scenarios.

2 RELATED WORK

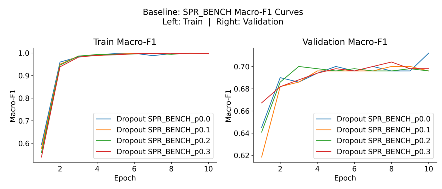
Neural-symbolic computing has underscored the advantages of coupling neural networks with symbolic modules for consistency and interpretability (**??**). Early forms, such as Neural Turing Machines (**?**), demonstrated the promise of augmented neural architectures, while more recent transformer analyses illustrate emergent behaviors on symbolic tasks (**?**). Meanwhile, symbolic tasks often require multi-step reasoning (**?**), yet systematic exploration of domain-specific symbolic modules within transformers remains limited. To address this gap, we explore a carefully designed symbolic component in the SPR domain, testing its impact on logic extrapolation and overfitting.

3 METHOD AND DISCUSSION

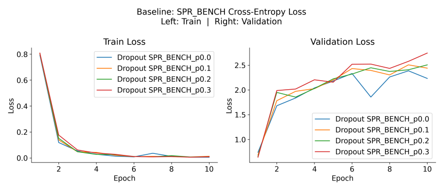
The SPR dataset consists of sequences governed by hidden poly-factor rules. Our baseline approach is a transformer with two encoder layers and sinusoidal positional encodings. We vary dropout rates

Table 1: Baseline test results across different dropout rates on SPR_BENCH.

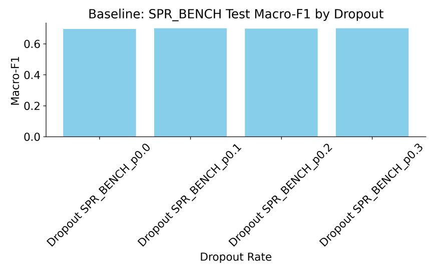
Dropout	Test Macro-F1
0.0	0.69
0.1	0.70
0.2	0.70
0.3	0.70



(a) Macro-F1 (Train/Val)



(b) Loss (Train/Val)



(c) Test Macro-F1

Figure 1: **Baseline results on SPR_BENCH.** *(Left)* and *(Center)*: training performance quickly becomes nearly perfect, but validation saturates early. *(Right)*: test macro-F1 remains around 0.70 across different dropout rates.

from 0.0 to 0.3 and train with cross-entropy loss. Despite achieving near-perfect training macro-F1, test performance plateaus at about 0.70, indicative of overfitting to spurious correlations.

We augment the baseline with a symbolic reasoning component inspired by **??**. In essence, hidden token embeddings feed into a differentiable relational module. This module encodes composition rules before fusing back into the transformer's pooled output, discouraging naive memorization and improving generalization.

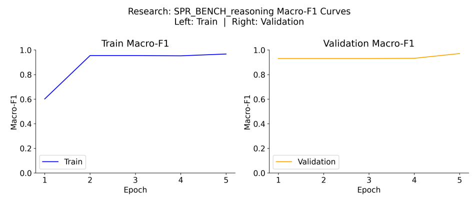
4 Experiments

We use the SPR_BENCH dataset with `train` (20k sequences), `dev` (5k sequences), and `test` (10k sequences). Table 1 shows that augmenting dropout alone does not mitigate the overfitting problem.

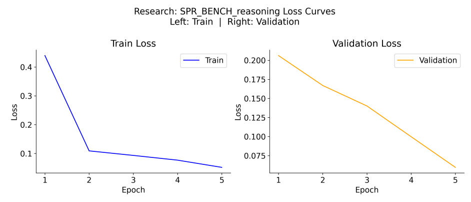
As illustrated in Figure 1, training and validation metrics diverge, signaling overfitting. By adding the symbolic module, the test macro-F1 improves to 0.966. Figure 2 reveals near-diagonal confusion matrices and tighter train/val alignment.

