BACHELOR DEGREE PROJECT IN MATHEMATICS, OPTIMIZATION
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2019

Control of inverted pendulum by learning

SA114X och SA115X

<Gustaf Holte and Måns Rasmussen>

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Abstract

The inverted pendulum attached to a cart moving in one dimension, also called a CartPole, is considered to be the most fundamental example on control theory and optimization. In this report the goal was to try and solve the unstable, non-linear stabilization problem using machine learning, more specifically using artificial neural networks. Using this to control the CartPole to be able to swing up and stabilize the pendulum where the mass of the cart as well as the mass and length of the pendulum was set randomly at the start of each simulation.

Furthermore, to find out how far it is possible to push the artificial neural network outside of the limits of training and still get a stable result. This was accomplished by doing thousands of simulations, coded in Python using packages like TFLearn, Tensorflow and the gym environment from openAI. The results showed that it is in fact possible to control the CartPole system using artificial neural networks, also showing that it is robust against new CartPole parameters going far outside of what the network had been trained for.

Keywords

- CartPole
- ANN - Artificial neural network
- Machine learning
- Inverted pendulum

Abstract

Den inverterade pendeln fäst på en vagn som kan röra sig längs en riktning, även kallad CartPole, anses vara det mest grundläggande exemplet inom reglerteknik och optimering. I denna rapport var målet att försöka lösa detta instabila, olinjära stabiliseringsproblemet med hjälp av maskininlärning, mer specifikt med ett artificiellt neuronnät. I början av varje simulation ändrades vagnens massa samt pendelns massa och längd slumpmässigt. Dessutom undersöktes hur robust modellen var, genom att ta reda på hur långt man kunde ändra CartPole parametrarna utanför de gränser som modellen har tränats för. Detta uppnåddes genom att göra tusentals simuleringar, allt kodat i Python med paket som TFLearn, Tensorflow och gym från openAI. Resultaten visade att det faktiskt är möjligt att kontrollera CartPole-systemet med hjälp av ett neuronnät, samt visade att det är en robust modell som klarar av nya CartPole parametrar som går långt utanför det område som neuronnätet hade tränats för.

Nyckelord

- CartPole
- ANN - Artificiellt neuronnät
- Maskininlärning
- Inverterad pendel

Acknowledgements

We would like to acknowledge our supervisor and examiner Professor Hu for helping us define the scopes of this project as well as providing guidance and assistance whenever we got stuck.

Authors

Gustaf Holte <gholte@kth.se>
Måns Rasmussen <manra@kth.se>

Place for Project

KTH Royal Institute of Technology
Stockholm, Sweden

Examiner

Professor Hu Xiaoming
Department of Mathematics and Optimization
KTH Royal Institute of Technology

Supervisor

Professor Hu Xiaoming
Department of Mathematics and Optimization
KTH Royal Institute of Technology

Contents

1 Introduction

The inverted pendulum mounted on a cart moving in one dimension (also known as a CartPole) has long been seen as a fundamental classroom example for an unstable, highly non-linear system. Which means the system can't be modelled successfully using only linear equations. To solve this problem, one first need to set up the non-linear physical model and then move on to designing a controller for the system. This can be done in several different ways, all with their own pros and cons. Some examples of controllers being: linear quadratic regulator (LQR), PID control, neural network control, etc. However, the main goal for this report will be to solve this unstable CartPole problem using machine learning, more specifically an artificial neural network controller.[1][2]

This report is written as a bachelor's degree thesis in mathematics at the undergraduate level, with a focus on optimization and system theory at the SCI school at the Royal Institute of Technology by the students Gustaf Holte and Måns Rasmussen.

1.1 CartPole

Figure 1.1 shows the setup of the cart with the inverted pendulum, called CartPole[1]. The cart has mass M, the pendulum has length L and the mass m and is fixed in the centre of the cart. The vector F is the force that pushes the carriage horizontally to swing the pendulum and keep it balanced in its upright position. The vectors H and V are the forces that the cart applies to the pendulum in horizontal and vertical directions.

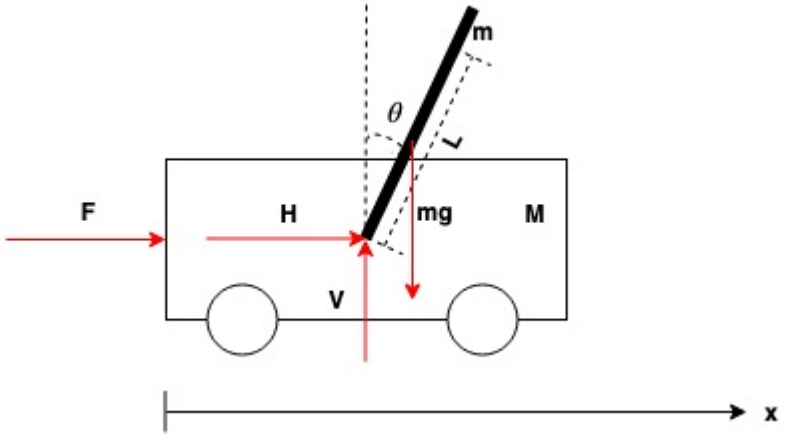


Figure 1.1: An overall illustration on the inverted pendulum and its components (without any frictional forces or the gravity force from the cart).

[1]A cart with a pendulum attached to its centre.

1.2 Artificial neural network

An artificial neural network, or ANN, is a framework that can support many different machine learning algorithms and is inspired of the structure of animal brains, like the human brain. An animal brain consists of millions of neurons all connected in an enormously complex network. A neuron is built up by four main parts, dentaries, a cell body, an axon and synaptic endings. The dentaries receive electrical signals from other neighbouring neurons and once it reach a threshold value, summed up in the cell body, it fires its own electrical signal through the axon to the next connected neurons though its synaptic endings.

