*Improve UnitTests for Temporal Memory Algorithm*

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Syed Mostain Ahmed  Email: syed.ahmed@stud.fra-uas.de | Name: Farjana Akter  Email: farjana.akter@stud.fra-uas.de | Name: Md. Shamsir Doha  Email: mohammad.doha@stud.fra-uas.de | Md. Nabuat Al Jahid  md.jahed@stud.fra-uas.de |

*Abstract*—Temporal memory algorithms have gained popularity as a promising approach for modeling temporal sequences in machine learning. This project aims to improve the unit test for the given temporal memory algorithm, which is based on the principles of the cortical column and the neocortex. The algorithm uses a sparse distributed representation of data and incorporates temporal context to predict future values in a sequence. We implemented improvements to the existing unit test, including the addition of more test cases with varying complexity and the implementation of cross-validation techniques for better evaluation of the algorithm's performance. We also optimized the implementation of the algorithm for improved efficiency and scalability.

Keywords— Temporal memory algorithm, Cortical column, Neocortex, Sparse distributed representation

# Introduction

Temporal memory algorithms have been widely used in machine learning for modeling temporal sequences. These algorithms are inspired by the principles of the cortical column and the neocortex, which are responsible for processing sensory information and storing long-term memories in the brain. The Temporal Memory algorithm is a well-known implementation of these principles, which has been used in various applications such as natural language processing, anomaly detection, and stock price prediction.

However, accurately evaluating the performance of the Temporal Memory algorithm can be challenging, especially when dealing with complex and noisy data. Therefore, improving the unit test for this algorithm is crucial for ensuring its accuracy and reliability. In this project, we propose several improvements to the existing unit test for the Temporal Memory algorithm. These improvements include the creation, removal, and update of synapses in distal segments, growth of new dendrite segments, activation of cells in columns, and detection/handling of duplicate active columns.

We also implemented learning and recalling patterns of sequences with different sparsity rates and the ability to initialize Temporal Memory with custom parameters such as the number of cells per column and number of column dimensions. Additionally, we adapted segments and increased the permanence of active synapses, limited the number of active cells per column, and retrieved winner cells from Temporal Memory Compute. Furthermore, we implemented least used cell selection and correct initialization of Connections object and used different parameters for existing unit tests to reinforce testing.

Overall, our project aims to enhance the reliability and accuracy of the unit test for the Temporal Memory algorithm, enabling more accurate evaluation of its performance. This improvement can help advance the development of more robust and reliable algorithms for modeling temporal sequences, benefiting various domains and applications.

# Methods

In this section, we describe the approach we followed to evaluate the performance of our Temporal Memory algorithm. Our objective was to test the algorithm's ability to learn and predict spatio-temporal patterns in a dataset. To achieve this, we performed a series of experiments using unit tests, which allowed us to systematically evaluate the algorithm's behavior under different conditions. We first describe the setup of our experiments, followed by a detailed explanation of the unit tests we designed and the results obtained.

