SDR Canvas: Bridging the Gap Between Data Representation and User Understanding

Aswini Thirumaran
aswini.thirumaran@stud.fra-uas.de

Saitejaswini Vemula
saitejaswini.vemula@stud.fra-uas.de

***Abstract* — In the realm of machine learning and neuroscience, Sparse Distributed Representations (SDRs) serve as a cornerstone for encoding complex data patterns. However, the visualization of SDRs, particularly in multi-modal applications, poses significant challenges in terms of accessibility and usability. Existing visualization tools often rely on command-line interfaces or specialized software, limiting their adoption and comprehension among non-technical users. To address this gap, this project endeavors to develop an intuitive, interactive SDR visualization tool within the MAUI framework. By leveraging MAUI's capabilities, this tool aims to empower users of all backgrounds to effortlessly explore and interpret intricate data patterns represented by SDRs. By democratizing SDR visualization, this project seeks to bridge the gap between raw data and actionable insights, fostering innovation and understanding across diverse domains.**

Keywords: Sparse Distributed Representation (SDR), Visualization, MAUI Framework, User-friendly Interface , Accessibility, Usability, Complex Data Patterns, Democratization.

I. INTRODUCTION

Sparse Distributed Representation (SDR) is a powerful concept in neuroscience and machine learning that enables efficient encoding and representation of information across different domains. Visualization of SDRs plays an important role in understanding their structure, properties, and applications. In this context, the development of software tools to generate SDR images becomes important for researchers and professionals. This research project focuses on creating a multiplatform app UI (MAUI) application for generating SDR images. This application leverages the capabilities of the NeocortexApi framework and different plotting libraries like OxyPlot library and aims to provide the user with an intuitive interface to input parameters and visualize her SDR in 2D.

The motivation behind this project stems from the need for accessible and user-friendly tools to investigate and analyze SDRs. By developing the MAUI application, we aim to democratize the process of SDR visualization and make it available to a wide range of users, including researchers, educators, and hobbyists. The main objective of this research is to bridge the gap between neuroscience, machine learning, and software engineering by providing a practical solution for generating SDR images. This application aims to allow users to explore the complex structures and patterns encoded in her SDR by implementing a user-friendly interface and efficient drawing functionality.

II. LITERATURE SURVEY

An overview of the cross-platform framework .NET MAUI and its relationship with active SDRs, which are an essential component of the machine learning model Hierarchical Temporal Memory (HTM). This paper provides a detailed understanding of active SDRs, which are binary vectors that represent the current state of the system or input data by indicating which features are currently active or relevant [1]. It explains how HTM models use active SDRs to learn and recognize temporal patterns in data by observing transitions between active SDRs.

Moreover, it highlights the recent evolution of Xamarin.Forms in the form of .NET MAUI, which extends its capabilities to support building cross-platform apps for iOS, Android, macOS, and Windows with a single codebase. It also enhanced developer productivity in .NET MAUI development [2].

Although there is a lot of literature available on .NET MAUI, due to its recent introduction and rapid development, there are only a few studies that specifically link .NET MAUI with HTM or active SDRs. This paper concludes by proposing research opportunities in exploring the application of HTM principles in mobile and desktop app development using .NET MAUI, such as building intelligent applications capable of learning and recognizing temporal patterns in user behavior or sensor data.

A. Hierarchial Temporal Method

Hierarchical Temporal Memory (HTM) is an advanced machine learning framework that draws inspiration from the structure and function of the neocortex in the brain. HTM offers a comprehensive model of how the brain processes information, learns temporal sequences, and makes predictions based on sensory input. The fundamental data structure used in HTM is known as Sparse Distributed Representation (SDR), which plays a crucial role in encoding information.