An artificial neural network is built up in the same way, with many neurons all of which is connected in different layers as shown in figure 2.1. The basic structure is to have an input layer, some hidden layers and one output layer. How an ANN learns and knows what to respond for a given input is by applying different weights between all the interlinking neurons. In the input layer, a set of signals is sent in which is later sent around in the inner layers in ways depending on these weights. Lastly, in the final layer using the same principal of summation as in biological neuron, the neuron with the largest sum of incoming signals is chosen as the output signal of the ANN. Which is used to decide which action is to be used, following that specific input given. Depending on the training data, there are mainly three different ways one can implement to train an ANN, reinforced, supervised and unsupervised learning.[3]

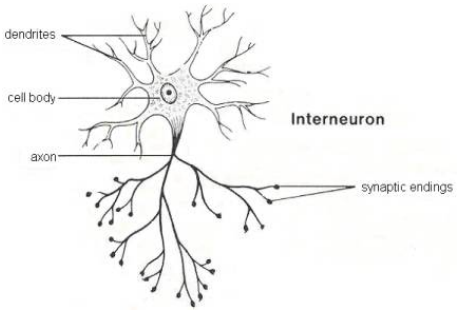


Figure 1.2: Illustration of a neuron.

In reinforced learning, a reward and scoring system is implemented in order to achieve a good network. The idea is that you want to get the ANN that aims to get such a high score as possible. Using a scoring system to be able to separate a good set of training data from the rest and later train the ANN on that good set. This is very useful when the achieved output is not directly known given a specific input.

In supervised learning, the ANN is trained on already known input- and output values. Thereby, telling the ANN how to behave given the known set of training data. A good example is in image recognition, where one can instruct the network on what each picture is meant to portrait. Thereafter having the ANN guess by itself, once trained.

Lastly, in unsupervised learning the ANN updates its weights based on the input data only. Not knowing the output or having another way of telling if the ANN is getting better (like with a point system), making it impossible to tell how accurate they are. Although many real-world problems can be applicable in this way.

When setting up an ANN, there are many different parameters that can be varied in order to get such an optimized network as possible for the given problem. Mainly by varying the parameters: learning rate[4], keep probability [5], epochs [6], numbers of neurons, layers, optimizers [7] and activation functions [8].

1.3 Problem description

The purpose of this report is to show that it is possible to control an unstable, non-linear CartPole system using an artificial neural network controller. Where the mass of the cart as well as the mass and length of the pendulum is set randomly at the start of each simulation. Doing this by first swinging up the pendulum from its stable starting position, pointing straight down. To its unstable position, pointing straight up, and balancing it there, using a force applied on the cart, for an extended period of time. Also, to investigate how different sized networks affect the performance of the ANN controller.

Furthermore, finding out how robust the artificial neural network is when the CartPole's parameters (pole length, pole mass and cart mass) go beyond the range of what the network has trained for.

1.4 Limitations

To limit the problem to a more manageable size, that can be solved on a regular computer (by 2019 standards) within our limited timeframe for this thesis, the following restraints where set for the CartPole system:

- The length of the track was set to ± 50 m.
- The CartPole have one rigid pendulum pole attached to its centre.
- Neglected any frictional forces.

The following parameters for the CartPole where varied and studied:

- Length of the pole.
- Mass of the pole.
- Mass of the cart.
- The number of neurons and layers of the ANN.

Lastly, the following where varied until a suitable combination where found, after which they where held constant for the rest of the study.

- The forces applied to the cart in horizontal direction.
- Score limit, to determine which training data that should be saved.
- Number of runs as well as the number of steps per run.
- The parameters for the ANN (see table 4.1 under network parameters)

2 Models

In this section the models used in the report are explained in more detail.

2.1 Physical model

The derivation for the physical model has been using the figure 1.1 found in section 1.1.[9]

Summing all the forces for the cart in horizontal direction:

$$\rightarrow: M\ddot{x} = F - H \tag{1}$$

Summing all the forces for the pole in horizontal direction:

$$H = m\ddot{x} + mLcos\theta\ddot{\theta} - mLsin\theta(\dot{\theta})^2 \tag{2}$$

Combining equation 1 and 2 gives:

$$(M+m)\ddot{x} + mLcos\theta\ddot{\theta} - mL\dot{\theta}^2 sin\theta = F \tag{3}$$

Summing all the forces perpendicular to the pole:

$$mg - V = mL\left(-sin\theta\ddot{\theta} - cos\theta(\dot{\theta})^2\right) \tag{4}$$

Newton's second law applied to the rotary motion of the pendulum provides:

$$mL^2\ddot{\theta} = mgLsin\theta + VLsin\theta - HLcos\theta \tag{5}$$

Combining equations 2, 4 and 5 along with some trigonometric relationships gives:

$$\ddot{\theta} = \frac{gsin\theta - cos\theta\ddot{x}}{L\left(\frac{4}{3} - m\frac{cos^2\theta}{M+m}\right)} \tag{6}$$

Finally, rewriting equation 3 gives:

$$\ddot{x} = \frac{F + L\dot{\theta}^2 sin\theta - mL\ddot{\theta}cos\theta}{M+m} \tag{7}$$

2.2 Artificial neural network

Neural networks can have various shapes and sizes with no telling which might be best suited for any problem one attempt to solve. Because of that there are only two ways to go. The first one is trial and error, finding network by testing many different versions. The second is using someone else's ANN that worked well for a similar problem.

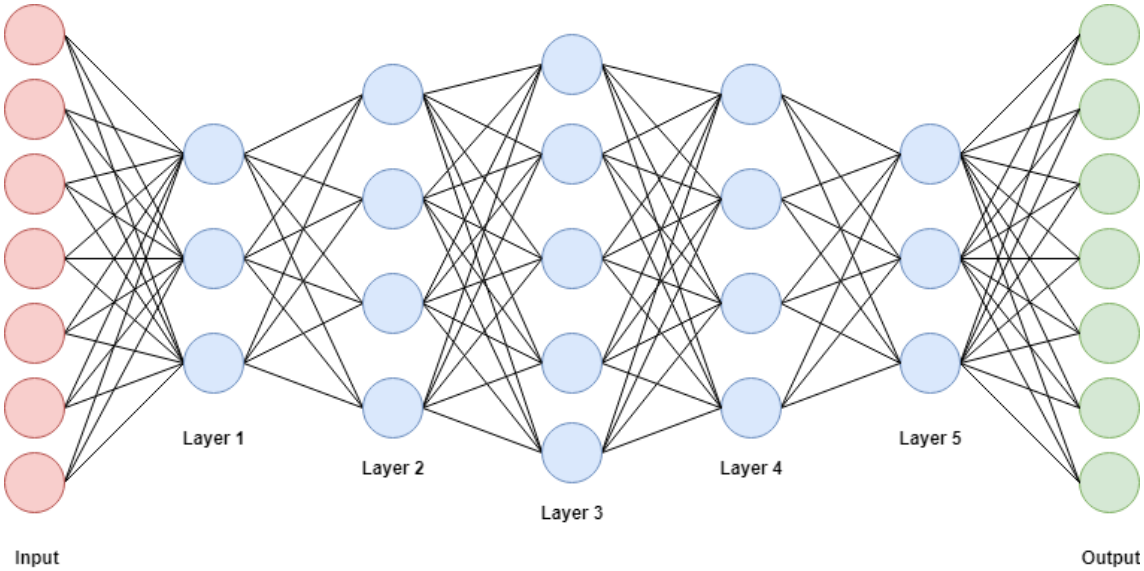


Figure 2.1: Sketch of the layout for the neural network that was used for the medium sized network (the hidden layers are not to scale).

