Hierarchical Temporal Memory and its advantages over some Machine Learning algorithms.

Chung Alvarez, Alex Steve

July 2021

Abstract

Machine learning has many different techniques which are not the most well known, but may be very useful for today's new challenges. Hierarchical temporal memory is one of these, multiple works based on this approach are shared here in order to let other people learn about this concept. We have found in different machine learning problems, such as computer vision, reinforcement learning, anomaly detection, continuous learning and classification, that the solutions based on the hierarchical temporal memory approach outperformed the state-of-the-art machine learning solutions or at least achieved very acceptable results.

Contents

List of Tables

List of Figures

1 Introduction

Machine learning (ML) was developed in order to let devices "learn" from collected data and make choices by their own [17]. This is performed by training neural networks with large input data-sets for different tasks such as classifying, abnormality detection on time series, processes automation for the industry, robots logic and so on. Since ML depends on large input data-sets, it is expensive to train its models. Furthermore, ML algorithms do not learn at all. For example, if we train a ML model to recognise animals from a picture, it will not learn the complete concept of each animal. It will just "learn" how the animal look like, based on the images used in the training phase. But it will need to train other models in order to "learn" how each animal sound, the way they move, etc [7]. Though ML has good results, this is not how our brains work. Human brains receive several inputs from our senses in order to build our perception from reality, and they don't need large inputs in order to infer what a new object might be. This is because neurons form connections between them and build not only one, but different models of one specific input. These models are connected to the inputs and when a new object comes in the data, it will activate some neurons which are already connected to a model, therefore, we can infer what the new data is. All of these processes are done in one specific part of the brain, called the neocortex. Hierarchical Temporal Memory (HTM) is another approach to achieve artificial intelligence. HTM is based in how human brains learn. Therefore, it may be less expensive than the actual machine learning models. It doesn't need very large data-sets in order to learn, so we may achieve less time training with HTM. This paper will show you the advantages HTM has over some ML algorithms. Hopefully, after reading it you will find HTM a better approach than others because of its results, as it is time and cost saving.

1.1 Historical background

As the usage of machine learning has increased through the years [12], every time more complex algorithms have been developed in order to achieve better results. Nonetheless, even more expensive hardware has been built for reaching a faster learning process with these algorithms. This is why reinforcement learning and deep learning were developed and also the reason of existence of NPUs [6], among other software and hardware technologies built for this purpose. These technologies are well known and have pretty good results in different situations such us recognising images, natural language processing, helping in business decision making, etc [12]. However, all of these processes don't act like we would like to do. Artificial intelligence is thought to make machines learn as humans do. To achieve this, Hierarchical temporal memory was created by Numenta. Numenta is a research company that have been studying how to apply the knowledge we have about how the brain works into machines, so that they can learn in a similar way to us. This model, based on the neocortex, is worth to be studied more since it may solve the problem of needing large data-

sets for training artificial intelligence models, and what machines can learn from it will have more information than what the majority of the actual algorithms, which require expensive computational features, can achieve. Some researches have been made comparing HTM with Artificial Neural Network (ANN) and Support Vector Machine (SVM) [5] for classifying objects (people) in a video, also comparing HTM with Reinforcement Learning (RL) [9] [15] which showed better rewards by using HTM applied to RL.

1.2 Hypothesis set to achieve the objective

The objective of this study was to study the advantages of Hierarchical Temporal Memory over some Machine Learning algorithms.
It is hypothesized that HTM algorithms have a better performance than ML algorithms. Since HTM models are base on the neocortex, which is the part of the brain where learning takes place, they should learn in a more similar way to humans than ML models do. So HTM should be less expensive and more accurate than most current ML algorithms. Another objective was to study the advantages of BrainBlocks over Scikit-Learn. BrainBlocks uses HTM methods to solve practical ML applications, so it makes it comparable to Scikit-Learn, which is one of the most known frameworks for ML. It may have some advantages over Scikit-Learn since it is based on HTM.

1.3 Objectives

The main objective was to study the advantages of Hierarchical Temporal Memory over some Machine Learning algorithms.
The specific objectives of the study were:

- To examine the benefits of HTM over ANN and SVM for classifying objects from a video stream.
- To examine how the application of HTM techniques improves Reinforcement Learning.
- To examine the performance of HTM in detecting anomalies in real time.
- To examine the performance of HTM in continuous online sequence learning.
- To study the advantages of an HTM based classifier (BrainBlocks Classifier) over ML classifiers (Scikit-Learn Classifiers).

1.4 Limitations of the study

- HTM is based on the study of the brain. Since we still don't know exactly how it works, we won't be studying the full potential of what HTM can do. We will just present the achievements of HTM until the present. The

HTM algorithm is, therefore, incomplete. However, what has been reached is sufficient to this study.

1.5 Relevance of the study

It is important to acknowledge other approaches of artificial intelligence in order to improve the most used ones. Furthermore, the research about HTM is still in progress, so we can contribute to develop better tools based on it. HTM is an artificial intelligence model based on the neocortex, so studying its advantages over actual ML techniques will let us discover other mechanisms to make machines actually learn. With the actual models of ML there are many computational limitations while HTM lets us achieve similar results with less computational power. It is also relevant for the study of image detection and natural language processing, since HTM models are richer than ML current models, so it can be more precise in those fields.[5][9][15]

2 Literature review

2.1 Hierarchical Temporal Memory

HTM is a computational model of the human neocortex, originally proposed by Jeff Hawkins [8]. As its name says, this model is hierarchical. It receives data according to its complexity, so most simple data inputs will be received by the neurons (also called cells) in the lower level, while the most complex ones will be received by the neurons at the top level. All of these neurons are connected between them, so all the inputs contribute to build the model of the object (Figure 1). Most HTM implementations have the following components:

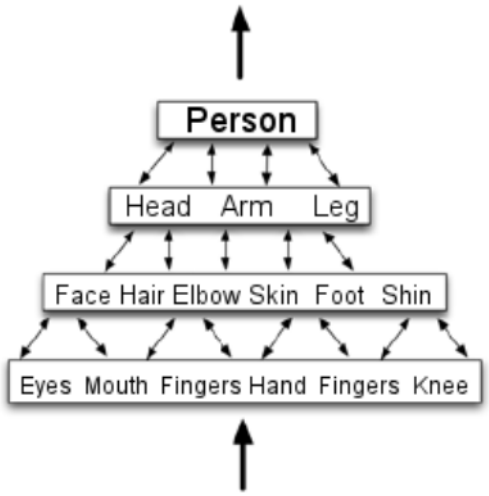


Figure 1: HTM hierarchy: a hierarchy comprised of four levels.
Source: Fig 3 [5]

Encoders, Spatial Pooler and Temporal Memory (Figure 2). All communication between them are made by Sparse Distributed Representations (SDRs) which are basically binary arrays where the cells with a 1 are called active cells.

Figure 2: HTM Components.**Source:** Fig 1a [3]

2.1.1 Encoders

Encoders are responsible of translating the raw input data into SDRs, so the encoders depend on the type of data you want the model to be trained with. They need to interpret similar inputs to partially overlapped SDRs.

2.1.2 Spatial Pooler

The Spatial Pooler ensures that all the SDRs that enter to the model have the same size and sparsity before they are passed to the temporal pooler [3]. It also ensures that the available capacity is optimally used. Finally, it chooses a subset of columns related to the input level and between them (Figure 3).

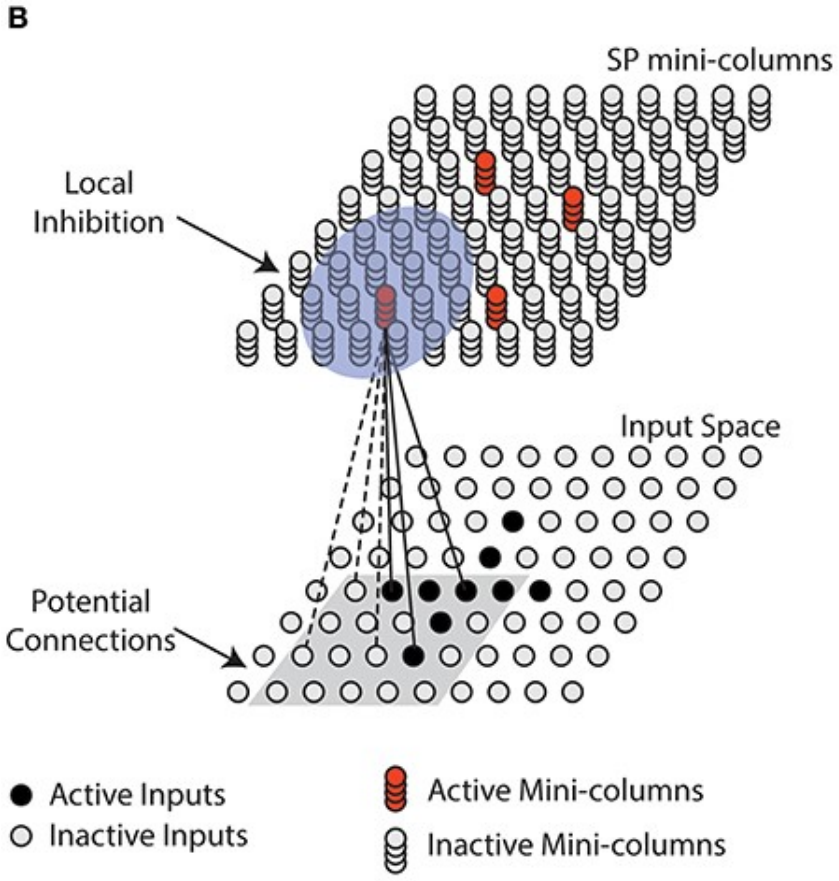


Figure 3: HTM Spatial Pooler.**Source:** Fig 1b [3]

2.1.3 Temporal Memory

The temporal memory decides what cells from the columns selected by the spatial pooler become active (gives a context). After that, it makes some cells become predictive based on their connections with the active cells and the previous interaction with the input (Figure 4). If the number of active cells connected to a cell is over a certain threshold, this cell will become predictive. In this component you can decide for how much time the model retains some information learned.

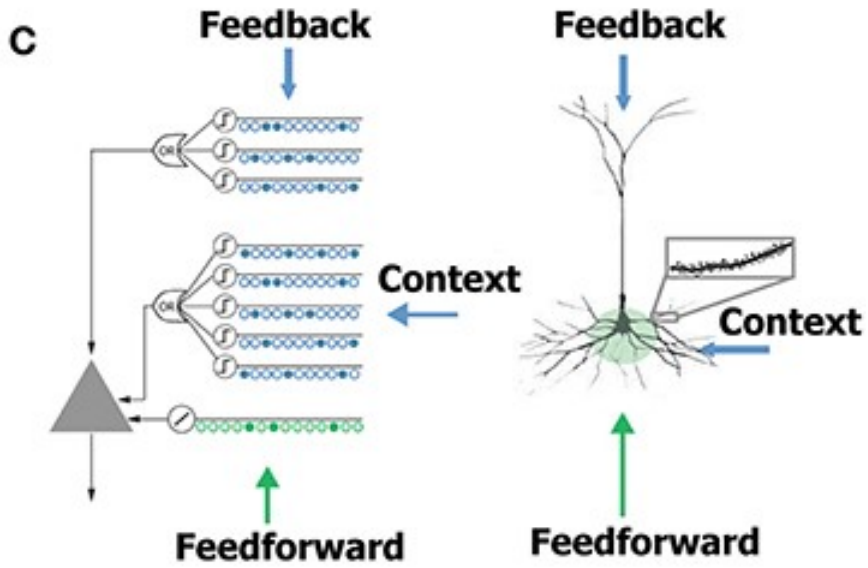


Figure 4: HTM Temporal Memory.**Source:** Fig 1c [3]

2.2 Machine Learning

Machine Learning is a natural outgrowth of the intersection of Computer Science and Statistics[12]. Its algorithms receive input data and produce some outputs by themselves, without being programmed to do so [4]. Three types of machine learning algorithms are: supervised learning, unsupervised learning and reinforcement learning.

