Comparative Analysis of Context-Aware and Context-Free Models for Sequence Prediction on the IPARC Dataset

Anonymous authors
Paper under double-blind review

Abstract

This project explores sequence prediction in the IPARC dataset, where grayscale input images are transformed into output images through a sequence of eight operations—dilation or erosion—using specific structural elements. We compare two models: one that encodes information about the previously predicted sequence using a hidden state, and another that predicts each transformation based only on the immediate previous operation and structural element. Both models employ logistic regressors for operation (binary classification) and structural element (8-class classification) prediction. Thus, the project evaluates the impact of incorporating sequential context on prediction accuracy.

1 Problem Statement

The IPARC dataset contains pairs of input and output images, along with a sequence of eight transformations applied successively to transform the input image to the output image. The challenge is to identify an Inductive Logic Programming (ILP) engine capable of handling the selection, iteration, and sequencing components outlined in the Böhm–Jacopini theorem.

We focus only on the Category A (CatA Simple) tasks, which consist of simpler data points. These tasks apply a sequence of eight transformations to grayscale images, where each element in the image matrix is either 0 or 1. Each transformation has two components: an operation (dilation or erosion) and a kernel (a 3x3 structural element). There are eight types of kernels, each linked to a specific operation. Every example includes four input-output image pairs, where the output images are generated by applying these transformations sequentially to the corresponding input images. The task associated with this problem is to predict the 8-step transformation sequence where each step of the sequence involves dilation or erosion of the image using a particular structural element (kernel).

The CatA Simple tasks have a fixed configuration - the first 4 operations are always dilations, and last 4 are always erosions. Additionally, the sequence of structural elements used for dilation and erosion is identical. This fixed configuration limits the number of unique examples that can be generated. As a result, approaches which are data intensive like neural networks are unlikely to generalize well. In this project, we explore the effectiveness of the relatively simpler Logistic Regression model for the problem of prediction of the transformation sequence.

Hypothesis: *A model that takes as an input the encoded information about previously predicted sequence elements will lead to higher prediction accuracy for both operations and kernels, compared to a model that only takes as inputs the immediately preceding operation and kernel, as it allows for better context encoding of sequential dependencies in the transformation process.*

2 Methodology

2.1 Dataset Preprocessing

The dataset generation script given in the GitHub repository (GenerateDataset.py) was used to generate 5000 tasks for the Category A (simple).

For each task in CatA Simple there are 4 pairs of input-output images, where each image is of size 15x15. To preprocess the data we first flattened each 15x15 image (input or output) into a 225 sized linear vector.

For each step in the transformation sequence associated with a particular input-output pair, we represent whether it is dilation or erosion using a binary flag (0 for dilation and 1 for erosion) and represent the structural element (kernel) involved in each step using a one-hot encoded vector of size 8. One-hot encoding is used because it treats each category as distinct, thereby avoiding the introduction of an artificial order that could bias the model.

Due to the small size of the dataset and a large number of features in the input and output images (225) we need to reduce the number of features to mitigate the risk of overfitting. Since the input and output image matrices are relatively sparse (for CatA Simple) we used Principal Component Analysis to reduce the number of features to 60. 60 features were chosen since this explains approximately 80 percent of the variance.