In figure 2.1 you find one of the neural networks that was used during the project. More specific this was the setup for the medium sized one (not to scale). It consists of seven layers. One input layer (red) as well as one output layer (green), both with seven nodes. This was keept as standard for all different ANNs being tested in this report. Between the input and output layers are the hidden layers (blue). All three different networks can be found in section 4, see table 4.2 for the small network, table 4.1 for medium sized and table 4.4 for the largest.

The most common way to create an ANN is to use a package such as TensorFlow or TFLearn. This makes it simple to set up an ANN, train it as well as use it for predictions in an application, without needing to code everything from scratch. TFLearn was used in this report and more information about the package and how to use it can be found on the web page ([online] available at: <http://tflearn.org/>).[10]

3 Method

The work behind the report consisted of three main phases. The first phase consisted of a literature study, going in-depth on the CartPole problem and artificial neural networks. Focusing on what they are and how they work, as well as what has been done so far and how it could be implemented on this inverted CartPole problem. The second phase was the execution phase, this is where the programming and various simulations were made (all coding where done in Python) in order to achieve an optimized network. The last phase of the work consisted of writing the report and preparing for the oral presentation of the project.

3.1 Creating training data

Since no existing training data for this specific problem (with our chosen CartPole parameters and forces) could be found, creating new data was the best option. Following the reinforced training methodology, a point system was set up to enable a classification of good and bad runs, without having to view the animation first. Which decreases the running time of the simulations significantly. Then doing several millions of runs with a random action for every step and finally saving all the state variables along with the specific action for every step of the runs that got a score above the score limit as training data.

3.2 Training data

The state variables for the CartPole (also the input to the ANN) was the following:

$$\begin{bmatrix} x & \dot{x} & \theta & \dot{\theta} & L & m & M \end{bmatrix} \tag{8}$$

Where x represents the position along the x-axis (in meters), with the zero positioned at the centre of the track. $\dot{x}$ is the velocity of the cart (meters per second), θ is the angle of the pendulum (radians), with zero being when the pendulum points straight up. $\dot{\theta}$ is the angular velocity (radians per second), L is the length of the pole (in meters), m is the mass of the pole (in kilograms) and lastly M is the mass of the cart (in kilograms). At the start of each run the CartPole parameters (L, m and M) where randomly reset to some value in the range specified, and kept constant throughout that run.

The force vector F (also the output to the neural network), was the following:

$$F = \begin{bmatrix} -300 & -200 & -100 & 0 & 100 & 200 & 300 \end{bmatrix} N. \tag{9}$$

For each step during the simulation, one force (or action) was selected randomly from the force vector and applied on the CartPole.

This specific force vector was found after multiple tries with different forces and was found to be the most suitable for this specific set up of CartPole parameters (see table 4.1).

Lastly, the accepted training data was saved as an hdf5 file (suitable for saving large amounts of data).

3.3 Point system

Using the reinforced training method, a point system as shown in figure 3.1 was created in order to further differentiate between good and bad runs. This specific point system was developed after testing different variations and proved to be a suitable point system for this application. This training data was later on used to train the ANN model.

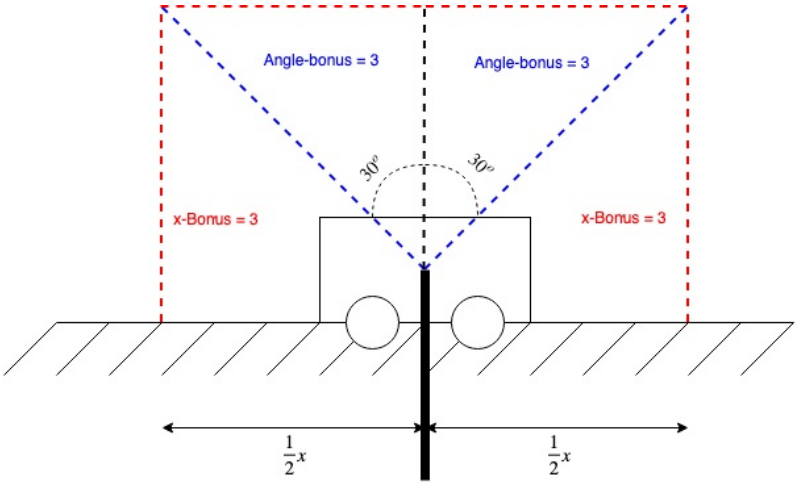


Figure 3.1: Sketch of the point system that was developed and used for the simulations.

Figure 3.1 shows how the different limits where set up for the point system. When the CartPole was inside of any of the ”bonus areas” the reward was multiplied by Angle-bonus or/and x-Bonus. However, if the CartPole was outside the ”bonus area” it just received one reward per time step. As for the Angle-bonus area, it was fixed to the cart and the Angle-bonus didn’t depend on the x position of the CartPole. No reward was given while the pole was below the ground.

The run continued as long as the pendulum did not cross the x-threshold (either to the left or to the right) or if the pendulum went below the ground ($\theta = \pm 90^o$, periodically), after it had been above first.

Lastly, when the run was done the total number of rewards per step was summarized and saved as the score for that run. If the score was above the score limit the training data was saved, containing all the state variables and actions for every step.

3.4 Training the artificial neural network

From the research done in phase one, it was clear that there is no exact combination of neurons and layers that works for each different problem one wishes to solve using machine learning. Therefore, three different sized ANN models were tried and compared (see section 4.2).

