Information Technology Course
Module Software Engineering
by Damir Dobric / Andreas Pech

# Analyse Image Classification Use Encoder Output Images

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

Niloy Sarker
niloy.sarker@stud.fra-uas.de

***Abstract*—The method of categorizing and labeling bunches of pixels or vectors inside a picture utilizing particular rules is known as image classification. It is the foremost significant component of advanced image analysis. The objective of this paper is to see into an approach to image classification that's based on an open Hierarchical Temporal Memory (HTM) form algorithm. The HTM Cortical Learning Algorithm (HTM CLA) Spatial Pooler (SP) could be a learning calculation propelled by the neocortical framework. Its objective is to comprehend the spatial design by producing a Sparse Distributed Representation code from the input (SDR). HTM may be a cognitive learning framework and a progressive naturally propelled show of the neocortex's structure that's utilized to maximize basic affecting parameters of the HTM demonstrate in arrange to anticipate more accurately when connected to encoder output pictures. We performed a set of tests in this paper to decide the most excellent parameter combination for accomplishing great and exact likeness between encoder output pictures.**

***Index Terms*—Spatial Pooler, Hierarchical Temporal Memory, Sparse Distributed Representation, neocortex.**

## I. Introduction

For decades, image classification and recognition have proven to be difficult tasks in artificial intelligence. Image classification is the process of categorizing images into one of several predefined classes. The system must be able to extract critical feature information that, when combined with training data, allows the system to recognize the item in the image. As a result, picture categorization can employ effective knowledge representation strategies. Hierarchical temporal memory (HTM) is used as a learning technique in this paper.

Hierarchical temporal memory (HTM) [1] is a machine learning algorithm inspired by the neocortex, the human brain's center for higher brain functions, and designed to learn sequences and predict spatiotemporal data. It should be capable of producing generalized representations for similar inputs in its idealized form. HTM can be used in edge computing devices to perform intelligent data processing. HTM semantically encodes input temporal data generated from various data sources as a sparse array known as sparse distributed representation (SDR). This encoded array is subjected to spatial pooling, which normalizes/standardizes the input data from various sources into a sparse output vector or mini-columns (columns of pyramidal neurons) with defined size and sparsity.

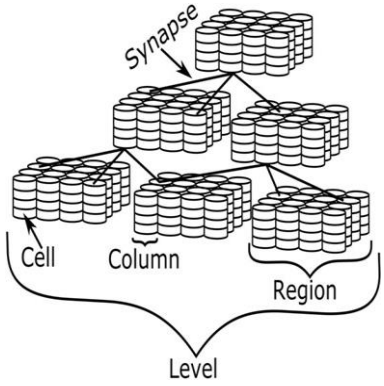



Fig. 1. Depiction of HTM, showing the various levels of detail Source [1]

## II. Hierarchical Temporal Memory(HTM)

A HTM network is built in the form of a tree-shaped hierarchy. Although uncommon, multiple parents per node are permitted, allowing the parents' receptive fields to overlap. Its structure is similar to that of cortical mini columns in that an HTM region is made up of numerous columns, each of which contains a large number of cells. A level consists of one or more regions. Levels are stacked hierarchically in a tree-like structure to build the entire network. In HTM, synapses are used to make connections, with proximal and distal synapses used to produce feedforward and adjacent connections, respectively, as shown in Figure 1. [2] It is possible to mix and match multiple HTM networks. Hierarchical organization is advantageous in terms of efficiency. Because patterns learned at each level of the hierarchy are reused when combined in interesting ways at higher levels, training time and memory usage are reduced significantly. HTMs automatically learn the best possible representations at each level based on the statistics of the input and the number of resources available. Giving a level more memory results in larger and more sophisticated representations, implying that fewer hierarchical levels are required. When you give a level less memory, it will build smaller and simpler representations, which may necessarily require the use of more hierarchical levels.

As shown in Figure 2, HTM has distinct progression levels. As crude information is entered into the framework at lower levels, it is prepared, and more specific data of the information is exchanged to the higher layers of the HTM. This implies

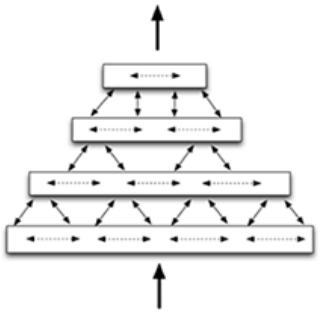


Fig. 2. Simplified diagram of HTM hierarchy [3]

