Computer vision using biomimicking AI system

ELIAS HÖGBOM ARONSSON
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Computer vision using biomimicking AI system

Elias Högbom Aronsson
el2020ar-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Fredrik Edman

Examiner: Erik Larsson

April 4, 2022

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The aim of this thesis was to develop a biologically inspired model for unsupervised learning on visual data. A preprocessor inspired by the retina was developed, which generated event-camera like data. This can be used as a low-cost alternative to an event-camera. Further, a neuron model was developed which showed ability to learn useful features from video data in combination with the preprocessor. A number of aspects of the system was also explored to probe its properties.

A data set consisting of videos of moving 3D-printed objects was created. This was used to compare the biological model in combination with a small LSTM network against a much larger convolutional LSTM, in classifying the different objects. This gave promising results, where the biological model performed as well or better in most aspects, compared with the large convolutional LSTM.

Acknowledgements

I would like to thank everyone from IntuiCell and the lab for their support and interest in the project. Without this guidance, the project would not have been possible. A special thanks to Udaya for all the time he spent explaining new concepts and guiding me. I would like to thank Henrik for his insights and for the opportunity to do this work. Further, I would like to express my gratitude to Fredrik for directing me through the process of a master thesis project, along with his help during the writing process.

Popular Science Summary

English

To recognize things in the world is something we do effortlessly. We can recognize a bird even if we have never seen that particular species or from that angle. For computers, this is not that easy. To define everything that makes a bird, a bird, in pixel values, is not a trivial task. Modern machine learning methods try to, instead get the computer to learn all these different aspects by itself. This has proven successful for many tasks, but requires large amounts of training data and computational resources. These methods also have problems generalizing to new situations.

The company IntuiCells AI technology was built on more general assumptions and inspiration from neuroscience, to address such problems. This thesis builds on IntuiCell technology, to see if this novel approach works in the field of computer vision.

First, two videos were animated and rendered, to test out different aspects of the model, as wells as to prove its promise. When this was completed a recording setup, consisting of a box with controllable LEDs and a motorized slider, where objects could be moved, was constructed. From the recordings, the model was trained to recognize different objects. This gave promising results. The model could recognize the different objects, even in some new situations. This shows a first promising result, of this novel approach to computer vision.

Svenska

Att känna igen saker i världen är något vi gör utan att ens fundera över det. Även på en ny plats, har vi inga problem att känna igen en fågel, trots att vi kanske aldrig sett just den arten eller en fågel från den vinkeln. För mjukvara däremot, är detta ett mycket svårt problem. Tänk dig att du ska definiera allt som gör en fågel till en fågel och inte till exempel en insekt. Det är inte helt lätt, då fåglar kan ha så många olika färger former och dessutom ses från olika vinklar och avstånd.

Moderna metoder försöker få program att själva lära sig dessa definition genom att träna dem på många exempel. Detta har lyckats bra för vissa områden, men kräver mycket datorkraft och många träningsexempel. Metoderna har dessutom

problem med att generalisera till situationer de aldrig sett tidigare. Utav dessa anledningar startades företaget IntuiCell, som jobbar med artificiell intelligens, som bygger mer på det vi vet om den mänskliga hjärnan, än de klassiska metoderna gör. Detta arbete genomfördes i samarbete med IntuiCell, med som mål att använda deras teknologi för att testa en helt ny metod för bildigenkänning.

Först animerades och renderades två träningsvideos, för att kunna utforska olika aspekter av modellen, samt för att initial undersöka om modellen verkade lovande. När detta var gjort konstruerades en inspelningslåda, där ljuset kunde kontrolleras och olika objekt kunde köras i bilden. Från inspelningarna tränades modellen till att känna igen de olika objekten. Detta gav lovande resultat. Modellen kunde känna igen objekten och även till viss del i nya situationer. Detta visar möjligheten till en helt ny typ av artificiell intelligens, med användning inom datorseende.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Object detection and tracking is a common problem in computer vision, with applications in a wide range of automation applications. Both the autonomous driving and video surveillance sectors are drivers in developments in this area [1]. The aim of this work is to test a novel approach to this problem using a bio-mimicking approach of the mammalian visual system. It builds on the IntuiCell artificial intelligence (AI) platform designed mainly for tactile and auditory data and extends it to the field of visual data.

Even the best current approaches to object detection in dynamic environments comes nowhere near what animals do effortlessly [2]. All current approaches also use vast amounts of labeled data, are computationally demanding and have problems generalizing to new situations; problems that does not apply to human visual perception. It is thus of great interest to investigate how well a system, built on knowledge from neuroscience, can perform.

In this work, the first processing steps of the biological visual system will be modeled. The resulting system will be explored to find its uses and properties, as well as be tested against a pure classical artificial neural network.

Chapter 2

Background and Theory

In this section the background literature on which this thesis is based will be discussed. First, the biological visual system, which the model will be built on will be explored. Then a discussion about event cameras, cameras that mimic retina processing in hardware, will follow. Finally, a short history and summery of the state-of-the-art methods in computer vision concludes this chapter.

2.1 IntuiCell

This master thesis was done in collaboration with IntuiCell, a company developing AI-technology, which builds much more faithfully on the functionality of a real brain compared to traditional AI. The company was founded by three neuroscience researches, one professor having more than 30 years of experience in the field and two postdoctoral researches.

2.2 The biological visual system

Visual information first enters the body through the eyes. Light travels thought the lens, which focuses it at the back of the retina. The retina consists of a dense network of neurons, with a layer of photosensitive cells at the back. The light thus has to travel through the neurons to reach the photoreceptor cells. Behind these cells there is a layer of pigmented epithelium cells, which stop the light from traveling any further. The visual signal is propagated from the photoreceptor cells, through the retinal neural network and into the brain via the optic nerve. The optic nerve projects to neurons in the thalamus. Most of these neurons send further projections to the visual cortex. An illustration of the system is shown in Fig. 2.1.

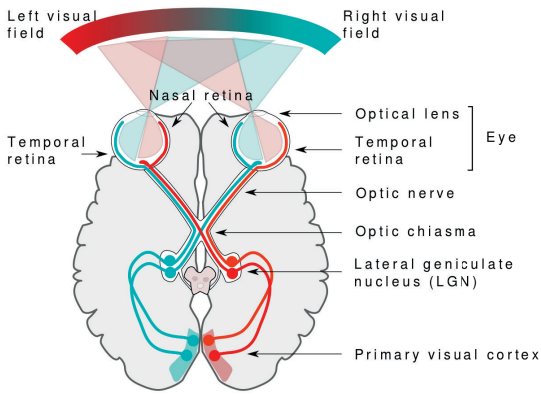


Figure 2.1: Illustration of the human visual pathway. (Source: Adapted from [3])

2.2.1 The retina

The retina is the innermost tissue layer in the eye, which is responsible for transforming incoming light into neural signals. It consists of a number of neuronal layers, which can be divided into three main once, the photoreceptor layer, the outer plexiform layer, and the inner plexiform layer [4]. The first layer consists of light sensitive photoreceptor cells, of two types, rods and cones [4]. The cones are sensitive to light of different wave lengths, giving color vision, while rods respond to a broad range of wave lengths, but with higher sensitivity down to a single photon [5]. Both types of photoreceptor cells convert incoming light into a change in membrane voltage, which results in a decrease in the tonic glutamate release in their axon terminals [5]. For simplicity, only rod cells will be modeled in this thesis. The rod cells projects onto two types of cells in the second layer, horizontal and bipolar cells [4]. An illustration of the cell connectivity is shown in Fig. 2.2.

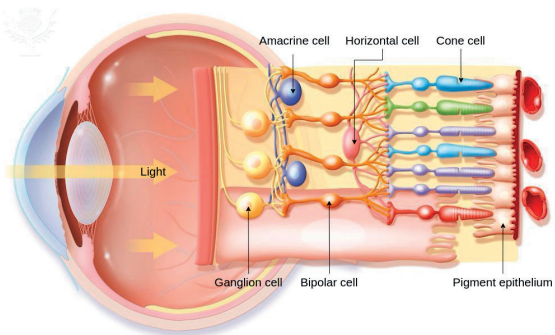


Figure 2.2: Illustration of the vertebrate retina. (Source: Adapted from [6])

The horizontal cells receive inputs from a number of photoreceptor cells. They

also exchange information by the presence of gap junctions between the horizontal cells [4]. These integrate their inputs relatively linearly [4]. This linear integration of signals from a number of receptor cells, along with lateral signal propagation, generates a spatio-temporal low pass filter of the visual information.

The bipolar cells can be divided into two general types, ON and OFF, which respond to increments and decrements in light intensity respectively [5]. This is done by comparing the difference between the inputs from projecting horizontal and photoreceptor cells [4]. The bipolar cells then project onto ganglion cells who's axons form the optic nerve. The optic nerve projects to neurons in the lateral geniculate nucleus (LGN), a thalamic relay nucleus [5].

This is, however, a very simplified model of the vertebrate retina. The retina contains cells which can be divided into at least 50 distinct cell types, with a great variety in morphology and function [7]. Recent studies [7] suggest that feature extraction and preprocessing is performed already in the retina, which opposes the old view of the retina as just a filter. In this work, the simplified model will be used as an inspiration for a preprocessor.

2.2.2 Lateral geniculate nucleus

The lateral geniculate nucleus (LGN), is commonly represented as a simple relay station to the cortex. This however, may again, be a great simplification. The LGN has been implicated to perform diverse functions such as attention, eye movement inhibition, signal amplification and contrast gain control [8]. In this work, both the ganglion and LGN neurons will be simplified down to a neuron model originally based on properties found in cuneate neurons, which are equivalent mechanosensory information processing neurons in the tactile system (see section 2.3). The cuneate neurons are feature extractors of tactile sensory information [9], [10].

2.2.3 Cortex

The LGN project mainly to the visual cortex. One view of the cortex's function is to find relationships between sensory input features and relate them to an internal model of the world [11]. The cortex is highly interconnected and mainly consists of two types of cells, principal cells and interneurons [11]. Principal cells release the excitatory neurotransmitter glutamate, while interneurons release the inhibitory neurotransmitter GABA. This could be modeled by the same cuneate based neuron model used for the LGN neurons, but connected in an interconnected network where some neurons would output positive signals and others negative. This is, however, outside the scope of this thesis, but could serve as inspiration for future work.

2.3 The Cuneate nucleus

The inspiration of the neuron model used in this work came in part from *in vivo* experiments done on the cuneate nucleus of cats [9], [10]. The cuneate nucleus is a mammalian brain stem nucleus which receives afferent sensory input from the skin on the upper body [12]. It has been found that cuneate neurons, despite having

similar receptive fields, respond to different stimuli, indicating that some feature extraction is done already in subcortical processing stages [9]. Further work has shown that just a few (4-8) synapses dominate the inputs to cuneate neurons, whereas the rest have relatively low weights [13].

2.4 Event cameras

There has been an effort to design cameras which already in hardware try to mimic the workings of biological retinas, i.e. retinomorphic cameras. These are often called event cameras or event-based vision sensors. They have been touted to have many advantages compared to traditional cameras and show promise in automation applications such as autonomous driving and quality control [14].

Instead of the traditional output of frames at a fixed frame rate like normal cameras, event cameras only output a signal if a pixel detects a change in luminosity above a threshold. This is done asynchronously between pixels, which allows higher temporal resolution on the order of µs or lower, while reducing data transfers by discarding information about statics. They also have high dynamic range of around 140 dB and low power consumption [15]. Only the most high-end traditional cameras can reach a temporal resolution on the order of µs. The $175,000 Phantom v2512 is an example of one such camera. Even at that price, it only has a dynamic range of 57.2 dB, while consuming 230 W [16]. This can be compared with the $5700 Davids 346, which has a similar latency, but a dynamic range of 120 dB and a power consumption of 140 mW [15], [17].

Due to the IntuiCell model's sensitivity to spatiotemporal dynamics, along with event camera's many advantages, they seem like a good fit for this project. However, they are still in their infancy and few commercial cameras exists and even fewer with software support and pretuning. Because of this, along with the lower price of a traditional camera, it was decided that the functions of the retina would be implemented in software with a traditional camera as a sensor, transforming the frame based data into event camera style data.

2.5 Traditional methods

There are a great variety of methods for object detection in video data. Early methods used handcrafted feature detection, which was paired with shallow neural networks. These suffered from poor accuracy and low robustness to noise [18].

With the development of deep learning came new approaches. Early development here took static image classifiers and used them frame by frame, creating a vector of features in time. Then a second neural net was trained on these feature vectors. This suffered from two prominent problems, unnecessary computational load and inability to cope with real-world situations such as motion blur, light and contextual inconsistencies [1].

A number of modern methods address these problems. Automatic feature extraction can be incorporated into the networks using deep networks. Recurrent neural network, such as long short-term memory (LSTM), is another approach which allows for memory in the network to be able to find spatiotemporal patterns.

Further, highly specialized architectures has shown great promise [1]. For both detection and segmentation i.e. localization of objects in a frame, all current state-of-the-art methods use deep convolutional networks (DCNN) [19]. They can be divided into two classes, single- and two-stage. Single-stage does both segmentation and classification in one network. Two-stage first uses a region proposal method and then classification by a convolutional neural network (CNN). Two-stage methods have higher accuracy, while single-stage are less computationally demanding. Mask R-CNN is a prominent two-stage network, while *You Only Look Once* (YOLO) is a well-known single stage method [19].