2.2.1 Supervised Learning

In supervised learning algorithms, the inputs and the outputs that correspond to each input are known, and you train neural networks to learn which output correspond to what inputs (Figure 5).

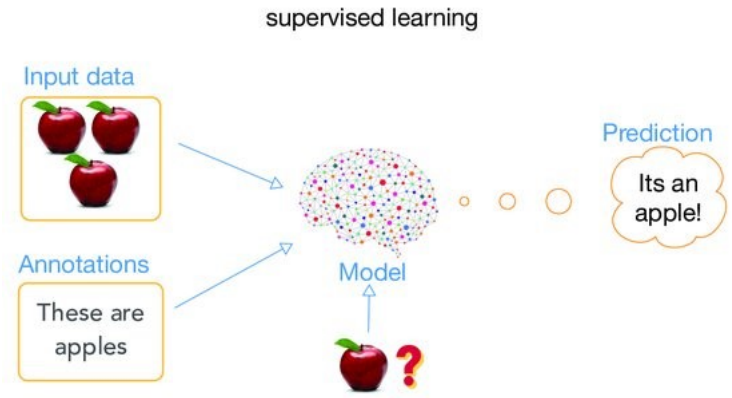


Figure 5: Supervised Learning.**Source:** Fig 1 [18]

2.2.2 Unsupervised Learning

In unsupervised learning algorithms, neural networks only receive the input data, and manage somehow to group it according to some similarities (Figure 6).

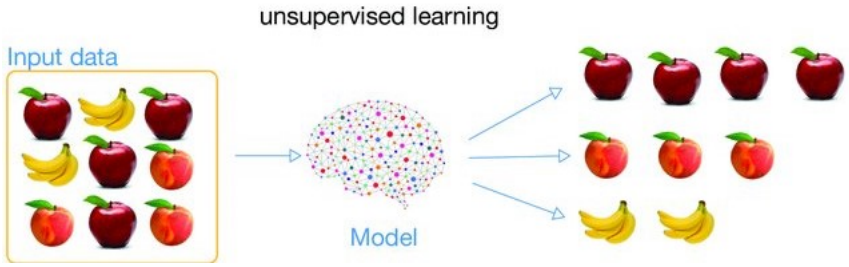


Figure 6: Unsupervised learning.**Source:** Fig 1 [18]

2.2.3 Reinforcement Learning

Reinforcement learning (RL) is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal [16]. The learner, called agent, needs to decide the actions to take in order to achieve the best final reward. After doing this, a policy is created. Policies describe which actions are going to be taken in each environment state, since they map the agent's actions with the environment states. The environment is where the agent receives the input data and take actions in response. Rewards are numeric signals sent by the environment each time the agent interacts with it, they tell the agent whether it took a right decision or not. For making the best decision, a value function

is also defined, which defines what is good after multiple interactions with the environment. A model is optional to be defined, it allows the agent to infer the next state and reward of the environment. Markov Decision Processes represent ideally RL in mathematics (Figure 7).

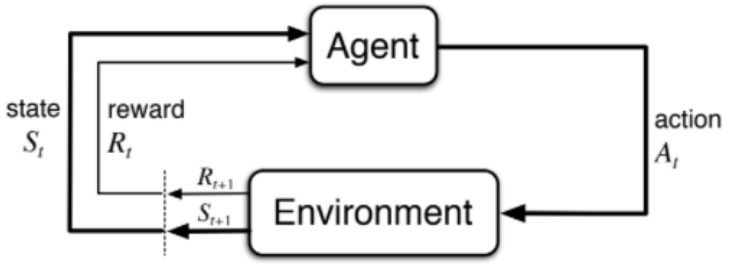


Figure 7: The agent–environment interaction in a Markov decision process.
Source: Fig 3.1 [16]

3 Experimental Review

3.1 Experiment 1: Object Tracking via HTM

There is an important aspect in a tracking system: to detect if the target is present on a frame [5]. This experiment was made by Fallas-Moya and Torres-Rojas. Figure 8 represents object tracking.

Figure 8: Object tracking: Graphical description.
Source: Reading University Computational.Performance Evaluationof Tracking and Surveillance. A dataset for Computer Visionand Pattern Recognition. 2009, cited in Fig 1 [5]

They compared HTM (called HTM Hard) with a mixed model of a Temporal Pooler connected with a K-Nearest Neighbor classifier (called HTM Soft), an Artificial Neural Network (ANN) and a Support Vector Machine (SVM). The case of ANN and SVM are similar, they receive the data directly to a classifier [5]. Figure 9 shows the box plot (as defined by Massart et al.) as a result of these four implementations [5].