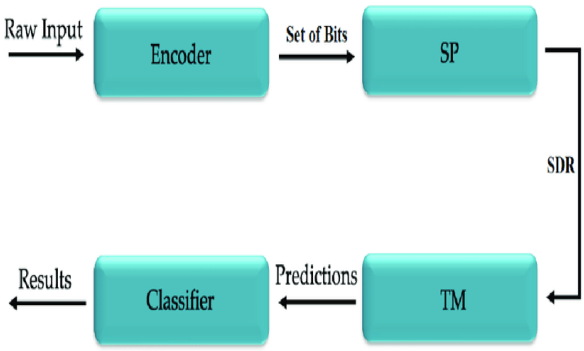



Fig. 3. HTM-network-topology [4]

that the higher the layer of the chain of command, the more prepared information and complex highlights are represented. In other words, lower-level hubs deal with straightforward events that change quickly and involve smaller spatial areas, whereas higher-level hubs deal with numerous events that were detected by lower-level hubs and are thus affected by a larger run of information. Higher level hubs, in particular, detect more complex causes, specifically designs of designs, which progress at a slower rate and can be found in larger spatial zones.

Figure 3 depicts how HTM operates from the beginning to the end of the method. The various components of the framework are depicted one by one as following. In any case, we did not use Transient Memory in this Extraordinary Extend. Input data from a sensor or information source, which can also be continuous and global, is semantically encoded into an inadequate cluster known as inadequate dispersed representation (SDR). This sparse cluster of encoded information enters the framework from the HTM's lowest layer.

## III. SPATIAL POOLER MODEL

In an HTM-System, the Spatial Pooler receives data from encoders and sends it to the Transient Memory. The primary goal is to convert the parallel data into a semantically similar SDR with predetermined sparsity. It consists of a series of mini columns arranged in a grid. A mini-column is made up of a collection of HTM Neurons that all have the same open area. The responsive field specifies which input bits can be used by a scaled down column. After initialization, the SP calculation for each modern input repeats through three stages:

1. Enhancing Coverage Calculation
2. Adaptation through determination as a means of learning.
3. Inhibition.

## IV. SPARSE DISTRIBUTED REPRESENTATIONS (SDR)

SDRs correspond to neocortical active neurons. An SDR is a massive array of bits, the vast majority of which are turned off (0s) and only a few are turned on (1s) (1s). The SDR is literally an array of 0's and 1's, and because the HTM concept is attempting to build a system based on the same principles of brain function, the SDRs are a mathematical representation of these sparse signals, which will likely contain around 2

## V. ENCODER

Hierarchical Temporal Memory (HTM) provides a versatile and organically precise system for forecasting, classifying, and detecting anomalies in a wide range of information types (Hawkins and Ahmad, 2015). HTM frameworks necessitate data input using Meager Distributed Representations (SDRs) (Ahmad and Hawkins, 2016). SDRs differ from standard computer representations such as ASCII for content in that meaning is encoded directly into the representation. An SDR is made up of a large number of bits, the majority of which are zeros and some of which are ones. Because each bit has a few semantic meanings, if two SDRs have more than a few overlapping one-bits, those two SDRs have similar meanings. Any data that can be converted into an SDR can be used in a wide range of applications that employ HTM systems. Thus, the first step in using an HTM system is to convert a data source into an SDR using a device known as an encoder. The encoder converts the information's native format into an SDR that can be fed into an HTM framework. The encoder is responsible for determining which yield bits should be ones and which should be zeros for a given input value in order to capture the critical semantic properties of the information. Comparable input values should result in highly overlapping SDRs. [5]

### *A. DateTime Encoder*

Before we get to DateTime Encoder, let's go over some important data-usage parameters. [7]

1. Range (R)

It is the difference between the highest and lowest values. In the case of DateTime Encoder, it is always 12 in the case of the month, 31 in the case of the day, 10 in the case of the year, 24 in the case of an hour, 60 in the case of minutes, and 60 in the case of seconds.

2.Width (W)

The width bit is always ON. The width of the output signal is determined by the number of bits used to encode a single value. To avoid centering issues, it should always be an odd

```
{
    { "W", 21},
    { "N", 1024},
    { "MinVal", now.AddYears(-20)},
    { "MaxVal", now},
    { "Periodic", false},
    { "Name", "DateTimeEncoder"},
    { "ClipInput", false},
    { "Padding", 5},
});
```

Fig. 4. Parameters of DateTime Encoder [7]

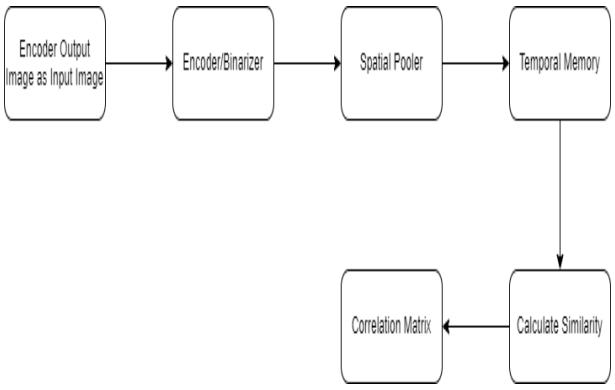



Fig. 5. HTM image classification

number. The algorithm defines it as 'W.' Simply put, width refers to the number of ones in a single encoder.

