DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Online Prediction of Motion Patterns using Hierarchical Temporal Memory

Lukas Tenbrink

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Online Prediction of Motion Patterns using Hierarchical Temporal Memory

Online Vorhersage von Bewegung mithilfe eines Hierarchischen Temporalspeichers

Author:	Lukas Tenbrink
Supervisor:	Dr.rer.nat Florian Röhrbein
Advisor:	Benedikt Feldotto
Submission Date:	15.2.2018

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.2.2018 Lukas Tenbrink

Acknowledgments

I'd like to thank my Advisor Benedikt for his great help streamlining this thesis and giving me clear input when I needed it. I'd also like to thank friend Leon and my mother for helping me sew the last remaining holes in this thesis shut.

Abstract

In this thesis, we perform a number of progressively more complex experiments that examine the utility of Hierarchical Temporal Memory (HTM) networks in the task of visual online motion pattern prediction. First we discuss the requirements for predicting visual motion patterns with HTM networks and build a framework that fulfills these requirements. In the first experiment we demonstrate a working system that is capable of predicting motion patterns for an input space describable by a stateless machine. We then expand upon this network to create a machine that can predict motion patterns that requires a stateful machine to predict. Next, we use data from repetitive tasks in real world camera captures as input and adjust the network to be able to predict motion in that input space. To examine the progress we compare the three iterations of the network and find that the last iteration performs best or similarly to the other two iterations in all experiments. In conclusion, the presented system is capable of performing varying motion pattern prediction tasks from visual data without requiring human adjustments. Finally we discuss what future work could be made to expand upon this system.

Contents

1 Motivation

In Robotics, interaction with humans is a very important component for many practical applications, to utilize strength and advantages of both human and robotic or algorithmic workers. Recently, Huber et al. have shown that workflows are absolved more quickly with active robot assistance than with no assistance [Hub+13] (Figure 1.1). They showed that any non-adaptive strategy examined (sensor-based trigger, averaged static times step-specific waiting times) have significantly higher waiting times for both the human performing the task, as well as the robot assisting with it. However, for a fully adaptive assistant to act at the times required for optimal waiting times for both actors, it cannot simply act whenever the human is already done with the task provided. This was shown by the experiment's sensor-based trigger assistant that acted exactly when the human was done with the previous task. In fact, an assistant that was ready to assist at the exact time the human was ready to receive assistance improved the workflow speed of the task even better than a perfect always-wait assistant, which can assist at any time it is required.

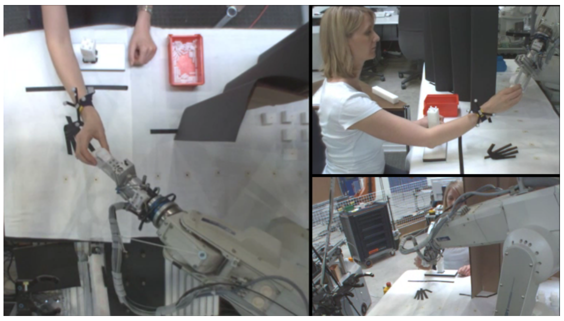

Figure 1.1: [Hub+13] Humans absolving a task with robot assistance

That means that an optimal assistant acts in a *predictive* manner. In highly vision-based tasks such as assembly of machine parts, accountancy or mechanical reparation, a motion prediction system has to be implemented that makes predictions based on visual data to capture all the information that is required for an accurate prediction based on the situation.

Robotics, however, is not the only application of motion prediction. One other application is the use in medical fields as a surveillance system of patients. The earlier the system can detect danger by predicting it, the quicker help can arrive for the patients, as well as offering the capability to collect data specific to the danger [Ugo+13]. Using a vision-based system especially enables the system to be non-intrusive and applicable to any patient. Motion prediction can also be used as part of smart anomaly detection systems in surveillance systems. If collected data can be minimized by stripping away unwanted or insignificant parts, analysis can be a lot more meaningful as well as easy for either humans or other systems.

Some approaches, both analytical and those utilizing neural networks, have already been implemented in an attempt to predict motion. However, those approaches lack two fundamental components essential for a system dealing with real-world data motion prediction. The first aspect is inherent temporality. To properly analyze and interpret temporally complex situations, e.g. with changing speeds or repeating frames, the system needs a temporal aspect in the algorithm rather than one simply added onto it after design. The second aspect is online learning. In the real world, situations change both short-term and long-term, and a good predictive system needs to be able to adjust to changes on the fly rather than needing to be re-adjusted and parametrized after set intervals of time to eliminate the need for a manual component to run the system.

Hierarchical Temporal Memory The technology HTM provides both online learning and inherent temporality and is therefor well-suited for online motion prediction. While running, a HTM's cells make constant predictions about the next step and their synapses to other cells are reinforced or punished according to the Hebbian Rule of Learning [Heb57]. Due to these constant next-step predictions, further predictions such as 5-step or 10-step predictions can be extrapolated using a weight matrix.

Another important aspect of HTMs is its usage of Sparse Distributed Representation (SDR)s for both input and output. SDRs are long binary vectors with a low number of on-bits, also called the Hamming Weight.

The capacity of an SDR c is lower than that of a conventional dense bit array c_d. Given the vector length n, they can be calculated as $c = \binom{n}{w} < 2^n = c_d$ where $n > w > 0$. However, in a trade-off, SDRs have a higher noise tolerance and can thus be considered stabler under varied inputs [AH16]. Using a threshold overlap for comparison, SDRs can tolerate extremely high noise values with low false positives in equality. Many systems using digital sensors, computer vision being one, are subjected to a lot of noise and can thus benefit from high noise tolerance.

Outline In this thesis we attempt to create a network that is capable of online prediction of visual motion in varying tasks. To create such a system, we first start with a HTM network designed to solve a very specific visual prediction task. We then progressively increase the difficulty of the predictions by introducing more complex aspects for the input data, and for each iteration adjust the network to produce optimal results in the new problem space. After each experiment we draw conclusions on the current state of the network and its capabilities and analyze the predictions made. Numenta claims that a HTM shares many traits with the way actual human brains compute their data [HAD10; HA16; HAC17]. It is therefore interesting to examine if predictions made by HTM networks also share aspects of predictions made by humans. If it is possible to make comparisons between aspects of the predictions made by the HTM networks in this thesis and humans, this thesis can hopefully contribute towards research into a better understanding of human intelligence itself. While it is still possible to perform complex algorithms on the data prior to encoding to ease the computational load on the network, to enable the HTM-human comparison, computations performed in this thesis are passed to the network in relatively raw forms with minimal pre-computation instead. This allows us to investigate how well the network is capable of performing all logical tasks required for the prediction itself. A system capable of solving very different tasks without individual adjustments would be very useful as it would save time on implementation of complex systems solving specific tasks.

2 State of the Art

Before we begin with the experiments examining the capabilities of HTM networks regarding visual prediction of motion patterns, we review other work that has been done in the same field. We also assess the state of Numenta Platform for Intelligent Computing (NuPIC), Numenta's implementation of HTM systems that is used in this thesis, including important metrics for evaluating the accuracy of predictions.

2.1 Related Work

In robotics, Visual human analysis, or "looking at people" [Gav99], has been one of the classic and most popular research fields, since it is essential for dynamic human-robot interaction. It is also a skill humans can do very well while our computers are, historically, very bad at it.

Motion prediction especially, coming with a broad range of practical applications, has been a popular study field for a long time. These include parts of systems for human-robot-interaction [Hub+13; Haw+13], full-body capture, surveillance, user interfaces or motion analysis [WRB11].

However, new technologies such as the Artificial Neural Network (ANN), algorithms and altogether different approaches regularly inspire new solutions and progress the field. We currently differentiate between 3 approach types:

analytical	logical approaches to prediction, linear extrapolation of movement
trained ANN	system trained to understand specific patterns
ANN with online learning	system with online learning to understand dynamic patterns

Extremely simple approaches, such as the constant pose predictor, can currently often outperform other very high-tech complex algorithms in metrics such as 0.4 and 1 second prediction error, where predictions are compared to the actual data. This gives research some very good baselines to compare state-of-the-art motion prediction to [MBR17].

These are recent approaches and other related work in the 3 approach types:

Analytical Analytical approaches for motion prediction were the first to be considered and have, compared to neural networks, a very long history, with a large number of approaches that so far turn out to be comparatively high-effort. Still, logical understanding of image sequences is important and can contribute to solving many related, albeit usually highly specialized, problems.

Some early approaches to the problem include an approach by Polana et al., where human action is described statistically by segmenting, normalizing and recognizing based on low-level features in repetitive motion [PN94], (Haar) wavelet coefficients as low-level intensity features [Ore+97], or active shape models to track contours [Coo+95]. Recognition of object movement with very distinct shapes (most importantly the hand) could easily be assessed by simple background subtraction, noise reduction and succinct shape analysis [Gav99].

Recently, Zhou et al. [Zho+15] have shown a simple baseline that concatenates features from visual questions' words and Convolutional Neural Network (CNN) image features that performs on-par with current purely ANN based approaches. Jabri et al. [JJM16] present a simple baseline that, while only recognizing whether complete image-question-answer triplet is correct rather than creating an answer itself, still achieves state-of-the-art performance compared to more complex approaches, suggesting a lack of visual grounding concepts for questions and answers. Lehrmann et al. [LGN13], too, show a simple baseline performing competitively. They implement a non-parametric bayesian network model as a prior of human pose, which can be used for pose tracking.

Huber et al. [Hub+10] developed an algorithm modeling human behavior, using a linear dependency to develop kalman filters in order to predict human behavior in assembly tasks.

Trained ANN While these analytical approaches are certainly useful for their intended purposes, it is becoming clear that prediction of motion patterns requires some completely different approach to become tractable. Qualitatively, the predictions made by these systems make little sense to humans examining them. For that metric, ANNs have recently shown a lot of promise. Quantitatively, however, ANNs still have a way to go before reliably outperforming the aforementioned analytical baselines - for example by .4 and 1 second prediction error metrics.

Fragkiadaki et al. [FLM15] introduced a network with LSTM-3LR (3 layers of Long Short-Term Memory cells) and ERD (Encoder-Recurrent-Decoder) that is able to produce plausible long-term human motion predictions. However, it requires specific tuning by introducing random noise during training, otherwise it quickly accumulates errors and loses its realistic capabilites.

Jain et al. [Jai+15] proposed a network called structural recurrent neural networks that introduces expert knowledge into neural network architectures and thus outperforms previous work both qualitatively and quantitatively. It utilizes semantic knowledge about the network as input, and dynamically assigns different recurrent neural networks to similar parts of the data.

