Learning Permutation Matrix Patterns in the NARMA Dataset with Spiking Neural Networks: A Spike-Timing-Dependent-Plasticity Based Approach

Akshay kumar Mundrathi , Stevens Johnson , Johnson P Thomas Ph.D.
Dept of Computer Science , Oklahoma State University , Stillwater , USA

Problem Statement

- While Spiking Neural Networks (SNNs) have shown promise for temporal pattern recognition due to their biological plausibility and event-driven processing, their application to structured financial time series remains underexplored. Traditional neural network models often struggle to capture the complex, nonlinear dependencies present in financial data and lack the interpretability and efficiency of SNNs

- This research investigates the effectiveness of SNNs in classifying structured patterns within financial time series, specifically using permutation matrices derived from the NARMA dataset based on S&P 500 segments. The study aims to evaluate the potential of SNNs with fixed synaptic weights for trend classification, identify current limitations, and lay the groundwork for more adaptive, neuromorphic approaches in financial data analysis.

Technical Approach

Let X={x1,x2,…,xN}denote the dataset of input samples, where each xi is a 5×5 binary permutation matrix representing a segment of financial time series data. The corresponding set of ground truth labels is Y={y1,y2,…,yN}, where each yi ∈{0,1,2,3,4} indicates the pattern class for matrix xi.

The network employs N=5 output neurons, each associated with a unique class, and a pre-trained synaptic weight matrix W∈R(5×25) that defines the connection strength from each of the 25 input features (flattened from the 5×5 matrix) to each output neuron.

The simulation is conducted over T=20 discrete time steps, with each neuron governed by a spiking threshold Pth =6, membrane potential dynamics, and a lateral inhibition mechanism for competitive spiking behavior.

The goal is to predict the class label for each input matrix and evaluate classification performance using standard metrics and visualizations.

Classes and Network Architecture

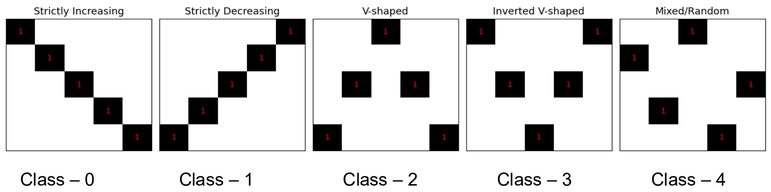

Figure 1: Pattern Classification classes

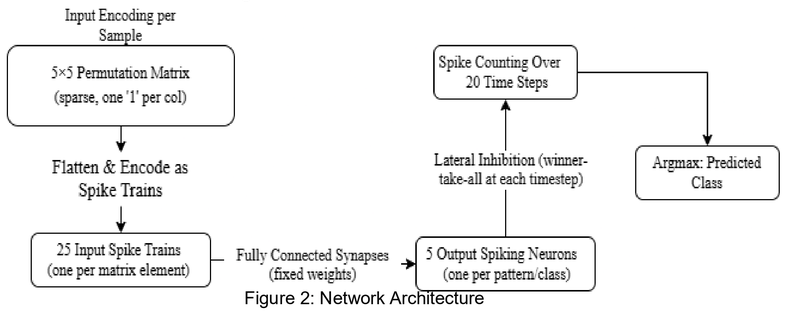

Figure 2: Network Architecture

Algorithm 1: Generate 5x5 Permutation Matrices from S&P 500 Time Series

Input: CSV file path (SP500.csv)
Output:
- List of binary permutation matrices (size 5x5)
- Corresponding labels

Begin:
1. Load and Clean Data:
   a. Read CSV file with columns: "date", "value"
   b. Convert "date" column to datetime format
   c. Set "date" as index
   d. Convert "value" column to numeric (handle errors)
   e. Interpolate missing values in the "value" column
2. Create Permutation Matrices:
   a. Initialize empty list `matrices`
   b. For i = 0 to length(data) - N, step N:
      i. Extract 5-day segment from values: segment = data[i : i + 5]
      ii. Rank the 5 values from smallest to largest (ranks range from 0 to 4)
      iii. Initialize a 5x5 zero matrix `M`
      iv. For each column j in 0 to 4:
          Set M[rank[j]][j] = 1
      v. Append matrix M to `matrices`
3. Assign Labels:
   a. Use the predefined label array: `labels[i]` corresponds to `matrices[i]`
4. Return:
- List of 5x5 binary matrices, Corresponding labels

Algorithm 2: Spiking Neural Network-Based Classification of Binary Permutation Matrices