3. Radius (r)

Non-overlapping representations exist for two inputs separated by more than the radius. In general, two inputs separated by less than the radius will overlap in at least some of their bits. It is known as 'Radius.' The radius determines how many bits overlap. Overlapping= radius – 1.

4. Maximum and Minimum Values

It constrains the input value to fall between the maximum and minimum values. They are denoted by the terms 'MinVal' and 'MaxVal.' Because the date and time are always within the global defined range, it is always fixed in DateTime Encoder. 5. Output bit count (n)

It is the total number of bits required to define an array based on the input data. It is defined by the letter 'n.' n = width * (radius / range)

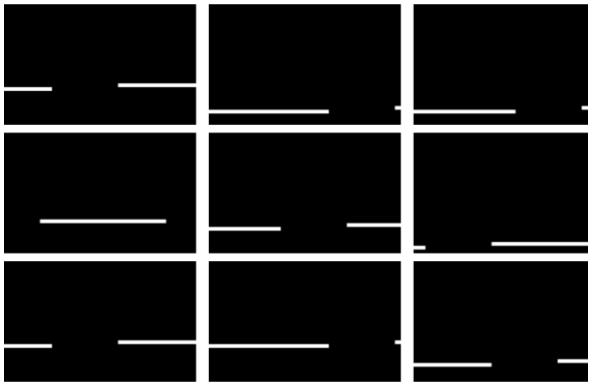
DateTime encoders encode a date and time using the encoding parameters specified in their constructor. A DateTime object is used as the input to a date encoder. [6]

## VI. Methodology

Previously, the image classification project was implemented in C.NET Core in the Microsoft Visual Studio 2022 IDE (Integrated Development Environment). The goal of this project is to create a program that uses the existing solution as a library to determine the spatial pooler parameter values that influence image learning, resulting in the best correlation matrix based on HTM learning, as shown in Figure 5.

### A. Input Dataset

This section describes the reference to the already implemented methodology for image classification, as it is used in this project for image learning and thus for further experiments. This project makes use of the DateTime encoder output images dataset. Figure 6 shows how we use different dates and times for different months.



Plot

Fig. 6. DateTime Encoder Output image dateset

### B. Image Binarization Process

An HTM evolves as a result of the data that is fed to it, as well as the way that data is presented to it. An encoder is a program that converts arbitrary input into a format that an HTM can understand. This must always be an SDR input. Each bit in the SDR represents the activation state of the columns from the previous area in the HTM ('0' or '1' for inactive and active, respectively). This input is then used as the HTM's next area's feedforward input.

### C. SDR Similarity Calculation

Following the training of each image in the learning phase, the next task is to compute the similarity between the SDRs of each trained image. The similarity is calculated between the active columns of the image's SDR. This similarity is defined in two ways: one is the similarity between images of the same shape, known as Micro similarity, and the other is the similarity between images of different shapes, known as Macro similarity. For example, comparing all images of the shape circle to itself gives the micro similarity and circle to rectangle gives the macro similarity. These micro and macro similarity values are combined to form a similarity matrix, which is used to analyze learning accuracy and further image class prediction.

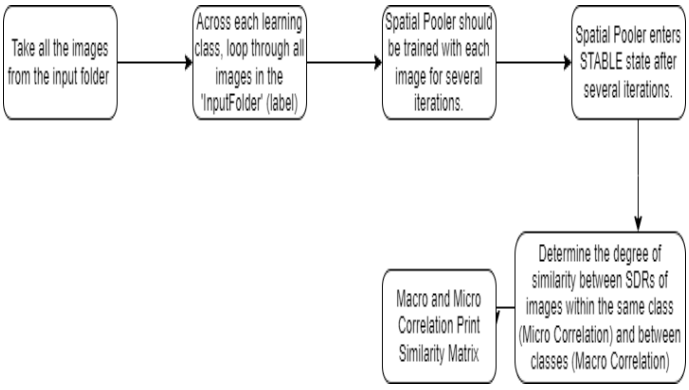



Fig. 7. The HTM Image Training Process Flow

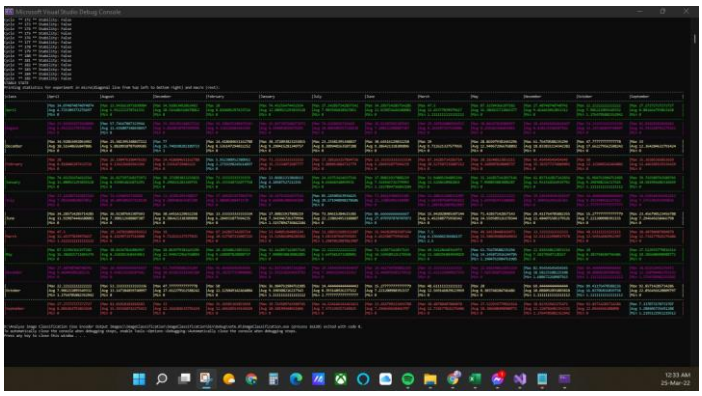


Fig. 8. The macro and micro similarity after training the system (Output Similarity)