Martinez et al. [MBR17] analyzed the previous [Jai+15; FLM15] recent deep recurrent neural network methods by looking at the architectures, loss functions, and training procedures. They proposed 3 changes that that resulted in motion prediction that can compete with other state-of-the-art approaches.

ANN with online learning There have not yet been many approaches to prediction of motion patterns. However, there has been some work proving the general capability of HTMs to deal with human poses and motion.

Zhang et al. [Zha+09] have implemented a two-stage approach to detection of people eating and drinking using a single accelerometer sensor attached to the participants' wrists. In the first step, the raw accelerometer data is interpreted into euler angles using the Extended Kalman Filter. A HTM is then used to classify the actions. The approach achieves the detection task with high accuracy. While non-visual, this shows that a HTM is a very viable approach to human action observation.

Zhituo et al. [ZHH12] presented a content-based image retrieval system utilizing multiple HTM classifiers. An image is classified, and from a database a semantically similar image is selected. Ugolotti et al. [Ugo+13] also developed SDR encoders specifically for HTM. They presented a four-camera detection and classification of multiple human activities at the same time, implemented using HTM. The focus is on rogue behavior detection: When a patient is in danger, it activates an accelerometer worn by the person to collect a more complete fall data profile for later analysis. This detection system is meant to lengthen the short battery life on the accelerometers.

To the best of our knowledge, there hasn't yet been any work examining the predictive aspect of HTM using a human-oriented visual motion setup as of yet.

2.2 A Brief Introduction to Hierarchical Temporal Memory

In their book, J. Hawkins and S. Blakeslee provide explanations on how certain aspects of the human brain operate from a state-of-the-art neurological point of view [HB04]. The neocortex in mammals is strongly associated with all conscious thought effort and many important functions that separate them from other, less intelligently regarded animals such as birds or reptiles. They thus proposes to start there when looking to implement intelligence in other systems.

The neocortex has a lot of different regions consisting of many neurons and synapses, but apart from their contextual place in the brain defined by how they're connected to other parts of the body, they all operate roughly by the same rules on the inside, indicating that there is a basic approach to its function that could be extracted and implemented in other systems. These regions are interconnected through nerve fibers in a hierarchy. Input is handled at sensory regions, which feed-forward into other regions, being associated with progressively more abstract and permanent concepts [HB04].

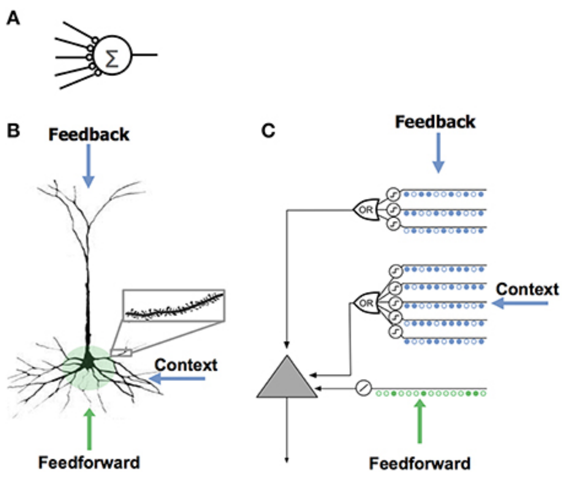

Figure 2.1: [HA16] Comparison between Inputs and Outputs of Biological Neurons and HTM Neurons

Hierarchical Temporal Memory The HTM approach to machine intelligence attempts to implement the neocortex's logic into a digital, well-defined algorithmic space. Each neuron can roughly be considered to be represented by a binary cell: 1 if firing, 0 if not (Figure 2.1). The initial states are provided by an encoder, which converts data into binary vectors stored into a sensor region. The vectors are fed into different kinds of regions, e.g. the Spatial Pooler and the Temporal Memory. Finally, classifiers are created for some sensor regions for evaluation, which convert the current state into readable data (Figures 2.2 and 2.3).

A HTM contains two important types of data mapping algorithms, the Spatial Pooler and the Sequence Memory.

Spatial Pooling The Spatial Pooler groups spatially similar data. In it, each cell is connected to a number of cells in an individual potential space from an input

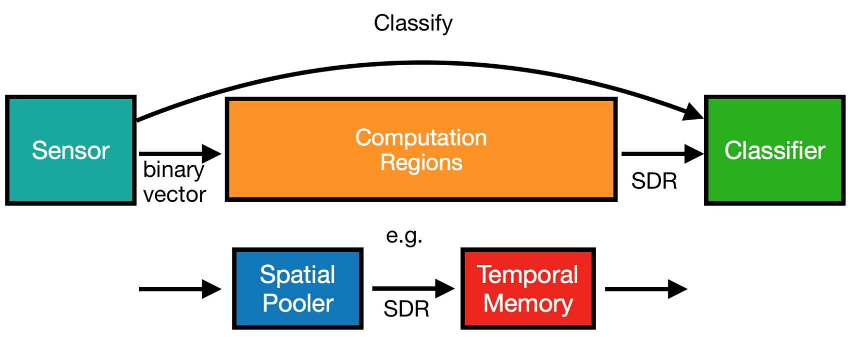

Figure 2.2: General Structure in a HTM: A sensor provides data, which is computed by regions such as the Spatial Pooler or the Temporal Memory, and finally the data is classified with regards to the Sensor Region to create a readable output

region. On each step, the cells have a chance to fire if sufficient input bits are activated. When firing, the cell feed-forwards its value to the output, and adjusts its synapses' connectivity to other cells according to the Hebbian Rule of Learning: Connections to cells that previously fired are strengthened, while connections to those that didn't are weakened. A firing cell inhibits other nearby cells from firing. An algorithm called boosting encourages unused cells to fire by counting their duty cycles, which ensures an efficient use of cells as well as preventing unnecessary overload of cells already in use, increasing effectiveness in learning different patterns.

Sequence Memory The Temporal Memory (Figure 2.4) groups temporally similar data. In it, each input-cell (now considered a mini-column) is split into multiple cells sharing the same distal connections (contextual: same level on the hierarchy), but differing in proximal connections (input: lower level on the hierarchy). When a column fires based on proximal connections, it is determined which cell from the column fires based on the distal connections. When no cell can be determined, the column bursts, firing all its cells. When firing, the column feed forwards its value to the output, and adjusts its synapses' connectivity to other columns according to the Hebbian Rule of Learning.

In addition to the two data mapping algorithms, there are two other important types of regions: Sensor regions and classifier regions.

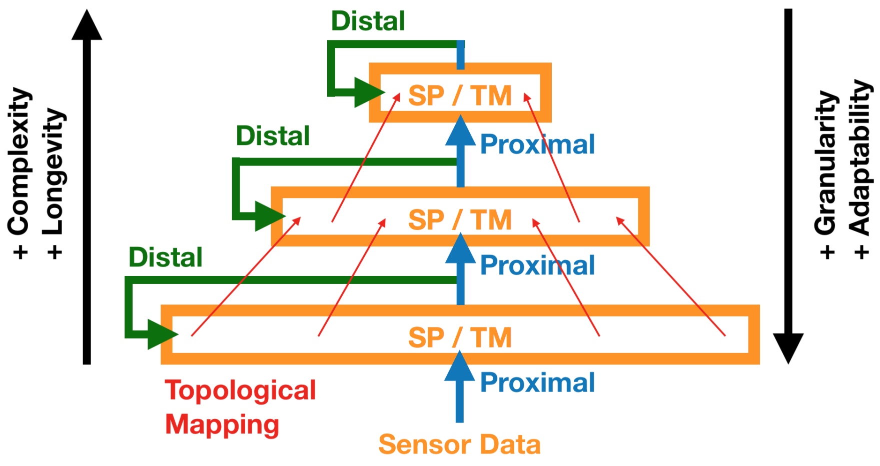

Figure 2.3: Hierarchy of a HTM: In general, hierarchies in a HTM get smaller and adjust slower the later they appear in the hierarchy. Each region layer in the hierarchy is spatially and possibly topologically mapped onto the next in a feed-forward input. Each layer also receives distal input from itself.

Sensors A sensor region contains input data fed into it by a sensor mapped by an encoder. The encoder usually has an input of itself, for example a table of scalar values or a camera input. It converts this sensory data into an SDR.

Like in the neocortex, only a fraction of cells is activated at one time, defining the concept of the input space approximately and optimally as an SDR.

Classifiers A classifier region is always connected to a sensor region. It therefore converts an SDR from another region into data resembling that provided by its sensor region. It can only classify data represented by previous states of the sensor region (buckets).

Encoders and SDRs To understand the properties and examine the quality of our encoders used in the experiments, we need to introduce a few metrics for their produced SDRs.

The sparsity is a good indicator on an SDR's quality - it should be low and preferably similar on each step [AH16]. The vector length n and Hamming Weight w can be used to calculate the sparsity s:

$$s = \frac{w}{n}$$

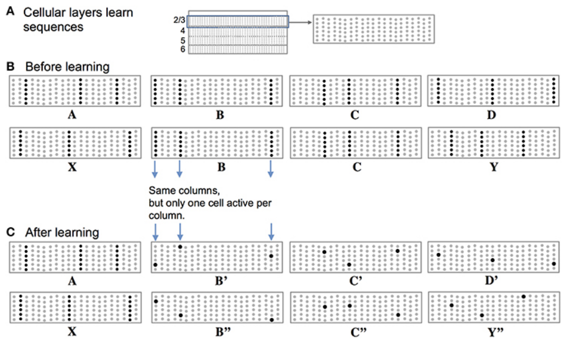
A Cellular layers learn sequences
2/3
4
5
6
B Before learning
A
B
C
D
X
B
C
Y
Same columns, but only one cell active per column.
C After learning
A
B'
C'
D'
X
B''
C''
Y''

Figure 2.4: [HA16] Temporal Memory in a HTM: A Temporal Memory can differentiate between different sequences sharing Inputs

A difference score can be used to compare different SDRs in the same model. This is especially useful to compare the accuracy of a prediction against the real data that is acquired later. We define the difference score d as the euclidean distance between two vectors v_1 and v_2:

$$d = \|v_1 - v_2\|$$

3 Setup

There are 3 key components required in the setup of the experiments: The implementation, its inputs and its outputs. In the following chapter we discuss what is required to begin with the experiments.