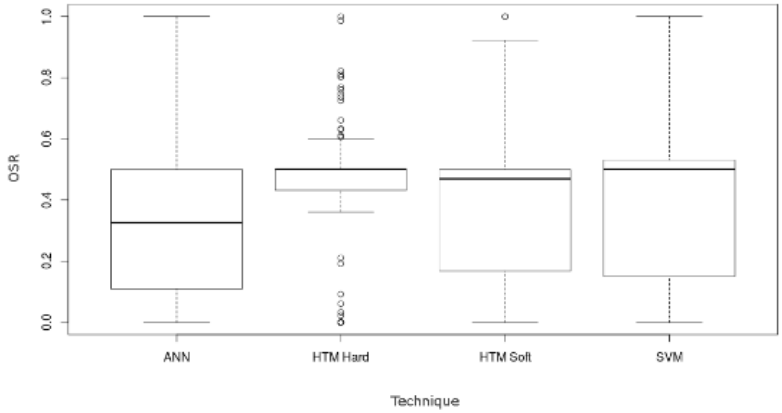


Figure 9: Boxplot for the significant factor Technique.**Source:** Fig 8 [5]

The significant factor technique is the one they use for extracting features from images [5], which will serve as the raw data for the HTM and as input for ANN

and SVM. As we can see in Figure 9, HTM Hard obtained the most accurate values from the four implementations since its box plot is thinner than the other three. This means the values from HTM Hard are in the smallest range, so it will have less failures during tests [5].

3.2 Experiment 2: A Reinforcement Learning Algorithm Based on HTM

This algorithm was called HTMRL [15] or RLHTM [9]. Koffi et al. compared the rewards obtained by an RLHTM agent with a RL agent. Figure 10 shows that better rewards were obtained by the RLHTM agent.

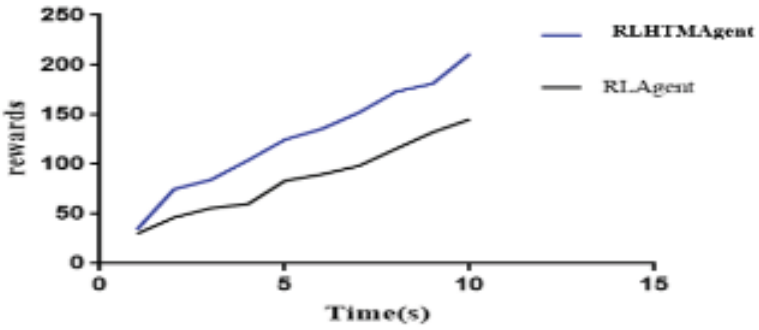


Figure 10: RLHTM learning rate versus RL agent.**Source:** Fig 4 [9]

To determine the accuracy of their method, they tested it against Q-learning (a RL algorithm) in two experiments: Traffic environment and Cartpole environment [9]. Figure 11 shows that RLHTM did converge while Q-learning did not.

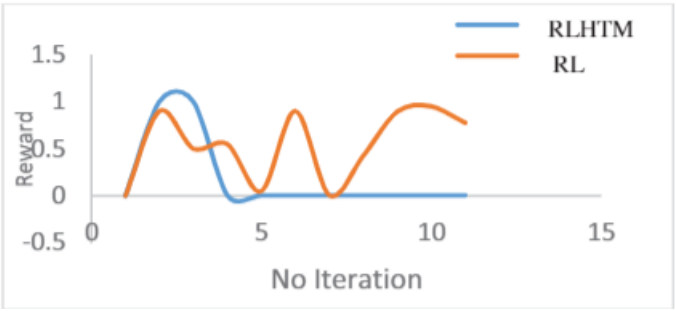
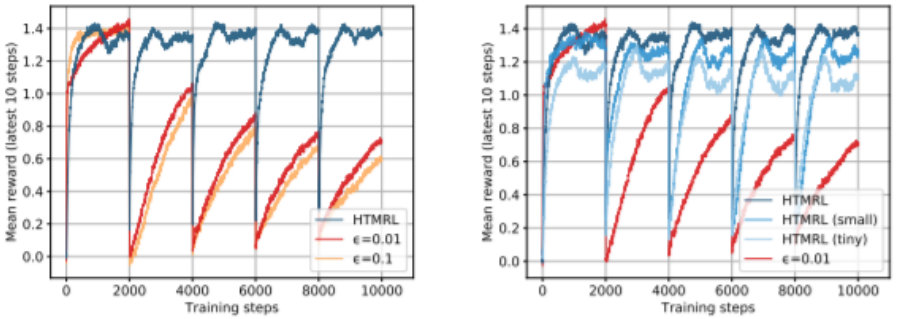


Figure 11: Time Episodic Comparison of RLHTM with RL Q-learning algorithm for the Traffic environment problem.
Source: Fig 6 [9]

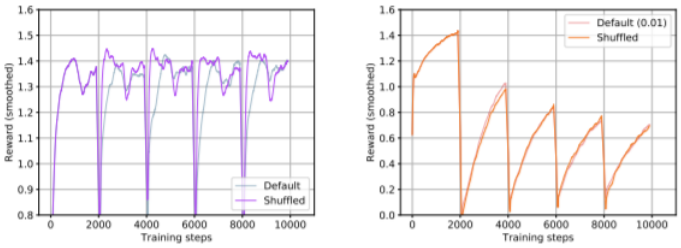
For the Cartpole environment problem they obtained that RLHTM achieved

100% accuracy for the two instances of the Cartpole they used, while Q Function only obtained 96% and 98% [9]. They concluded that their algorithm (RLHTM) outperformed the current reinforcement-learning algorithm [9].
Struye et al. compared their HTMRL with an ϵ-greedy learner in a variant of the bandit environment. Figure 12a shows HTMRL achieved better rewards than the ϵ-greedy, furthermore, figure 12b shows that even restricting the dimension of the HTMRL, it still obtained better rewards. Figure 13 shows that when shuffling instead of re-initialising the arms, HTMRL can leverage previous knowledge while ϵ-greedy cannot [15].



