**Methodology**

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.

Hierarchical Temporal Memory (HTM)

In Hierarchical Temporal Memory (HTM), Sparse Distributed Representation (SDR) is used to represent data. SDRs are binary vectors with many dimensions, with each dimension corresponding to a specific feature or property of the input data. Only a small fraction of dimensions are active (set to 1) in any given SDR, while the rest are inactive (set to 0). This sparse activation property enables SDRs to encode complex patterns while maintaining low-dimensional representations efficiently.

Active Sparse Distributed Representation (SDR)

An active SDR is a specific instance of an SDR in which a subset of its dimensions is set to 1, and the remaining dimensions are set to 0. The activation pattern of an SDR represents the presence or absence of certain features or properties in the input data. The active dimensions in an SDR indicate which features or properties are currently present or relevant in the input.

Purpose of SDR SVG

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.

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 is 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. `maxCycles`: This seems to be a variable representing some maximum number of cycles.`activeCellsColumn.Count`: This seems to be accessing the `Count` property of the `activeCellsColumn` collection. This might represent 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 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.

To generates 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.

Implementation:

This project implements the

Start Page

View Output

Download SVG

Set input values

Get SDR Values

Attach txt or csv file