3.1 Software / Implementation

Experiment Framework After initial setup, each experiment runs through a main control loop that is adjusted to run at a set frame rate. Each step, the current image is captured, the network run and the prediction extracted, and values shown to the user for evaluation.

The framework used to interface with NuPIC is the Online Prediction Framework (OPF). The OPF is the python framework of Numenta created to interface with the core implementation of NuPIC. For one, it contains important methods making it simple to interact with NuPIC using python. However, it also contains additional functionality that is designed to aid the user in common tasks performable with the NuPIC framework, such as error metrics, prediction interpretation and an easy setup of networks.

The execution flow is set up as:

```
# Set Up
read_command_arguments()
register_components()
network = setup_network()
pause(until=key_is_pressed)

n = 0
last_predictions = []

while True:
    start_time = current_time()

    # Run the step
    image = Camera.capture()
    prediction = network.run(n, image)

    # Interface to the user
    show(blue=image,
        green=prediction,
        red=last_predictions[n - prediction_steps])
    check_key_presses()
    last_predictions.append(prediction)

    # Prepare for the next step
    frame_time = current_time() - start_time
    # If any frame computation takes longer than the frame_time,
    # we skip ahead to the next planned frame
    n += floor(frame_time * frame_time);
    sleep(frame_time % frame_time)
}
```

3.1.1 HTM Network

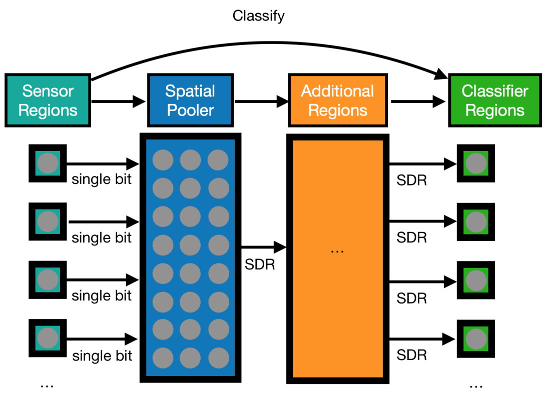

Figure 3.1: HTM Network Structure for the Experiments

In the network (Figure 3.1), each pixel has one corresponding sensor and one classifier region. When running a step, the camera input is captured and split up into individual pixels, each of which is assigned onto its corresponding sensor region's data source. These are linked to a Spatial Pooler as feed-forward. Depending on the experiment, more regions might be inserted at this point. Finally, the data from the last regions is feed-forwarded to each classifier region. Each experiment may slightly vary from this setup. The parameters for each region are discussed in the individual experiments' sections.

3.2 Network Inputs

For each experiment, a 32x32 size ($n = 1024$) image sensor in binary color space is attached to the network's sensors to compute. The method of image capture differs from experiment to experiment. In experiments capturing real-time data, 5 frames per second are captured and computed.

We chose this resolution as lowering it below 32x32 increases the ambiguity of human visual observations, while increasing it significantly increases rendering times. It should be possible to run these networks at different resolutions, although some parameters might have to be adjusted to optimize against the new resolution.

3.3 Network Outputs

For human visual observations, the captured image data is shown in *blue* and overlayed with the 5-step predictions computed by the network. The prediction created in the current frame is shown in *green*. The prediction created 5 frames earlier, thus predicting the current frame, is shown in *red*.

For evaluation, we calculate the distance score between the image matrices of the current input and the input predicted 5 steps earlier and average it with a number of other recent distance scores to reduce noise in prediction accuracy. An average distance of 0 is to be desired.

3.4 Changes to the OPF

Many of the OPF's methods are restricted to single-value functionality, as opposed to binary vectors as required in this thesis. For example, it only supports the prediction and classification of just one value per sensor region. Therefore it is not possible to predict multiple distinct values in the same region. To be able to create a setup capable of predicting motion using binary vectors, we need to make some adjustments to the OPF that are not relevant to the thesis' data or evaluation, but nonetheless required for its execution.

These are the required changes:

1. The SDR Pass-through encoder only allows prediction of a single value. A custom Encoder needs to be implemented to allow for prediction of binary vectors.
2. Spatial Pooler regions are restricted to input from a single region. A custom spatial pooler region needs to be implemented which allows input from multiple regions.
3. Spatial Pooler regions don't support topology. Topology support needs to be implemented for Spatial Pooler regions.

4 Experiment 1: Running Bar

The first experiment is designed to test the capability of a HTM to handle topological input for an online 5-step prediction.

4.1 Setup

In the experiment, a bar runs from left to right, one pixel per step, cycling back to the start at the end. This means the animation has 32 distinct repeating frames and could be described by a stateless machine.

To make the experiment topological, frames must have large overlap sets with similar frames, and small overlap sets with unsimilar frames. To ensure this, the bar is set to a width of 3 pixels ($w = 96$). Exact same frames share all input bits, 1-step distant frames share $\frac{2}{3}$ of the input bits, and 2-step distant inputs share $\frac{1}{3}$ of the input bits. All other frames have no overlap. No additional information, such as frame numbers, are given to the network for the prediction.

4.1.1 Input Details

The input vector is fully digitally computed. No additional sensors are required.

The input vector length $n = 1024$ and the bar set cardinality $w = 96$ enable us to calculate the constant sparsity s:

$$s = \frac{w}{n} = 0.09375$$

Frames share 2 cells with the frame before, thus also giving us a constant frame to frame difference score d:

$$d = \|v_0 - v_1\| = 8$$

4.1.2 Network Details

In this experiment, each frame has a deterministic following and leading frame, eliminating the need for a temporal component. Therefor, the network consists of only the

Sensor Regions, a Spatial Pooler, and the Classifier regions. A full list of parameters can be found in appendix section .1.

The HTM network is set up to be very minimal in order to get it to allocate exactly one cell per state to allow for a high accuracy prediction. With a highly minimal setup, we can closely examine which parameters influence what aspects of the prediction.

Spatial Pooler The Spatial Pooler is, in general, defined by very high values. The boost strength of 100 ensures that all cells are used very efficiently - that is, each of the 32 cells is responsible for interpreting exactly one bar position. Other values such as the potential connection ratio of 1 and the global inhibition are required to enable this. However, it also makes the network highly specialized to this task. When other parameters such as the cell count are modified, prediction accuracy very quickly deteriorates. Other parameters are optimized against performance but not required at specific values.

Classifier It is possible to speed up or slow down learning speed when adjusting the *alpha* value of the classifier: A higher value (to its maximum of 1) speeds it up, while a lower value slows it down.

4.2 Observations

These are our observations made from visual and statistical analysis of the execution of this experiment:

Observation 1.1	The bar starts at the leftmost position, moving to the right one pixel each step. At this point, each pixel gets activated once and begins constantly predicting its activation, ignoring the position of the bar (Figure 4.2).
Observation 1.2	After the first cycle on the second step, the HTM begins predicting the bar's 5-step future position to be 5 pixels to the right of the bar.
Observation 1.3	Every prediction left of the bar is weakened as passes each pixel, with the notable exception of the first 5 pixels sharing the same starting point in decrease (Figures 4.3 and 4.4).
Observation 1.4	After each cycle except the first, just before completion, the prediction flashes the whole screen (i.e. weaker prediction) and the bar loses accuracy, predicting an additional pixel to the right, subtracting the chance from the correctly predicted left pixel (Figure 4.5).

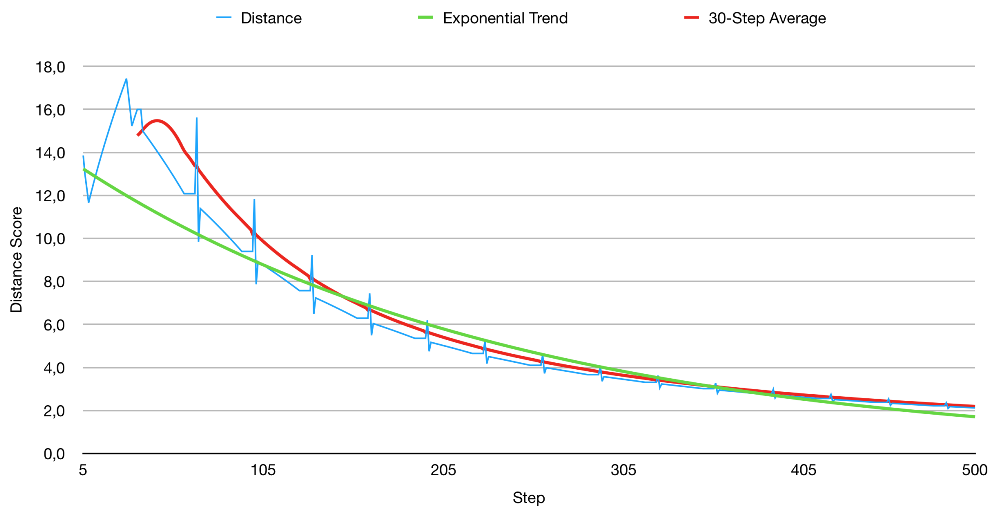

Figure 4.1: Learning Progression

Different steps of the experiment. Blue: Input Data; Green: 5-Step Prediction

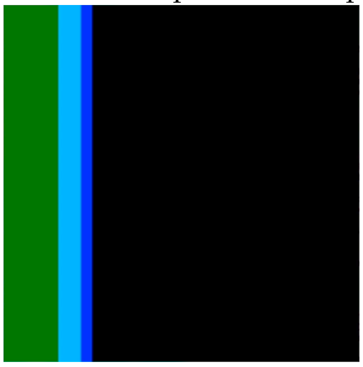
Figure 4.2: The 6th step of the 1st cycle

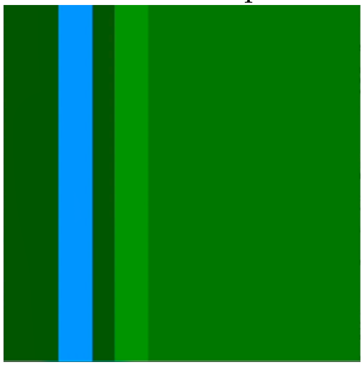
Figure 4.3: The 6th step of the 2nd cycle

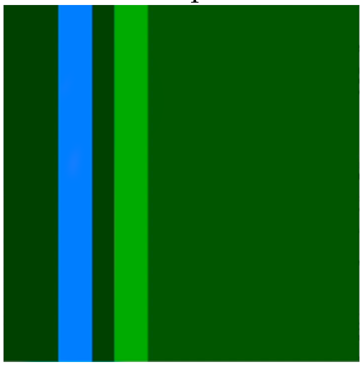
Figure 4.4: The 6th step of the 3rd cycle

On the first step of the next cycle, the prediction instead flashes dark (i.e. stronger prediction), and the bar loses accuracy, prediction an additional pixel to the left, subtracting the chance from the correctly predicted right pixel (Figure 4.6). Afterwards the prediction continues as usual (Figure 4.7).