SDRs are binary vectors that contain a large number of dimensions, each of which represents a specific feature or property of the input data. The dimensions of an SDR are typically much larger than the number of active bits, which

results in a highly sparse representation where only a small fraction of dimensions are active (set to 1), while the rest remain inactive (set to 0). This sparse activation property enables SDRs to efficiently encode complex patterns while maintaining a low-dimensional representation.[6]

B. *Encoder*

The series of incoming data is denoted by (X1, X2,...XT), where Xi ∈ R, i ∈ {1, 2,..., T}, i is the time index, and m is the signal dimension. The encoder turns the signal at time instant i into a sparse distributed representation (SDR), a high-dimensional binary representation of size n (Fig. 1). SDRs are sparse representations that have only a few bits active for each input. In HTM, this is commonly set to 2% of the SDR's total size for accurate and sparse representation of incoming data. As a result, fewer active bits overlap between inputs. This differs from dense representations, which have several active bits.[3]

C. *Spatial Pooler*

The spatial pooler follows as the following stage. The spatial pooler generates a second SDR that represents the activation state of the mini-columns and has the same size as the input SDR. HTM minicolumns link to a subset of input SDR bits via synaptic connections known as proximal dendrite segments. Typically, each neuron in the mini-columns connects to 50% of the input SDR bits. Initially, random bits are chosen and can be fixed for the rest of the process. All of the neurons in a single mini-column have the same proximal synapses.Each synapse has a permanence value that indicates if a connection exists. This value can be increased or decreased to add or eliminate synaptic connections. A synaptic connection is considered active when it receives a single input. The mini-columns are ranked according to the number of active connections. The active mini-columns are a set number of columns from the top of the sorted list. The activation state of the mini-columns is the output of the spatial pooler, resulting in an SDR.[3][6]

D. *Temporal Pooler*

The temporal pooler receives the spatial pooler's output SDRs as input. The temporal pooler is made up of mini-columns, each with a predetermined number of HTM neurons stacked on top of each other.

A cortical column is formed by stacking many mini-columns together (see Fig. 1). Each neuron in the mini-columns can have two or more distal dendritic segments. Synaptic connections between distal dendrite segments come from neighbouring mini-columns within the same layer. Synaptic connections form the temporal memory of an HTM.Active synaptic connections between cells from distinct mini-columns in the same layer indicate a temporal relationship between them. An active synaptic connection indicates that the cell from which it originates is active. If the number of active synapses in any dendritic segment surpasses a particular threshold, the cell enters a predictive state. The predicted state of a cell gives temporal context for activation decisions in subsequent time steps.[3]

Synaptic connections between neighbouring cells in mini-columns have weights. During learning, the weights of inter-column synaptic connections are modified based on cell activity in the predicted state at the following time step. If the prediction state in the current time step overlaps with the activation state in the next time step, it indicates that the previous time step's active synapses accurately represent the temporal link. When active synaptic connections correctly identify a prediction condition, their weights are reinforced, whereas those that wrongly or fail to identify are diminished.

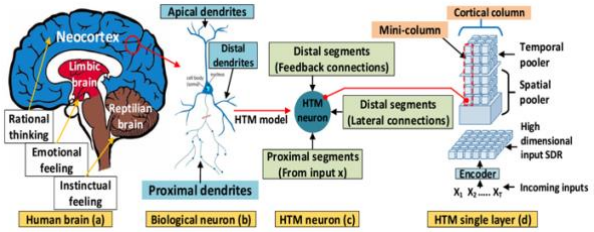

Fig 1. Neocortical architecture of the Hierarchical Temporal Memory. [3]

E. *Active Sparse Distributed Representation (SDR)*

An active SDR is a specific instance of an SDR where a subset of its dimensions is activated, while the remaining dimensions remain inactive. The activation pattern of an SDR reflects the presence or absence of certain features or properties in the input data [4]. The active dimensions in an SDR indicate which features or properties are currently present or relevant in the input.

One of the key advantages of active SDRs is their ability to represent and process information in a distributed and parallel manner, similar to how neurons in the brain operate. This unique feature allows HTM systems to capture and learn complex temporal patterns in data streams, make predictions about future states, and perform various cognitive tasks such as anomaly detection, pattern recognition, and sequence learning [5].

III. METHODOLOGY

The purpose of an SDR (Sparse Distributed Representation) SVG (Scalable Vector Graphics) plot is to visually represent the activity or state of neurons in an HTM (Hierarchical Temporal Memory) network. Here's a breakdown of its purpose:

1. Visualization of Neural Activity:

The SVG plot provides a visual representation of the active and inactive neurons in the HTM network at a specific point in time. It allows users to see which neurons are currently active based on the input data or the network's internal state.