The technical field is both wide and rapidly developing. There will, thus, not be room to go into details in all methods, but due to the fact that all modern methods are built on DCNNs, they have some common drawbacks. They need large amounts of labeled training data, which can both be laborious and expensive to gather [20]. Training more complex networks also require large amount of computational power. The balance between overfitting and underfitting is a big problem in all approaches to both classification and regression problems [21].

Chapter 3
Methods

This chapter will first look at the model of the visual system. It will then go into the details of a classical neural network, which was used as a comparison to the visual system model. The chapter continues with the methods of data generation. Finally, the experiments conducted to test the models will be explained, which concludes this chapter.

3.1 Model of the biological visual system

The processing system consisted of a retina inspired preprocessor which transform video input into event camera like data, i.e. only information about changes. Modeled neurons received inputs from a number of pixels in the output of the preprocessor. These connections were called afferent synapses and the set of connections, to one neuron, the receptive field. Their initial weights (seed weights) were randomly distributed, but were changed during training by a model of hebbian learning. The inputs to the neurons were processed to mimic the response of biological neurons, and the resulting membrane potential was used as an output. A high-level diagram of the system can be seen in 3.1.

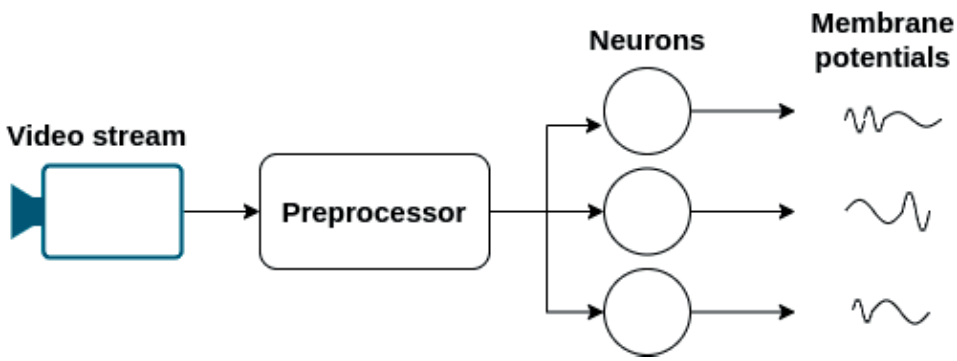


Figure 3.1: High-level diagram of the visual system model.

3.1.1 Retina inspired preprocessor

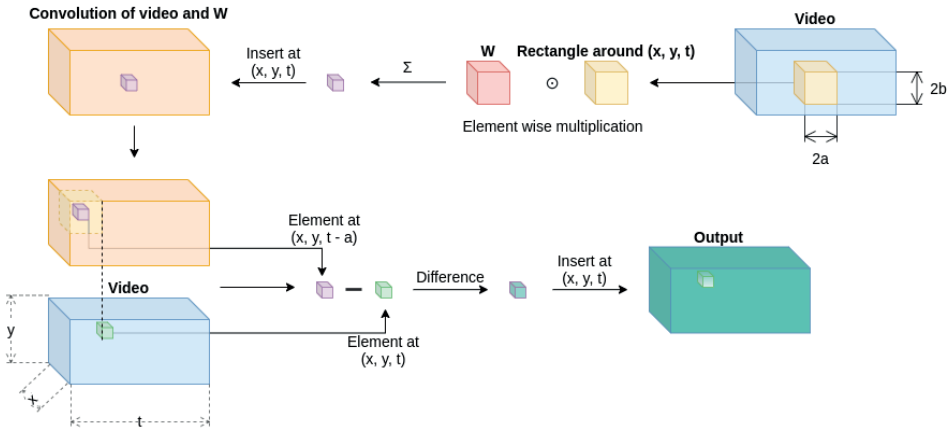


Figure 3.2: Diagram of the preprocessor, which took the convolution of an input video with the weight W and then for each element computed the difference between this and an element from the original video at the corresponding spatial position, (x, y), but at the next time point, t. The blocks represent video data as three-dimensional arrays, where two were spatial (x and y) and was temporal (t).

To mimic the visual processing in the human retina, a preprocessor was developed. The purpose for this was to highlight the dynamics in input data, amplifying changes and excluding statics. It first transformed the video into a three-dimensional array, with two spatial dimensions and one temporal. Then a 3D-convolution of this array with a 3D-array of weights was computed. The weights could be tuned to give different emphasis on temporal or spatial changes. The difference of each pixel in this convoluted array and the corresponding pixel in space in the original video array was calculated. For a pixel (x, y, t) in the convoluted array, C, with a weight array of size $2a$ in the time dimension and original video array V, the difference calculation was given by;

$$O_{x,y,t} = C_{x,y,t-a} - V_{x,y,t} \tag{3.1}$$

where, O was the resulting output array. The time $t - a$ was used for the convoluted array to compare the video with a low passed version of the past, to find changes. If the same time point had been used for both arrays, the changes would have been excluded instead.

This, thus, calculated the difference of each pixel in the video with the mean of its neighboring pixels in time and space. This is similar to comparing each pixel to a low pass filtered version of its surroundings, but could be weighted differently for different distances and dimensions.

A schematic of this process can be seen in Fig. 3.2, which illustrates the convolution on the upper row and then the subtraction calculations below that.

The blocks represent the 3D-arrays, where x and y were the spatial dimensions and t was the temporal dimension.

3.1.2 Neuron model

The neuron model was based on the electrochemical properties of biological neurons. The aim was to capture the information processing and learning functions of neurons, while being as simple as possible, to limit computational resource needs [22].

The model was a simplified version of the method described in [10]. Instead of three ion channels, it only used one for Ca^{2+}, but tried to capture the voltage dynamics of all of the ion-channels.

The input activity was calculated as the sum of synaptic inputs times their weights. This was multiplied by a scalar called the accumulator, meant to represent the amount of available ions around the cell. The accumulator was charged by Ca^{2+} outflow and discharged by Ca^{2+} inflow. The input activity A, to a neuron, was thus calculated by:

$$A = \left(\sum W_{x,y} \cdot a_{x,y}\right) \alpha \tag{3.2}$$

where, $a_{x,y}$ was the pixel value at position (x, y) in a frame after preprocessing, $W_{x,y}$ was the corresponding synaptic weight and α was the accumulator value. With the inputs (F) and weights (W) as matrices, the first part of this calculation is also illustrated graphically in Fig. 3.3. For the full calculation, the sum of $F \odot W$ times α would also have to be computed.

Figure 3.3: Graphical illustration of the calculation of activity in each synapse for a frame, F, and synapse weights, W.

From the input activity, the Ca^{2+} inflow was modeled by a RC-circuit described by two equations. The first described the resulting membrane potential change, (V^1);

$$V_i^1 = \frac{1-dt}{\tau} \cdot V_{i-1}^1 + \left(\frac{dt}{\tau}(A_i - A_{i-1})\right) e^{-t/\tau} \tag{3.3}$$

where, τ was a time constant, dt was the time step, t was the time and i was the iteration step. The Ca^{2+} current, $I_{Ca^{2+}}$, was then given by;

$$I_{Ca^{2+}} = \frac{V}{R} \tag{3.4}$$

where, R was a membrane resistance constant.

The voltage change due to outflow of Ca^{2+} was modeled by a second RC-circuit. This was meant to represent the process of ion-channel and ion-pump recruitment processes. It was described by the equation;

$$V_i^2 = \frac{1-dt}{\tau} \cdot V_{i-1}^2 + \left(\frac{dt}{\tau}\right) e^{-t/\tau} \tag{3.5}$$

Post spike generation, an after-spike-hyperpolarization (AHP) was modeled by;

$$AHP = \begin{cases} \int I_{Ca^{2+},in} - \int I_{Ca^{2+},out}, & \text{if } \int I_{Ca^{2+},in} - \int I_{Ca^{2+},out} < 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

The final membrane potential (V_m) was given by;

$$V_m = A + I_{Ca^{2+}} + AHP \tag{3.7}$$

3.1.3 Learning

The learning protocol was adapted from the methods described in [23]. The learning was based on two parts. First, a threshold was calculated. The membrane potential was then subtracted by this threshold, to normalize it against baseline activity. Secondly, the activity in each synapse was compared with the whole neuron's membrane potential and adjusted based on how closely they followed each other.

Threshold

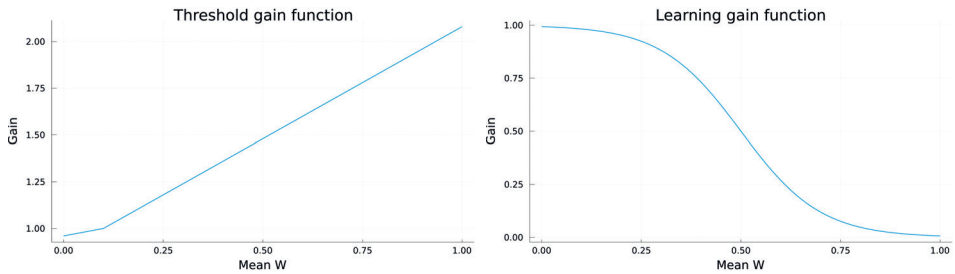
The threshold, T, was set by the mean membrane potential over a time window, times a gain function, used to control the learning rate based on the mean of the synaptic weights. The threshold was thus given by;

$$T = \overline{V_m} \cdot g_T(\overline{W}) \tag{3.8}$$

where $\overline{V_m}$ was the mean activity for some time window and $g_T(\overline{W})$ was a gain function, used to specify a set point for the mean of the synaptic weights, $\overline{W}$. The $g_T(\overline{W})$ used, was a two sloped linear function, defined by the equation;

$$g_T(\overline{W}) = \begin{cases} k_1 \cdot \overline{W} + (1 - k_1 \cdot \overline{W}^*)), & \text{if } \overline{W} < \overline{W}^* \\ k_2 \cdot \overline{W} + (1 - k_2 \cdot \overline{W}^*)), & \text{if } \overline{W} > \overline{W}^* \end{cases} \tag{3.9}$$

where the set point, $\overline{W}^* = 0.10$, and the slopes, $k_1 = 0.4$ and $k_2 = 1.2$, giving the line seen in Fig. 3.4a.



(a) Gain function g_T, used to set polarity threshold. **(b)** Gain function g_l, for learning, to limit weight growth.

Figure 3.4: Gain functions.

Weight change

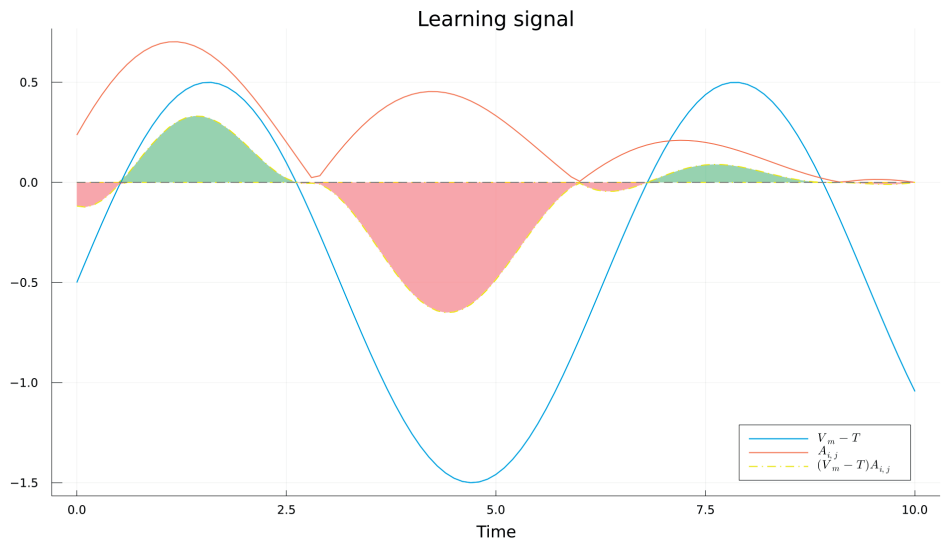


Figure 3.5: Learning signal in yellow, with fill indicating positive (green) or negative (red) dW, calculated from $V_m - T$ in blue and a synaptic input in red.

The idea of the learning protocol was to increase the weights of synapses, whose activity followed the membrane potential. This would increase the weight of synapses

with correlated inputs, which drove the neuron's membrane potential. The weight change, $dW_{i,j}$ for a synapse (i, j), was given by;

$$dW_{i,j} = (V_m - T) \cdot A_{i,j} \cdot g_l(W_{i,j}) \tag{3.10}$$

$$g_l(W_{i,j}) = 1 - \frac{1}{1 + e^{-10 \cdot (W_{i,j} - 0.5)}} \tag{3.11}$$

where $g_l(W_{i,j})$ was a gain function. As seen in Fig. 3.4b, it was an inverse sigmoid function, which was used to keep weights bounded. When a weight approached 1 the gain would approach 0.

An example illustration of a learning signal without g_l, with a sinusoidal V_m, $T = 0.5$ and $A_{i,j}$ as the absolute of a decreasing sinusoidal with a small phase shift from V_m, can be seen in Fig. 3.5. The green areas show the positive integral of the learning signal, while the red show the negative. This illustrates the importance of the threshold, which set how closely a synapse's activity had to follow V_m to be potentiated.

The weights were not instantly updated, but instead, the mean of dW for a time window was added to the current weight each time step.

A diagram over the whole visual system model is shown in Fig. 3.6.