For the ANNs, the input and output size of the first and last layer in all of the networks was set to size seven so the data could be saved in a symmetrical matrix in a hdf5 file. The network itself was saved using TFLearns built in save function.

The TFLearn package was used to set up the ANN (see figure 2.1 as well as table 4.2 and 4.4), create the network, train the network (using the training data created as mentioned in section 3.2) and lastly, when trained, to use the trained ANNs to predict the next coming actions. Based on what state the CartPole was in the time step just before.[10]

3.5 Simulation environment

The simulation environment was set up in Python using openAI's ”gym” package that has the CartPole environment as described in section 1.1 with the physical equations as written in section 2.1. The semi-implicit Euler method was used in order to update the positions and velocities of the CartPole using the accelerations given by equations 6 and 7 with the time step tau = 0.02s, the gravitational constant was set to 9.8 m/s^2.[11]

4 Results

The results obtained below was a result of several hours of simulations using the following settings as displayed in table 4.1. The code itself can be found in the GitHub repository located in appendix.

–CartPole parameters–	**–Value–**
Cart mass (M)	1-2 [kg]
Pole mass (m)	0.1-1 [kg]
Pole length (L)	1-2 [m]
–Simultation parameters–	**–Value–**
Number of runs	10^7
Number of runs per step	500
Score limit	600
Time-step (tau)	0.02
–Network parameters–	**–Value–**
Learning rate	10^{-3}
Keep probability	0.8
Epoch	3
Activation	relu
Optimizer	adam

Table 4.1: The parameters used when creating the training data and figures in section 4.1. Cart mass, pole mass and pole length were randomly chosen in the range of values specified at the start of each run.

4.1 Training data

When creating the training data set, only the runs above score equal to 600 was saved as training data to be used in section 4.2. The training data set was created by using entirely random actions for each step, as described in section 3.2. This gave a file size of around 0.5 GB using score limit of 600 points, saving 33 081 runs out of 10 million, or 0,33 %. The mean score was 149.8 points and a mean time length (also corresponds to the mean number of iterations per run) of 83.6 steps, as seen in figure 4.1d. Since the time step (tau) was equal to 0.02s, this corresponded to an average time the pole was in movement above ground (before the run was terminated) to 1.7s. The score data seems to decrease exponentially as seen in figure 4.1d.

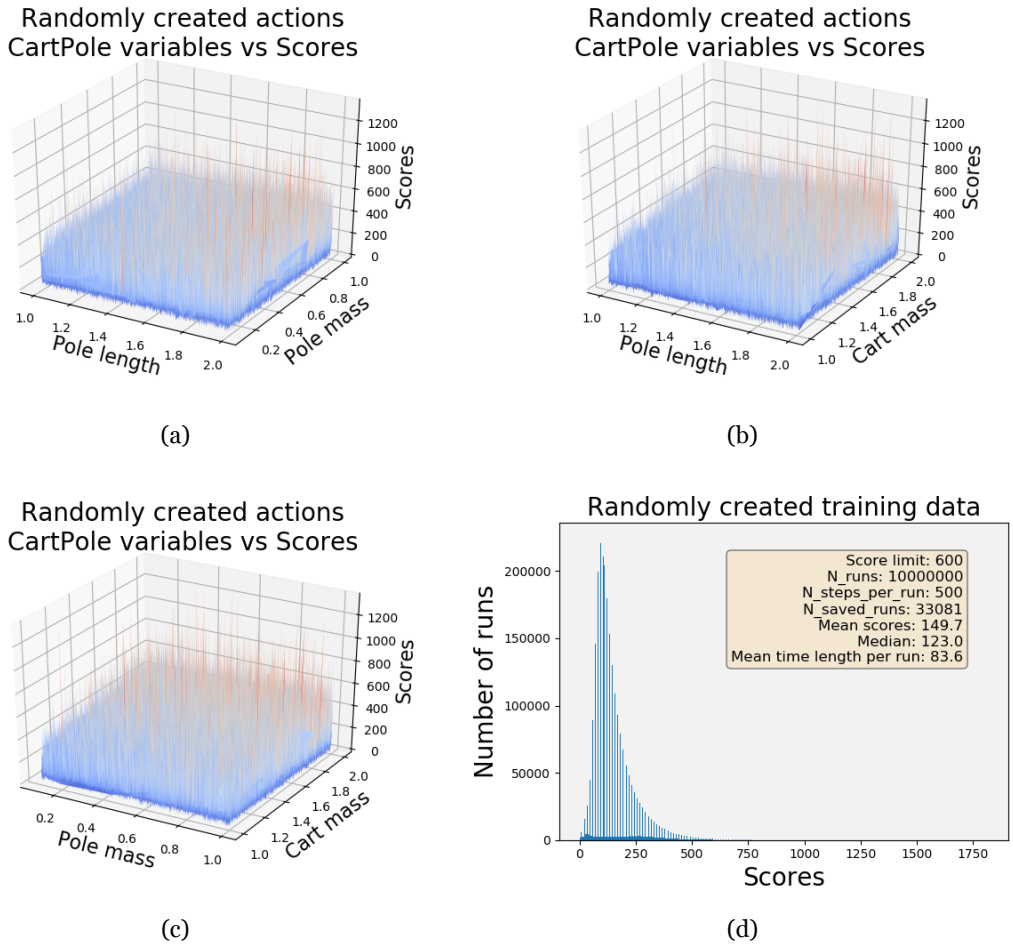


Figure 4.1: Shows the results recieved using randomly created actions with score limit 600 and steps per run equal to 500 for ten million simulations.

In figure 4.1 a)-c) it is possible to see how the scores where distributed for different combinations of the CartPole parameters. It clearly shows a random pattern with no obvious groups or formations taking place. The main bulk of the points lies around the mean at 150 points, but some peaks stand out in red with around 1 000 points, see figure 4.1a, 4.1b and 4.1c.

4.2 Simulating different sized neural networks

Using the same training data from section 4.1, three different sized neural networks where trained and compared to each other. Each one being running for 10 000 simulations with a maximum step length of 500 steps, as well as having the same constants and parameters as in table 4.1. With only the size and shape of the neural network being varied between them.