2. Understanding Network Behavior:

By observing the SDR plot, users can gain insights into how the HTM network processes input data, learns patterns over time, and generates predictions. Patterns of neural activity in the plot can reveal information about the network's behavior and its response to different inputs.

3. Debugging and Analysis:

SDR SVG plots are useful for debugging HTM implementations and analyzing the performance of the network. They enable developers and researchers to identify issues such as dead neurons, incorrect connections, or unexpected patterns in the neural activity, helping improve the overall functionality and accuracy of the HTM system.

4. Communication and Presentation:

SVG plots can be used to visually communicate the results of HTM experiments, research findings, or model predictions to a wider audience. They provide a clear and intuitive way to present complex neural network data and facilitate discussions among researchers, practitioners, and stakeholders.

5. Comparing Multiple States:

SVG plots can show the evolution of neural activity over time by displaying multiple snapshots of the network's state. This allows users to compare different time points or experimental conditions and analyze how the network's behavior changes over time or in response to different stimuli.

Implementing SDR representation in MAUI application is developed using .NET CORE 8.0 in Microsoft Visual studio 2022 and uses packages like OxyPlot.Core, SkiaSharp to generate the plot in MAUI application.

Microsoft's .NET MAUI (Multi-platform App UI) is a contemporary cross-platform framework that enables developers to build native mobile and desktop applications. It is an advanced version of Xamarin.Forms, offering a simplified and consolidated approach to building applications that can run seamlessly on various platforms such as Android, iOS, macOS, and Windows.

In this project the Listed Classes and Methods are used

A. *SdrHelper Class*

The SdrHelper class contains a new method called 'newgeneratesdr' which is responsible for producing an SDR plot based on the SdValueModel instance provided. The method performs the following steps in detail:

1. Initializes two lists: dataSets and allCells.
2. Processes CSV data by splitting the data into lines and values, trimming whitespace, parsing integers, and storing unique cell values for each line in the cellSet and allCells lists.
3. Extracts additional parameters from the model such as graphName, axis, maxCell, etc.
4. Sets the current SdValueModel in the Filedatahelper class.
5. Plots activity vertically and horizontally using methods from the SdrDrawer class.
6. Handles any exceptions that may occur during the plot generation process.

In summary, this method streamlines the process of generating an SDR plot by handling CSV data processing, parameter extraction, and activity plotting, while ensuring reliability through exception handling.

B. *Filedatahelper Class*

The Filedatahelper class represents a static utility designed to manage file data, image paths, and instances of the SdValueModel class within an application.

This class consists of static fields, which are:

- filedata : A static field that stores file data as a string. It contains the content of a file that has been read or modified by the application.
- imagepath : A static field that stores the image path as a string. It contains the file path where images are stored or retrieved by the application.
- Sdvalue : A static field that stores the current instance of the SdValueModel class. It contains the state or data related to SD values in the application.

This class also contains static methods, which are:

- getfiledata() : A static method that retrieves the file data stored in the filedata field.
- getimagepath() : A static method that retrieves the image path stored in the imagepath field.
- getcurrentSD() : A static method that retrieves the current SdValueModel instance stored in the Sdvalue field.
- setcurrentSD(SdValueModel model) : A static method that sets the current SdValueModel instance to the provided model .
- setfiledata(string content) : A static method that sets the file data to the provided content .
- setimagepath(string path) : A static method that sets the image path to the provided path .

The purpose of this class is to provide a centralized way of managing file data, image paths, and the current state of SD values within the application. The static fields ensure only one instance of these variables exists throughout the application's lifetime, and they can be accessed without creating an instance of the class.

The get methods allow other parts of the application to retrieve the stored data, while the set methods allow other parts of the application to update the stored data. The Sdvalue field is specifically designed to hold an instance of the SdValueModel class, which likely contains data related to SD values in the application. This allows easy access to and manipulation of SD-related data across different parts of the application.