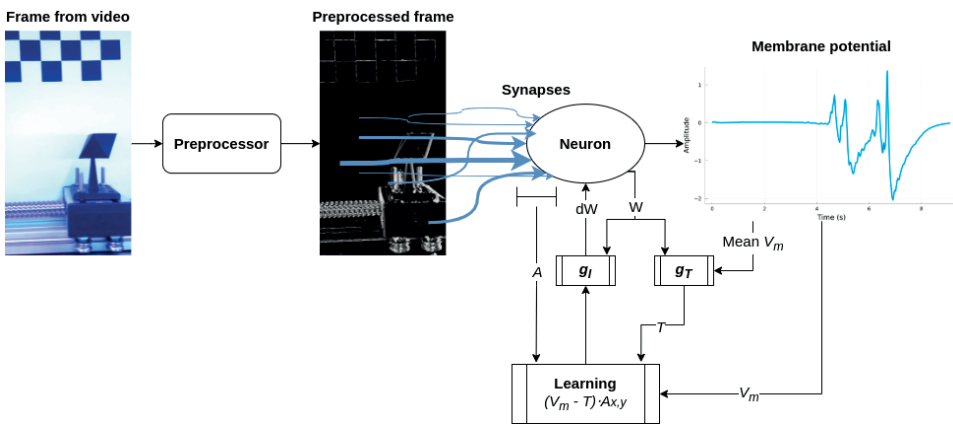


Figure 3.6: Diagram over the whole visual system model. To the left, a frame from a training video is shown. The video feed is then shown being fed through the preprocessor, generating an event camera style "frame". The blue arrows, from this, indicate connections between a pixel and the neuron, with the widths representing different synapse weights. The neuron block represents the input processing, generating a membrane potential. Below this, the different parts of the learning mechanism are represented.

3.2 Classical neural networks

Classical neural networks were used for two reasons. First, a LSTM was used to classify neuron outputs. Any classifier could have been used here, but a LSTM was chosen for its ease of use with multi variable time series classification.

Secondly, a convolutional LSTM (ConvLSTM) was used to compare the results of the neuron model with a more traditional ANN structure.

Accuracy was used to compare the different methods. According to ISO 5725, accuracy is defined as,

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.12}$$

where, TP = true positive, TN = true negative, FP = false positive and FN = false negative [24].

3.2.1 LSTM classifier

The LSTM classifier consisted of a four layer structure as shown in Fig. 3.7. First, a masking layer was used for the network to be able to handle different length inputs in the same batch. Then a LSTM layer was used, because it can find relationships at different time scales. Next, a dropout layer was added to counteract overfitting. Finally, a five unit dense layer, with softmax, was used as the output layer. The resulting network had 3493 parameters.

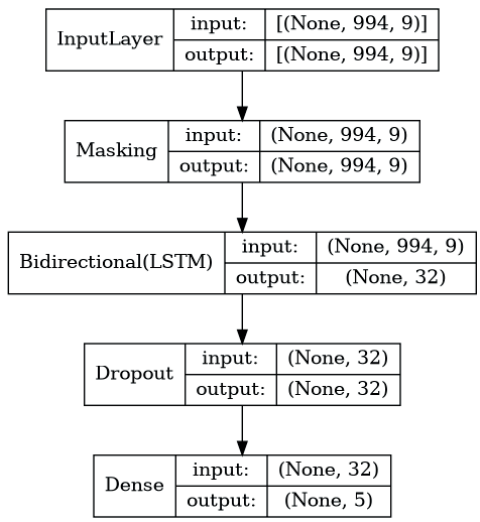
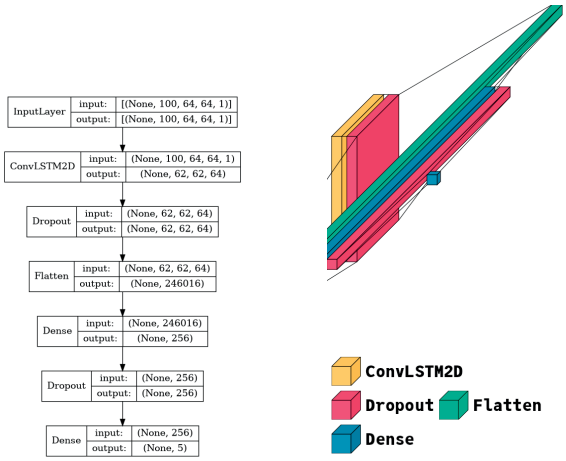


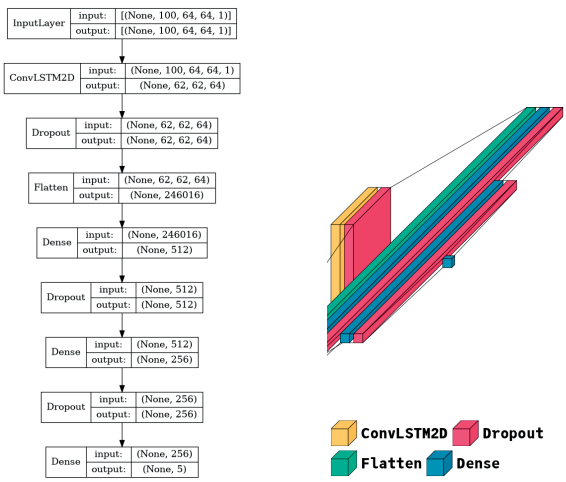
Figure 3.7: Diagram over LSTM classifier structure.

3.2.2 Convolutional LSTM



(a) Diagram **(b)** Illustration

Figure 3.8: Diagram and illustration of the eight layer ConvLSTM structure.



(a) Diagram **(b)** Illustration

Figure 3.9: Diagram and illustration of the eight layer ConvLSTM structure.

A convolutional LSTM was used to classify the videos directly. Two different versions were implemented, one with six layers and one with eight layers. Both of these had the same initial structure. First, a convolutional LSTM layer was used, for its ability to find spatio-temporal patterns. Then a flatten layer was used to

re-dimension the inputs to one dimensional vectors. After this, two dense layers were used for the six layer version and three dense layers for the eight layer version. Between each of the neural layers, there were a dropout layer as well.

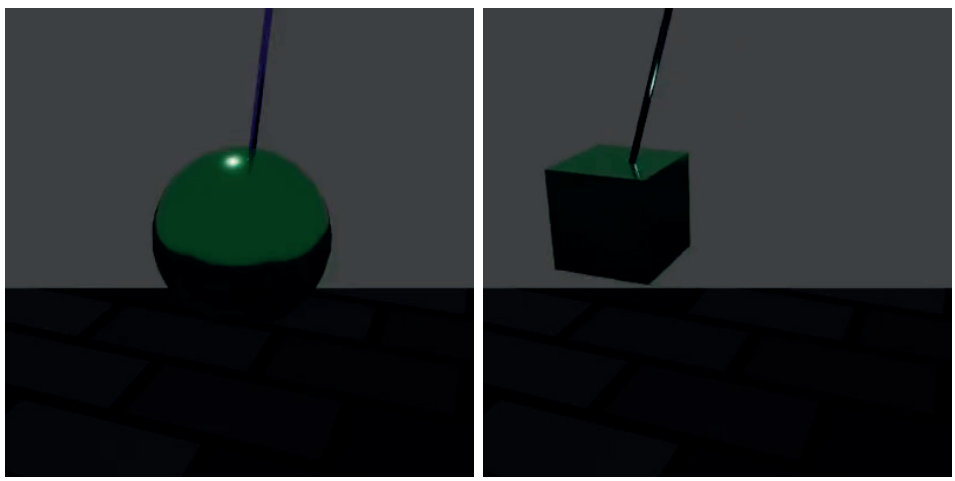
The six layer network had 63,131,653 parameters, while the eight layer network had 126,113,315 parameters. These were large models compared with the LSTM classifier, but the number of parameters were in the same order of magnitude as state-of-the-art models such as the 60 million parameters YOLOv4 model [25]. Illustrations and diagrams of the two networks can be seen in Figs. 3.8 and 3.9, for the six and eight layer networks, respectively.

3.3 Data set generation

Two types of video data was used in this project. First, rendered videos were used. After proving some success of the model, a hardware set-up was constructed, which was used to record videos.

3.3.1 Video renderings

Videos were rendered in Blender, a free and open source modeling and rendering software, to be able to test the model during implementation and to prove some promise of the model before building a hardware recording setup. Two simple videos were created to see if the model could be trained to pick up both common and different features from these. Both videos were 4 s clips of a swinging pendulum. The only two differences between the videos were the swinging objects, which was a cube in one video and a sphere in the other, and the texture of the suspension rods. One frame from each video can be seen in Fig. 3.10.



(a) Sphere **(b)** Cube

Figure 3.10: Frames from two training videos rendered in Blender.

3.3.2 Recording setup

To be able to create training videos of moving objects in a controlled environment, a hardware setup was designed and built. It was decided that in this setup we should be able to control the light intensity and direction, move an object or camera and induce controlled vibration of the camera. The recording setup was designed in SolidWorks, a computer-aided design (CAD) software. A rendering of the base assembly, without a cover and LEDs, can be seen in Fig. 3.11.

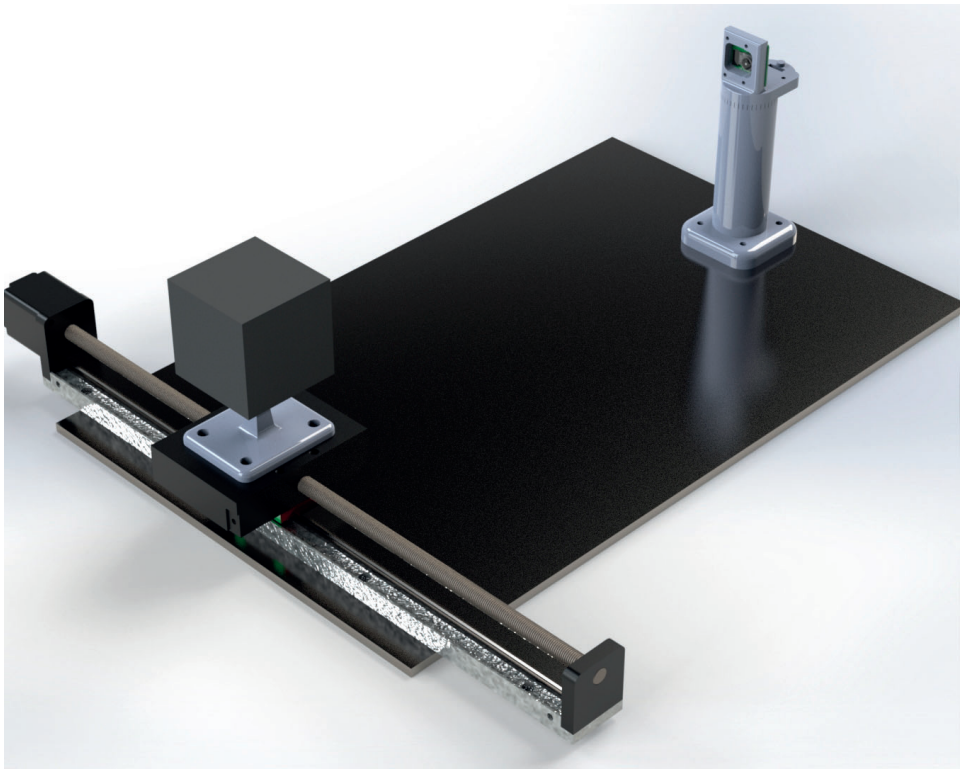


Figure 3.11: Rendering of CAD design of recording system, without cover and LEDs.

The baseplate, seen in black in the rendering, was machined out of 10 mm thick brass, as a frame for all components as well as a dampener, to limit unwanted vibrations. The camera mount was designed to be turnable in increments of 16°, by a crown gear mechanism, as seen in Fig. 3.12.

The camera mount also had a mounting point for a vibration motor, which can be seen in the figure as the pin and holes behind the camera on the upper part.

Figure 3.12: Camera mount consisting of two parts to be able to adjust direction in increments of $16°$.

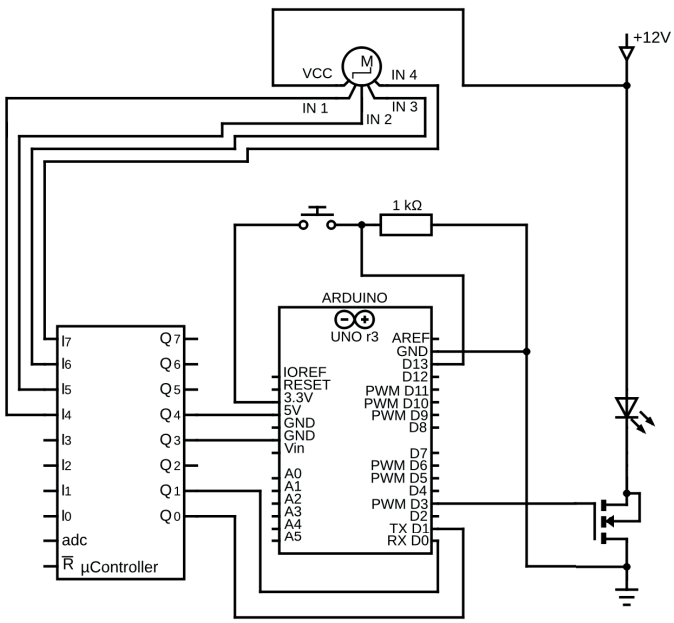


Figure 3.13: Wiring diagram of Arduino controlling LED intensity, motor controller and monitoring end stop switch. For simplicity, only one out of two end stop switch circuits and one out of three LED circuits are shown.

