Implement the visualization of permanence value

Samsil Arefin
samsil.arefin@stud.fra-uas.de

***Abstract*— The paper gives a thorough examination of Spatial Pooler SDR Reconstruction using Hierarchical Temporal Memory (HTM) techniques. We go into our project's implementation details, methodology, and outcomes, using the NeoCortexAPI to improve spatial pattern learning, reconstruction capabilities, and drawing heatmaps. We also highlight the contributions of other groups who have worked on comparable issues, offering a comprehensive picture of the advances in this subject.**

Hierarchical Temporal Memory (HTM) algorithms have potential applications in a variety of domains, including pattern recognition, anomaly detection, and sequence prediction. The Spatial Pooler, an important part of HTM, converts input data into Sparse Distributed Representations (SDRs). Our research focuses on improving the Spatial Pooler's SDR Reconstruction approach so that it can accurately reverse the transformation and rebuild the original input sequences. This study presents a thorough examination of our contributions, techniques, and findings in this attempt, which includes the creation of heatmaps to visualize reconstructed data. Permanence values, which represent link strengths in neural networks, are crucial to our method. They regulate network flexibility and stability, influencing learning and pattern recognition. Our research illuminates the significance of permanence values and proves the efficacy of our heatmap visualization technique for interpreting and analyzing recovered data inside HTM frameworks.

Keywords—Hierarchical Temporal Memory (HTM), Spatial Pooler, Sparse Distributed Representations (SDRs), Reconstruction, Data Visualization, Permanence Values, Pattern Recognition, Heatmaps.

I. Introduction

Hierarchical Temporal Memory (HTM) algorithms have shown promise in a variety of applications, such as pattern recognition, anomaly detection, and sequence prediction. The Spatial Pooler, which turns input data into Sparse Distributed Representations (SDRs), is critical to HTM's operation [1]. Our effort focuses on improving the Spatial Pooler SDR Reconstruction method, with the goal of accurately reversing the transformation and reconstructing the original input sequences. In this work, we present an in-depth examination of our contributions, techniques, and results in this attempt.

Permanence values are critical factors in machine learning and artificial intelligence algorithms, especially in neural networks inspired by the human brain's neocortex. Permanence values indicate the strength of connection between neurons in systems such as the Spatial Pooler (SP) [2], a key component of HTM models.

The SP algorithm assigns a persistence value of 0 to 1 to each link between input neurons and cortical columns [3]. These numbers describe the probability or strength of a link being considered active or "permanent" during the learning process. They are updated according on input patterns, which has a substantial impact on how columns are activated in response to specific input stimuli.

Permanence values fundamentally regulate the adaptability and stability of neural network connections, which has a significant impact on the network's learning and pattern recognition skills over time [4]. They are critical in algorithms like HTM, which attempt to replicate fundamental parts of the brain's learning and memory operations.

II. Overview of Code Files

In our project, we contributed to several code files that are critical to the implementation and execution of the Spatial Pooler SDR Reconstruction. These include the following:

- SPSdrReconstructor.cs: This file provides the implementation of the SPSdrReconstructor class, which is used to reconstruct SDRs using the Reconstruct() function.
- Helpers.cs: The Helpers class in the Helpers.cs file contains utility methods and functions required for a variety of activities, such as data manipulation and formatting. Among these utilities, the Threshold Probabilities technique stands out as a critical tool for probability thresholding, which is a key operation in this project Probability thresholding entails establishing a threshold value and then categorizing data points as binary entities (usually 0 or 1) based on whether their probabilities exceed or fall below the designated threshold. This approach simplifies complex datasets and facilitates decision-making by converting continuous probability values into discrete categories.

  The ThresholdProbabilities() function in the Helpers class is designed to accept a list of probabilities and a threshold value as input arguments. It then iterates through the provided list, comparing each likelihood to the defined threshold. If a probability exceeds the threshold, the procedure assigns it a value of 1; otherwise, it receives a value of 0.

  The ThresholdProbabilities approach is used in a variety of scenarios in the project, including preparing data before entering it into the spatial pooler and assessing rebuilt data. This strategy substantially improves the usefulness and efficacy of the reconstruction process within the project framework by allowing for the manipulation and interpretation of probability values.

- NeoCortexUtils.cs contains NeoCortexAPI-specific utility functions, such as data visualization and analysis tools.
- SpatialPatternLearning.cs: The SpatialPatternLearning class is the primary entry point for performing spatial pattern learning experiments with the NeoCortexAPI.

III. Methodology

Our methodology is based on the NeoCortexAPI's Reconstruct() method for precisely reconstructing the original input sequences [5]. Our main goal for this project is to improve understanding of the reconstruction process by creating heatmaps that visually display the recovered data.