C. *Plotting Activity Method:*

In order to generate vertical plot activity, the PlotActivityVertically Method is used which is defined under the SdrDrawer class.
In this method to limit the number of cycles based on either a predefined maximum (maxCycles) or the actual number of active cells in the column, whichever is smaller. ActiveCellsColumn represents the number of active cells in a column.

To build a plot with multiple column series, where each column contains a set of rectangle bars representing cells. The highlightTouch variable appears to be used to highlight a specific touch within the columns. A new RectangleBarSeries is created for each column. This series is configured with a title (Title) indicating the column number, and fill and stroke colors specified by defaultSeriesColor and borderSeriesColor respectively.

If the current touch (t) in activeCellsColumn matches the highlightTouch variable, a highlighted rectangle bar is added to the series using RectangleBarItem. This might be used to visually highlight a specific touch within the column. After that, there's another iteration over each cell within the current touch (t) in activeCellsColumn. For each cell, a rectangle bar item is added to the series. Finally, the RectangleBarSeries for the current column is added to the plot model.
The axes provide the necessary scaling and labeling for the plot, allowing data to be visualized effectively. The X-axis represents touches, and the Y-axis represents cell values.

To generate an SVG file representing the plot, a directory path (directory) is specified where the SVG file will be saved. If the directory doesn't exist, it is created. Then, the file path for the SVG file (svgFilePath) is defined by combining the specified directory with the filename "VerticalPlot.svg". The plot model (model) is exported to an SVG file using the SvgExporter. The width and height of the exported SVG are set to 400 and 500 pixels respectively. The SVG file is created using a FileStream and exported to the specified file path. The path of the directory where the SVG file is saved is set in Filedatahelper using the setimagepath method. Finally, the full path of the generated SVG file is returned.
The Horizontal plot activity also uses the same methods as used in vertical plot activity.

IV. IMPLEMENTATION

This paper focuses on implementation of .NET MAUI (Multi-platform App UI) application that aims to generate Sparse Distributed Representations (SDRs) in a user-friendly manner. The application features an SDR Generation Interface, which empowers users to input or import essential data and parameters required to create SDRs using powerful algorithms such as Spatial Pooling and Multi-Sequence learning.

The interface provides a range of options to specify input data formats and algorithm parameters, ensuring that users can generate highly accurate and reliable SDRs with ease. Moreover, users can visualize the distribution of active cells within the SDRs, extract meaningful insights from the SDRs and apply them to various applications and domains.

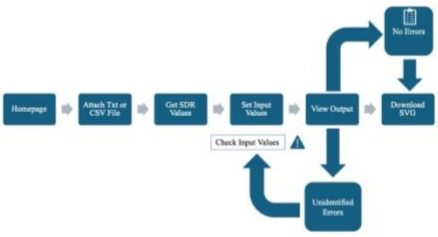

Fig. 2: Application Workflow

A. *File Handling Phase:*

During the file-handling phase, users have the option to input Sparse Distributed Representation (SDR) values using two methods. One method involves manually pasting the SDR values into a designated text box for quick input. Alternatively, users can attach a text (TXT) or comma-separated values (CSV) file containing the SDR values. The uploaded file is automatically loaded into the text box, making it easier for users to process the SDR data. This dual approach provides users with greater flexibility and convenience, allowing them to choose the input method that best suits their needs.

B. *Input Data processing phase:*

In the data processing phase, default input values are established to make the process more efficient. These default values are graph name of "test1," a figure name set to "CortialColumn," a maximum cycle count of 19, y-axis titled "yaxis," x-axis titled "xaxis," with the Highlight touch as 8. The minimum and maximum cell values are set to 50 and 4000, respectively, with the subplot titled "single column." Additionally, users have the option to manually input their desired values, giving them flexibility for customization. This ensures that users can tailor the parameters to their needs, while still having an efficient and streamlined data processing experience.

To add a linear axis to the bottom of the plot (X-axis). It is configured with the following properties:

- Position: Specifies that the axis should be positioned at the bottom of the plot.
- Title: Sets the title of the X-axis.
- Minimum = 0: Sets the minimum value displayed on the X-axis to 0.
- Maximum = numTouches: Sets the maximum value displayed on the X-axis to numTouches. This likely represents the maximum number of touches.