(a) HTMRL vs. ϵ-greedy (b) HTMRL with restricted dimensions

Figure 12: Performance of HTMRL and ϵ-greedy for a 10-armed stochastic bandit, re-initialising its arms every 2000 steps.
Source: Fig 3 [15]



(a) Arm shuffling (HTMRL) (b) Arm shuffling (ϵ-greedy)

Figure 13: Performance of HTMRL and ϵ-greedy for a 10-armed stochastic bandit, shuffling its arms every 2000 steps.
Source: Fig 4 (c,d) [15]

We can conclude again from this that the HTM based RL algorithm obtained better learning results than the RL approach now with ϵ-greedy. Struye et al. concluded these results have strengthened the value of the HTM system.

3.3 Experiment 3: Real-time anomaly detection for streaming data and its application in manufacturing systems.

In order to compare the HTM anomaly detection algorithm with other ML algorithms, The Numenta Anomaly Benchmark (NAB) has been designed [1]. Table 1 shows the comparison made in [1] of average NAB scores for different ML algorithms and the HTM algorithm.

Source	Anomaly Type	Numenta HTM	CAD OSE	KNN CAD	Relative Entropy	Twitter AdVec	Skyline	Sliding Threshold
Artificial	Spatial only	0.70	**0.84**	–0.06	0.78	0.55	–0.39	–1.00
	Temporal only	0.11	0.08	–0.13	–0.52	**0.52**	–0.38	–0.90
	Spatial + Temporal	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Online advertisement clicks	Spatial only	**0.75**	0.21	0.54	–0.55	–0.03	–1.00	–0.38
	Temporal only	0.53	**0.83**	0.54	–0.53	–1.00	–1.00	–1.00
	Spatial + Temporal	**0.47**	**0.47**	0.37	–0.15	0.10	–0.47	0.41
AWS server metrics	Spatial only	0.61	**0.74**	0.54	0.11	0.28	0.40	–0.42
	Temporal only	**0.70**	0.29	0.03	0.53	–0.66	–0.77	–1.33
	Spatial + Temporal	**0.29**	0.20	0.01	–0.23	–0.45	–0.14	–0.34
Miscellaneous known causes	Spatial only	0.19	**0.33**	0.23	0.13	0.00	–0.38	–0.41
	Temporal only	–0.60	–0.79	**0.18**	–1.00	–0.91	–1.14	–0.96
	Spatial + Temporal	**0.32**	–0.15	–0.34	0.30	–0.48	–0.79	–0.92
Freeway traffic	Spatial only	0.83	0.85	–0.06	0.62	0.85	**0.87**	–0.38
	Temporal only	0.55	**0.86**	–1.44	0.75	–1.00	–1.00	–1.00
	Spatial + Temporal	0.51	**0.80**	0.26	0.50	–0.27	0.40	–0.83
Tweets volume	Spatial only	0.38	**0.74**	0.10	0.64	0.04	–1.46	–0.17
	Temporal only	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Spatial + Temporal	0.34	**0.43**	0.29	0.13	0.20	–0.25	0.05
Total Average		**0.40**	0.39	0.16	0.10	–0.03	–0.28	–0.36

Table 1: Comparison of average NAB scores for some algorithms across data from different kinds of sources provided by the benchmark. **Source:** Table 3 [1]

We can see that the HTM algorithm has the highest average score.
Based on this work, a more recent study made for predicting machines failure was made in [10]. The results of this study are shown in Tables 2 and 3. In this work, it was demonstrated that HTM outperforms the current state-of-the-art methods for detecting anomalies in both bearing and 3D printer failure data

Anomaly Detector	Scoring Profile			Runtime (s)
	Standard	Low FN	Low FP	
TM-HTM+HD (Ours)	**67.05**	**73.33**	56.57	4728
HTM+HD (Ours)	66.38	71.93	55.33	5792
Windowed Gaussian	64.70	70.50	57.35	336
HTM+LP (Ours)	64.03	69.12	**57.47**	21084
HTM (Ours)	59.75	66.24	47.63	4277
TM-HTM+LP (Ours)	54.12	61.53	43.03	18508
TM-HTM (Ours)	54.39	63.33	32.47	3156
Etsy Skyline [35]	47.53	51.51	43.75	742632
CAD-OSE [33]	46.88	52.81	40.96	3589
EXPoSE [32]	41.75	44.80	36.96	5575
Threshold-Based	37.75	43.75	25.21	125
Relative Entropy [34]	34.97	37.05	32.94	806
LSTM [31]	33.99	38.13	28.38	43698
KNN-CAD [36]	32.31	43.06	4.69	4393
Random	3.06	9.16	0.00	233
BC [37]	0.00	0.00	0.00	10270
Null	0.00	0.00	0.00	235

Table 2: Normalized NAB scores for anomaly detection on the bearing failure dataset.
Source: Table 1 [10]

Anomaly Detector	Scoring Profile			Runtime (s)
	Standard	Low FN	Low FP	
HTM+HD (Ours)	63.03	**73.18**	42.23	5813
LSTM [31]	**64.76**	71.43	**51.34**	16414
TM-HTM+HD (Ours)	58.01	68.13	44.14	3364
Windowed Gaussian	57.42	67.99	49.31	189
HTM+LP (Ours)	54.56	65.45	36.76	10062
CAD-OSE [33]	54.69	63.27	34.72	1573
KNN-CAD [36]	53.76	65.55	23.48	3375
HTM (Ours)	49.21	62.25	31.16	2713
TM-HTM+LP (Ours)	45.31	57.90	21.93	7456
TM-HTM (Ours)	41.43	54.43	13.86	1724
Relative Entropy [34]	41.78	52.49	15.96	668
Etsy Skyline [35]	39.16	47.84	19.91	146165
BC [37]	21.33	23.64	17.55	1946
Random	11.34	24.75	3.74	49
EXPoSE [32]	6.04	6.20	6.04	7136
Threshold-Based	0.00	0.00	0.00	54
Null	0.00	0.00	0.00	49

Table 3: Normalized NAB scores for anomaly detection on the 3D printer dataset.
Source: Table 2 [10]

with minimal to no pre-processing or application-specific tuning, achieving an execution time of the same order of these methods [10]. They concluded that HTM-based anomaly detection would be a practical solution for a wide range of industrial predictive maintainance applications [10].