Input:
1. Dataset of 5×5 binary matrices X = {x_1, x_2, ..., x_N}
2. Correct labels Y = {y_1, y_2, ..., y_N}, where y ∈ {0,1,2,3,4}
3. Pre-trained weight matrix W ∈ $\mathbb{R}^{5x25}$
4. Simulation time T = 20, threshold P_th = 6
5. Number of output neurons N = 5

Output: Predicted class for each input matrix

Begin:
1. For each output neuron j = 1 to N:
   - Set membrane potential P_j = 0,t_rest_j = 0, spike_count_j = 0
2. For each input matrix x_i in X:
   a. Preprocess:
      - pot = rf(x_i)
      - v = flatten(pot)
      - S = encode_stochastic(v, T)
   b. Reset neuron states:
      - For each neuron j = 1 to N:
        1. P_j = 0, spike_count_j = 0, t_rest_j = 0
   c. For t = 1 to T:
      i. For each neuron j = 1 to N:
         - If t ≥ t_rest_j:
           1. P_j = P_j + W_j • S[:, t]
           2. If P_j > P_rest:
              - P_j = P_j - D
      ii. Lateral Inhibition:
         - If no neuron has spiked yet and max(P_j) > P_th:
           1. winner = argmax_j P_j
           2. For all k ≠ winner: P_k = P_min
      iii. Spike Check:
         - For each neuron j:
           1. If P_j ≥ P_threshold:
              - spike_count_j += 1
              - P_j = P_reset
              - t_rest_j = t + t_ref
   d. Classification:
      - y_pred = argmax_j spike_count_j
      - Store y_pred, y_i, spike_count

End

Results

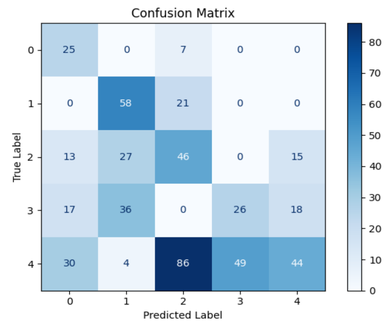

Figure 3: Confusion Matrix

TABLE I
EVALUATION METRICS

Metric	Value
Success rate	0.381
Always-firing	0.151
False positives	0.467
No learning	0.000
Inverse learning	0.259

Table 1: Evaluation Metrics

- The confusion matrix reveals that the Spiking Neural Network (SNN) demonstrates selective success in recognizing certain pattern classes, but also exhibits significant misclassification, particularly for more complex or overlapping patterns. For example, class 1 (second row) is most accurately identified, with 58 correct predictions, but there is notable confusion between classes 2, 3, and 4. Class 4, in particular, is frequently misclassified as class 2 (86 times) and class 3 (49 times), indicating difficulty in distinguishing these patterns.

- The evaluation metrics further quantify these trends. The overall success rate (accuracy) is 0.381, reflecting that just over one-third of the samples are correctly classified. The false positive rate is relatively high at 0.467, and the "always-firing" metric (0.151) suggests that in some cases, multiple neurons spike equally, leading to ambiguous predictions. Notably, the "no learning" metric is 0.000, indicating that the network always produces a response for each input. The "inverse learning" metric (0.259) points to a specific challenge where the network confuses class 4 with other classes and vice versa.

In summary, while the SNN is capable of capturing certain structured trends, its overall classification performance is limited by overlapping class representations and the fixed-weight architecture. These results highlight the need for larger, more diverse datasets and adaptive learning mechanisms (such as synaptic plasticity) to improve discrimination among complex financial patterns.

Discussion and Conclusion

Discussion

- The Spiking Neural Network demonstrated promising ability to recognize clear and well-defined temporal patterns in financial time series data. Using permutation matrices as input, the model leveraged spike-based computation to capture temporal dependencies in an interpretable and energy-efficient manner. This highlights the potential of biologically inspired architectures for structured time series classification.

- However, the network's performance was limited by fixed synaptic weights and a relatively small dataset, resulting in moderate accuracy and confusion among overlapping classes. These limitations suggest that adaptive learning mechanisms and larger, more diverse datasets are essential to improve robustness and generalization in real-world financial applications.

Conclusion

- This study validates the feasibility of applying SNNs to classify structured financial patterns, achieving selective success on the NARMA dataset derived from S&P 500 data. The results demonstrate that SNNs can serve as interpretable and efficient models for temporal pattern recognition in financial analytics.

- To advance this work, future efforts should focus on incorporating online learning rules such as spike-timing-dependent plasticity, expanding datasets, and exploring hybrid models. These enhancements will help overcome current limitations and unlock the full potential of neuromorphic computing in practical financial time series analysis.