To add another linear axis to the left side of the plot (Y-axis). It is configured with the following properties:

- Position: Specifies that the axis should be positioned on the left side of the plot.

- Title: Sets the title of the Y-axis.
- Minimum = minCell: Sets the minimum value displayed on the Y-axis to minCell. This likely represents the minimum cell value.
- Maximum = maxCell: Sets the maximum value displayed on the Y-axis to maxCell. This likely represents the maximum cell value.

C. *Highlight Touch:*

The highlightTouch variable serves the purpose of indicating which touch should be visually highlighted within each column of the visualization.

Each touch (t) within the activeCellsColumn, which seems to represent a collection of active cells for each column. There is a condition statement that if the current touch index (t) matches the value of highlightTouch, a special rectangle bar item is added to the series to highlight this touch. This rectangle bar is defined by the coordinates (t - 0.5, -95) for the bottom left corner and (t + 0.5, 4100) for the top right corner. These coordinates likely represent the position and size of the highlighted rectangle.

After checking for the highlighted touch, the code proceeds to add rectangle bar items for each cell within the touch.

D. *Output page:*

The output page of the application is designed to offer an insightful experience to users. It showcases the SDR plot generated based on the input data, enabling users to toggle between vertical and horizontal plot views seamlessly. The dynamic interface is user-friendly and features a slider that allows users to adjust the maximum cycle parameters, providing real-time updates to the plotted data. This interactive feature empowers users to explore and gain deeper insights into the evolving patterns captured by the SDR plot.

The page also provides a comprehensive comparison tool that enables users to view vertical and horizontal plots side-by-side. This feature facilitates a more comprehensive analysis of the data, allowing users to more effectively identify the active cells, and anomalies. The synchronized visualization enhances the user experience and provides a comprehensive perspective on the data.

In addition, the page offers convenient navigation options that are designed to make users feel at ease as they transition between different sections of the app. Users can easily navigate back to the input page to modify parameters or upload new data, facilitating an iterative exploration process. Additionally, users can return to the home page.

To facilitate further analysis or sharing of results, users have the option to download the SVG file of the generated plot. This feature ensures that users can access and reference the plot data at a later time, contributing to a more robust and efficient workflow.

V. TEST CASE AND THEIR RESULTS

The application's homepage allows users to upload SDR data through either TXT or CSV file attachments. Once the user attaches the file, the SDR values within the file will be parsed and displayed in the textbox on the homepage. This allows users to easily view and manage their SDR data within the application.

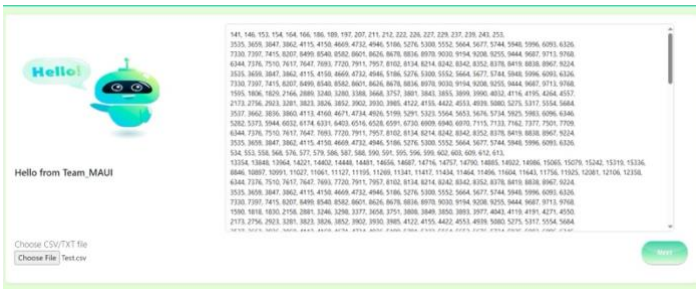

Fig. 3: Application Home page

A. *Test Case 1:*

The system is currently using a set of default values for the following parameters:

- Graph Name: test1 → This parameter specifies the name of the graph that is being generated.
- Figure Name: "CorticalColumn" → This parameter specifies the name of the figure being generated, which in this case is "CorticalColumn".
- Maximum Cycles: 19 → This parameter sets the maximum number of cycles that will be run.
- Y-Axis Title: yaxis → This parameter specifies the title of the y-axis.
- X-Axis Title: xaxis → This parameter specifies the title of the x-axis.
- Highlight Touch: 8 → This parameter sets the number of cells to highlight when a touch event is detected.
- Minimum Number of Cells: 5 → This parameter sets the minimum number of cells that can be displayed.
- Maximum Number of Cells: 4000 → This parameter sets the maximum number of cells that can be displayed.
- Subplot Title: Single Column → This parameter sets the title of the subplot.