3.4 Experiment 4: Continuous online sequence learning and its application in oil and gas pipeline data prediction and monitoring.

In 2016, two experiments were carried out to compare HTM with state-of-the-art ML models for sequence learning tasks [2]. The first experiment was the high order sequence prediction with artificial data and the second one was the prediction of New York city taxi passenger demand.

3.4.1 High order sequence prediction with artificial data

First, they compared HTM with ELM, TDNN and LSTM [2]. As we may see from Figure 14, HTM achieves one of the highest prediction accuracy. Besides, after changes in the data stream, HTM recovers faster than the state-of-the-art ML models.

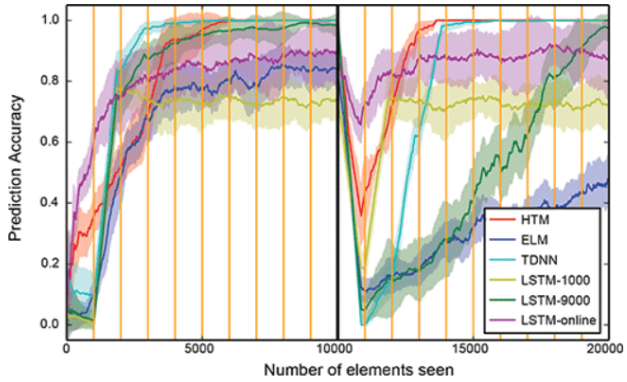


Figure 14: Prediction accuracy of HTM (red), LSTM (yellow, green, purple), ELM (blue), and TDNN (cyan) on an artificial data set.
Source: Figure 5 [2]

Second, they compared HTM, LSTM and ELM for simultaneous multiple predictions [2]. As we may see in Figure 15, HTM outperforms the state-of-the-art ML models' predictions accuracy.

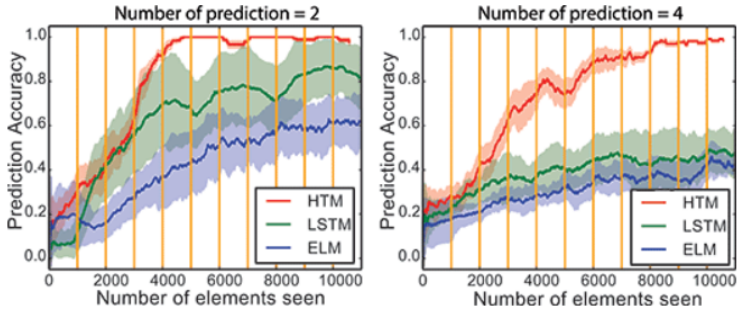


Figure 15: Performance on high-order sequence prediction tasks that require two (left) or four (right) simultaneous predictions.
Source: Figure 6 [2]

Then, they compared HTM with LSTM using the same dataset, but adding noise after 12000 elements. Figure 16 shows that the performances of the two

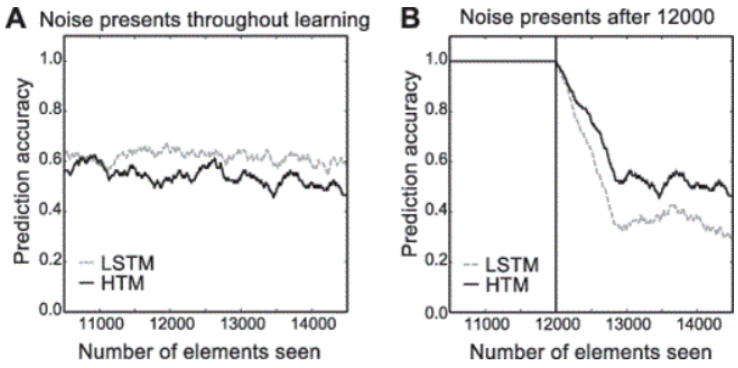


Figure 16: (A) Prediction accuracy over learning with the presence of temporal noise for LSTM (gray) and HTM (black). (B) HTM and LSTM are trained with clean sequences. Temporal noise was added after 12,000 elements.
Source: Figure 8 [2]

models drop, but LSTM has worse performance than HTM [2].

3.4.2 Prediction of New York city taxi passenger demand

The goal of this experiment was to compare the performance of HTM with other sequence learning techniques in real-world scenarios [2].

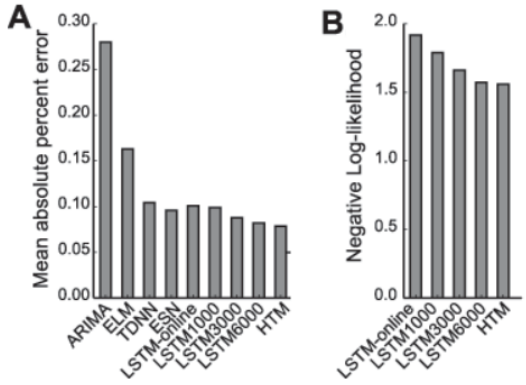


Figure 17: Prediction error of different sequence prediction algorithms using two metrics: mean absolute percentage error (A), and negative log likelihood (B).
Source: Figure 10 (b,c) [2]

As we can see from Figure 17, HTM sequence memory had comparable performance to LSTM on both error metrics. Both techniques had a much lower error

than ELM, ESN, and ARIMA, as seen in Figure 17 (A). We also must note that HTM sequence memory achieves this performance with a single-pass training paradigm, whereas LSTM requires multiple passes on a buffered data set [2].
Finally, they tested how fast the algorithms adapted to changes in the data. Figure 18 shows that the prediction error of HTM quickly dropped back in around two weeks, while the LSTM prediction error stayed high much longer [2].