Observation 1.5 This effect gets weaker each cycle, exponentially approaching 0 (Figure 4.1).

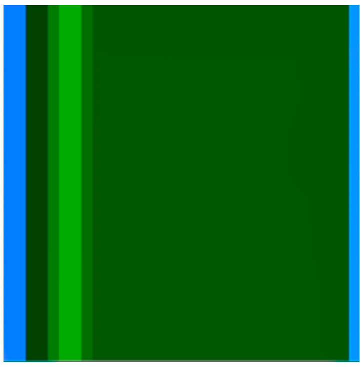
Figure 4.5: The on flash

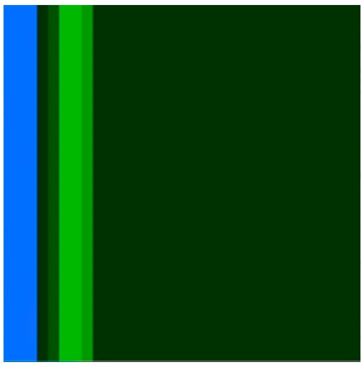
Figure 4.6: The off flash

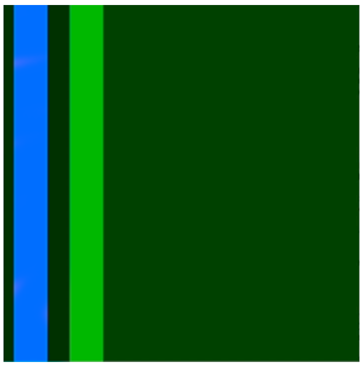
Figure 4.7: Cycle Continuation

After a few cycles (depending on classifier alpha), the bar is constantly predicted in the right location without any ambiguity. All bar pixels are predicted to be approximately 1, while all other pixels are predicted to be approximately 0.

4.3 Evaluation

In the first cycle, each pixel that was passed is colored green (Observation 1.1). This can be explained by the network not having any concept of topology at first, and each step (save for the overlapping pixels) being a completely new input for it. Observation 1.2 confirms this assumption, since when the network sees inputs it has seen before, it starts correctly predicting future positions of the bar. As the correlation between bar and future pixel states is reinforced, prediction strength in other places slowly decreases. The first and second pixel to the right of the bar share an input overlap with it, which explains why they share the prediction decrease along with the exact bar pixels (Observation 1.3).

After each cycle, the screen flashes first bright and then dark. This can be explained by the fact that the bar begins at this position, and thus the step before the origin shares its prediction with the beginning of the animation, which introduced the network with random values. This is confirmed with Observation 1.5, as the effect weakens over time until it is no longer recognizable.

This experiment shows the capability of a HTM to predict a simple animation with high accuracy even with a minimal number of cells and segments. For optimal SDRs, there should be a low and preferably constant Hamming Weight each step [AH16]. In this experiment, this could be achieved with a constant sparsity of 0.09375.

The performance was acceptable for our purposes - it can be run on a modern home computer without any problems at the set 5 frames per second, with the network computing the predictions online.

5 Experiment 2: Cyclic Photo Series

With the general capability of handling topological input for an online 5-step prediction proven, we decide to expand upon other aspects of the input that the system will need to handle if we want to predict motion with visual camera input.

The key aspect missing in Experiment 1 to show if HTM networks are capable of handling real-world data is complexity. A setup like experiment 1 can easily be predicted using a simple automaton based solely on the position of the bar, but real world data cannot so simply be predicted as it is both spatially and temporally much more complex.

5.1 Setup

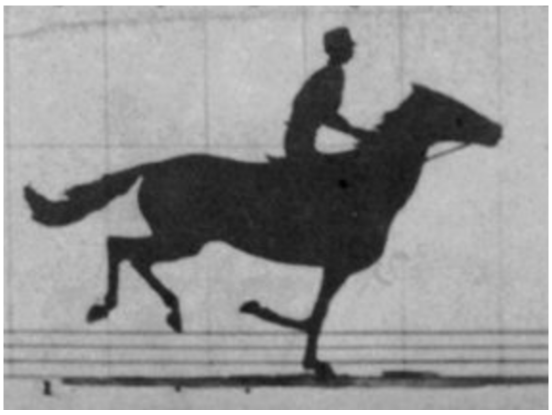

Figure 5.1: [Muy78] Single frame of Eadweard Muybridge's "The Horse in Motion"

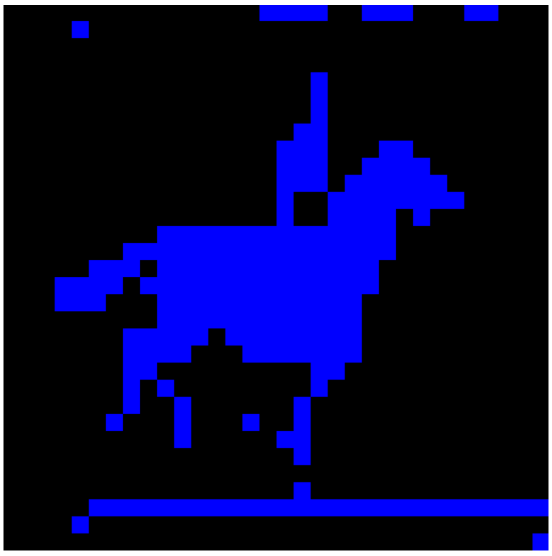

Figure 5.2: Single frame encoded as input data

To examine whether we can realistically introduce real-world data, we must add to both the spatial and temporal complexity of the input. To add to spatial complexity, we introduce Eadweard Muybridge's photos "The Horse in Motion" (Figure 5.1), a series of pictures showing a horse and its rider in various steps of motion. With the photos Muybridge showed that the human eye can perceive motion even with a series of pictures shown in quick succession. As they are actual photos, they most closely resemble what we want real-world input data to look like and can thus be used to examine whether the network is capable of handling the complexity. To add to temporal complexity, we choose a method that forces the network to utilize a temporal aspect for optimal prediction even without adding noise to the input: The video will be played forwards and backwards in succession. This ensures that while the input stays spatially consistent and interpretable by the human eye, it contains repeating frames that prevent a good prediction based solely on the current input data.

The final input consists of 18 repeating frames and could be described by a state machine.

5.1.1 Input Details

To encode the image into a binary vector, it is first scaled into size (32x32 pixels). We then convert the color space to grayscale and finally apply a threshold of $value < 150$ (Figure 5.2).

The input vector length $n = 1024$ and each of the images Hamming Weights w allow us to calculate the minimum sparsity s_{min}, the maximum sparsity s_{max} and the average sparsity $\bar{s}$:

$$s_{min} = \min \frac{w}{n} = \frac{214}{1024}; \; s_{max} = \max \frac{w}{n} = \frac{360}{1024}$$
$$\bar{s} = \overline{(\frac{w}{n})} = \frac{242}{1024}$$

We can also calculate the difference scores between frames for future comparison. The minimum difference score d_{min}, the maximum difference score d_{max} and the average difference score $\bar{d}$ calculate as such:

$$d_{min} = \frac{8}{1024}; \; d_{max} = \frac{14}{1024}$$
$$\bar{d} = \frac{11}{1024}$$

5.1.2 Network Details

In this experiment, the state cannot be purely determined from just the shown image, in contrast to experiment 1. This means a Temporal Memory is required for accurate prediction. It is inserted in between the Spatial Pooler and the Classifier regions.

A full list of parameters can be found in appendix section .2.

Spatial Pooler In addition to the Temporal Memory region, a few parameters are required at specific ranges to allow for accurate prediction of the reversal of the image sequence. The boost strength must range between about 5 and 20, and the column count cannot be too low. With differing parameters the network might continuously predict both leg positions at once, or, at worst, always the last position the legs assumed, which with a repeating animation is always incorrect. Since the input is now topologically more complex than in the previous experiment, we change the Spatial Pooler to reflect this aspect. It is switched to a topological setup by restricting each cell in its available input space.

Temporal Memory The values in the Temporal Memory are mostly not required at specific values. Each of the values can contribute to a quicker learning speed when set to an optimal value. Most significantly, the segments per cell must be above 2 and the max seq length must be above about 20 to ensure good results. An optimization algorithm suggests a largely independent optimization per variable is possible.

5.2 Observations

These are our observations made from visual and statistical analysis of the execution of this experiment:

Observation 2.1	At the very beginning, difference scores for prediction and actual state have a very large range (Figure 5.3.
Observation 2.2	The visual prediction roughly equates to an accumulation of previously seen inputs.
Observation 2.3	After a full cycle, the difference scores begin to exponentially improve and approach 0.
Observation 2.4	As the difference scores improve, the difference score noise does as well.
Observation 2.5	At first, one can observe a second pair of legs moving opposite the first. It is corrected through more cycles (Figures 5.4 through 5.6).