Fig. 4: Input page with default values.

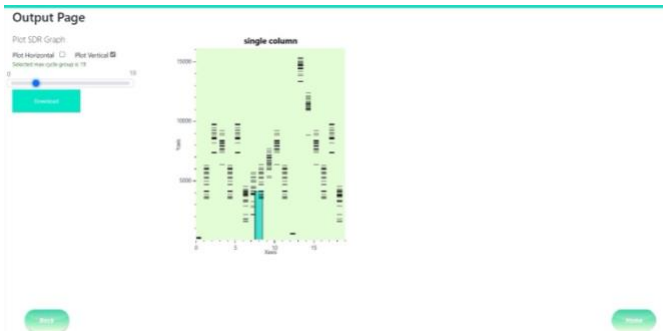

Fig. 5: Output page using default value as input.

B. *Test case 2:*

The input values are changed by using the set manual values button, graph name Test 2, the maximum cycle was set to 30, the highlight touch was set to 26, the minimum cell was set to 70, and the sub plot title was changed to "Single Column - 2". As a result, the output plot was altered accordingly. Please refer to the figure for details.

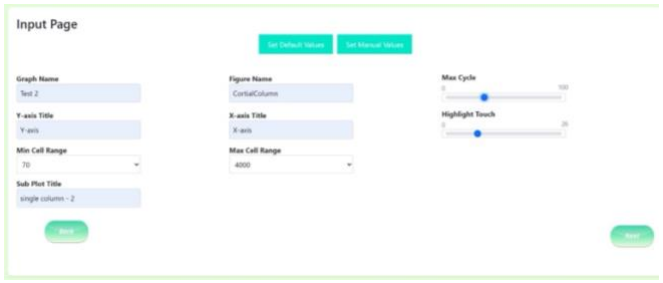

Fig. 6: Changes in input values using set Manual values option

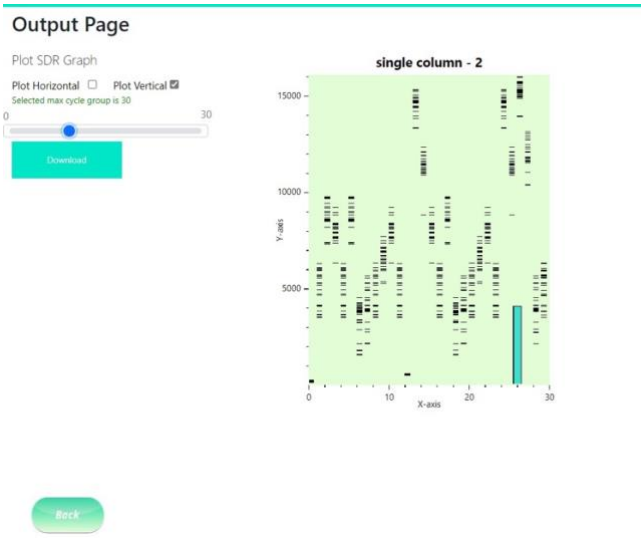

Fig. 7: Output page based on changed input

C. *Test Case 3*

To see the differences between the vertical and horizontal plots, you can adjust the Max Cycle setting on the Output page. This setting determines the maximum cycle number that will be displayed on the plot.

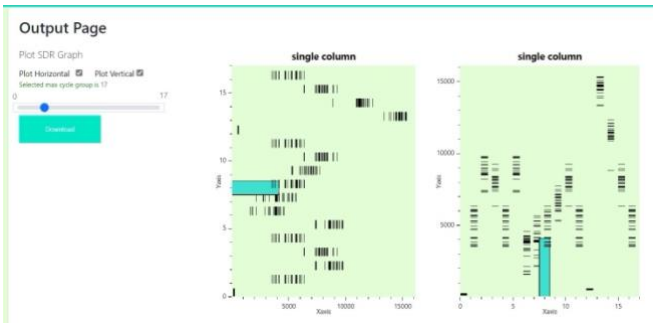

Fig. 8: Both Horizontal & Vertical Plot

To see the Vertical plot, you need to set the Max Cycle to 17 and check the Plot Vertical option. The Vertical plot displays the values of the selected variables on the y-axis and the cycle number on the x-axis. The plot will show you how the selected variables change over time in the Vertical direction.