* *Testing New Segment Growth when multiple matching segments found* []: The purpose of this test is to verify the growth of a new dendrite segment when multiple matching segments are found. The TemporalMemory object is initialized and a set of default parameters are applied to create a Connections object. The TemporalMemory is then initialized with the Connections object. Next, a set of active columns and cells are created, and multiple matching segments are created in the Connections object for the first active cell. The TemporalMemory is then instructed to compute based on the active columns, and the test asserts that a new segment is grown for the active cell with two synapses. This test is important because it verifies the ability of the TemporalMemory to dynamically adapt to changing input patterns and grow new dendrite segments when necessary.
* *Testing value of Synapse Permanence Updating When Matching Segments are Found*: The purpose of the Test Synapse Permanence Update When Matching Segments Found unit test is to verify that the permanence of synapses in a matching dendritic segment are updated when the segment is activated. Firstly we initialize the TemporalMemory and Connections object. The Parameters object is used to set the permanence decrement parameter, which is applied to the Connections object. In the test, two sets of columns and cells are used: the previous active columns and cells, and the current active columns and cells. Two matching segments are created in the Connections object, each containing synapses to the previous active cells and a synapse to a different cell. The previous active columns are then computed using the TemporalMemory object, followed by the current active columns. The test then verifies that the synapse permanence of the matching synapses is updated correctly. In a nutshell, this unit test verifies that the TemporalMemory algorithm can update the permanence of synapses in matching dendritic segments correctly.
* *Testing TemporalMemory Algorithms initialization*: TestCellsPerColumn [], TestCustomDimensionsAndCells [], and TestColumnDimensions [] are unit tests written to test the initialization of the TemporalMemory class with custom parameters. The first test checks if the class initializes correctly with a custom number of cells per column, while the second test checks if it initializes correctly with custom column dimensions and cells per column. The third test checks if the class initializes correctly with a custom number of column dimensions. To perform the tests, the TemporalMemory class is initialized with a Connections object and the Parameters object, which contains the default parameters for the temporal memory algorithm. The custom parameters are then set using the Set() method of the Parameters object and applied to the Connections object using the apply() method. The TemporalMemory class is then initialized using the initialized Connections object. To check if the initialization was successful, the number of cells in each column is counted and compared to the expected value using the Assert.AreEqual() method. If the number of cells is equal to the expected value, the initialization is considered successful. These tests help to ensure that the TemporalMemory class is properly initialized with the desired custom parameters, which is necessary for the proper functioning of the temporal memory algorithm. Note that this an extension of an original existing test method [].
* *Testing Recycle Least Recently Active Segment To Make Room For New Segment*: We enhanced this existing test [] by giving different various data by DataRow attribute. This tests the behavior of the Temporal Memory algorithm in recycling the least recently active segment to make room for a new segment. The test method is parameterized with different sets of previously active and currently active columns to test the algorithm's behavior in different scenarios. The Temporal Memory object is initialized with a set of default parameters, and the connections are set up with a fixed number of cells per column and a maximum number of segments per cell. The test method then simulates previous and current sets of active columns using the Temporal Memory object's Compute method. The test asserts that the least recently active segment is replaced by the new segment after exceeding the maximum number of segments per cell. The test verifies that the new segment's synapses are disjoint with the replaced segment's synapses by checking that the presynaptic cells of the synapses in the old and new segments are not shared. Overall, this test method ensures that the Temporal Memory algorithm's behavior in segment recycling is consistent and operates correctly in different scenarios.
* *Testing New Segment Add Synapses To All Winner Cells*: This is also an existing test [] enhanced by us to test it against different dataset. This test, named "TestNewSegmentAddSynapsesToAllWinnerCells" [], verifies the behavior of the algorithm when new segments are added to winner cells. The test is parameterized with the number of previously active columns and the number of currently active columns. The method is initialized as usual. The method creates two arrays, previousActiveColumns and activeColumns, which represent the columns that were previously active and the currently active columns, respectively. The method then computes the TM algorithm on the previousActiveColumns and activeColumns arrays, and retrieves the winner cells from the compute cycle. The method asserts that the number of previous winner cells matches the number of previously active columns and that the number of current winner cells matches the number of currently active columns. The method then retrieves the distal dendrites and synapses for the first winner cell in the current winner cells list. The method asserts that the number of segments is 1 and that the number of synapses is 4, which is the default value for the maximum number of new synapses per segment. Finally, the method retrieves the presynaptic cells for each synapse and sorts them in ascending order. The method asserts that the presynaptic cells match the previous winner cells in the same order, which verifies that the new segment has added synapses to all of the previous winner cells.
* *Testing Destroy Weak Synapse On Wrong Prediction* []: This is one of the existing tests [] we modified with different datasets to ensure the tests true potential in testing the feature of Temporal Memory algorithms behaviour in destroying weak synapses. Our modified test checks if weak synapses are correctly destroyed when an incorrect prediction is made with different datasets. The method takes in a double value for the permanence of the weak synapse and creates a TemporalMemory object along with its required connections and parameters. It then sets up a previous active column, a set of previous active cells, and an active column with an expected active cell. A distal dendrite segment is created with synapses connected to the previous active cells and a weak synapse connected to the fourth previous active cell with the specified permanence. The method then calls the Compute method of the TemporalMemory object with the previous active column and the active column as input. After the computation, the method asserts that the distal dendrite segment has correctly destroyed the weak synapse, leaving only the strong synapses. The unit test method is executed with various values of weakSynapsePermanence using the [DataRow] attribute. This allows for multiple test cases to be executed in a single test method.
* *Test Adding Segment To Cell With Fewest Segments* [] *:*This existing test [] was previously implemented with only one set of data but we now modified it to run with three different datasets. The purpose of the test is to verify the behavior of the Temporal Memory algorithm under various conditions. The test checks if adding a new distal segment to the cell with the fewest segments grows the segment on the correct cell. The test is repeated 100 times with different random seeds to ensure the behavior is consistent across multiple runs. The test uses a mock Connections object and sets up a Temporal Memory object with default parameters. It then creates an array of previous active columns and cells, an array of currently active columns, and an array of non-matching cells. Two new distal segments are created on non-matching cells and connected to two of the previous active cells. The Temporal Memory algorithm is then run, and the number of segments, number of segments on specific cells, and number of synapses on each segment are checked for correctness. The test then checks if the segment was grown on the cell with the fewest segments and ensures that the correct columns are activated. The test utilizes the DataRow attribute to test the method with different input data, in the form of weak synapse permanence values or random seeds. Finally, the test checks whether the distal segment grew on both cell 1 and cell 2 at least once during the 100 runs.
* *Test Adapt Segment To Max* []*:* We modified the exiting TestAdaptSegmentToMax [] to test the ability of the TemporalMemory class to adapt the permanence value of a synapse in a distal dendrite segment to the maximum value specified in the HTM configuration parameters with different values with the help of [DataRow] attribute. The test is performed with different initial permanence values of the synapse, and the expected permanence value after adaptation is also specified in the test. The test method creates a new instance of the TemporalMemory and Connections classes, initializes them with default parameters, and creates a new distal dendrite segment with a synapse connecting it to a cell. The AdaptSegment method of the TemporalMemory class is then called with the specified parameters to adapt the permanence value of the synapse to the maximum value. The test method asserts that the permanence value of the synapse is equal to the expected value within a tolerance of 0.1. The test is repeated with the same segment and cell, and the AdaptSegment method is called again to ensure that the permanence value remains at the maximum value. The test method again asserts that the permanence value of the synapse is equal to the expected value within a tolerance of 0.1. So, this test method verifies that the TemporalMemory class is able to correctly adapt the permanence value of a synapse to the maximum value specified in the HTM configuration parameters.