4.2.1 Small network

Neural Network	Input	7
	Layer 1	21
	Output	7

Table 4.2: The neural network design used when trying a smaller network.

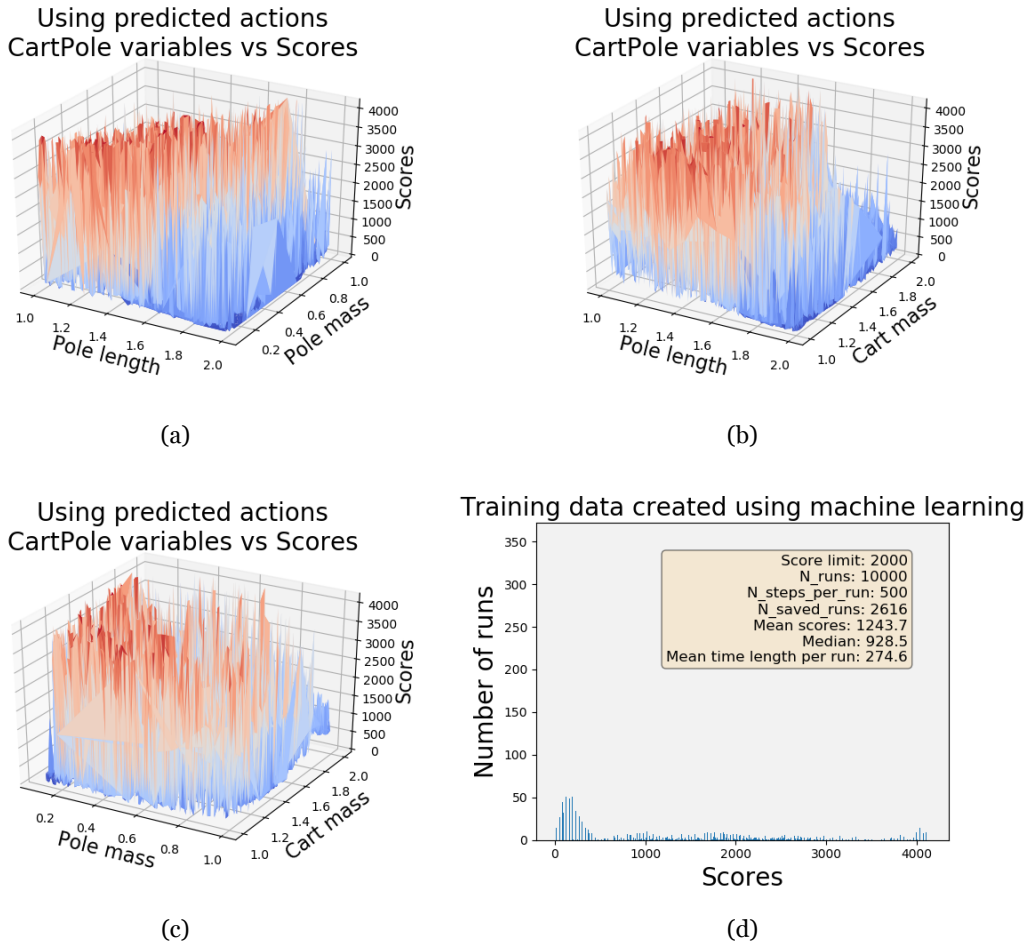


Figure 4.2: Shows the results received using the small sized neural network with steps per run equal to 500 for 10 000 simulations. The score limit was set to 2 000, saving training data above that limit.

The small network received a mean score of 1 247.6 points which is significant higher than the mean score achieved from figure 4.1, that was a result of only random actions. The mean time length per run was 273.7 steps which corresponds to an average time the pole was in movement above ground (before the run was terminated) to 5.5s.

4.2.2 Medium network

Neural Network	Input	7
	Layer 1	128
	Layer 2	256
	Layer 3	512
	Layer 4	256
	Layer 5	128
	Output	7

Table 4.3: The neural network design used when trying a medium sized network.

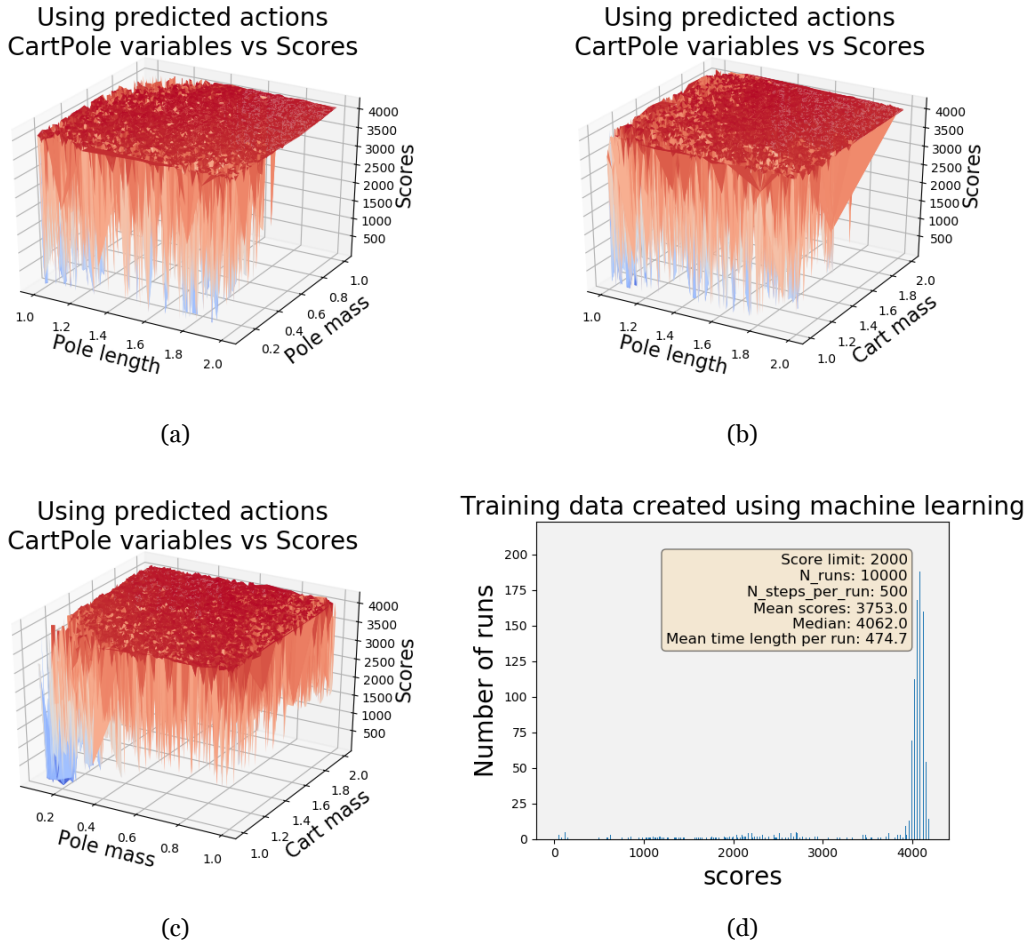


Figure 4.3: Shows the performance received using the medium sized neural network with steps per run equal to 500 for 10 000 simulations. The score limit was set to 2 000, saving training data above that limit.