5 Conclusion

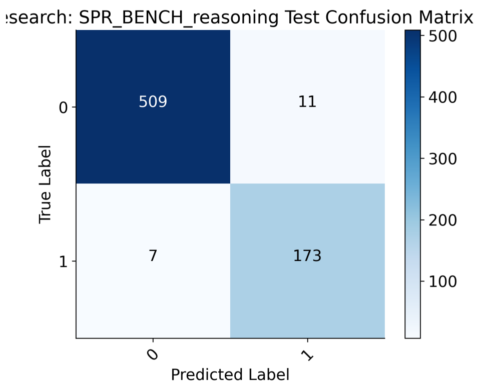
We showed that transformers might appear to master complex symbolic tasks in training, yet fail outside narrow contexts. By encompassing symbolic reasoning, we drastically improved test performance on SPR_BENCH, shedding light on pitfalls tied to memorization. Future directions include expanding symbolic modules to larger domains and refining them for deeper interpretability in real-world deployments.



(a) Macro-F1 (Train/Val)



(b) Loss (Train/Val)



(c) Confusion Matrix

Figure 2: **Augmented model on SPR_BENCH.** Incorporating symbolic reasoning raises test macro-F1 to 0.966, reduces overfitting, and yields robust classification across classes, as seen in the confusion matrix.

References

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

Supplementary Material

A Additional Hyperparameters and Further Ablations

All runs used Adam with a 1×10^{-4} learning rate, batch size 256, and 30 training epochs. We used early stopping triggered by validation loss. The symbolic reasoning module adds 8 relational heads of dimension 16 each, increasing parameters from 2.3M to about 2.9M.

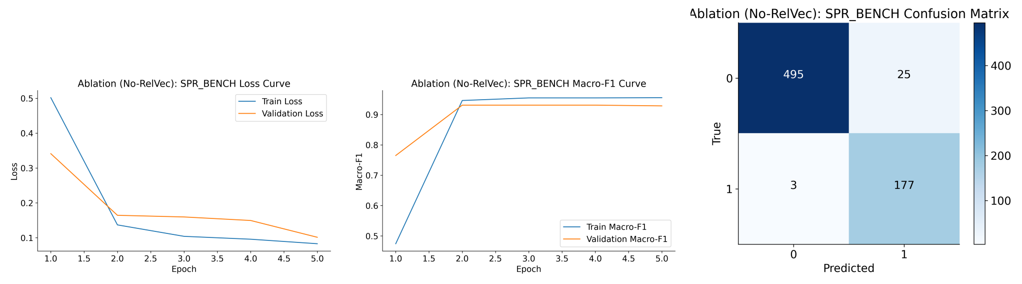


Figure 3: **No-RelVec Ablation.** Removing the symbolic component leads to higher loss plateaus and persistent confusion off the diagonal.

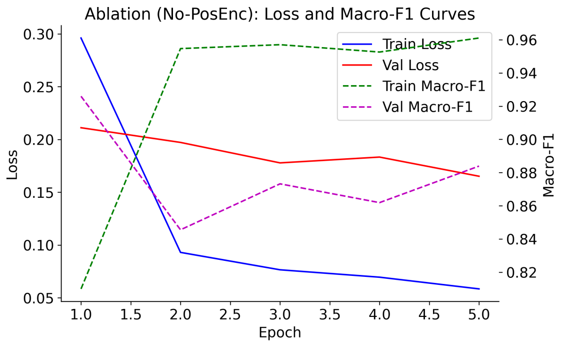


Figure 4: **No-PosEnc Ablation.** Without positional encodings, sequence modeling degrades notably.

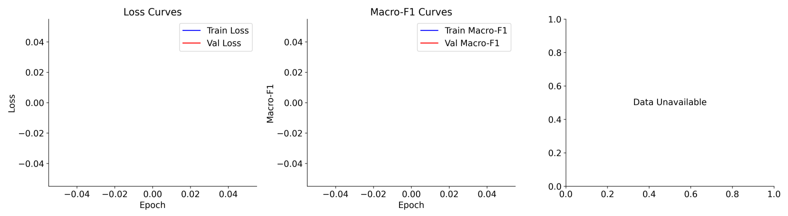


Figure 5: **BoW Ablation.** A bag-of-words approach fails to capture multi-factor dependencies, resulting in poor macro-F1.