# Results

This Part of the text describes results of your works. There can only be mentioned references, MUST point back to Methods and Intro chapter. No more external references.

Code examples must be provided to demonstrate how to use the algorithm/module. Provide a reference to more unit tests, which show the same in more detail. Also provide all diagrams with comments and reference to unit tests, which generate diagrams.

# Discussion

Conclusion of your work should be precise and concise. How was the project, what is done, what is the result... There can be discussion on further work and direction.

# Ease of Use

## Selecting a Template (Heading 2)

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the A4 paper size. If you are using US letter-sized paper, please close this file and download the Microsoft Word, Letter file.

## Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

# Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

## Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

## Units

* Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as “3.5-inch disk drive”.
* Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
* Do not mix complete spellings and abbreviations of units: “Wb/m2” or “webers per square meter”, not “webers/m2”. Spell out units when they appear in text: “. . . a few henries”, not “. . . a few H”.

may use the solidus ( / ), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

*a**b* 

Note that the equation is centered using a center tab stop. Be sure that the symbols in your equation have been defined before or immediately following the equation. Use “(1)”, not “Eq. (1)” or “equation (1)”, except at the beginning of a sentence: “Equation (1) is . . .”

## Some Common Mistakes

* The word “data” is plural, not singular.
* The subscript for the permeability of vacuum **0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter “o”.
* In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
* A graph within a graph is an “inset”, not an “insert”. The word alternatively is preferred to the word “alternately” (unless you really mean something that alternates).
* Do not use the word “essentially” to mean “approximately” or “effectively”.
* In your paper title, if the words “that uses” can accurately replace the word “using”, capitalize the “u”; if not, keep using lower-cased.
* Be aware of the different meanings of the homophones “affect” and “effect”, “complement” and “compliment”, “discreet” and “discrete”, “principal” and “principle”.
* Do not confuse “imply” and “infer”.
* The prefix “non” is not a word; it should be joined to the word it modifies, usually without a hyphen.
* There is no period after the “et” in the Latin abbreviation “et al.”.
* The abbreviation “i.e.” means “that is”, and the abbreviation “e.g.” means “for example”.

An excellent style manual for science writers is [7].

# Using the Template

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

## Authors and Affiliations

**The template is designed for, but not limited to, three authors.** A minimum of one author is required for all report articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

### For papers with more than three authors: Add author names horizontally, moving to a third row if needed for more than 8 authors.

### For papers with less than three authors: To change the default, adjust the template as follows.

#### Selection: Highlight all author and affiliation lines.

#### Change number of columns: Select the Columns icon from the MS Word Standard toolbar and then select the correct number of columns from the selection palette.

#### Deletion: Delete the author and affiliation lines for the extra authors.

## Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is “Heading 5”. Use “figure caption” for your Figure captions, and “table head” for your table title. Run-in heads, such as “Abstract”, will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed.

## Figures and Tables

For adding object other than text (tables, equations, graphs, figures, code…), **there must be at least one cross reference** to it. Figure 1 is an example

#### Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1”, even at the beginning of a sentence.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)



Figure 1 Example Figure Caption

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

## Code References:

Referencing Code in your text should be avoided unless necessary. In such cases it can be inserted as a listing as shown in **Error! Reference source not found.**

Listing 1 Code Reference Example

Console.WriteLine(“Referencing code”, var);

// using tab can be replaced with 4 spaces

Do not pass code as image. When referring to variable in **Error! Reference source not found.**, italics should be used for example *var.* Code flows and logic should be presented better as Graph or Diagram instead of words.

Code Block which is too big to put in the textbox can be reference as Listing 2.

Listing 2 Unit Test [EncodeDateTimeTest](https://github.com/ddobric/neocortexapi/blob/0348ffb99739ddf8c8c3a875f8162a18073938ca/source/UnitTestsProject/EncoderTests/DateTimeEncoderExperimentalTests.cs#L34-L49)

public void EncodeDateTimeTest(int w, double r, …)

{

…

DateTimeEncoderExperimental encoder = new…

var result = encoder.Encode(input);

…

Assert.IsTrue(result.SequenceEqual(expected…

}

##### Acknowledgment *(Heading 5)*

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. K. Elissa, “Title of paper if known,” unpublished.
5. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
7. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

**This report template contains guidance text for composing and formatting technical reports. Please ensure that all template text is removed from your report prior to submission to the examination office. Failure to remove template text from your paper may result in your paper being degraded.**

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an MSW document, this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord “Format” pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.