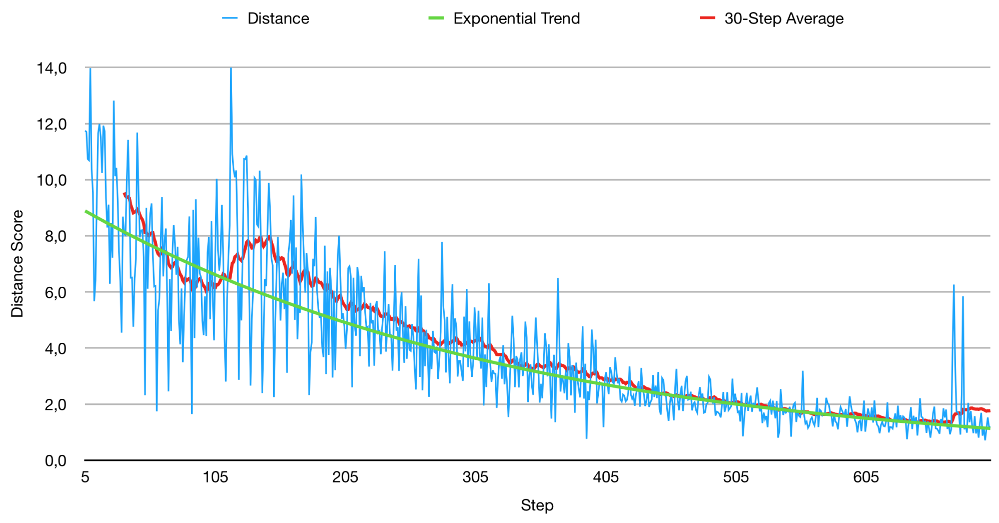

Figure 5.3: Learning Progression

Different steps of the experiment. Blue: Input Data; Red: Prediction 5 steps earlier

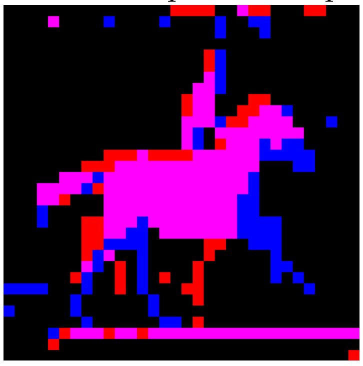

Figure 5.4: Step 5 of the Experiment

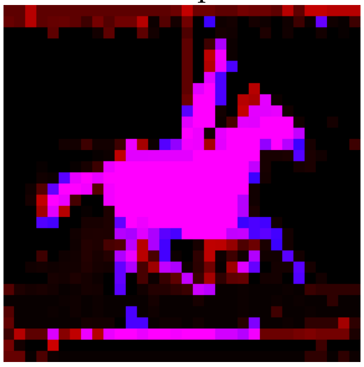

Figure 5.5: Step 50 of the Experiment

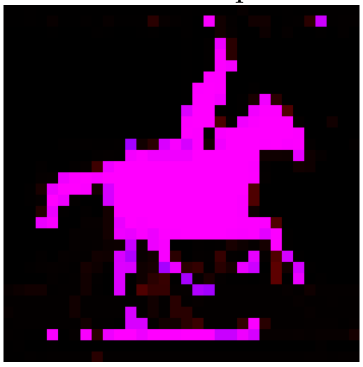

Figure 5.6: Step 500 of the Experiment

5.3 Evaluation

At the very beginning, the prediction is roughly equal to the states that the network has seen previously (Observation 2.1). This is to be expected, as seen before in Observation 1.1, because the network can confidently predict pixels that stay the same value over time (Observation 2.2) - i.e. have high overlaps in between all frames and are thus regarded as semantically similar to each other. However, the pixels that do change cause noise in the prediction until the first cycle is complete. After the first cycle, noise and prediction accuracy increase (Observation 2.3). This can also be observed visually by the horse's legs appearing blurry in Figure 5.5.

Observation 2.5 is significant because it highlights the importance of a temporal aspect in the network. Especially with differing values in boost strength, Spatial Pooler resolution or Temporal Memory column count, the network can quickly lose its ability to predict the inversion of the animation. A good set of parameters however can give the network the ability to correctly predict even inversions of an animation.

After about 22 cycles / 396 steps there is little noise left and the network has extremely accurate predictions for the animation (Figure 5.3). This suggests that the network is capable of handling more complex animations than the one used in Experiment 1, both spatially and temporally.

The experiment required better tweaking of parameters compared to Experiment 1 to reach good accuracy after a minimal amount of time. However, apart from a few parameters integral to predicting the animation reversal, most merely improved upon the speed of learning the pattern. In terms of performance, it is possible to compute this experiment with 5 frames per second on a modern home computer.

6 Experiment 3: Real World Data

With the complex inputs both spatially and temporally proven to work with our setup, we decide to expand upon the final missing aspect of motion prediction: Input variability. This is arguably the most important and most difficult aspect since without it, simpler prediction algorithms would suffice to predict small input sets.

6.1 Setup

The experiment is split into repeating iterations of the same movement. On each iteration, an arm is moved across the frame from right to left, taking a total of about 5 seconds to move through it. This action is performed in repetitive fashion to ensure an easily predictable input, though it is not machine guided - no timer, guideline or reference is used to ensure the statisticity of the action. The captured data is thus completely subjected to human motion and not describable by a state machine. The input contains no repeating iterations, though repeating frames are theoretically possible with a low chance.

6.1.1 Camera Details

To capture image data, a camera mounted to a stand and pointed down towards a white table with a distance of 58cm (Figure 6.1 and 6.2).

6.1.2 Encoder Details

To encode each image into a binary vector, we want to make sure the algorithm does as little pre-interpretation on the image to avoid evaluating two algorithms at once (the encoding algorithm as well as the network). At the same time, though, it needs to be evaluatable. The algorithm needs to produce very similar results on every run, not influenced by different lighting or backgrounds, so that the input data always reflects the actions performed with as little alteration as possible. To ensure this, we use a mask derived from a threshold on a background subtraction. We capture the view of the camera (Figure 6.3), blur the image to reduce noise (Figure 6.4), subtract it from a previously determined background reference (Figure 6.5) and threshold the colors

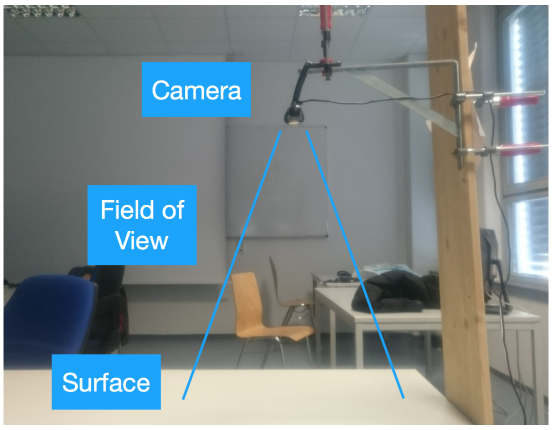

Figure 6.1: Experiment Setup: The Camera is mounted 5cm above a white table and directed towards it

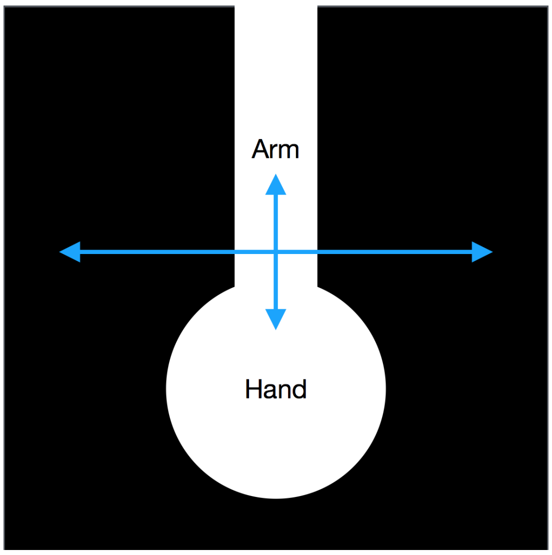

Figure 6.2: Abstraction of the captured image: An arm is moved through the frame

(Figure 6.6), and finally scale and combine the color channels to get the final encoded image (Figure 6.7).

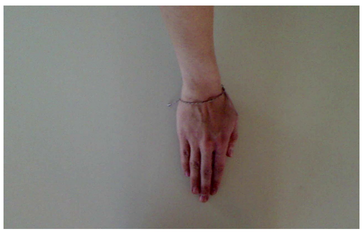

Figure 6.3: The raw image captured by the camera

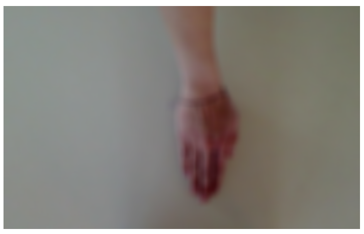

Figure 6.4: 32x32 Blur on the captured image

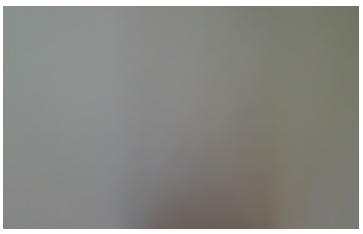

Figure 6.5: Blurred Reference Background

Since the difference between the background and the current image is used without pre-interpretation as input, sparsity and difference scores vary among frames. However, since there is only one object involved in the experiment, the sparsity varies only between the mean sparsity (arm fully in frame) and 0 (arm out of frame). The difference score varies depending on the arm speed, the value of which denotes semantic data in itself (arm position change between frames) and is thus acceptable as well. Since we

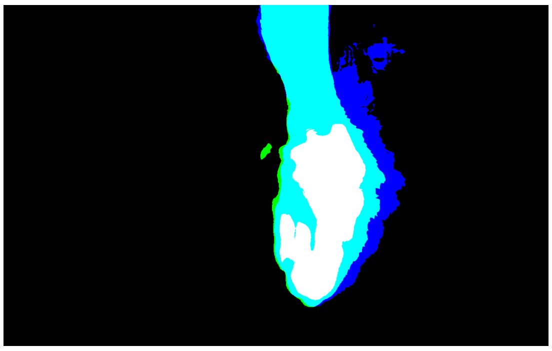
Figure 6.6: 0.01 Threshold Difference of the captured image and background

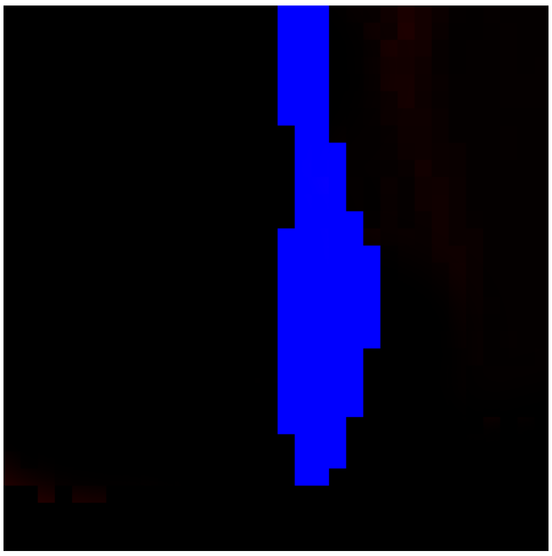
Figure 6.7: Encoded Data for the Network

only want to predict the position of the arm, it is acceptable to have a low sparsity and predict few patterns when it is not in frame [AH16]. In some frames, the image has a Hamming Weight of 0; in others, the arm and arm are in full frame, taking up about 10% of the total image size in pixels. Since the speed varies only slightly and no frames are skipped, however, the difference scores stay roughly the same.