2.2 Model 1 - With Hidden State

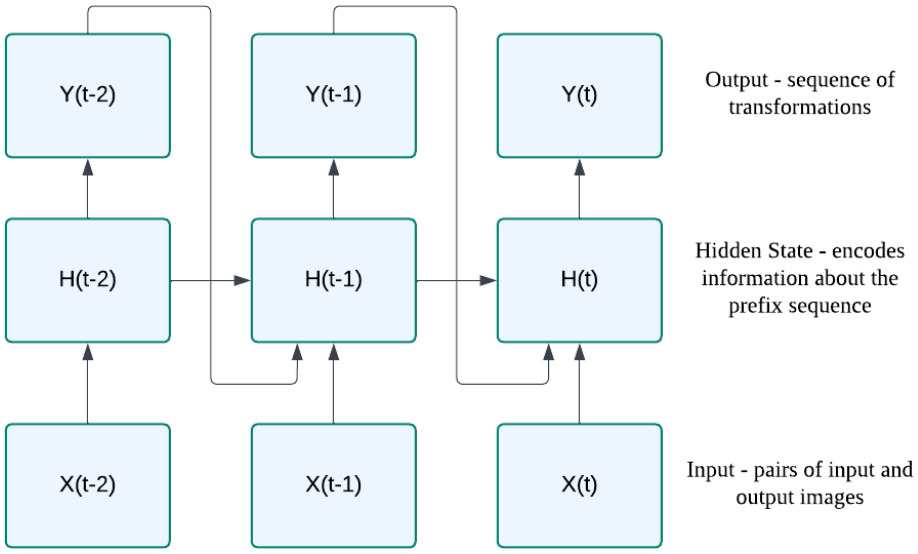

Figure 1: Hidden State Representation for Sequence Prediction

2.2.1 Logistic Regressors

We use two Logistic Regression models - one for predicting the operation (dilation or erosion) and the other for predicting the Kernel (the type of the structural element). The inputs to each Logistic Regressor is the input-output image and a hidden state (which encodes the information regarding the previous transformations observed in the sequence).

The regressor which predicts the next operation (dilation or erosion) acts as a binary classifier and is trained using Binary Cross Entropy loss. The other logistic regressor acts as a multi-class classifier (8 classes, one for each possible structural element) and hence uses Cross Entropy loss for training.

2.2.2 Updation of Hidden State

The hidden state is computed using Linear Regression - it updates the hidden state vector using the previous value of the hidden state vector, previous operation, and the previous kernel. Therefore, the overall input to the model consists of: four input-output image pairs, the previous operation, the previous kernel and the

hidden state. Initially, since the previous kernel and operation are not available, they are initialized as zero vectors.

2.3 Model 2 - Without Hidden State

Here, we again make use of two logistic regression models, with their inputs being the input-output image pairs, the immediately previous operation and kernel predicted. The notable difference here is that the hidden state is not maintained. We initialize the previous kernel and the previous operation as zero vectors. The loss functions used for each of the Logistic Regressors remain the same.