The camera used was a Raspberry Pi Camera Module v2 [26]. It was chosen

for its low price of $25 and ease of use. It had a resolution of 720p at 60 frames/s [26]. To be able to accurately move objects, for the model to be trained on, a linear actuator, consisting of a stepper motor and a threaded rod was used. This was controlled by an Adafruit featherwing motor controller, marked as μcontroller in the diagram in Fig. 3.13. The motor controller in turn was connected to an Arduino. The Arduino also controlled the intensity of three LED-strips, placed in the cover box, by power width modulation control of the gate voltage of one MOSFET per strip. This can be seen as the right most circuit in the diagram. The LED-strips were placed, one on each width of the cover and one behind the camera facing the actuator. The right and center LED-strips are shown in Fig. 3.14 and marked as 9.

The Arduino had one additional purpose, to receive inputs from one micro switch at each end of the actuator, stopping the motor before it would run into the frame. This was also used to zero the position of the object. This part of the circuit can be seen in the center above the Arduino in the diagram.

Five different objects in geometrical shapes were 3D-printed to be used as stimuli, along with a mount for attaching these to the actuator carriage. The hardware setup, with the stimuli placed in the center and the actuator carriage along with the object mount in the back, can be seen in Fig. 3.14

3.3.3 Recording protocol

With this setup, four light conditions and five speeds were chosen, as seen in table 3.1. The light conditions were named, l, r, lc and lcr, where the letters stand for which of the LED-strips were turned on during the recording. Here l stands for left, r for right and c for center.

For all videos the stimulus was recorded moving from one end of the slider to the other and back again. Four different sets of recordings were done. In the first, called the fixed parameter set, 10 videos for each of the stimuli were recorded at light condition lc and a speed of 3.27 cm/s. These were used to validate the robustness of the model against real-time noises.

In the second set, called the varying speed video set, five videos were recorded at each of the five speeds for each of the stimuli. This was used to explore the response of the model to a varying parameter, as well as to see if the responses were similar enough to be able to classify the stimuli from the neuron outputs.

For the third video set, called the varying light video set, five videos were recorded at each of the five light conditions, with the torus as stimuli, traveling at a speed of 3.27 cm/s. This was used to test the robustness of the model to different light conditions.

The final video set was only used to test the classification accuracy of the neuron outputs with a LSTM classifier. It consisted of one video for every given sensing parameter (speed and light condition) combination for each of the stimuli.

Table 3.1: Chosen parameter settings for data recordings. For light conditions, the letters indicate which LED/LEDs were lit, where l, c, and r corresponds to left, center and right LEDs respectively.

Speed cm/s **:**	1.24	1.80	1.89	3.27	4.00	4.50
Light condition:	l	r	lc	lcr	-	-

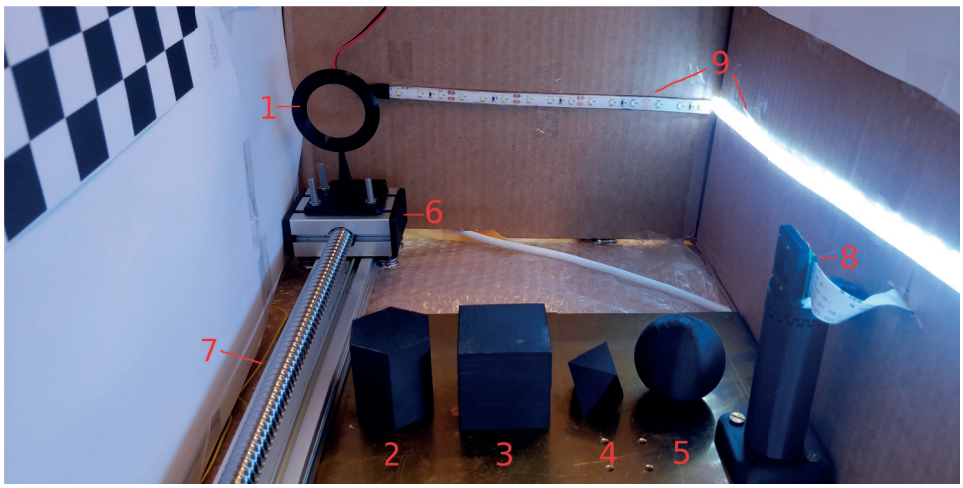


Figure 3.14: Image of the five stimuli in the recording setup. 1 - sphere, 2 - pentagonal prism, 3 - cube, 4 - octahedron, 5 - sphere, 6 - actuator carriage, 7 - actuator, 8 - camera on mount and 9 - LEDs.

3.4 Model experiments

Initially, the model was tested on the rendered data to determine the potential of the model, before constructing a recording set-up. First, the effect of different synaptic weights, along with the shapes and densities of connected synapses, were explored. Then the effect of added noise to the neuron output and learning was investigated.

3.4.1 Experiments with the rendered video data

Effect of different synapses and their weights

The first aspect of the model that was explored, was the effect of different initial synaptic weights, called seed weights. The neurons were only connected to a subset of the pixels in the videos due to computational constraints. The set of connected synapses was called the receptive field. A non connection between a

pixel and a neuron was modeled as a synaptic weight of 0 and was excluded from any computations. The seed weights were created by giving a number of synapses a random value. These values were taken from a log-normal distribution with $\mu = 0$ and $\sigma = 0.2$ and scaled by dividing them by 8.

Firsts, the model was tested to see that the same seed weights and receptive field with the same input generated the same weight change. This was done by giving six neurons the same receptive fields and seed weights and training them on 200 repetitions of each of the two rendered videos.

Then the same was done, but with different seed weights, to see if different weights, but the same receptive field, would generate different learning outcomes. A third variation of this was tested, were the neurons had the same seed weights and shape of the receptive field, but shifted to different positions. This was done to prove that the learning was dependent on the inputs and not dominated by the seed weights.

The effect of receptive field density was also explored. 12 neurons with 400 randomly selected synapses from within a circle, with increasing radius from 4 to 48 pixels, were trained for both 200 and 400 repetitions of the rendered videos. For the first two neurons, 400 synapses could not fit within the circle so these only had 45 and 193 synapses respectively.

From this, a method for capturing the entire frame by a number of neurons was developed and tested. The frame was divided into a grid of six rectangles, one per neuron, and each neuron were given 1200 random synapses from pixels in its corresponding area.

Effect of noise

Two experiments were conducted to test the model's robustness to noise. First, a neuron was trained for 200 repetitions of each of the rendered videos and then the videos plus different levels of Gaussian noise was inputted to the neuron. The second experiment used six neurons with the same receptive field and seed weights. These were trained again for 200 repetitions of the videos, but with different amounts of Gaussian noise added, from $\sigma = 0$ to $\sigma = 0.5$.

Effect of preprocessor

To test the effect of the preprocessor, one set of 12 neurons were trained on the raw videos and one set on preprocessed videos for 600 repetitions.

3.4.2 Evaluation with experimental data

First, 12 neurons were trained on one set of recorded videos of each of the five stimuli, at a speed of 3.27 m/s and light condition lc. The receptive fields were uniformly, randomly distributed in a square area for half the neurons and a circular area for the other half. This was done to not bias the neurons to find features constrained by only one type of border of the receptive fields.

The outputs of these neurons were then used for a number of experiments. First, the variance in response of one of these neurons for inputs of videos of the same stimulus and recording setting was compared to inputs of videos of the

different stimuli, but still with the same recording parameters. This was also compared with the response to inputs with videos of the same stimuli, but in different light conditions. Finally, this was compared to neuron outputs from inputs with videos of the same stimuli but traveling at the different speeds. All of this was also done with the same neuron, but before it had been trained.

A second way of doing these comparisons was used. Here, the LSTM classifier (see section 3.2.1) was trained to first classify the stimulus from the outputs of the same 12 neurons, both before and after training, and the fixed parameter video set as inputs. This classifier was tested with the outputs of the neurons with the varying light condition video set and the varying speed video set. This was done to see how well the model in combination with the LSTM could generalize to new situations. The LSTM classifier was also trained to classify the outputs with the varying speeds video sets as inputs, as well as with all the videos as inputs.

The same classifications were also performed with the ConvLSTM (see section 3.2.2) directly on the raw videos.

Chapter 4
Results

4.1 Results from rendered input videos

4.1.1 Impact of different receptive fields and seed weights

Because the model is deterministic, the same seed weights and inputs should give the same learning output every run of the model, which also seems to be the case. In Fig. 4.1 the initial and final weights for six neurons can be seen as heatmaps, where each pixel represents a weight corresponding to the same pixel in an input video. Darker pixels represent a higher weight. The six neurons with the same seed weights and receptive fields, all ended up with the same distribution of final weights, as expected with a deterministic model with the same inputs.

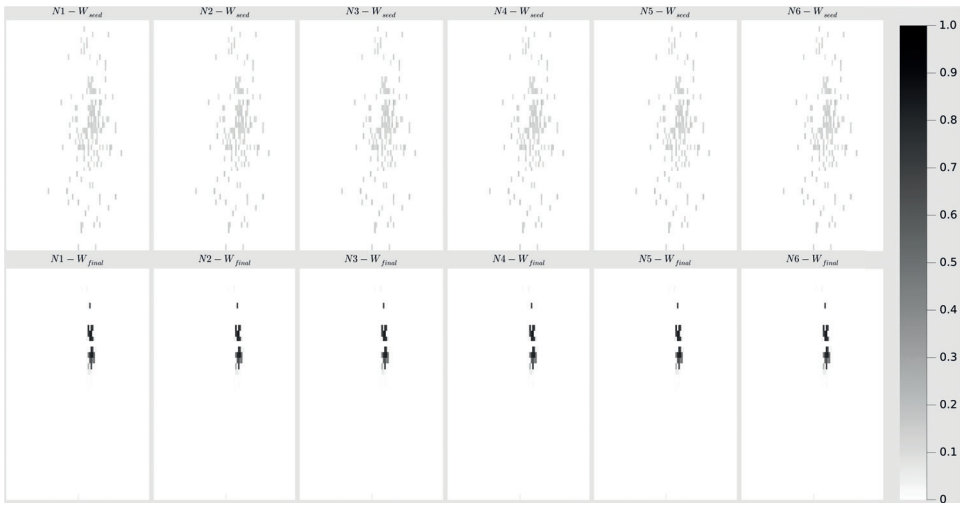


Figure 4.1: Heatmap of seed weights on the upper row and final weights on the lower row, for six neurons with same seed weights.

Running the model with the same receptive fields, but with different seed weights also gave very similar final weights, as seen in Fig. 4.2. Because the input

is very simple there are not a lot of features to pick up at any one place, so the neurons may be learning the information that is there. There is some variation though, which might be useful in more complicated situations, where a lot of different things can happen in the same region.

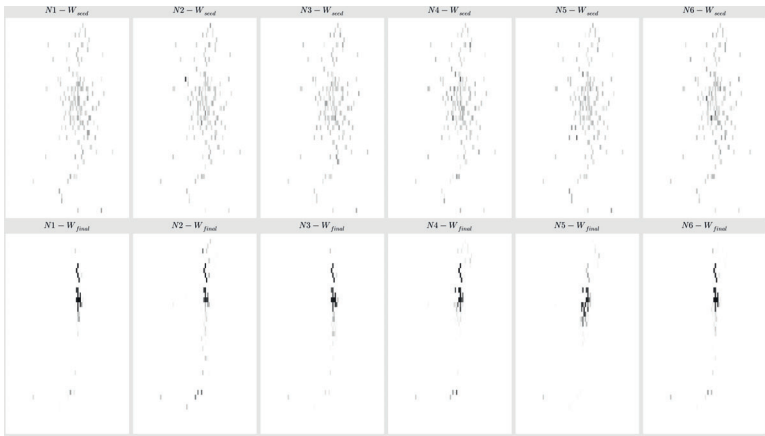


Figure 4.2: Heatmap of seed weights on the upper row and final weights on the lower row, for six neurons with same synapses, but with different initial weights.

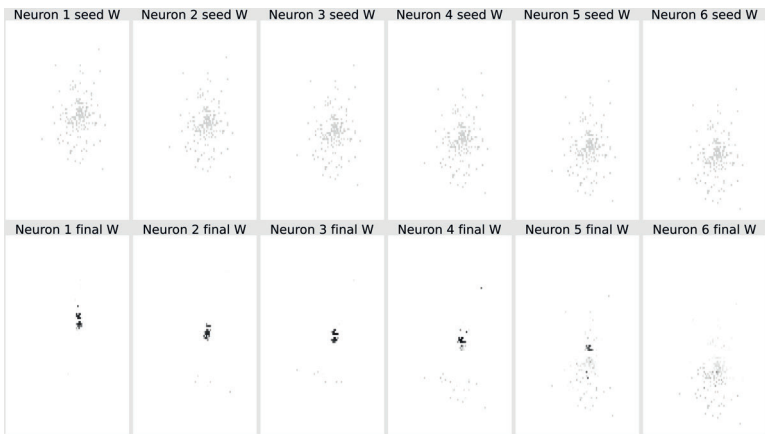


Figure 4.3: Heatmap of seed weights on the upper row and final weights on the lower row, for six neurons with synapses in the same shape and weights, but at different positions.

Using the same weights and shape of connected synapses, but at different positions, resulted in different final weights. Fig. 4.3 shows the initial and final weights of six neurons with connected synapses in the same shape and initial weights, but at lower positions, for each of the neurons moving right in the figure. Now, the neurons have potentiated different weights. Neurons 1-4 seem to have

found a part of the pendulum rod, where the density of connected synapses were high.