6.1.3 Network Details

In this experiment, two components are added to the previous network setup: An untopological Spatial Pooler is inserted between the topological Spatial Pooler and the Temporal Memory, which additionally receives feed forward input from two new sensor regions that encode the grey value center x and y of the raw input using a scalar encoder. As the arm is the only object in the encoded camera input, the center values represent its position and we thus predict that it will help the network understand the action faster if it receives the position as well. The center is encoded using a $w = 31$ and a $radius = 5$.

Another region that we introduce in this experiment is the edge filtered sensory region. It receives the same input as the sensory pixel region, but applies an edge filter to it, blurs and finally thresholds the result again. This results in thicker edges, so that edges at similar positions share bits and thus semantic meaning for the network. However, feeding this either as the sole input for the network, or as an additional feed-forward in the same way as the center sensory regions, significantly worsened the prediction accuracy with different permutations of parameters. We thus do not include the edge filtered sensory region in the final network.

Most of the parameters contribute to quicker and more stable learning speeds, and a lot of them have minimal effects. With a growing number of parameters, this is to be expected as the network always has other data sources to base its predictions on. Since the actions in this experiment never change, except for minor variations in each iteration, the permanence decrease values are low compared to the increase values. This is the case in all of the computation regions. A full list of parameters can be found in appendix section .3.

Joining Spatial Pooler To encourage cells in this Spatial Pooler to make use of both raw image- and center data at once, the joining spatial pooler is untopological.

Temporal Memory To ensure the network can learn a full iteration of movement and wait adequately long for the arm to come into frame after the previous one, relatively large numbers in sequence length, infer backtracking length and learn backtracking length are required. Though the network can make acceptable predictions with small numbers in infer and learning backtracking count, high values are required for very good predictions. Since the network is required to adjust its predictions based on the speed of the arm in frame, we set the max number of synapses to a high amount compared to the column count.

6.2 Observations

These are our observations made from visual and statistical analysis of the execution of this experiment:

Observation 3.1	At first, the prediction roughly equates to an accumulation of previously seen inputs. This improves after the first iteration (Figures 6.9 through 6.11).
Observation 3.2	After almost a full iteration, a predicted arm can be observed on the right of the image.
Observation 3.3	This prediction isn't on the far right but instead a little bit indented to the left.

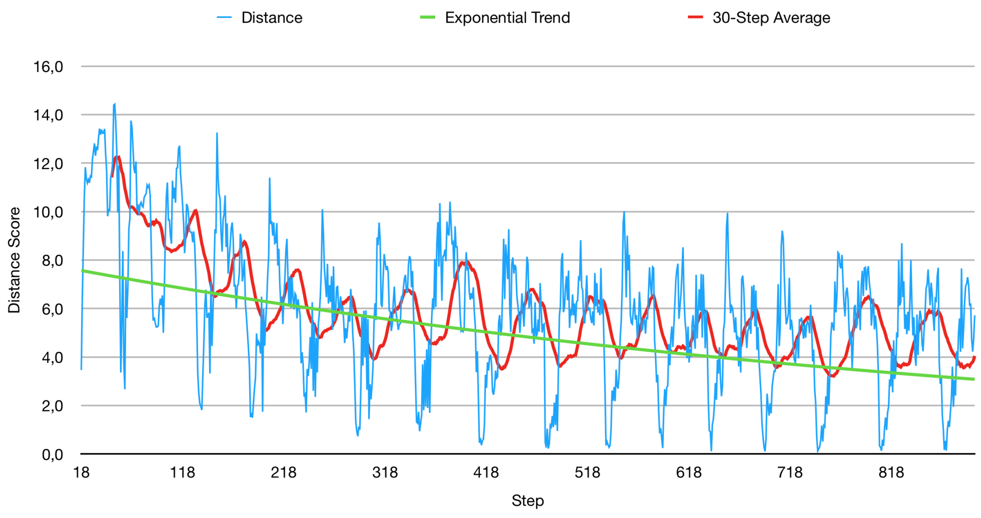

Figure 6.8: Learning Progression

Different steps of the experiment. Blue: Input Data; Red: Prediction 5 steps earlier

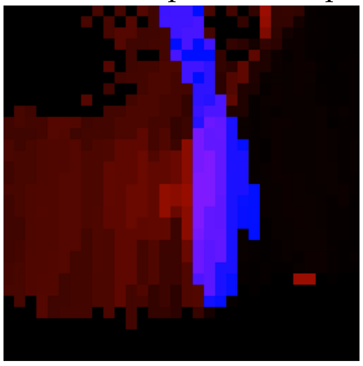

Figure 6.9: Iteration 2 of the Experiment

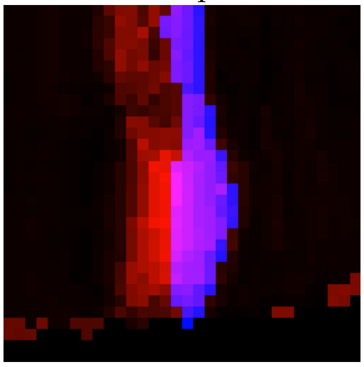

Figure 6.10: Iteration 4 of the Experiment

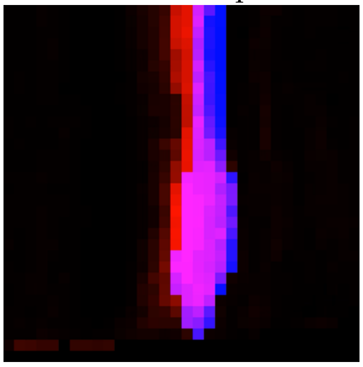

Figure 6.11: Iteration 8 of the Experiment

Observation 3.4 If pixels that where never activated before activate in an input, they behave roughly similar to Observation 3.1, though adapts more quickly.

Observation 3.5 At first, difference scores never reach near-zero values, but after a few iterations they regularly do.

Observation 3.6 The difference scores never approach 0, as opposed to the previous experiments (Figure 6.8). They also maintain a lot of noise.

Observation 3.7 The predicted arm is on average a little bit larger than the actual arm and blurs towards the outward (Figure 6.10).

6.3 Evaluation

Observation 3.1 is a repeated observation from Observations 1.1 and 2.2: After initial activation, the network seems to continue to predict the activation of most pixels until they are activated a second time.

The new network setup greatly improves learning speed and prediction accuracy as opposed to a single Spatial Pooler linked to a Temporal Memory. The center sensory regions help speed up learning speed and accuracy by a significant amount, though they are not required for good prediction results. The edge filters significantly worsens the result with different permutations of hierarchies and parameters and were thus deactivated.

Observation 3.2 highlights the influence of the temporal memory in the network. Since any given iteration takes roughly the same time, the network has an idea of when to predict the arm coming into frame. Since the prediction is set to 5 steps / 1 second into the future, the shown prediction can never be on the far right (Observation 3.3), since the arm will have moved to the left already after the 5 steps following the initial step with the arm coming into frame has passed.

Since the network has no initial sense of topology, when a sensory region activates for the first time, it behaves as if the animation had just started, regardless of whether previous iterations had already passed. Other cells are at later points already committed to prediction specific positions of the arm, however, which leads to quicker learning for the newly activated pixels according to the Hebbian rule of learning (Observation 3.4).

A prediction approaching perfect distance scores of 0 are to be preferred, of course, but it is to be expected that this is unrealistic as a lot of complex factors are involved in human motion. To reach a maximum prediction accuracy with minimal information, the network must make trade-offs with outwards pixels because of varying motion speed by the arm. This approach shows in Observations 3.6 and 3.7. However, the network also learns where to make trade-offs and where it can confidently predict specific values. After a few iterations it learns that there is always a minimum pause when the arm disappears from the screen - which is caused by the arm moving back to the other side of the camera. This optimization is shown by Observation 3.5, where the network slowly begins to confidently predict all pixels to be inactive.

After about 20 iterations / 1000 steps, the network has already reached stable average distance scores of about 5 and continues to further improve after more iterations. It seems to converge to an average distance of about 3.5.

The results of this experiment show that the setup utilized in this experiment is suitable for real-world data and can manage realistic predictions within a small number of iterations. The parameters did not have to be tweaked to the same level of exactness of experiment 2, possibly because the temporal component was more similar to experiment 1 than experiment 2 in complexity. However, the performance went further down from experiment 2, disabling the capability to run this experiment in real-time (a frame took approximately 300 milliseconds to compute on a modern home computer).

7 Network Capability: An Examination

The evaluation of experiment 3 shows that the network is now capable of handling real-world data. We will draw a comparison between the different iterations of the network, and test the last iterations against different kind of tasks to examine the extent of its capability.

7.1 Comparison of the Networks

Compared performance of the 3 developed Networks over the different experiments

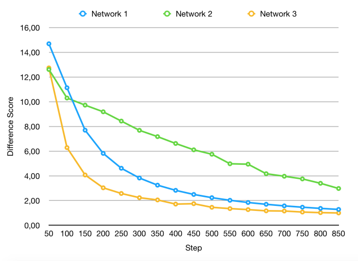

Figure 7.1: Experiment 1

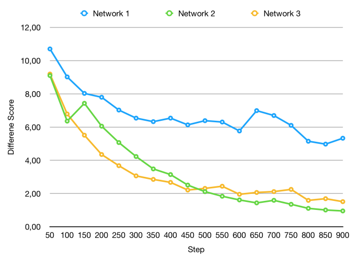

Figure 7.2: Experiment 2

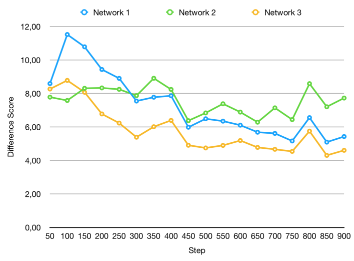

Figure 7.3: Experiment 3

To investigate a possible improvement in prediction accuracy over the previous iterations of the network, we compare them to each other using the distance score in all 3 experiment setups.

The comparison between all 3 networks in different tasks shows that with each change over the former experiment, the setup gets better at handling different tasks with greater speed (Figures 7.1 through 7.3). Since the input for experiment 1 is very similar to the input in experiment 3, its network handles that input data surprisingly well. However, it doesn't doesn't allow for a wider range of tasks, as its performance in experiment 2 shows: Its average distance scores never improves beyond a general average of 5.5. The second network handles the first two experiments decently but doesn't make strong improvements with the data from experiment 3. The third network can handle all 3 tasks with relative ease, and even comparable performance to the networks specifically designed to accomplish those tasks.

7.2 Additional Experiments

To examine the capability range of the third network, we will also perform additional experiments and test its learning progression within that input space.