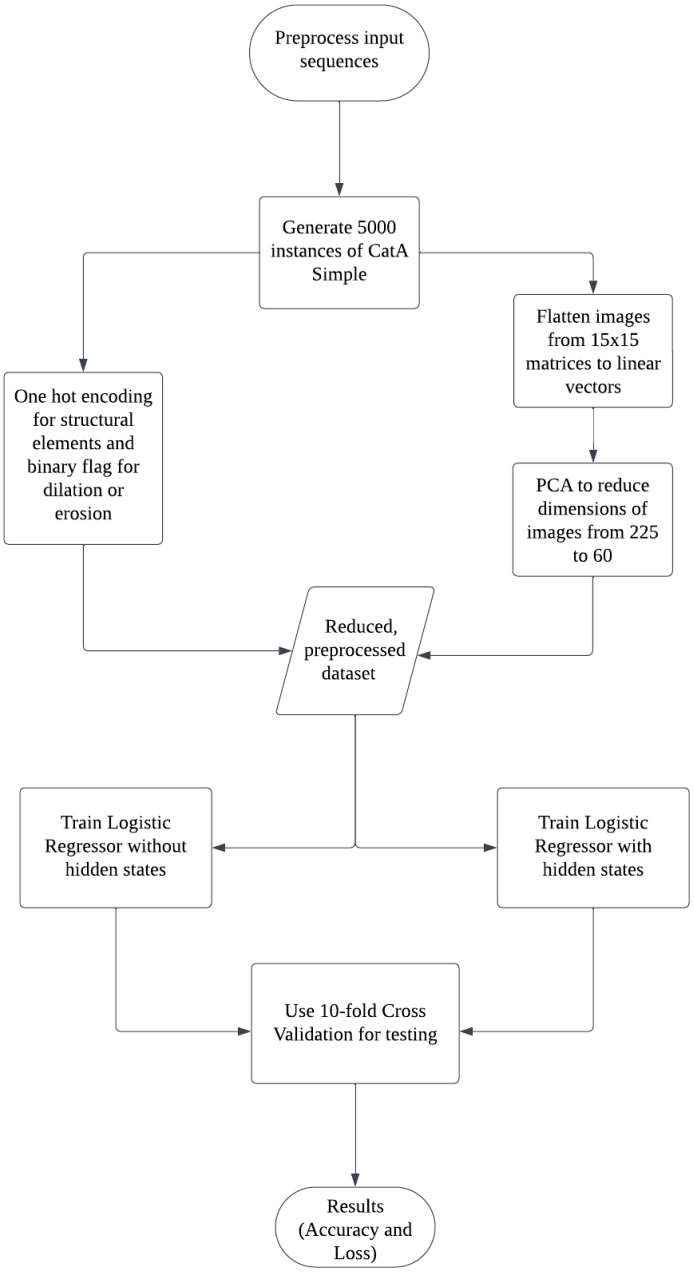

Figure 2: Step-by-step depiction of our methodology

3 Experimental Results and Validation

We conducted training for both models using a dataset of 5000 instances from the CatA Simple IPARC dataset. The models were trained for 40 epochs, as the loss values stabilized and showed no significant decrease beyond this point. Furthermore, we used 90/10 test-train split with Adam as the optimizer. For evaluation of the models, we use K-fold cross validation, with the number of folds set to 10. We plot the loss and accuracy for both the test and train sets over the 40 epochs.

3.1 Train and Test Loss Plots

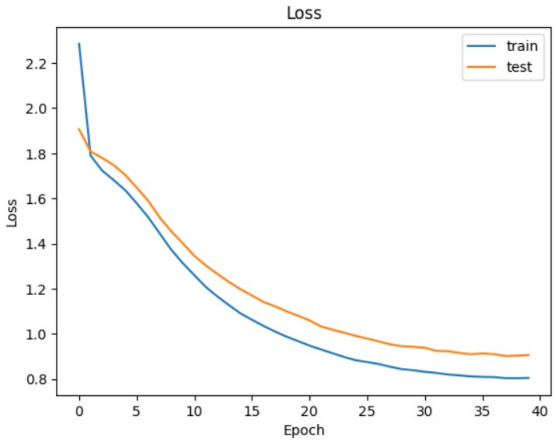

Figure 3: Loss Profile for Model 1 using a hidden layer with Logistic Regression

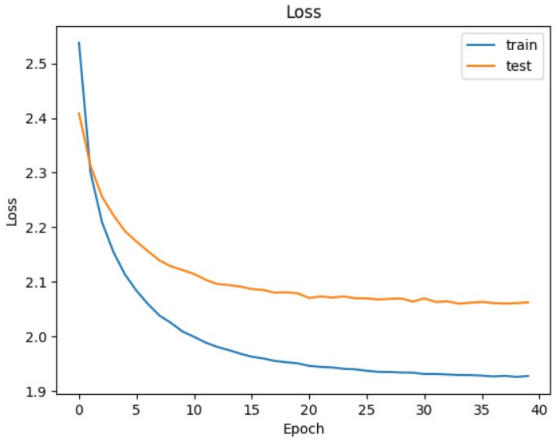

Figure 4: Loss Profile for Model 2 not using a hidden layer with Logistic Regression

We can see significant differences between the results of the two models trained on the IPARC dataset. The model which maintains the hidden state (Model 1) shows notably better kernel and operation accuracy and has much lower loss than Model 2.

3.2 Operation Accuracy Train and Test Plots

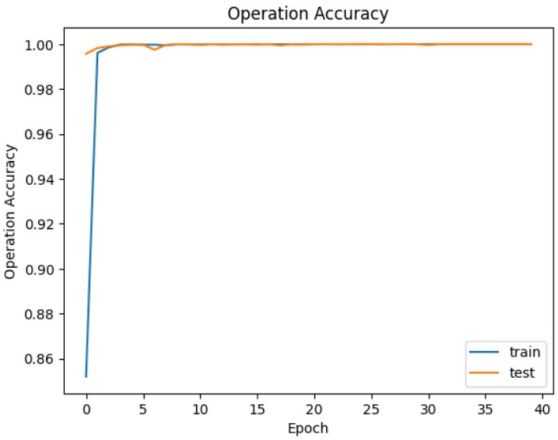

Figure 5: Operation Accuracy for Model 1 using a hidden layer with Logistic Regression

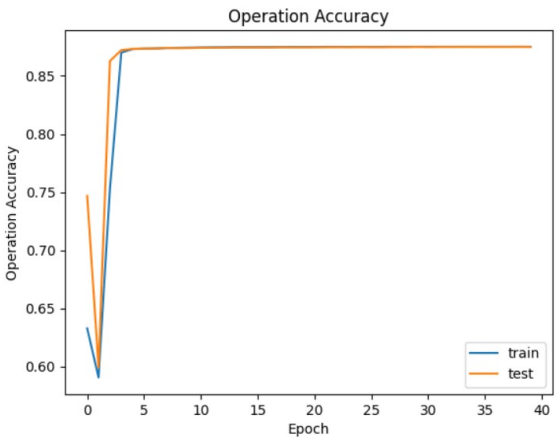

Figure 6: Operation Accuracy for Model 2 not using a hidden layer with Logistic Regression

Looking at the loss plots, we observe a larger generalization gap in Model 2. The larger gap indicates that Model 2 is overfitting to the training data and is not generalizing well to unseen data.