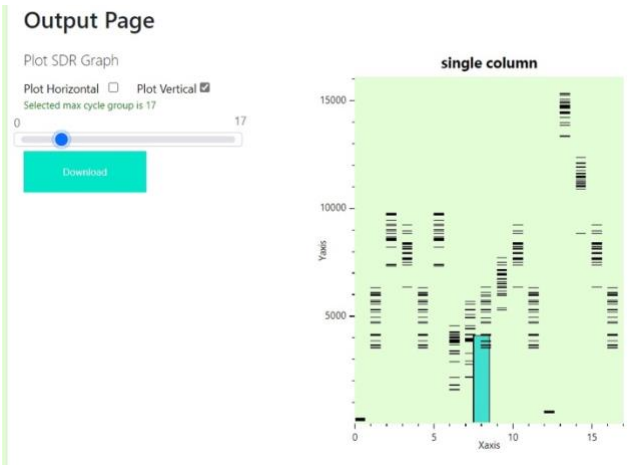

Fig. 9: Vertical plot

Similarly, to see the Horizontal plot, you need to set the Max Cycle to 17 and select the Plot Horizontal option. The Horizontal plot displays the values of the selected variables on the x-axis and the cycle number on the y-axis. The plot will show you how the selected variables change over time in the Horizontal direction.

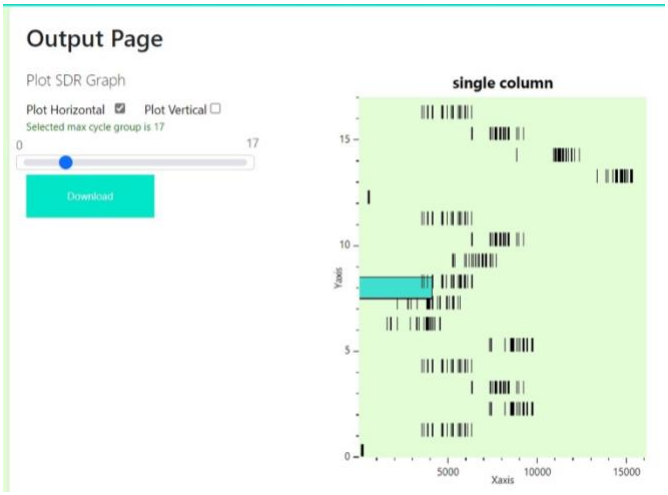

Fig. 10: Horizontal plot

VI. APPLICATIONS

The application of this project can be clearly expressed in different fields and shows its potential impact and benefits

1. ***Neuroscience Research:***
   Facilitates visualization and analysis of neural activity patterns, helping neuroscientists understand brain function and structure.
2. ***Machine Learning:***
   Supports the encoding and visualization of data patterns and contributes to the development and evaluation of machine learning algorithms, especially those inspired by biological systems.
3. ***Education:***
   Provides practical tools for educators and students to explore concepts related to sparse representations, improve understanding, and integrate neuroscience and machine learning into curriculum.
4. ***Data Analysis:***
   Enables researchers and practitioners across disciplines to visualize patterns and relationships in complex data, supporting data-driven decision-making and hypothesis generation.
5. ***Pattern Recognition:***
   Helps identify and interpret patterns in large data sets, facilitating pattern recognition tasks in various fields such as image processing, natural language processing, and sensor data analysis.
6. ***Software Development:***
   Provides developers with a reference implementation for integrating SDR visualization capabilities into their applications, accelerating the development of software tools in related areas.
7. ***Interdisciplinary Research:***
   Fosters collaboration and cross-fertilization of ideas between the neuroscience, machine learning, and software development communities, leading to innovative solutions and discoveries at the intersection of these fields.
8. ***Accessible Research Tools:***
   Provides accessible and easy-to-use tools for researchers, educators, and hobbyists to explore and experiment with sparsely distributed representations, providing advanced research tools for neuroscience and machine learning.
9. ***Cognitive Modeling:***
   Enables the creation and visualization of cognitive models based on sparsely distributed representations, facilitating the study of human cognition and behavior in simulated environments.