7.2.1 Circular Motion

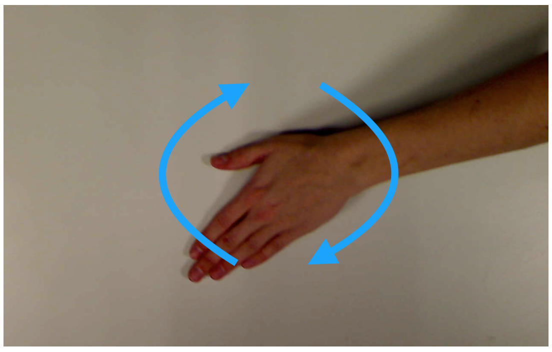
Figure 7.4: Setup of the Circular Motion Experiment

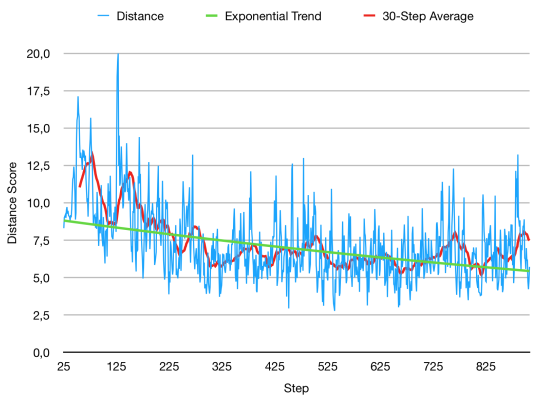

Figure 7.5: Learning Progression of the Circular Motion Experiment

This experiment tests if the network is capable of predicting a different task to Experiment 3. In this Experiment, the arm is not moved from right to left; rather, it stays in the center of the frame and does a circular motion (Figure 7.4). Like in experiment 3, this motion is not machine-guided and thus completely subject to human motion.

Evaluation In the trend (Figure 7.5) there is a significant improvement over time as the network learns the pattern of the moving hand action. However, since the center of the circulation was not marked, the rotation center drifted over time which results in worse prediction accuracy from the network. Still, both noise and average value of the distance score improve over time and suggest the network is decently capable of predicting this action without any additional tuning.

7.2.2 Varying Speeds

To test how well the experiment adjusts to even larger differences in motion, we introduce a randomized factor into the setup of Experiment 3. The arm also moves

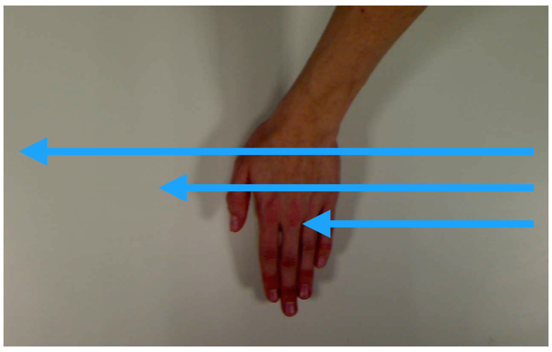
Figure 7.6: Setup of the Varying Speeds Experiment

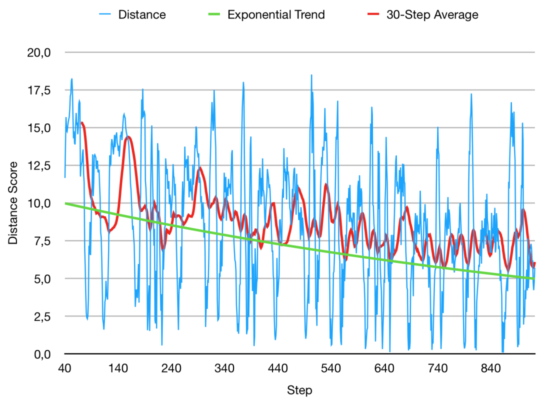

Figure 7.7: Learning Progression of the Varying Speeds Experiment

across the frame from right to left, but rather than trying to keep a roughly constant speed, it is moved with highly varying speeds (Figure 7.6). Thus, iteration length ranges from about 1 to 10 seconds. Overlap Scores also strongly differ between iterations in this experiment.

Evaluation While not immediately visible when observing just the raw distance score data, the trend and the 30-step average show a clear improvement overtime in this experiment (Figure 7.7). It can therefor be regarded as being an acceptable prediction system of this experiment. It is however notable that the network requires a lot more time and has a lot more noise in the accuracy of its predictions compared to the results of Experiment 3. This can be explained by the additional randomized variable introduced to make prediction harder.

7.2.3 Two Hands

In this experiment we test how well the network deals with two hands in the frame at the same time, and how well it deals with a reversal of motion similar to Experiment 2. The hands are first moved into the frame and then continuously move back and forth while staying roughly parallel to each other (Figure 7.8). Like in the other experiments, no machine guidance was used and a high variance in actual motion is to be expected.

Evaluation Even after a lot of iterations, no clear trend is visible in the prediction of this motion (Figure 7.9). This suggests that the network is well-suited for predicting a repetitive real-world task with a reversal of motion.

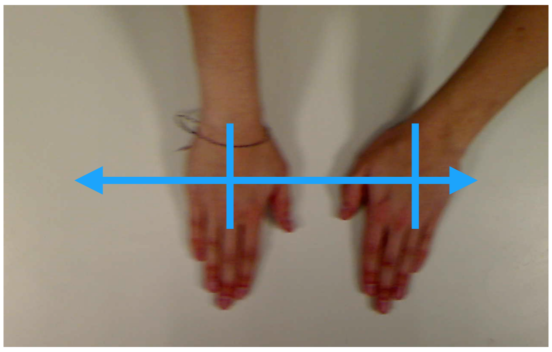
Figure 7.8: Setup of the Two Hands Experiment

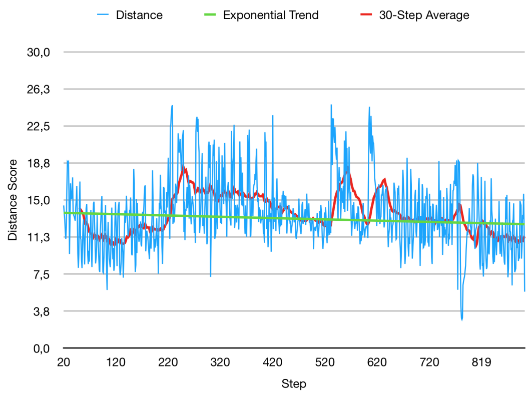

Figure 7.9: Learning Progression of the Two Hands Experiment

7.2.4 Basic Assembly

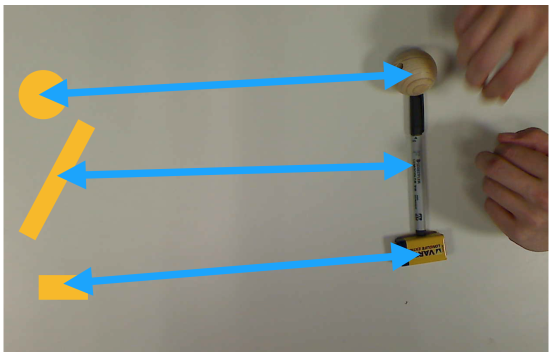
Figure 7.10: Setup of the Basic Assembly Experiment

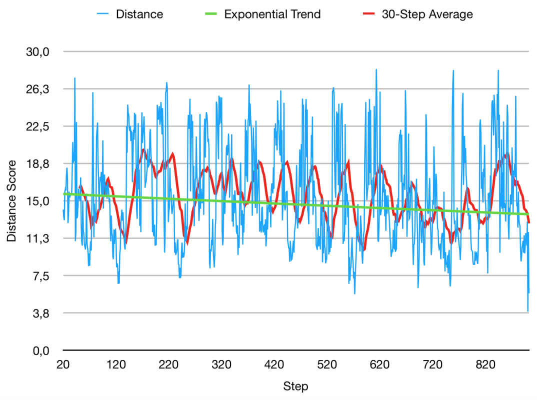

Figure 7.11: Learning Progression of the Basic Assembly Experiment

This experiment introduces a complex task for the machine to predict. On each iteration an object is assembled (Figure 7.10) and then disassembled. The assembly consists of 3 distinctly shaped objects lying in a set order on one side with a random orientation. The pieces are then grabbed and moved to the other side of the frame, where they are assembled in a set order and orientation (though with varying positions of the final assembled object). Finally, the object is disassembled and the parts moved to the other side of the frame again.

Evaluation For this experiment, the distance scores of prediction are a lot higher than in all previous experiments (Figure 7.11). However, the data shows clear peaks and caves in performance, which suggests that the network picks up on certain parts of the assembly which it can predict really well, while it is a lot worse at other parts. The trend shows a very slight downwards slope, but it is likely that it will not be able to improve much beyond its current state. This suggests that the network is not well-suited for the prediction of complex tasks such as assembly of machine parts.

8 Conclusion

We presented a system using a HTM network that was capable of predicting a simple, repetitive animation that would be predictable by a stateless machine. We then expanded upon the system to make it capable of predicting a cyclic animation that requires a stateful machine to predict. We extended the final system to enable it to predict a repetitive task from real-world data.

The results from Experiment 3 as well as those from the additional experiments show that the system presented in this thesis is capable of predicting a number of different tasks within the problem space of real-world visual human motion prediction. These tasks include running a hand over the screen at semi-constant or varying speeds, and rotating the hand inside the frame without a set pivot point. It surpasses simple baselines such as the 1-second constant pose predictor [MBR17] in these tasks and confirms Zhang et al.'s [Zha+09] and Ugolotti et al.'s [Ugo+13] findings that systems using HTMs are viable approaches to human action observation. The system does not require specific adjustments for each task, expert knowledge or complex algorithms modifying the input data for reasonable predictions. This makes it well-suited for use in this problem space as the sole major logical component of a prediction system.

8.1 Future Work

The concept of a HTM is adjusting with future discoveries in neurological science, and its parameters, algorithms and general structure are subject to change in the future. Numenta's implementation NuPIC might change along with it, and it might also not stay the only implementation of this concept. With future implementations, it is possible to review the problems presented in this thesis to evaluate whether distance scores can be minimized further and whether similar systems can produce better predictions.

It is likewise plausible that the performance of these implementations improves, allowing for larger-scale vision based data such as bigger resolutions, better frame-rates, or a larger color space. Predictions made by this network might improve with more data available to it. It might also be possible to test longer-term predictions when short-term predictions such as 1 second predictions are optimized.