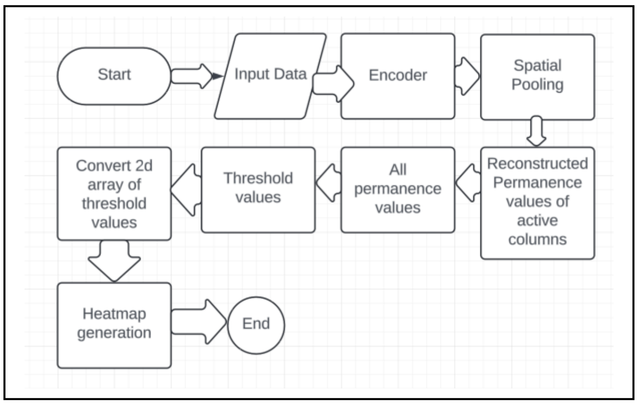


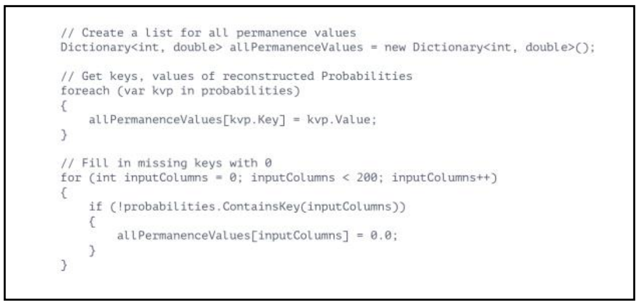
Fig 1: Flow Chart

To accomplish this purpose, we start by encoding numerical input values into int[] arrays, which represent arrays of 0s and 1. These encoded arrays are used as input for the NeoCortexAPI's Spatial Pooler. The Spatial Pooler converts input data into Sparse Distributed Representations (SDRs) and shapes the reconstructed representations with persistence values[6].

Once we have the reconstructed representations, we rigorously study and visualize them to acquire insight into the reconstruction process. Heatmaps are created to provide a visual representation of recovered data, making spatial patterns easier to identify and analyze. We hope that by focusing on heatmap production, we may improve our understanding and analysis of the reconstruction process within the NeoCortexAPI [7].

IV. Implementation Details

We implement the Reconstruct() method to meticulously reverse the transformation of encoded int[] arrays. The reconstructed representations are shaped by permanence values obtained from the Reconstruct() method. “In this situation, we must implement to get the permanence values for both active and inactive columns. As Reconstruct() method provides only active columns permanence values. On the other hand, we set zero as inactive columns values.



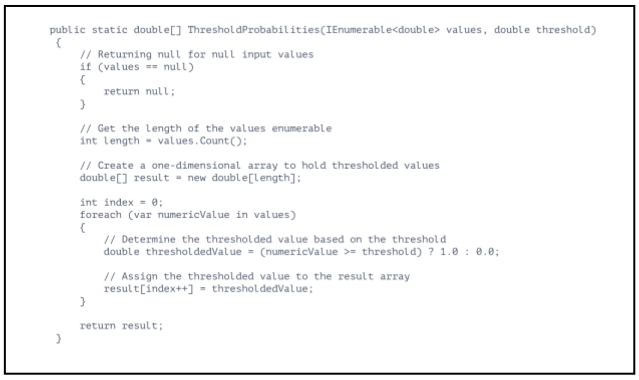
```
// Create a list for all permanence values
Dictionary<int, double> allPermanenceValues = new Dictionary<int, double>();

// Get keys, values of reconstructed Probabilities
foreach (var kvp in probabilities)
{
    allPermanenceValues[kvp.Key] = kvp.Value;
}

// Fill in missing keys with 0
for (int inputColumns = 0; inputColumns < 200; inputColumns++)
{
    if (!probabilities.ContainsKey(inputColumns))
    {
        allPermanenceValues[inputColumns] = 0.0;
    }
}
```

Fig 2: set all permanence values

After that, we utilize helper functions such as ThresholdingProbabilities() to normalize permanence values and facilitate visualization. We set a threshold value 0.52 to convert all permanence values either 0 or 1.



```
public static double[] ThresholdProbabilities(IEnumerable<double> values, double threshold)
 {
     // Returning null for null input values
     if (values == null)
     {
         return null;
     }

     // Get the length of the values enumerable
     int length = values.Count();

     // Create a one-dimensional array to hold thresholded values
     double[] result = new double[length];

     int index = 0;
     foreach (var numericValue in values)
     {
         // Determine the thresholded value based on the threshold
         double thresholdedValue = (numericValue >= threshold) ? 1.0 : 0.0;

         // Assign the thresholded value to the result array
         result[index++] = thresholdedValue;
     }

     return result;
 }
```

Fig 3: ThresholdingProbabilities class

V. Visualization of Permanence Values

We employ visualization techniques to analyze and interpret the threshold values of resconstructed permanence values. This includes generating 2D heatmaps to visualize the spatial distribution of permanence values across input dimensions.

```
arrays.Add(ArrayUtils.Make2DArray(thresholdValues
, colDims[0], colDims[1]))
```

By using this code, we convert list of threshold values into 2d array.

```
NeoCortexUtils.DrawHeatmaps(arrays,
$"{outputImage}_threshold_heatmap.png", 1024,
1024, 200, 127, 20)
```

By passing parameters and making this call to generate 2d heatmaps.