For Model 1, we achieved a near 1 test accuracy for operation prediction, which was a binary classification task to choose between dilation and erosion. This is because in all the tasks generated, the first 4 operations are dilation and the next 4 operations are erosion. This is a constant pattern which the model easily picks up on in the initial few epochs and hence achieves near perfect accuracy for operation.

However for Model 2, we only achieved 87.5% test accuracy as the model does not retain the contextual information from the previous steps. This limitation prevents the model from recognizing when the count of dilation operations has reached four as it only takes the immediately previous operation and kernel as the information. Since it does not encode the cumulative count of dilations, it is unable to precisely identify the switching point from dilation to erosion.

3.3 Kernel Accuracy Train and Test Plots

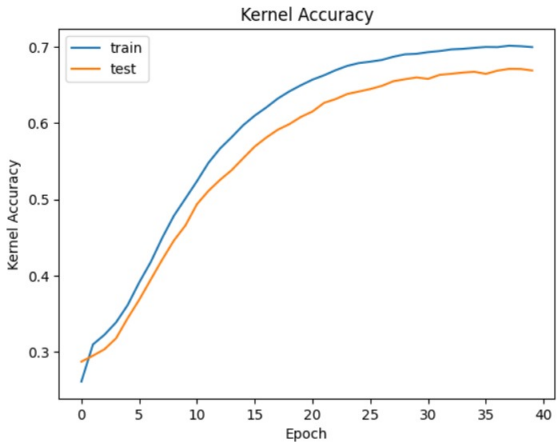

Figure 7: Kernel Accuracy for Model 1 using a hidden layer with Logistic Regression

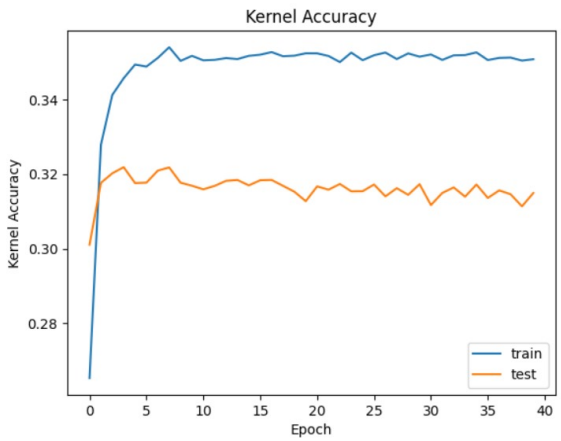

Figure 8: Kernel Accuracy for Model 2 not using a hidden layer with Logistic Regression

The most remarkable difference between the two models is in terms of kernel accuracy, which is correctly predicting the type of structural element (*SE1* to *SE8*). Hence, this a multi-class (8) classification problem as compared to binary classification problem in case of operation prediction. While Model 1 achieves a test kernel accuracy of 66.8%, Model 2 is only able to achieve 34.6% test kernel accuracy. This implies that encoding the contextual information about the steps predicted so far in the sequence in the form of a hidden state has a significant advantage.

In contrast, Model 2 lacks this capability and relies solely on the immediate previous operation and kernel, without retaining a broader understanding of the sequence. As a result, Model 2 struggles to identify patterns and key transitions that require a cumulative awareness of the sequence, leading to notably lower performance in kernel prediction. Also, in case of Model 2, convergence seems to be noisy, which suggests that Model 2 has high variance.