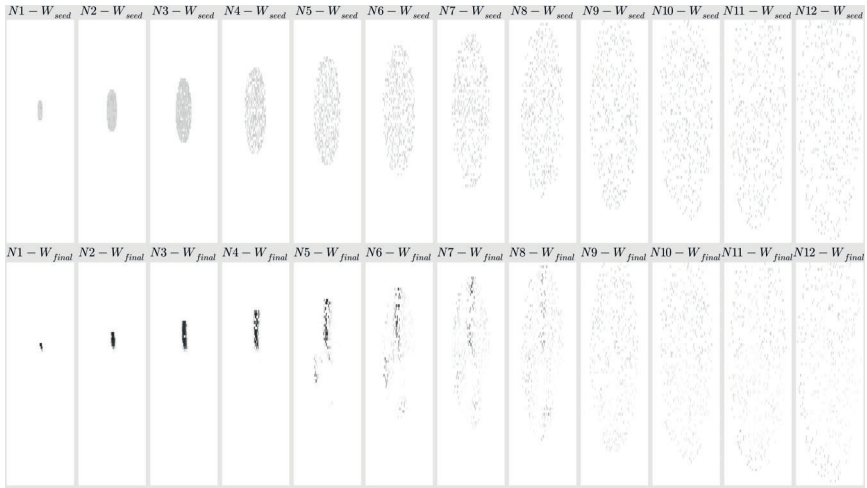


Figure 4.4: Heatmap of seed weights on the upper row and final weights on the lower row, for 12 neurons with randomly selected synapses withing a circle, with increasing radius for each neuron. The neurons were shown each video 200 times.

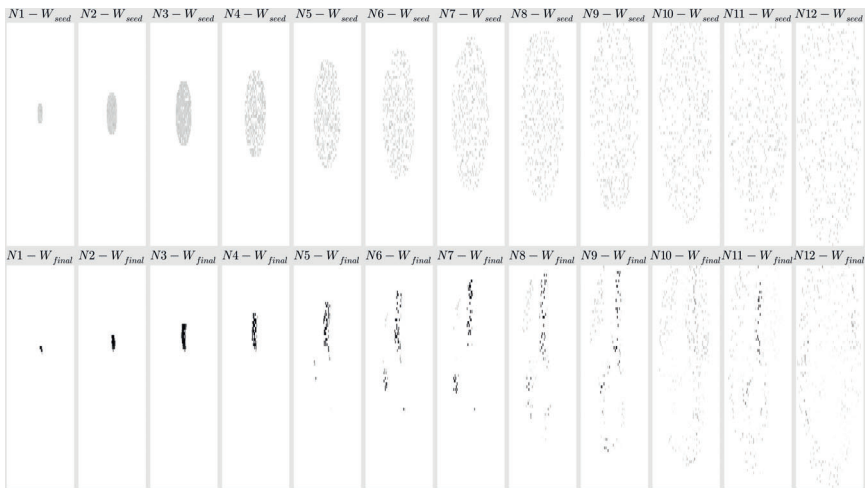


Figure 4.5: Heatmap of seed weights on the upper row and final weights on the lower row, for 12 neurons with randomly selected synapses withing a circle, with increasing radius for each neuron. The neurons were shown each video 400 times.

The effect of synapse density can be seen in Fig. 4.4. Here neurons with dense and confided synapses ended up with a dense confided number of high weights,

while neurons with less dense synapses ended up with less potentiated final weights, but from a bigger area.

This does, however, assume a constant number of learning iterations. As shown in Fig. 4.5, running the model for twice (400 compared with 200 repetitions) as long resulted in potentiation of weights even in the neurons with less dense synapses, showing that learning rate is dependent on the density of connected synapses. This is reasonable, as closer pixels are probably more correlated.

As shown in Figs. 4.1, 4.2 and 4.3, non-uniform distributions of connected synapses seems to bias the neurons to potentiate weights in the higher density regions. In Figs 4.4 and 4.5 the result of uniform receptive fields are shown. As shown in Fig. 4.6, the entire visual field can be captured, by be divided into square regions with one neuron connected to a uniform distribution of pixels for each area. Here the visual field was divided into six squares, with one neuron per square receiving 1200 random synapses from its corresponding area.

With this approach, the neurons seems to pick up weights building up features recognizable by eye. Neuron one looks to have learned the perimeter of the sphere. Neurons two, four and six looks to have learned different parts of the tiled floor and neurons three and five looks to have learned the rod at two different angles.

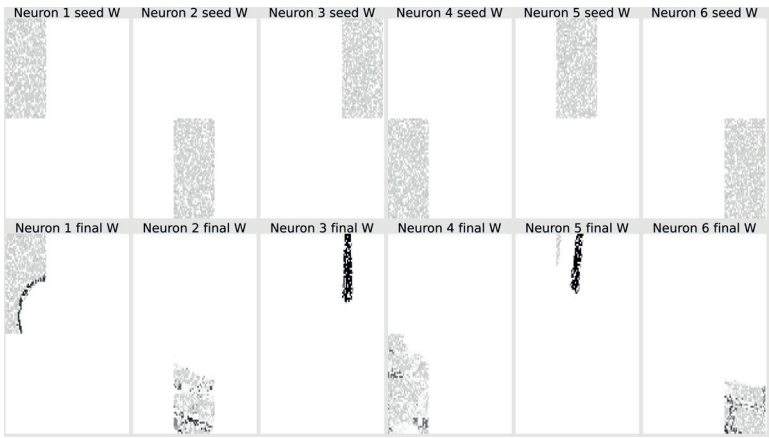


Figure 4.6: Heatmap of seed weights on the upper row and final weights on the lower row, for six neurons with 1200 randomly selected synapses within different square areas, for each neuron.

4.1.2 Effect of noise

To test the sensitivity of the model to noise, two experiments were conducted. First a neuron was trained for 200 repetitions of each video and then the videos plus different levels of Gaussian noise were inputted to the neuron. The inputs and resulting membrane voltage can be seen in Fig. 4.7. The magnitude of the membrane voltages varies, but the general shape and temporals of the responses looks to be conserved thought out all noise levels. The thick blue line in Fig. 4.7 shows the response to the video without any added noise.

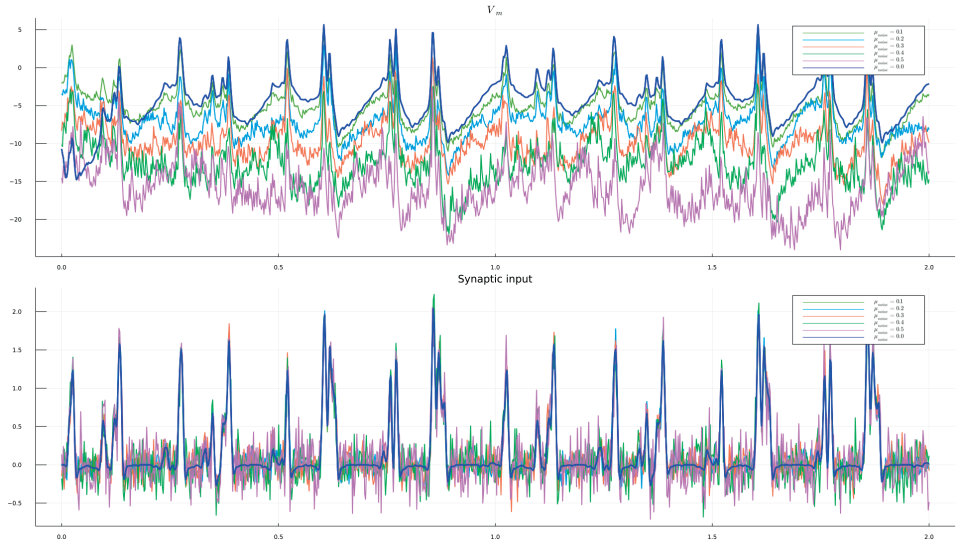


Figure 4.7: Membrane voltage, V_m for the same neuron, with different level of noise added to the input video. Before this the neuron had been trained for 200 repetitions of each video. The thick blue lines show input and V_m without added noise.

As seen in tab. 4.1 the cross-correlation of neuron outputs at different noise levels, with no noise, is less affected by the noise than at least the mean of the cross-correlations of each pixel time series between the original video and the video with added noise.

Table 4.1: Mean cross-correlations of each pixel time series from the original videos with the videos along with different amounts of Gaussian noise added, along with cross-correlations of neuron outputs from the original video inputs with outputs from video inputs with added noise.

Noise level - σ:	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Video (%):	64	50	40	33	28
Neuron output (%):	97	90	79	58	46

The second experiment used six neurons with the same receptive field and seed weights. These were trained again for 200 repetitions of the videos, but with different amounts of Gaussian noise added, from $\sigma = 0$ to $\sigma = 0.5$. In Fig. 4.8 it can be seen that despite the noise, all neurons potentiated the same weights.

Looking at the time series evolution of the weights for each neuron in Fig. 4.9, it can be seen that with more noise the weights move more erratically, but overall the same weights move in the same direction, regardless of noise level.

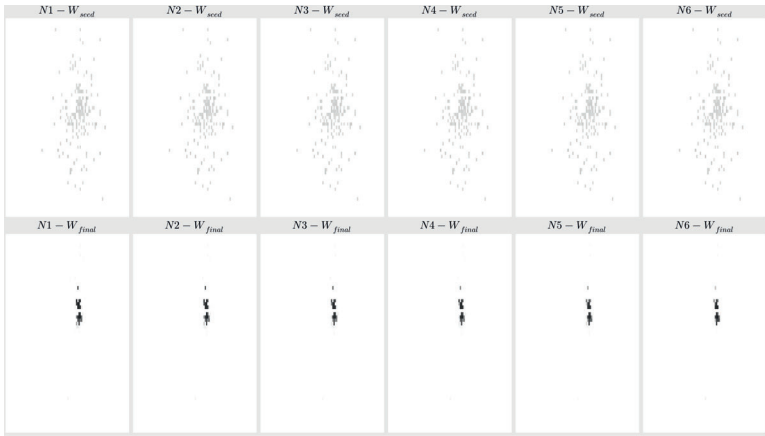


Figure 4.8: Heatmap of seed weights on the upper row and final weights on the lower row, for six neurons with identical seed weights, but trained with increasing amount of Gaussian noise ($\mu = 0$) added to the input, for each neuron. Neuron 1 had no added noise, while the remaining neurons had added noise with standard deviation increasing in increments of 0.1, meaning neuron 6 had added noise with $\sigma = 0.5$.

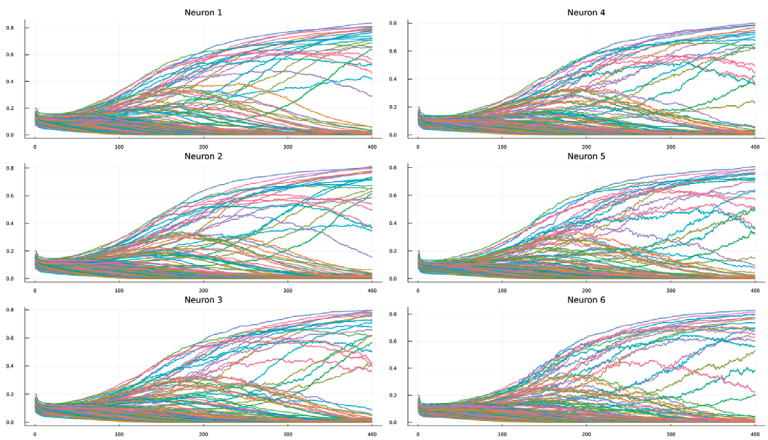
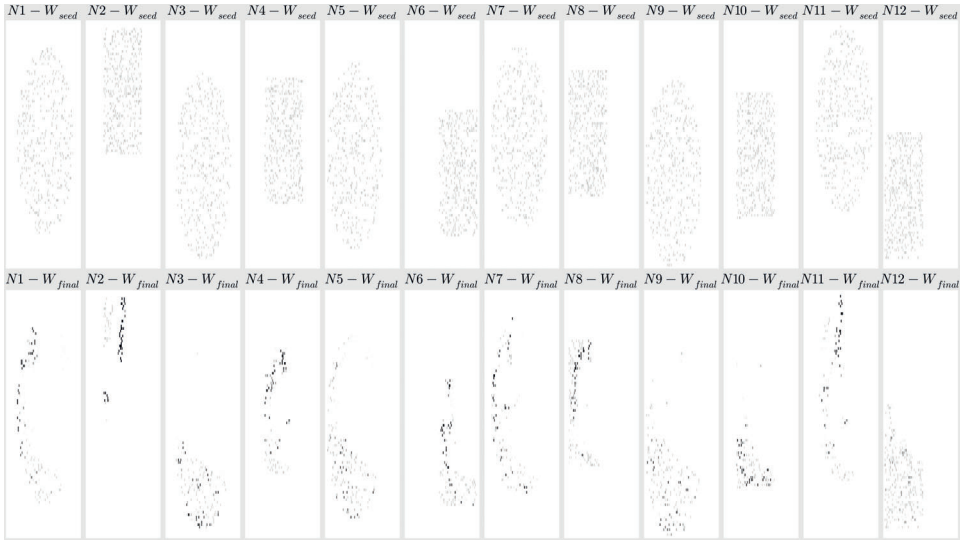


Figure 4.9: Time series of weights for six neurons with identical seed weights, but trained with increasing amount of Gaussian noise added to the input, for each neuron. Neuron 1 had no added noise, while the remaining neurons had added noise with standard deviation increasing in increments of 0.1, meaning neuron 6 had added noise with $\sigma = 0.5$.