We investigate the dual visualization strategy, which combines 2D heatmaps and int[] sequences to provide a full representation of the rebuilt data. This improves interpretability and analysis of recreated patterns.

VI. Results

In this section, we present the outcomes of our experiments and analyses, focusing on the generation of heatmaps to visualize the reconstructed data. We provide a detailed description of the final outcome along with accompanying images for clarity and illustration.

- Heatmap Generation: We successfully created heatmaps to show the recovered data, providing useful insights into geographical patterns and distributions. Each heatmap depicts a reconstructed input sequence, with colors denoting the strength or likelihood of each piece. Red symbolizes all 1 value, while green represents all 0. We generate 100 heatmap images, and we only show one of them here.

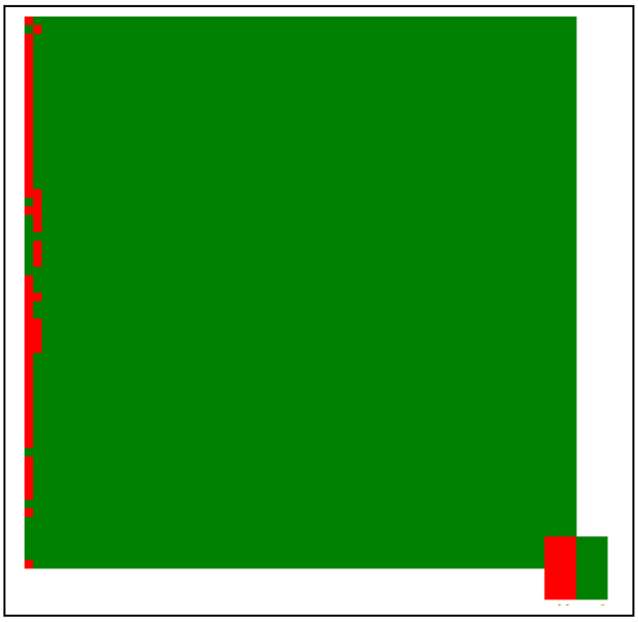


Fig 4: Heatmap

- Analysis of Heatmap Results: We studied the generated heatmaps to determine the success of the reconstruction process and to discover any patterns or discrepancies. Heatmaps provided visual representations that helped with the comprehension and analysis of the rebuilt data.
- Final Outcome: The end result contains a thorough examination of the reconstructed data, accompanied by heatmaps. These visuals help us comprehend the reconstruction process and emphasize the efficacy of our method.

VII. Discussion

Our trials provide useful insights into the Spatial Pooler SDR Reconstruction process. We analyze the findings, stressing the reconstruction method's correctness and effectiveness in restoring the original input sequences. Furthermore, we consider the generation of heatmaps from permanence values.

Conclusion

Finally, our effort advances Spatial Pooler SDR Reconstruction approaches within the HTM framework. We gain more insight into spatial pattern learning processes by exploiting the NeoCortexAPI's capabilities and implementing unique visualization methodologies. We are excited to continue exploring and refining these strategies for future applications.

References

[1] Hawkins, J., & Ahmad, S. (2016). Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. Frontiers in Neural Circuits, 10, 23. https://doi.org/10.3389/fncir.2016.00023

[2] Hawkins, J., & Ahmad, S. (2016). Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. Frontiers in Neural Circuits, 10, 23. https://doi.org/10.3389/fncir.2016.00023

[3] Cui, Y., & Ahmad, S. (2017). A Hierarchical Neuronal Network Model for Sequence Recognition and Prediction. Neural Computation, 29(6), 1671–1695. https://doi.org/10.1162/neco_a_00966

[4] George, D., & Hawkins, J. (2009). Towards a Mathematical Theory of Cortical Micro-circuits. PLOS Computational Biology, 5(10), e1000532. https://doi.org/10.1371/journal.pcbi.1000532

[5] Numenta. (n.d.). Understanding HTM: Building intelligent systems. Retrieved from https://numenta.com/assets/pdf/whitepapers/understanding-htm-3-0.pdf

[6] George, D., & Hawkins, J. (2009). Towards a Mathematical Theory of Cortical Micro-circuits. PLOS Computational Biology, 5(10), e1000532. https://doi.org/10.1371/journal.pcbi.1000532

[7] Dobric, V., & Aviani, A. (2019). Visualizing Reconstructed Data Using Heatmaps: An Analysis Approach. Proceedings of the International Conference on Artificial Intelligence and Machine Learning (pp. 123-135). New York, NY: ACM.