The medium network received a mean score of 3 753.0 points which is even higher than the mean score achieved from figure 4.1 and 4.2, that was a result of only random actions. The mean time length per run was 474.7 steps which corresponds to an average time the pole was in movement above ground (before the run was terminated) to 9.5s.

4.2.3 Large network

Neural Network	Input	7
	Layer 1	250
	Layer 2	500
	Layer 3	750
	Layer 4	1 500
	Layer 5	750
	Layer 6	500
	Layer 7	250
	Output	7

Table 4.4: Shows the performance received using the large neural network with score limit 2 000 and steps per run equal to 500 for 10 000 simulations.

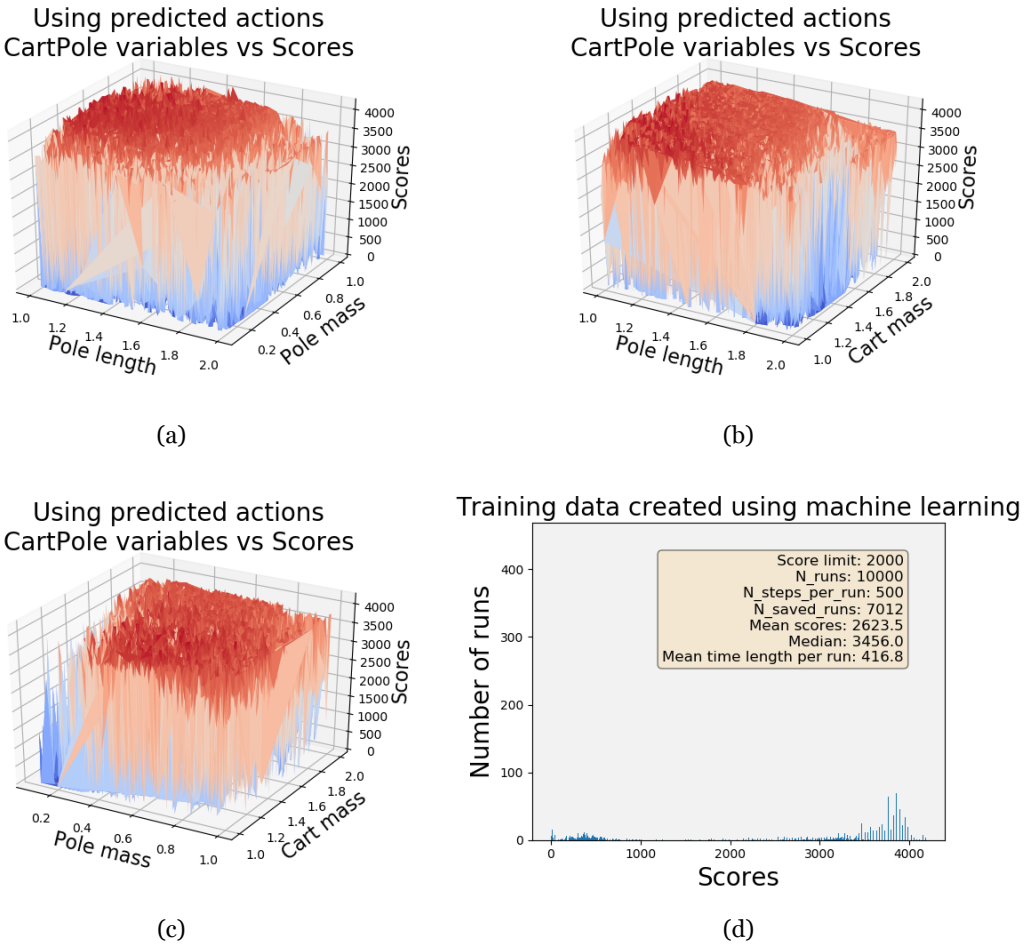
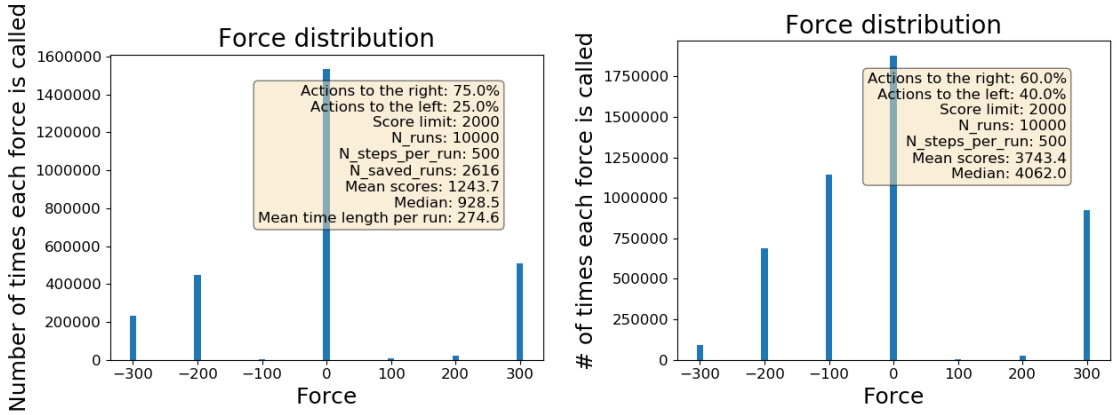


Figure 4.4: Shows the results received using the large sized neural network with steps per run equal to 500 for 10 000 simulations. The score limit was set to 2 000, saving training data above that limit.