## VII. Experiment

In this section, we have described various cases of experiments performed on images of datetime encoders from various dates and times from various months. The experiments are carried out to investigate the variation in similarity caused by changing the HTM parameters. After obtaining the best similarity matrix, experiments are carried out to test the accuracy of image prediction. The results of all experiments presented in this work are achieved through the use of local inhibition.

**Learning Phase:** Figure 7 depicts the flow of image training steps in the HTM image classification system. We used DateTime encoder output images with dimensions of 128x128 pixels for the training process. The first experiments were carried out using the default parameters from the htmconfig.json file. Each image is trained in a series of iteration steps, which are specified by the argument iterationSteps. The images are trained until the spatial pooler achieves a stable state, which is managed by the HomeostaticPlasticityController class (HPC). Its goal is to start the learning process by putting the Spatial Pooler in a newborn state. At this point, the boosting is very active, but the spatial pooler is unstable. When the SDR generated for each input becomes stable, the HPC will fire an event to notify the code that the spatial pooler has reached a stable state. The minimum number of cycles (iterationSteps) is calculated automatically by trainingImagesPath.Length × newBornStageIterations

The number of cycles required by spatial pooler to become stable during the learning process ranged from 90 to 380, depending on the number of images in each learning class and the HTM parameters specified in the htmconfig.json file. We examined the results of these experiments. Fetch all images contained within the Folder Path Across each learning class, loop through all images in the 'InputFolder' (label) Spatial Pooler should be trained with each image for several iterations. Spatial Pooler enters STABLE state after several iterations. Determine the degree of similarity between SDRs of images within the same class (Micro Correlation) and between classes (Macro Correlation) Print the similarity matrix of Macro and Micro Correlation to see how changing the spatial pooler parameters affects image classification.

.

## VIII. Results

The training cycles ranged between 90 and over 380 depending on the HTM parameters. First, the similarity of two images from the same category is compared for several HTM parameters, to determine how similar or different these images are found by HTM. Following that, the similarity of images from various categories is discussed, followed by a comparison of the similarity results of images from one category to images from another. Figure 8 depicts the macro and micro similarities prior to training the system. Here we can see how similar the binarized outputs of dataset images are to one another.

## IX. Discussion

Advances in our understanding of how our brains work biologically may lead to new and revolutionary approaches to achieving true machine intelligence, which is the goal of the HTM theory. We used the Spatial pooler-based image classification process and properties. The Spatial pooler has a versatile coding schema that can be applied in real-world machine learning applications. For the input image dataset, we use the DateTime encoder. This project sought to investigate a new learning algorithm for classifying labeled images that mimicked the human neocortex. If we start with an infant, they begin to learn using unlabeled data. Even in the future, the majority of the world's input data will be unclassified. However, we used a categorized set of images to train the model in this project. As a result, future research will focus on training the spatial pooler with uncategorized images.

## References

[1] J. Mnatzaganian, E. Fokoue´ and D. Kudithipudi, "frontiers in Robotics and AI," [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2016.00081/full.

[2] E. N.-A. I. Panov, "Hierarchical Temporal Memory with Reinforcement Learning," Hierarchical Temporal Memory with Reinforcement Learning, 2020.

[3] J. Hawkins, ”Hierarchical Temporal Memory (HTM) Whitepaper,” Hierarchical Temporal Memory (HTM) Whitepaper, 2011.
[4] R. S.-T. L.-A. A.-J. Machado, ”Hierarchical Temporal Memory Theory Approach to Stock Market Time Series Forecasting,” Advances in Public Transport Platform for the Development of Sustainability Cities, 2021.
[5] Purdy, S. (1970, January 1). [PDF] encoding data for HTM systems: Semantic scholar. undefined. Retrieved March 30, 2022, from https://www.semanticscholar.org/paper/Encoding-Data-for-HTM-Systems-Purdy/14552052a0ee729539298e4f8eb42292c0eb1622?p2df
[6] Numenta, ”Encoders - NuPIC 1.0.3 documentation,” Sphinx 1.5.3 , 2017.
[7] Ddobric. “DDOBRIC/Neocortexapi: C.Net Core Implementation of Hierarchical Temporal Memory Cortical Learning Algorithm.” GitHub, https://github.com/ddobric/neocortexapi.