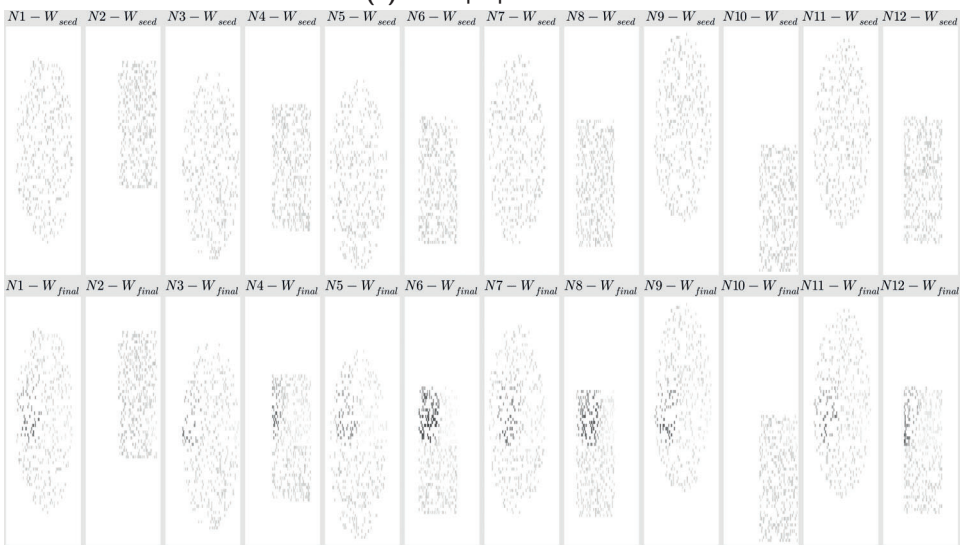
These two experiments show that the model is robust against noise, at least for this simple input. Even with a signal-to-noise ratio of 2 in the input, the neuron output is similar in shape to the output from an input of the pure signal. The

learning is also very similar in neurons receiving pure input to neurons receiving input with added Gaussian noise.

4.1.3 Effect of preprocessor



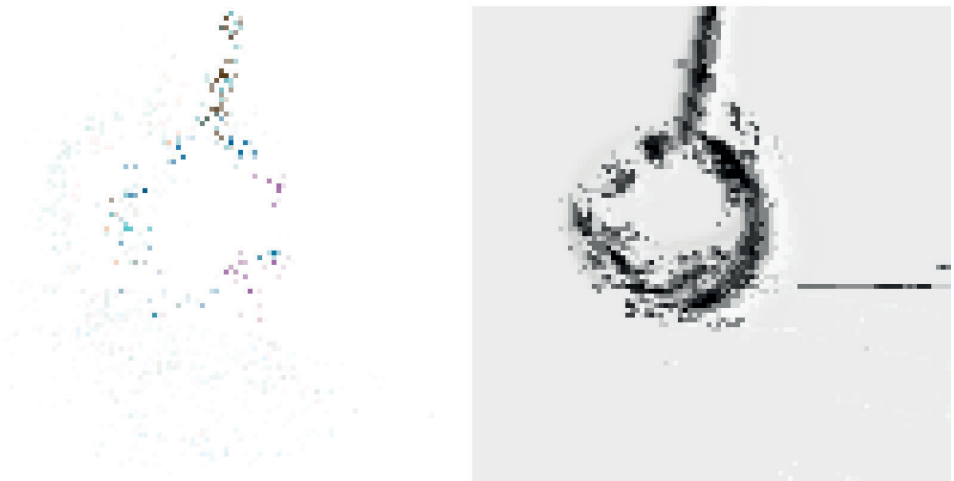
(a) With preprocessor.



(b) Without preprocessor.

Figure 4.10: Heatmap of seed weights on the upper row and final weights on the lower row, for 12 neurons trained with and without the preprocessor.

To test the effect of the preprocessor, one set of 12 neurons were trained on the raw video and one set on the preprocessed video for 600 repetitions. As seen in Fig. 4.10 the neurons receiving preprocessed data picked up more distinct features and learned faster compared with the neurons receiving raw video. Further, with the preprocessor, the neurons seemed to pick up more diverse features.



(a) Sphere.

(b) Cube.

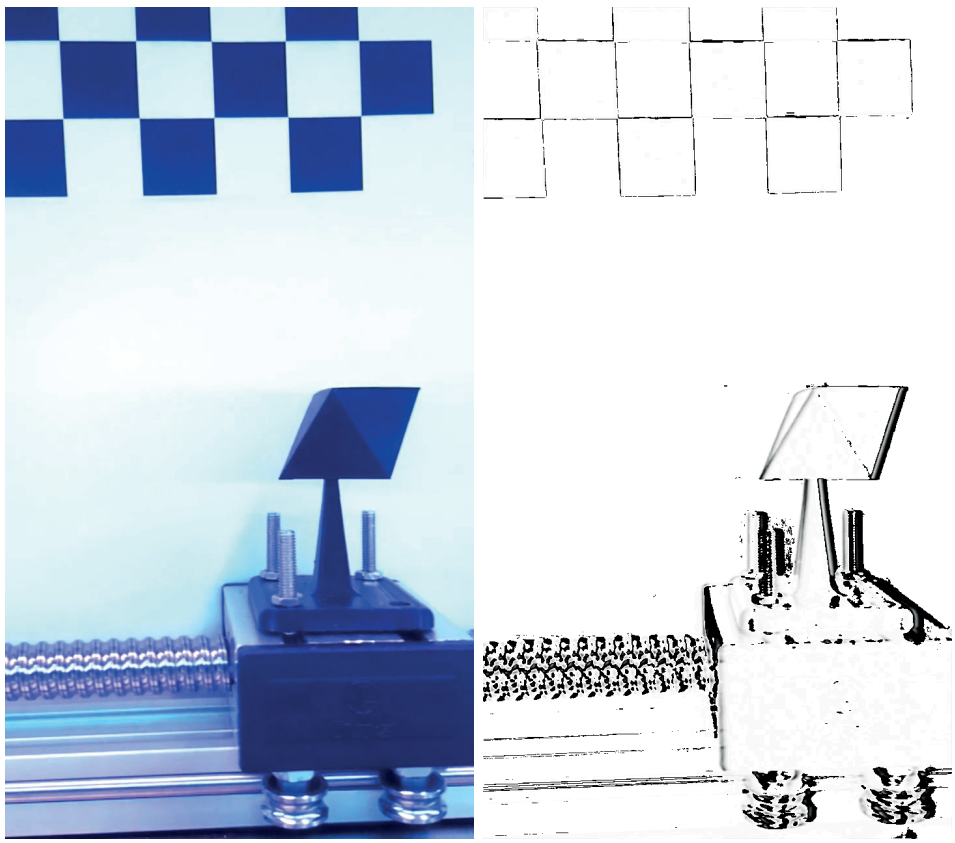
Figure 4.11: A single frame from each of the two rendered videos fed through the preprocessor to the right in each sub-figure. The left images, in each sub-figure, show inputs to the same neurons trained on preprocessed data as above. Each color represent input to one neuron and the intensity represents the amount.

Fig. 4.11 shows one frame from each rendered video after the preprocessor,

along with that frame multiplied by the neuron weights. This shows the input from each pixel to a neuron. Each color represents input to one specific neuron, and the intensity represents the amplitude of the input. The neurons used are the same as above, trained on preprocessed data. It can be seen that the two stimuli activate different neurons and differing amounts, showing that the neurons have picked up different features, some common to both stimuli, but some unique to one.

4.2 Results from experimental data

In Fig. 4.12b an example of a frame after preprocessing of one video from the recording set-up, is shown. The preprocessor seems to reliably detect edges and filter out small static variations in the background.



(a) Original. **(b)** After preprocessor.

Figure 4.12: Single frame of recorded video with octahedron as stimulus.

Training 12 dynamic neurons on one set of recorded videos of each of the five stimuli, at a speed of 3.27 cm/s and light condition lc, generated final weights

which seems to have some structure. The seed weights were uniformly, randomly distributed in a square area for half the neurons and a circular area for the other half. This was done to not bias the neurons to find features constrained by only one type of border of the receptive fields. However, as seen in fig 4.13 the shape of the receptive fields did not noticeably impact the conformation of high final weights.

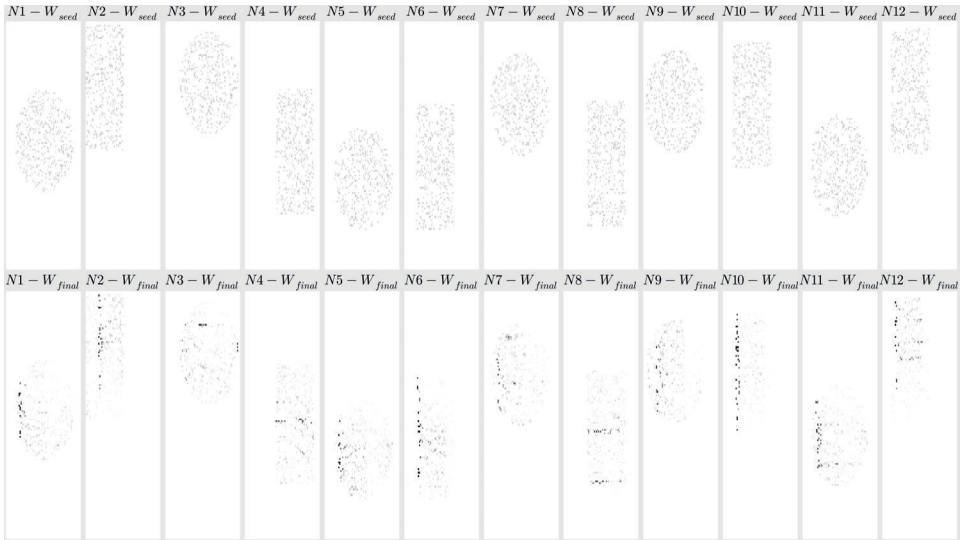
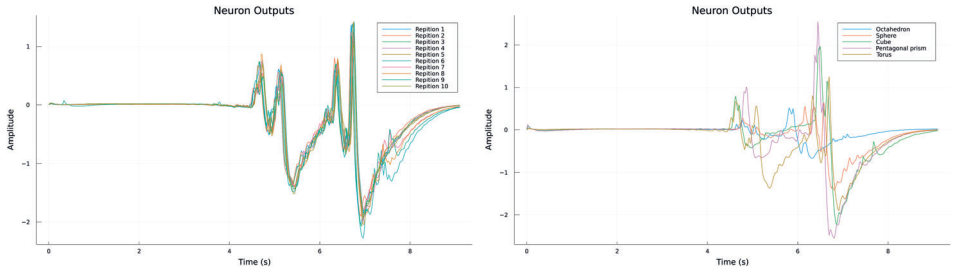


Figure 4.13: Heatmap of seed weights on the upper row and final weights on the lower row, for 12 neurons with 400 randomly selected synapses within different square or circular areas, for each neuron. The neurons were trained 250 times on one video of each object from the recording setup.

4.2.1 Repetitions

Inputting 10 different video of the same stimulus with the same recording settings, to one of the trained neurons, gives similar responses. In Fig. 4.14a this can be seen with the torus as the stimulus. Comparing this with Fig. 4.14b it can be seen that the variation of responses are greater between stimuli than between repetitions with the same stimulus, showing that the outputs are sensitive to the stimulus.

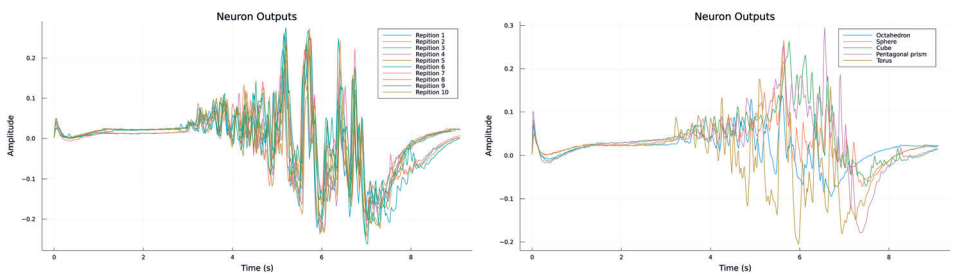


(a) Outputs from inputs of repetitions of the tours with the same recording settings.

(b) Outputs from inputs of the different stimuli with the same recording settings.

Figure 4.14: Neuron outputs, from one pre-trained neuron, given different inputs recorded with the same parameter settings ($V = 3.27\,\mathrm{cm/s}$ and light condition = lc).

Fig. 4.15 shows the outputs of the same neuron, with the same inputs as in Fig. 4.14, but before the neuron had been trained. Here, the outputs from inputs of repetitions of the same stimulus looks more varied as compared with the outputs after training. The responses for inputs of different stimuli also looks more similar to each other than the outputs after training did. This indicates that the neurons learn useful weights for differentiating the different stimuli. This can also be seen from the pair wise cross correlations. As seen in tab. 4.2 the mean cross correlation between one repetition and the others are higher for trained neurons than untrained ones. However, the mean cross correlation between the output with the torus as stimuli and the rest is lower for untrained neurons. On the other hand the outputs looks less structured and more chaotic here, which would result in the lower cross correlation, and could be sensitive to small changes in the input, giving worse information representation.



(a) Outputs from inputs of repetitions of the tours with the same recording settings.

(b) Outputs from inputs of the different stimuli with the same recording settings.

Figure 4.15: Neuron outputs, from one untrained neuron, given different inputs recorded with the same parameter settings ($V = 3.27\,\mathrm{cm/s}$ and light condition = lc).