The large network received a mean score of 2 623.5 points which is not as high as the medium sized network. The mean time length per run was 416.8 steps which corresponds to an average time the pole was in movement above ground (before the run was terminated) to 8.3s.

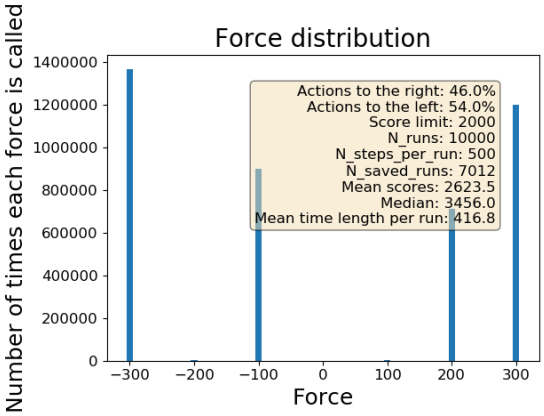
4.2.4 Force distribution

When running the simulations in section 4.2 for the small, medium and large ANNs the distribution of the predicted actions where saved and plotted as can be seen in figure 4.5.



(a) Small network

(b) Medium network



(c) Large network

Figure 4.5: Shows the force distribution received using small, medium and large sized neural network from the simulations, with steps per run equal to 500 for 10 000 simulations. The score limit was 2 000, saving training data above that limit.

4.3 Robustness

The robustness was tested using the neural network from section 4.2 with the best overall performance. Which was the medium sized network. The robustness was tested by changing the CartPole parameter range to the values in table 4.5, keeping the other parameters the same as before (see table 4.1 and 4.3).

CartPole parameters	Value	
	Test 1	Test 2
Cart mass (M)	0.1-10 [kg]	0.1-7 [kg]
Pole mass (m)	0.1-10 [kg]	0.1-7 [kg]
Pole length (L)	0.1-10 [m]	0.1-7 [m]

Table 4.5: The CartPole parameters used when running the simulations in section 4.3. Cart mass, pole mass and pole length were randomly chosen at the start of each run in the range of values specified.

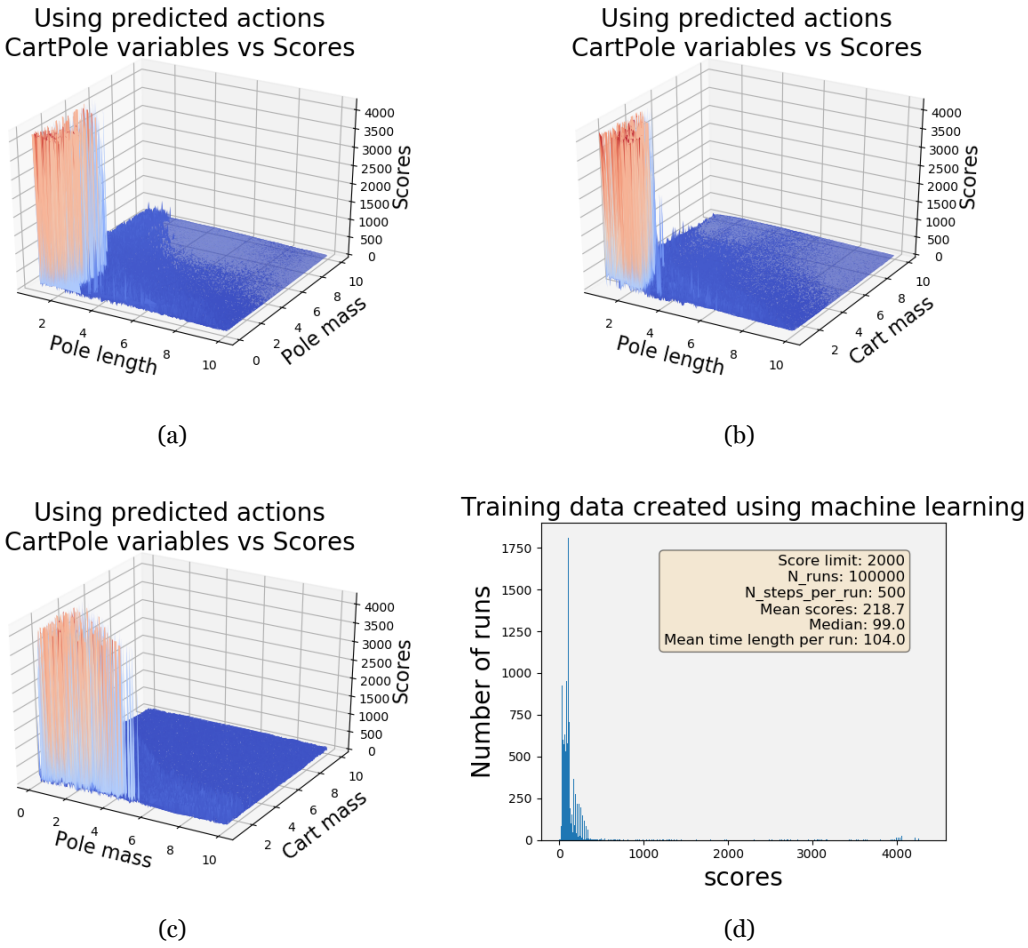


Figure 4.6: Shows the trained medium sized network's performance using and 500 steps per run and 100 000 runs with the CartPole parameters (m, M and L) ranging between 0.1 and 10. The score limit was set to 2000, saving training data above that limit.

Figure 4.6 shows that once the CartPole parameters goes towards 10, the scores tend quickly towards zero. Another simulation was made with a narrower range of the CartPole parameters (test 2 in table 4.5), as can be seen in figure 4.7.