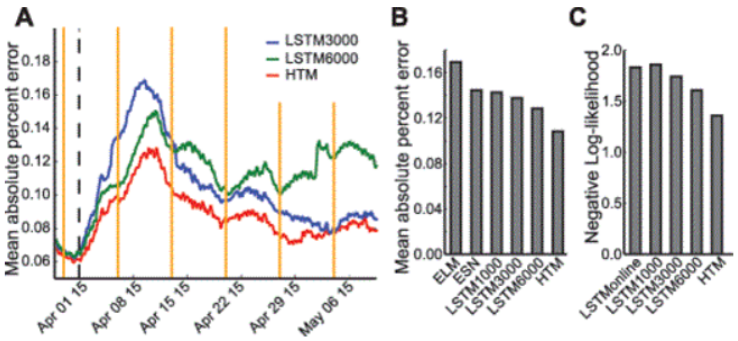


Figure 18: (A). The mean absolute percent error of HTM sequence memory (red) and LSTM networks (green, blue) after artificial manipulation of the data (black dashed line). (B, C). Prediction error after the manipulation.
Source: Figure 11 [2]

As we can see, HTM sequence memory has better accuracy on both the MAPE and the negative log-likelihood metrics [2].
They concluded that HTM showed promising results on real-time sequence learning problems [2].
A more recent study was made in 2018 by Onukwugha and Osegi. They compared HTM-SP (Hierarchical Temporal Memory Spatial Pooler) with the OS-ELM (Online Sequential Extreme Learning Machine).
Table 4 shows the comparison of the improvement factors of HTM-SP over OS-ELM using sig, sin and rbf activations.

Improvement Factors		
$MAPE_{sig}$/ $MAPE_{HTM\text{-}SP}$	$MAPE_{sin}$/ $MAPE_{HTM\text{-}SP}$	$MAPE_{rbf}$/ $MAPE_{HTM\text{-}SP}$
1.297	1.297	1.300

Table 4: Improvement of the HTM-SP over the OS-ELM activation function variants.
Source: Table 1 [13]

The results clearly indicated that the HTM-SP would outperform the OS-ELM on each activations [13].

4 Results

We used the BrainBlocks framework (`https://github.com/the-aerospace-corporation/brainblocks`) based on HTM to compare its classifier with some Scikit-Learn classifiers [14]. Figures 19 and 20 show the comparison between the five classifiers over ten different data-sets.

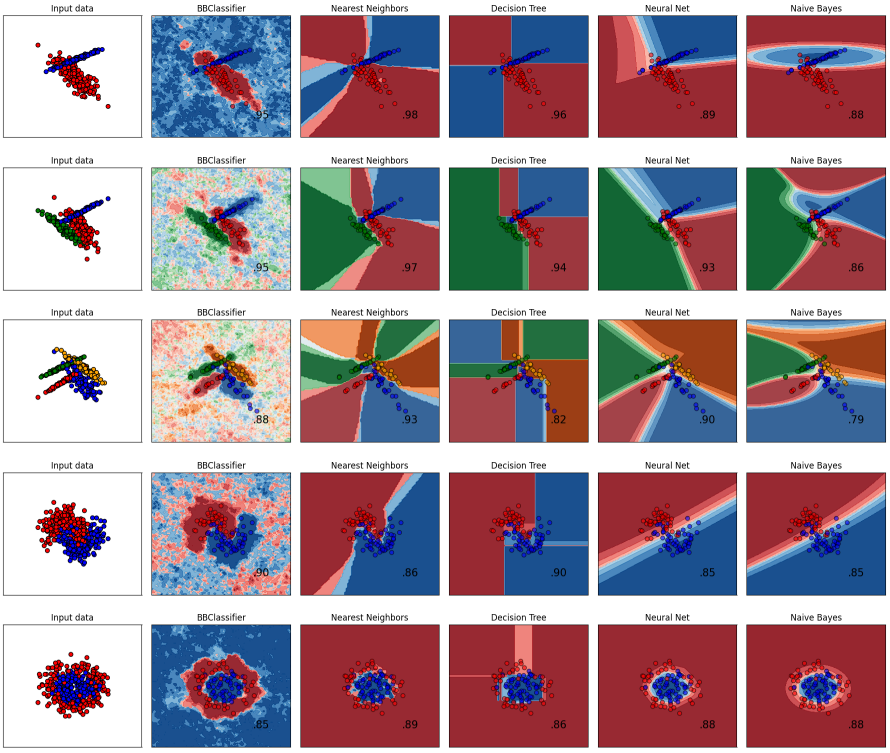


Figure 19: Comparison between Brainblocks classifier and Scikit-Learn classifiers data-sets 1-5 with scores.