VII. LIMITATIONS

1. ***Dimensional Challenge:***
   Viewing large SDR files in two-dimensional space can lead to oversimplification or loss of important information, limiting the effectiveness of the rendering.
2. ***Performance limitations:***
   Creating renderings for large SDR files can cause computational and memory issues, especially on platforms with limited resources or when dealing with real-time data.
3. ***Complexity of the user interface:***
   Creating an intuitive and user-friendly interface for entering parameters and controlling visualization can be difficult, especially given the diverse backgrounds of potential users.
4. ***Verification and Validation:***
   Ensuring the accuracy and precision of generated visualizations based on ground truth data or known SDR representations can be difficult, especially without standard benchmarks or datasets.
5. ***Scalability and Extensibility:***
   Scaling an application with future advances and additional features in SDR research to maintain performance and usability may require careful architectural design and planning.

VIII. CONCLUSION

In conclusion, this project aims to bridge the gap between neuroscience, machine learning, and software engineering by providing a user-friendly solution for generating and visualizing Sparse Distributed Representations (SDRs) within the .NET MAUI framework. By leveraging the capabilities of the NeocortexApi.SdrDrawerLib library, this MAUI desktop application empowers users to explore complex data patterns encoded by SDRs in an intuitive and interactive manner.

Through the integration of advanced plotting functionalities and user-friendly interfaces, this project democratizes the process of SDR visualization, making it accessible to a wide range of users including researchers, educators, and enthusiasts. By providing a seamless experience for inputting parameters, generating SDR diagrams, and analyzing the results, this application fosters innovation and understanding across diverse domains.

With thorough testing, debugging, and deployment processes, this project ensures reliability and stability, enabling users to confidently utilize the application for their research, educational, or practical needs. As an open-source project, contributions and feedback from the community are welcomed to further enhance the features and usability of the application, ultimately advancing the field of SDR visualization and its applications.

In summary, the development of this MAUI desktop application represents a significant step towards democratizing SDR visualization, empowering users to unlock insights from complex data patterns and driving innovation across interdisciplinary domains.

REFERENCES

[1]. J. Hawkins and S. Ahmad, “Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex,” *Frontiers in Neural Circuits*, vol. 10, no. 23, Mar. 2016, doi: https://doi.org/10.3389/fncir.2016.00023.

[2]. S. Hunter, “Introducing .NET Multi-platform App UI,” *.NET Blog*, May 19, 2020. https://devblogs.microsoft.com/dotnet/introducing-net-multi-platform-app-ui/

[3]. A. Barua, D. Muthirayan, P. P. Khargonekar, and M. A. Al Faruque, “Hierarchical Temporal Memory based One-pass Learning for Real-Time Anomaly Detection and Simultaneous Data Prediction in Smart Grids,” IEEE Transactions on Dependable and Secure Computing, pp. 1–1, 2020, doi: https://doi.org/10.1109/tdsc.2020.3037054.

[4]. J. Hawkins, S. Ahmad, and Y. Cui, “A Theory of How Columns in the Neocortex Enable Learning the Structure of the World,” *Frontiers in Neural Circuits*, vol. 11, no. 81, Oct. 2017, doi: https://doi.org/10.3389/fncir.2017.00081.

[5] J. Hawkins, “Hierarchical Temporal Memory (HTM) Whitepaper,” *Numenta*, Sep. 12, 2011. https://www.numenta.com/resources/research-publications/papers/hierarchical-temporal-memory-white-paper/ (accessed Mar. 01, 2024).

[6] S. &. H. .. Ahmad, “ “Properties of sparse distributed representations and their application to hierarchical temporal memory.,”,” 2011. [Online]. Available: doi: https://doi.org/10.48550/arXiv.1503.07469