Looking at the similarity matrix of outputs from the videos of the different stimuli as inputs, in Fig. 4.16, it can be seen that the sphere, cube and pentagonal

prism all are highly correlated. The most differentiated output comes from the octahedron.

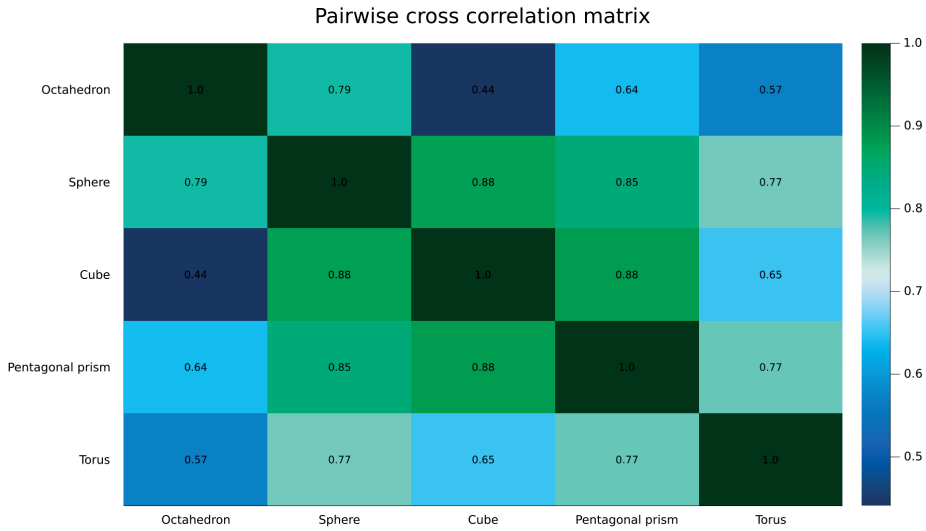


Figure 4.16: Similarity matrix of outputs from one pre-trained neuron when shown the different stimuli. The pairwise cross-correlations are indicated, by the color and number, where 1 indicates 100% similarity.

Training a LSTM neural network, with 9 units, to classify the stimuli from the neuron outputs resulted in 100 % accuracy on validation data, showing that the outputs are diverse enough for each stimulus, while being robust to at least the small changes between repetitions, to classify accurately. Doing the same on the outputs of untrained neurons did however result in an accuracy of 95 %, showing that even untrained neurons could give rather high accuracy in this simple experiment.

4.2.2 Different light conditions

The outputs from the trained neurons, with the same stimulus, at different light conditions, are more similar than that of different stimuli at the same light condition. This can be seen both from the cross correlations, as seen in tab. 4.2, but also by comparing Fig. 4.17 with Fig. 4.15b. The mean cross correlation between outputs, from inputs with the same stimulus, in light condition lc, with the others light conditions, is almost as high as the mean pair wise cross correlation of the outputs from repetitions of the same stimuli and recording parameters.

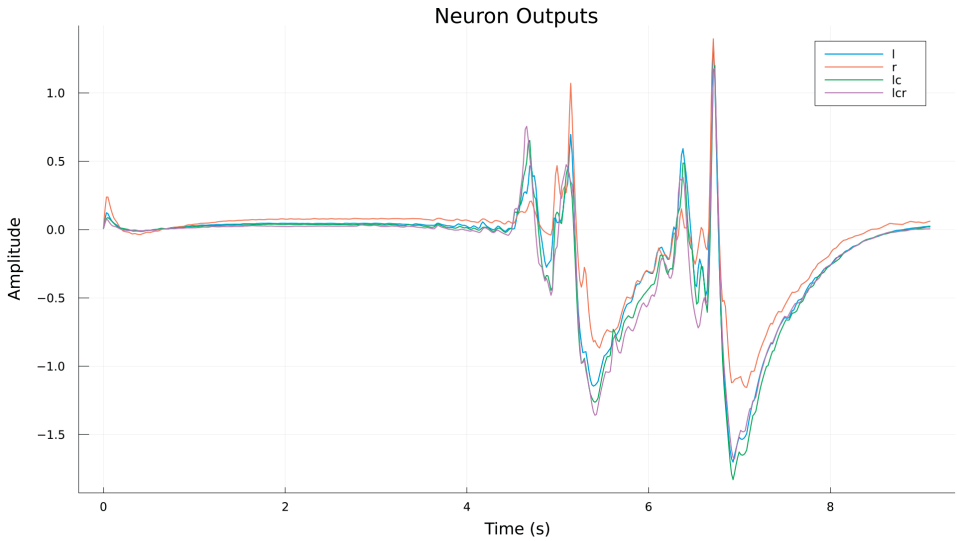


Figure 4.17: Neuron outputs, from one trained neuron, given input videos of the torus in different light conditions. ($V = 3.27$)

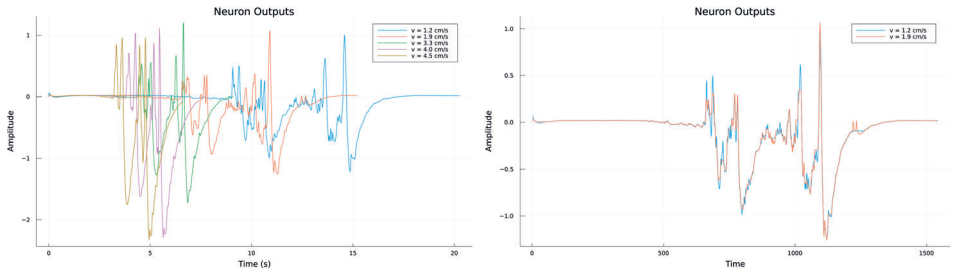
Using the first trained LSTM network described above, but with neuron outputs from input videos in the three, by the network, novel light conditions and the torus as stimulus, resulted in an accuracy of 58 %. Here all the confusions were with the sphere and in the two lower light conditions. With light only shining from the right, resulted in especially bad performance, with all predictions being for the sphere. In contrast, the higher light condition with all LEDs on, resulted in 100 % accuracy. The system thus seems more sensitive to lower light conditions than those in the training set. The torus and sphere are also similar and in lower light conditions the inner perimeter of the torus might be less prominent, making it even more similar to the sphere.

Using the second LSTM network trained on untrained neurons' outputs, resulted in even worse accuracy, predicting octahedron for all outputs from untrained neurons with the torus as stimuli in the novel light conditions. This again shows that the neurons learned useful weights.

4.2.3 Different speeds

The outputs of again the same trained neurons as above, but with the torus at different speeds as inputs, resulted in very different outputs, for the different speeds. Looking at Fig. 4.18a it could be suspected that these are however just warped in time. However, looking at the mean of pairwise cross correlations of the output, from an input video with the stimuli moving at the lowest speed, with up-sampled outputs from input videos with the higher speed settings, as seen in tab. 4.2, shows even lower cross correlation than between different objects. This could be a result of nonlinearities and instead using dynamic time warping gives a lot higher mean cross correlation. This could, though, be argued to just be peak matching, but looking at Fig. 4.18b, the outputs after dynamic time warping looks to follow each

other closely with some deviations in amplitude and a few small peaks missing.



(a) Raw outputs. **(b)** Dynamic time warping.

Figure 4.18: Neuron outputs, from one trained neuron, given input videos of the torus traveling at different speeds.

Again using the first trained LSTM, but now to classify neuron outputs from video inputs, at, to the LSTM, novel speed settings resulted in an accuracy of 65 %. Here both the two higher speed settings had higher accuracies of 88 % respectively. The lower speed settings instead resulted in accuracies of 28 % and 32 % for the speed settings of 1.24 cm/s and 1.89 cm/s respectively. This indicates that the system is more sensitive to lower velocities than what is has been trained on. Here however, both the neurons and the LSTM has only seen one velocity setting during training.

Training a new LSTM model to predict the stimuli from the neuron outputs of input videos recorded at the different speed settings resulted in an accuracy of 96 %. This shows that the neuron outputs are still distinct enough for different stimuli, even at varying speeds, to be classified with some accuracy. Doing the same, but instead training the LSTM to classify the speed of the stimulus, resulted in an accuracy of 68 %. This show that the neurons are more sensitive to the stimuli than to which speed they travel at, at least in combination with the LSTM. On the other hand, the neurons were only trained on different stimuli and had never seen different speeds, so this may be specific to these neurons.

Table 4.2: Mean pairwise cross correlations of the first output with the rest, for different varying sensing parameters, except for first the column, where all recording parameters were kept the same. For "V - **US**" the outputs for different speeds were up-sampled using linear interpolation, while for V - **DTW** they were dynamically time warped instead.

Varying:		**Non**	**Light**	V **- US**	V **- DTW**	**Stimuli**
	Mean:	98.4%	97.5%	55.2%	95.8%	74.5%
Trained:	Std:	0.77	1.2	22.7	3.06	9.88
	Mean:	90.3%	86.9%	44.5%	88.5%	68.5%
Utrained:	Std:	5.97	5.86	15.8	5.70	8.77

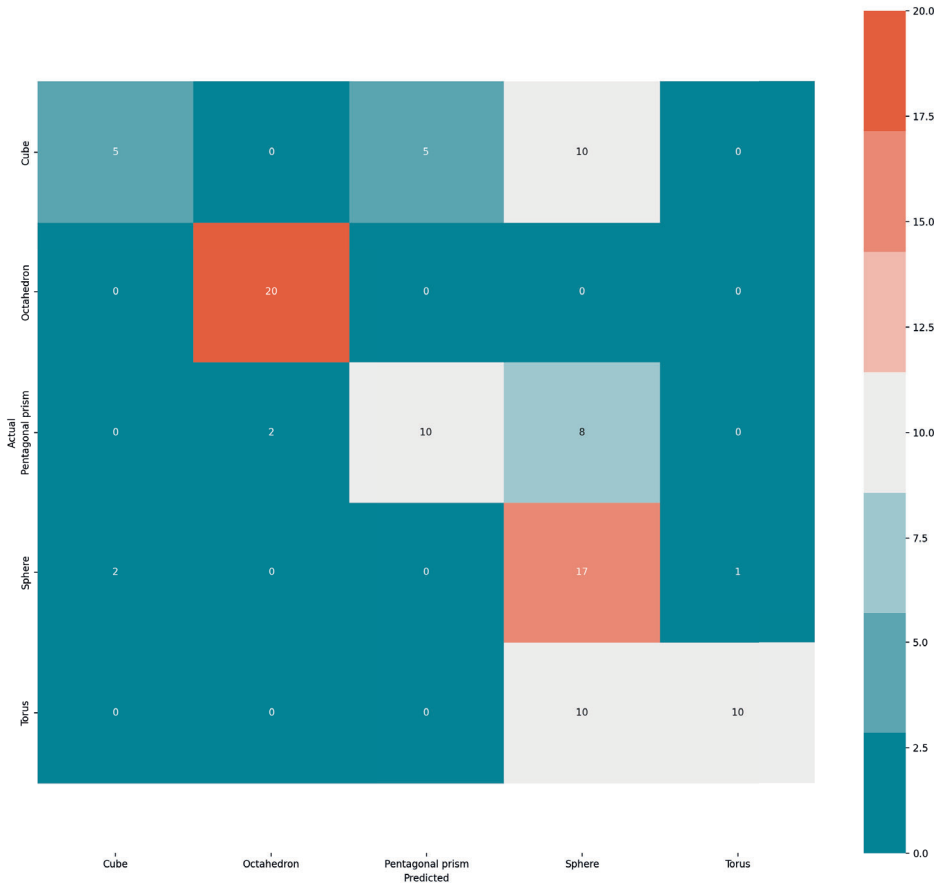


Figure 4.19: Confusion matrix for LSTM trained to classify the stimuli from the neuron outputs, with the 10 videos per stimuli with fixed recording parameters, as inputs, but tested on videos with stimuli moving at novel speeds.

4.2.4 Combination

Doing the same LSTM classification as before but, with neuron outputs from all the recorded data, resulted in an accuracy of 79 % on validation data. This shows the model can generalize to situations which it has never seen before. Looking at the confusion matrix in Fig. 4.20, it can be seen that the most common confusion is between the cube and the pentagonal prism. These are similar in size and outer perimeter when seen in profile, so this confusion does not seem to be random. The same is true for the second most common confusion, which is between the torus and the sphere.

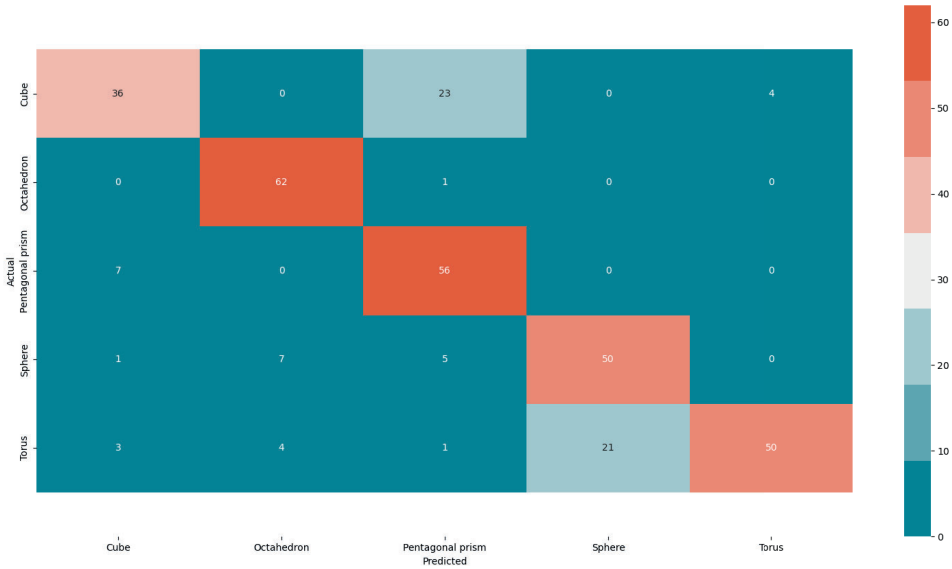


Figure 4.20: Confusion matrix from LSTM trained to classify the stimuli from the outputs of 12 pre-trained neurons with all the recorded training data as inputs.