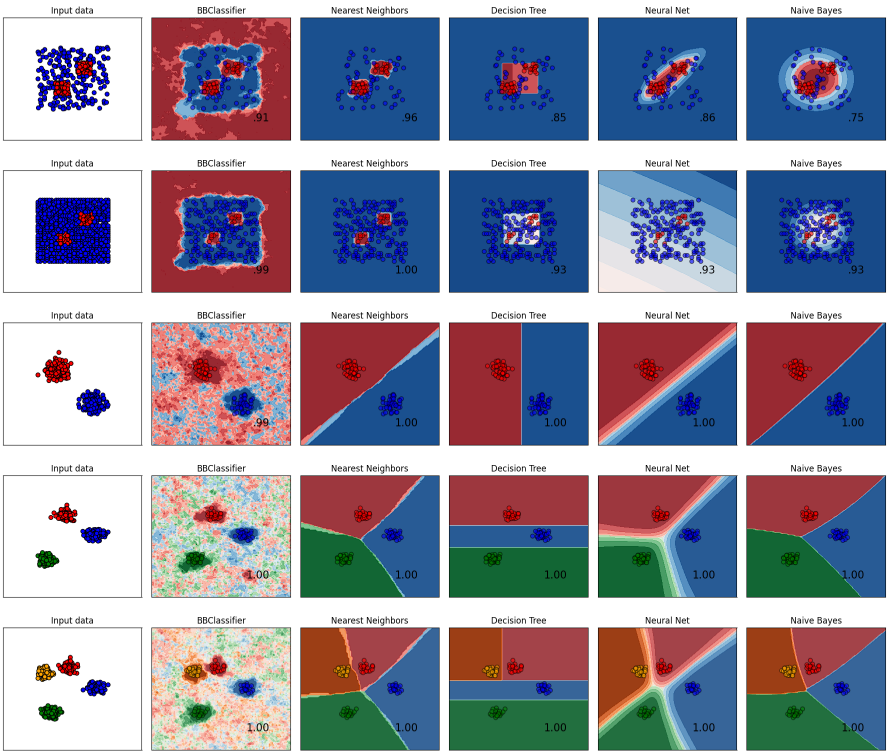


Figure 20: Comparison between Brainblocks classifier and Scikit-Learn classifiers data-sets 6-10 with scores.

From Figures 19 and 20 we can tell that the BBClassifier precision is one of the bests, having a score of 90 or above in 8 of the 10 data-sets, the only classifier that achieves this from the well-known ML ones is the Nearest Neighbors Classifier, which means it is flexible to classify different data-sets with almost the same precision. Furthermore, the BBClassifier gives a large "region of ignorance" where many of the other classifiers give high confidence for spaces of data of which it has no experience.

5 Conclusions

We have shown multiple scenarios where HTM has an acceptable performance, and even a better one than state-of-the-art ML algorithms. Most of these scenarios involve learning from data received in real time, which is the strongest characteristic of HTM, since this algorithm has no need of batch processing. Therefore, this approach based on the neocortex should be studied more in order to discover the most of its capabilities and limitations.

References

[1] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.04.070. URL https://www.sciencedirect.com/science/article/pii/S0925231217309864. Online Real-Time Learning Strategies for Data Streams.

[2] Y. Cui, S. Ahmad, and J. Hawkins. Continuous Online Sequence Learning with an Unsupervised Neural Network Model. *Neural Computation*, 28(11): 2474–2504, 11 2016. ISSN 0899-7667. doi: 10.1162/NECO_a_00893. URL https://doi.org/10.1162/NECO_a_00893.

[3] Y. Cui, S. Ahmad, and J. Hawkins. The htm spatial pooler—a neocortical algorithm for online sparse distributed coding. *Frontiers in Computational Neuroscience*, 11:111, 2017. ISSN 1662-5188. doi: 10.3389/fncom.2017.00111. URL https://www.frontiersin.org/article/10.3389/fncom.2017.00111.

[4] I. El Naqa and M. J. Murphy. What is machine learning? machine learning in radiation oncology. page 3–11, 2015. doi: 10.1007/978-3-319-18305-3_1.

[5] F. Fallas-Moya and F. Torres-Rojas. Classifying via hierarchical temporal memory. August 2019.

[6] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. Reinhardt, A. Caulfield, E. Chung, and D. Burger. A configurable cloud-scale dnn processor for real-time ai. pages 1–14, 06 2018. doi: 10.1109/ISCA.2018.00012.

[7] J. Hawkins and R. Dawkins. *A Thousand Brains: A New Theory of Intelligence (English Edition)*. Basic Books, 2021. URL https://numenta.com/a-thousand-brains-by-jeff-hawkins.

[8] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin. Biological and machine intelligence (bami). Initial online release 0.4, 2016. URL https://numenta.com/resources/biological-and-machine-intelligence/.

[9] T. Koffi, T. Cai, T. Epalle, and B. Mensa-Bonsu. A novel reinforcement learning algorithm based on hierarchical memory. pages 1–5, 11 2020. doi: 10.1109/ITIA50152.2020.9312239.

[10] A. Malawade, N. Costa, D. Muthirayan, P. Khargonekar, and M. A. Al Faruque. Neuroscience-inspired algorithms for the predictive maintenance of manufacturing systems. 02 2021.

[11] D. Massart, J. Smeyers-Verbeke, X. Capron, and K. Schlesier. Visual presentation of data by means of box plots. *LC-GC Europe*, 18:215–218, 04 2005.

[12] T. Mitchell. The discipline of machine learning: Carnegie mellon university. *Carnegie Mellon University, School of Computer Science, Machine Learning Department*, 2006.

[13] C. Onukwugha and E. Osegi. An integrative systems model for oil and gas pipeline data prediction and monitoring using a machine intelligence and sequence learning neural technique. 08 2018. doi: 10.20944/preprints201808.0194.v1.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[15] J. Struye, K. Mets, and S. Latré. Htmrl: Biologically plausible reinforcement learning with hierarchical temporal memory, 2020.

[16] R. Sutton and A. Barto. *Reinforcement Learning*. Amsterdam University Press, Amsterdam, Países Bajos, 2 edition, 2018. URL `https://mitpress.mit.edu/books/reinforcement-learning-second-edition`.

[17] M. TM. Machine learning. *New York: McGraw-Hill*, 1997.

[18] M. Yan, K. Liu, Z. Guan, X. Xinkai, X. Qian, and H. Bao. Background augmentation generative adversarial networks (bagans): Effective data generation based on gan-augmented 3d synthesizing. *Symmetry*, 10:734, 12 2018. doi: 10.3390/sym10120734.