3.4 Effect of Dataset Size on Loss and Accuracy

Dataset Size	Train Loss	Test Loss	Train Operation Accuracy	Test Operation Accuracy	Train Kernel Accuracy	Test Kernel Accuracy
100	0.81692	2.86575	0.99998	0.99124	0.59463	0.28750
500	0.81456	2.67334	0.99994	0.99774	0.70537	0.37654
1000	0.76633	1.76280	0.99996	0.99980	0.72570	0.47640
2000	0.54803	1.18153	0.99987	0.99999	0.80302	0.59934
5000	0.52963	0.90062	0.99999	0.99995	0.81063	0.66849

Table 1: Train and Test Data for Model with Hidden State

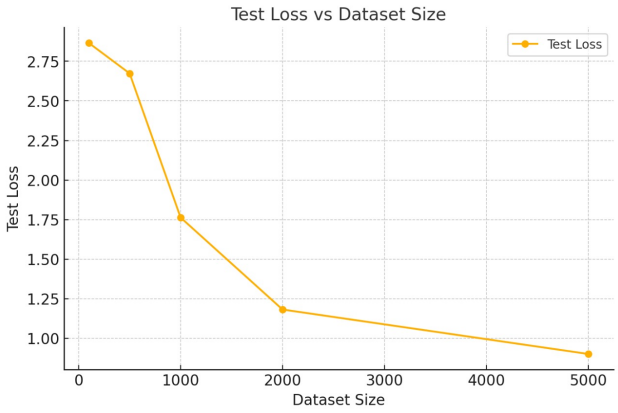

Figure 9: Test Loss as a function of Dataset size

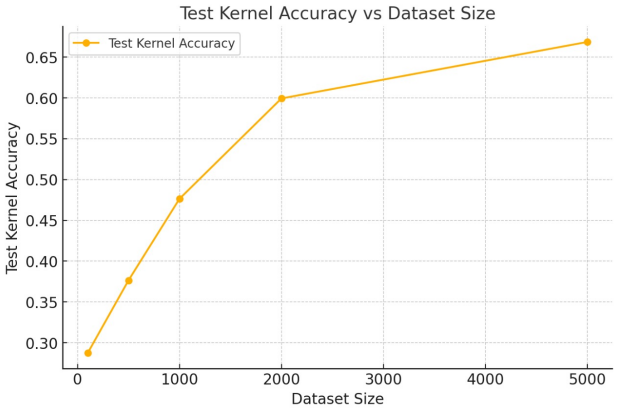

Figure 10: Test Kernel Accuracy as a function of Dataset size

We chose dataset sizes of 100, 500, 1000, 2000, and 5000 instead of uniformly varying sizes so as to better understand the learning behavior of the model when training data is highly constrained and observe the transition points where adding more data significantly improves performance.

We observe significantly lower kernel accuracies in case of smaller datasets because they lack sufficient examples and diversity to generalize well on the dataset and learn complex patterns. This is particularly important for kernel prediction which requires learning subtle differences between the various 8 structural elements. Additionally, with insufficient training examples the hidden state encoding may fail to represent meaningful context about the prefix of the sequence predicted so far reducing its effectiveness.

Even with a dataset size of 100, the test kernel accuracy is 28.75%, which is significantly above the baseline of 12.5% (random guessing among 8 structural elements). This shows that the model is still able to learn some meaningful patterns even with extremely limited data.

Furthermore, while the initial increase in dataset size from 100 to 2000 results in significant improvements in both kernel accuracy and test loss, the rate of improvement slows considerably when increasing the dataset size from 2000 to 5000, that is, it gives diminishing rate of returns. This suggests that the model is reaching its capacity for generalization. Secondly, we are generating our dataset for CatA Simple category - which places significant restrictions on the number and variety of unique examples that may be generated. Hence when we generate larger datasets of this type, a greater number of data points are repeated, which reduces the improvement in accuracy by using larger datasets.

4 Conclusion and Future Work

In conclusion, we are able to predict the sequence using our model that incorporates encoded information of previously predicted elements in the sequence with approximately 67% accuracy as compared to 34% accuracy obtained by a model which does not take contextual information of previously predicted steps into account using a hidden state. Hence, our initial hypothesis that a model that takes into account encoded information about previously predicted sequence elements will lead to higher prediction accuracy for both operations and kernels, compared to a model that only takes as inputs the immediately preceding operation and kernel, is shown to be true.

However, we have only tested our model using CatA Simple category datasets, and in its current state our model does not generalise well to the Inductive Logic Programming task associated with the ARC dataset. We can in principle generalize the model to work with more complex datapoints - having irregular order of dilation and erosion operations and having longer, variable sequence lengths, which is an interesting problem that can be worked on in future research.

Also, currently we remove a lot of attributes of interest in the preprocessing stage of our dataset - we lose spatial information about the pixels by flattening our input and output images. The model in its current state fails to understand the actual transformations performed by the sequence on the input image. Hence we sacrifice significant information by preprocessing our dataset, which could otherwise be used to incorporate vital domain knowledge.