Finally, there are a lot of possible configurations of the HTM hierarchy, including the introduction of additional sensors that provide pre-interpreted data based on analytic algorithms or expert knowledge. This kind of data might improve the learning speeds of the network and even allow it to solve larger problem spaces.

Appendices

.1 Experiment 1 - Network Parameter Details

Table .1: Spatial Pooler

boostStrength	100
columnDimensions	[sensor_cam.width]
globalInhibition	1
numActiveColumnsPerInhArea	1
inputDimensions	[sensor_cam.width * sensor_cam.height]
potentialPct	1.
potentialRadius	sensor_cam.width * sensor_cam.height
synPermActiveInc	0.5
synPermInactiveDec	0.5
synPermConnected	0.1
wrapAround	1
seed	50
spatialImp	cpp

Table .2: Classifier

alpha	0.5
maxCategoryCount	2
steps	5

.2 Experiment 2 - Network Parameter Details

Table .3: Spatial Pooler

boostStrength	10
columnDimensions	[20, 20]
globalInhibition	0
numActiveColumnsPerInhArea	-1
localAreaDensity	.15
inputDimensions	[sensor_cam.width, sensor_cam.height]
potentialPct	.25
potentialRadius	4
synPermActiveInc	0.1
synPermInactiveDec	.1
synPermConnected	0.1
wrapAround	0
seed	50
spatialImp	cpp

Table .4: Temporal Memory

activationThreshold	20
minThreshold	3
cellsPerColumn	6
columnCount	400
globalDecay	0.0
initialPerm	1
connectedPerm	.1
maxAge	0
maxSegmentsPerCell	4
maxSynapsesPerSegment	250
newSynapseCount	20
outputType	normal
pamLength	6
permanenceDec	0.001
permanenceInc	0.2
predictedSegmentDecrement	0.0
maxInfBacktrack	10
maxLrnBacktrack	10
maxSeqLength	60
seed	50
inputWidth	1
temporalImp	cpp

Table .5: Classifier

alpha	0.01
maxCategoryCount	2
steps	5

.3 Experiment 3 - Network Parameter Details

Table .6: Center Sensors

minval	0
maxval	sensor_cam.width / sensor_cam.height
w	31
radius	5

Table .7: Image Spatial Pooler

boostStrength	.001
columnDimensions	[20, 20]
globalInhibition	0
numActiveColumnsPerInhArea	-1
localAreaDensity	.15
inputDimensions	[sensor_cam.width, sensor_cam.height]
potentialPct	.25
potentialRadius	3
synPermActiveInc	0.3
synPermInactiveDec	.01
synPermConnected	0.1
wrapAround	0
seed	50
spatialImp	cpp

Table .8: Joining Spatial Pooler

boostStrength	5
columnDimensions	[256]
globalInhibition	1
numActiveColumnsPerInhArea	-1
localAreaDensity	.12
inputDimensions	[230 + 230 + 400]
potentialPct	.25
potentialRadius	20
synPermActiveInc	0.1
synPermInactiveDec	.01
synPermConnected	0.1
wrapAround	0
seed	50
spatialImp	cpp

Table .9: Temporal Memory

activationThreshold	20
minThreshold	3
cellsPerColumn	5
columnCount	256
globalDecay	0.0
initialPerm	1
connectedPerm	.1
maxAge	0
maxSegmentsPerCell	4
maxSynapsesPerSegment	500
newSynapseCount	10
outputType	normal
pamLength	6
permanenceDec	0.001
permanenceInc	0.1
predictedSegmentDecrement	0.0
maxInfBacktrack	120
maxLrnBacktrack	120
maxSeqLength	60
seed	50
inputWidth	1
temporalImp	cpp

Table .10: Classifier

alpha	0.01
maxCategoryCount	2
steps	5

Glossary

ANN Artificial Neural Network. 4, 5, *Glossary:* Artificial Neural Network

Artificial Neural Network is any neural network designed and implement by humans.. 1

CNN Convolutional Neural Network. 5, *Glossary:* Convolutional Neural Network

Convolutional Neural Network is a type of artificial neural network.. 1

Hierarchical Temporal Memory is a biologically oriented type of neural network.. 1

HTM Hierarchical Temporal Memory. iv, 2–4, 6–10, 12, 15, 16, 18, 20, 37, 46, 47, *Glossary:* Hierarchical Temporal Memory

Numenta Platform for Intelligent Computing is an implementation of a HTM.. 1

NuPIC Numenta Platform for Intelligent Computing. 4, 11, 37, *Glossary:* Numenta Platform for Intelligent Computing

OPF the Online Prediction Framework. 11, 13, *Glossary:* The Online Prediction Framework

SDR Sparse Distributed Representation. 2, 6, 9, 10, 14, 18, *Glossary:* Sparse Distributed Representation

Sparse Distributed Representation is a representation of data characterised by a low ratio of on-bits.. 1

The Online Prediction Framework is a python interface for NuPIC.. 1

List of Figures

List of Tables

Bibliography

[AH16] S. Ahmad and J. Hawkins. "How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites." In: *CoRR* abs/1601.00720 (2016).

[Coo+95] T. Cootes, C. Taylor, D. Cooper, and J. Graham. "Active Shape Models-Their Training and Application." In: *Computer Vision and Image Understanding* 61.1 (1995), pp. 38–59. ISSN: 1077-3142. DOI: `https://doi.org/10.1006/cviu.1995.1004`.

[FLM15] K. Fragkiadaki, S. Levine, and J. Malik. "Recurrent Network Models for Kinematic Tracking." In: *CoRR* abs/1508.00271 (2015). arXiv: `1508.00271`.

[Gav99] D. Gavrila. "The Visual Analysis of Human Movement: A Survey." In: *Computer Vision and Image Understanding* 73.1 (1999), pp. 82–98. ISSN: 1077-3142. DOI: `https://doi.org/10.1006/cviu.1998.0716`.

[HA16] J. Hawkins and S. Ahmad. "Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex." In: *Frontiers in Neural Circuits* 10 (2016), p. 23. ISSN: 1662-5110. DOI: `10.3389/fncir.2016.00023`.

[HAC17] J. Hawkins, S. Ahmad, and Y. Cui. "A Theory of How Columns in the Neocortex Enable Learning the Structure of the World." In: *Frontiers in Neural Circuits* 11 (2017), p. 81. ISSN: 1662-5110. DOI: `10.3389/fncir.2017.00081`.

[HAD10] J. Hawkins, S. Ahmad, and D. Dubinsky. "Hierarchical temporal memory including HTM cortical learning algorithms." In: *Techical report, Numenta, Inc, Palto Alto http://www.numenta.com/htmoverview/education/HTM_CorticalLearningAlgorithms.pdf* (2010).

[Haw+13] K. P. Hawkins, N. Vo, S. Bansal, and A. F. Bobick. "Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration." In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Oct. 2013, pp. 499–506. DOI: `10.1109/HUMANOIDS.2013.7030020`.

[HB04] J. Hawkins and S. Blakeslee. *On Intelligence*. Times Books, 2004. ISBN: 0805074562.

[Heb57] D. O. Hebb. *The Organizations of Behavior: a Neuropsychological Theory*. Chapman and Hall, 1957.

[Hub+10] M. Huber, A. Knoll, T. Brandt, and S. Glasauer. "When to assist? - Modelling human behaviour for hybrid assembly systems." In: *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. June 2010, pp. 1–6.

[Hub+13] M. Huber, C. Lenz, C. Wendt, et al. "Predictive Mechanisms increase Efficiency in Robot-supported Assembies: An Experimental Evaluation." In: *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication*. Gyengju, Korea, 2013.

[Jai+15] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. "Structural-RNN: Deep Learning on Spatio-Temporal Graphs." In: *CoRR* abs/1511.05298 (2015). arXiv: 1511.05298.

[JJM16] A. Jabri, A. Joulin, and L. van der Maaten. "Revisiting Visual Question Answering Baselines." In: *CoRR* abs/1606.08390 (2016). arXiv: 1606.08390.

[LGN13] A. M. Lehrmann, P. V. Gehler, and S. Nowozin. "A Non-parametric Bayesian Network Prior of Human Pose." In: *2013 IEEE International Conference on Computer Vision*. Dec. 2013, pp. 1281–1288. DOI: 10.1109/ICCV.2013.162.

[MBR17] J. Martinez, M. J. Black, and J. Romero. "On human motion prediction using recurrent neural networks." In: *CoRR* abs/1705.02445 (2017). arXiv: 1705.02445.

[Muy78] E. Muybridge. *The Horse in Motion*. 1878.

[Ore+97] M. Oren, C. Papageorgiou, P. Sinha, et al. "Pedestrian detection using wavelet templates." In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 1997, pp. 193–199. DOI: 10.1109/CVPR.1997.609319.

[PN94] R. Polana and R. Nelson. "Low level recognition of human motion (or how to get your man without finding his body parts)." In: *Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*. Nov. 1994, pp. 77–82. DOI: 10.1109/MNRAO.1994.346251.

[Ugo+13] R. Ugolotti, F. Sassi, M. Mordonini, and S. Cagnoni. "Multi-sensor system for detection and classification of human activities." In: *Journal of Ambient Intelligence and Humanized Computing* 4.1 (Feb. 2013), pp. 27–41. ISSN: 1868-5145. DOI: 10.1007/s12652-011-0065-z.

[WRB11] D. Weinland, R. Ronfard, and E. Boyer. "A survey of vision-based methods for action representation, segmentation and recognition." In: *Computer Vision and Image Understanding* 115.2 (2011), pp. 224–241. ISSN: 1077-3142. DOI: `https://doi.org/10.1016/j.cviu.2010.10.002`.

[Zha+09] S. Zhang, M. H. Ang, W. Xiao, and C. K. Tham. "Detection of Activities by Wireless Sensors for Daily Life Surveillance: Eating and Drinking." In: *Sensors* 9.3 (2009), pp. 1499–1517. ISSN: 1424-8220. DOI: `10.3390/s90301499`.

[ZHH12] X. Zhituo, R. Hao, and W. Hao. "A Content-Based Image Retrieval System Using Multiple Hierarchical Temporal Memory Classifiers." In: *2012 Fifth International Symposium on Computational Intelligence and Design*. Vol. 2. Oct. 2012, pp. 438–441. DOI: `10.1109/ISCID.2012.253`.

[Zho+15] B. Zhou, Y. Tian, S. Sukhbaatar, et al. "Simple Baseline for Visual Question Answering." In: *CoRR* abs/1512.02167 (2015). arXiv: `1512.02167`.