4.2.5 Classical ANN

Using the smaller ConvLSTM network to classify the stimuli in the fixed parameter video set (light condition = lc and $v = 3.27\,\mathrm{cm/s}$), resulted in a maximum accuracy of 80 % after 10 trails. The best trained networks consistently confused either the sphere and the torus or the pentagonal prism with the sphere. The first is similar to the neuron model and is expected due to the stimulus similar shapes. The network, with the second confusion, could however correctly classify all examples of the torus in new light conditions. This method is thus more robust to light changes than the neuron model. This might be because the preprocessor filter out most things except for edges, and the edges are very differently highlighted in different light conditions.

With an accuracy of 49 % for novel speeds, the ConvLSTM performed worse than the neuron model. As seen in Fig. 4.21 the predictions were also less systematic and more spread out as compared with the neuron model, seen in Fig. 4.19.

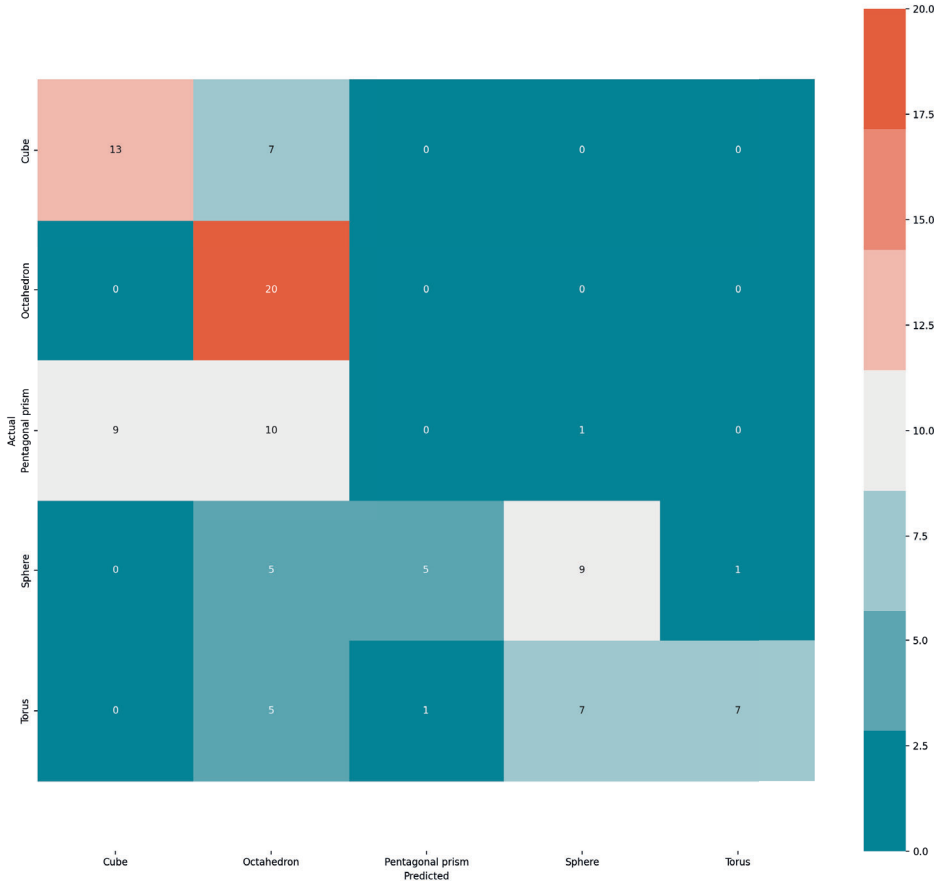


Figure 4.21: Confusion matrix for ConvLSTM trained to classify the stimuli directly from the 10 videos per stimuli with fixed recording parameters, but tested on videos with stimuli moving at novel speeds.

The eight layer ConvLSTM network could be trained to accurately (100 %) classify all the videos recorded at the fixed parameter settings. It also correctly classified the torus in all the novel light conditions, but performed even worse at novel speeds. For novel speeds, the resulting accuracy was 39 %.

This shows that the neuron model performs better in all aspects, except for in new light conditions, compared with the smaller ConvLSTM. The larger ConvLSTM performed as good on the validation data, better on the new light conditions, but worse on novel speeds, compared with the neuron model. The neuron model can, thus, despite having a fraction of the parameters, perform at least on par with these classical ANNs. It also trained faster on the same hardware (Intel® Core(TM) i7-9750H CPU @ 2.60GHz), taking 12 min for the neurons to learn and then 7 min for the LSTM, while the smaller ConvLSTM took 80 min. The ANNs could of course be trained much faster on a GPU. The neuron model is hard to com-

pare with this, because it was not implemented to be GPU comparable and might not be as serializable. On the other hand, the neuron model was implemented in a high-level programming language (Julia [27]), during the course of a master thesis and could be optimized much more, while the ANNs were implemented in TensorFlow.

Even larger and more optimized ANNs could probably perform even better, but with more parameters, overfitting become more problematic and training times increases.

Chapter 5
Conclusions and Future Work

5.0.1 Conclusions

The biologically inspired system demonstrates a novel, yet promising method for object classification in video data. With very little computational power and unlabeled training data, compared with traditional methods, it is able to pick out useful features in a dynamic environment. The system performed on pare with a classical ANN consisting of a convolutional LSTM. The LSTM was better at generalizing to new light conditions, but worse with new speeds. On the CPU our model also trained faster. This shows that a biologically inspire model with orders of magnitude fewer parameters can perform the same tasks as an ANN, at least for these experiments.

The preprocessor, paired with a cheap camera, shows a low price option to get retinomorphic like data without an expensive event camera. This data also show benefits in learning, at least to this system, compared with normal frame based videos. The results are promising, although further work is needed to prove the system in useful applications.

5.0.2 Future Work

This work has proven the usefulness of a biomimicking approach to computer vision in an experimental setting. Future work could thus continue to extend the model to more general applications in real-world situations. For more complex tasks, a multi-layer network of neurons might prove useful and would be an interesting continuation. Deployment of the model with continuous live learning in an embedded system could be an additional aspect to explore.

Bibliography

[1] Haidi Zhu et al. "A Review of Video Object Detection: Datasets, Metrics and Methods". In: *Applied Sciences* 10.21 (2020). ISSN: 2076-3417. DOI: 10.3390/app10217834. URL: https://www.mdpi.com/2076-3417/10/21/7834.

[2] N. V. Kartheek Medathati et al. "Bio-inspired computer vision: Towards a synergistic approach of artificial and biological vision". In: *Computer Vision and Image Understanding* 150 (2016), pp. 1–30. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2016.04.009. URL: https://www.sciencedirect.com/science/article/pii/S1077314216300339.

[3] Miquel Perello Nieto. *Human visual pathway*. 2015. URL: https://en.wikipedia.org/wiki/Visual_system#/media/File:Human_visual_pathway.svg (visited on 10/14/2021).

[4] Christoph Posch et al. "Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output". In: *Proceedings of the IEEE* 102.10 (2014), pp. 1470–1484. DOI: 10.1109/JPROC.2014.2346153.

[5] J. B. Demb and J. H. Singer. "Functional Circuitry of the Retina". In: *Annual review of vision science* 1 (2015), pp. 263–289. ISSN: 0896-6273. DOI: https://doi.org/10.1146/annurev-vision-082114-035334. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5749398/.

[6] BSIP. *Retina, Drawing*. 2016. URL: https://quest-eb-com.ludwig.lub.lu.se/search/181_769761/1/181_769761/cite (visited on 10/14/2021).

[7] Tim Gollisch and Markus Meister. "Eye Smarter than Scientists Believed: Neural Computations in Circuits of the Retina". In: *Neuron* 65.2 (2010), pp. 150–164. ISSN: 0896-6273. DOI: https://doi.org/10.

1016/j.neuron.2009.12.009. URL: https://www.sciencedirect.com/science/article/pii/S0896627309009994.

[8] Theodore G. Weyand. "The multifunctional lateral geniculate nucleus". In: *Reviews in the Neurosciences* 27.2 (2016), pp. 135–157. DOI: doi:10.1515/revneuro-2015-0018. URL: https://doi.org/10.1515/revneuro-2015-0018.

[9] Henrik Jörntell et al. "Segregation of Tactile Input Features in Neurons of the Cuneate Nucleus". In: *Neuron* 83.6 (2014), pp. 1444–1452. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2014.07.038. URL: https://www.sciencedirect.com/science/article/pii/S0896627314006497.

[10] Udaya B. Rongala et al. "Intracellular Dynamics in Cuneate Nucleus Neurons Support Self-Stabilizing Learning of Generalizable Tactile Representations". In: *Frontiers in Cellular Neuroscience* 12 (2018), p. 210. DOI: 10.3389/fncel.2018.00210. URL: https://www.frontiersin.org/article/10.3389/fncel.2018.00210.

[11] K. Harris and T Mrsic-Flogel. "Cortical connectivity and sensory coding". In: *Nature* 503 (2013), pp. 51–58. DOI: https://doi.org/10.1038/nature12654.

[12] Corinna Darian-Smith and Karen M. Fisher. "4.24 - Plasticity of the Somatosensory System After Injury". In: *The Senses: A Comprehensive Reference (Second Edition)*. Ed. by Bernd Fritzsch. Second Edition. Oxford: Elsevier, 2020, pp. 382–398. ISBN: 978-0-12-805409-3. DOI: https://doi.org/10.1016/B978-0-12-809324-5.24206-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128093245242065.

[13] Fredrik Bengtsson et al. "Integration of Sensory Quanta in Cuneate Nucleus Neurons In Vivo". In: *PLOS ONE* 8.2 (Feb. 2013), pp. 1–11. DOI: 10.1371/journal.pone.0056630. URL: https://doi.org/10.1371/journal.pone.0056630.

[14] Christoph Posch et al. "Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output". In: *Proceedings of the IEEE* 102.10 (2014), pp. 1470–1484. DOI: 10.1109/JPROC.2014.2346153.

[15] Guillermo Gallego et al. "Event-based Vision: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. ISSN: 1939-3539. DOI: 10.1109/tpami.2020.3008413. URL: http://dx.doi.org/10.1109/TPAMI.2020.3008413.

[16] Phantom High Speed. *v2512*. Retrieved 2021-10-04. URL: https://www.phantomhighspeed.com/products/cameras/ultrahighspeed/v2512.

[17] iniVation. *DAVIS346*. Retrieved 2021-10-04. URL: https://shop.inivation.com/collections/davis346/products/davis346.

[18] Zhong-Qiu Zhao et al. *Object Detection with Deep Learning: A Review*. 2019. arXiv: 1807.05511 [cs.CV].

[19] Ekim Yurtsever et al. "A Survey of Autonomous Driving: Common Practices and Emerging Technologies". In: *IEEE Access* 8 (2020), pp. 58443–58469. ISSN: 2169-3536. DOI: 10.1109/access.2020.2983149. URL: http://dx.doi.org/10.1109/ACCESS.2020.2983149.

[20] L. Brigato and L. Iocchi. *A Close Look at Deep Learning with Small Data*. 2020. arXiv: 2003.12843 [cs.LG].

[21] J. Lever, Krzywinski, and N M. Altman. "Model selection and overfitting". In: *Nat Methods* 503 (2013), pp. 51–58. DOI: https://doi.org/10.1038/nmeth.3968.

[22] Udaya B. Rongala et al. "A Non-spiking Neuron Model With Dynamic Leak to Avoid Instability in Recurrent Networks". In: *Frontiers in Computational Neuroscience* 15 (2021), p. 43. ISSN: 1662-5188. DOI: 10.3389/fncom.2021.656401. URL: https://www.frontiersin.org/article/10.3389/fncom.2021.656401.

[23] Udaya B. Rongala et al. "Cuneate spiking neural network learning to classify naturalistic texture stimuli under varying sensing conditions". In: *Neural Networks* 123 (2020), pp. 273–287. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2019.11.020. URL: https://www.sciencedirect.com/science/article/pii/S0893608019303880.

[24] *Accuracy (trueness and precision) of measurement methods and results*. Standard. Geneva, CH: International Organization for Standardization, Mar. 2020.

[25] Lining Hu. and Yongfu Li. "Micro-YOLO: Exploring Efficient Methods to Compress CNN based Object Detection Model". In: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,* INSTICC. SciTePress, 2021, pp. 151–158. ISBN: 978-989-758-484-8. DOI: 10.5220/0010234401510158.

[26] Raspberry Pi. *Raspberry Pi Documentation*. Retrieved 2021-10-04. URL: https://www.raspberrypi.org/documentation/accessories/camera.html.

[27] Jeff Bezanson et al. “Julia: A Fresh Approach to Numerical Computing”. In: *SIAM Review* 59.1 (2017), pp. 65–98. DOI: 10.1137/141000671. eprint: https://doi.org/10.1137/141000671. URL: https://doi.org/10.1137/141000671.

Printed by Tryckeriet i E-huset, Lund 2022

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2022-864
http://www.eit.lth.se