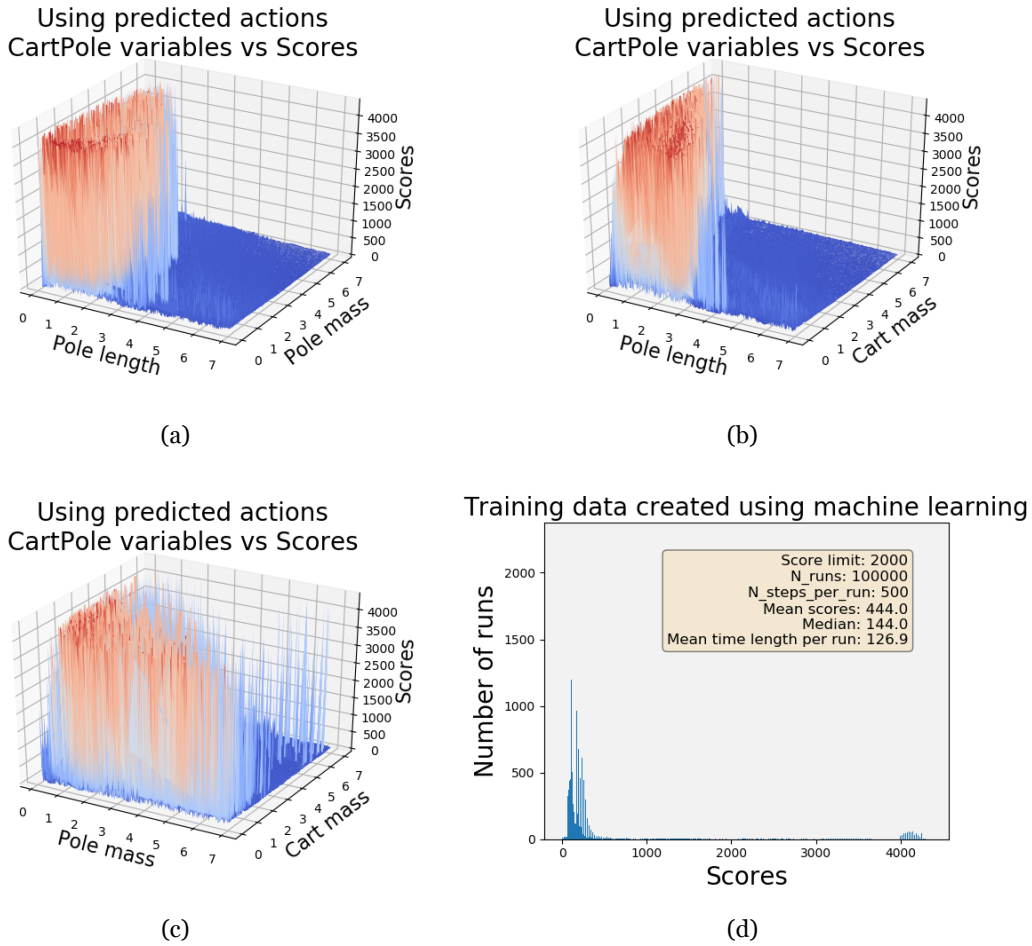


Figure 4.7: Trained network's actions using 500 steps per run and 100 000 runs, with the CartPole parameters (m, M and L) ranging between 0.1 and 7. The score limit was set to 2000, saving training data above that limit.

Figure 4.7 shows that the network seems to be able to handle pole masses and cart masses reaching up to around 6 kg, as well as pole lengths up to 4 m before the scores start to decrease rapidly.

5 Discussion

Just by watching the results received from the simulations and the figures (see section 4), it is clear that an artificial neural network is a functional and suitable controller to stabilize the unstable, non-linear inverted CartPole system. Both in the phase of swinging up the pole as well as balancing it in its upright position for a longer period of time. The medium sized network performed the best and showed to be a robust network, being able to handle new CartPole parameters going beyond the range it had been trained for.

As seen from figure 4.1, received from just using randomly chosen actions, compared to figure 4.3, received from the simulation using the medium sized network. The mean scores went from 149.8 to 3 753.0 points, a 25 times higher mean score, while the mean time length per run went from 84 to 475 steps. Corresponding to an increase from 1.7s to 9.5s in real time. Why it did not go all the way to 500 steps or 10.0s was a result of there being no reward given while the pole was swinging up beneath the ground.

5.1 Artificial neural networks

Reading from the plots in figure 4.2,4.3 and 4.4. We can see that the medium size network proved to be the most optimal since it had the overall best scores and longest mean simulation time. It is also possible to conclude based on the shape of the subfigures a)-c), that the medium sized network performed the best. Showing almost a complete red ”square roof”, lying close to the maximum score one run could achieve, being less than 4 500 points, for almost all different combinations of CartPole variables. Having some low scores for smaller CartPole variables shows that the network is not ”bulletproof”, or optimized, for all different simulations and situations. However, looking at the subfigures a)-c) for the small- and large networks one can clearly see that they do not handle all different CartPole variable combinations as well as the medium sized network did. Therefore, the medium sized network showed to have a more optimal combination of number of neurons and layers than the other networks. It also showed that a larger network do not always correspond with a better network. This is an especially good result for anyone who wants to use ANN in real life applications, since a larger sized network takes longer time and more computational power to train and use. Which was the main reason an even larger network then the large network was not tested.

However, the medium sized network is not necessarily or most likely, the most optimal network for this CartPole problem. Changing the parameters like learning rate, keep probability, epochs, activation- and optimizer functions can result in achieving a better network. Having said that, this was not the main objective of this report and is one thing that should be investigated further.

Section 4.2.4 shows that the force distribution between the small and medium network where quite similar in the overall distribution. Although, the small network almost never used any force of -100 N while the medium sized network did, see figure 4.5a. The large network however had an completely different distribution of the forces used, with almost none of them being 0 N or -100 N, see figure 4.5c. A common feature between all of the

three networks where that the action 100 was almost never used. Although not being an exact answer, one reason for this pattern could be interpreted viewing the animation of the simulations. There one can see that the CartPole almost always swings up the same way. Swinging up the pole anti clockwise before stabilizing it. The medium sized network also used the most actions of zero force, which could be a result of the pole being better balanced, since it does not need as many corrections.

5.2 Robustness

When comparing the graphs in figures 4.6 and 4.7 to the ones in figure 4.3, one can see some big differences. It shows that when the CartPole parameters (cart mass, pole mass and pole length) is in the range that the network had been trained for, it still receives high scores around 4 000 points. But when the CartPole variables are increased above the values of what the network had been trained for. The mean score is shifted to the left (comparing the histograms in figures 4.3, 4.6 and 4.7) meaning a lower overall score. Going from a mean score of 3 753.0 points in figure 4.3, to 444 points in figure 4.7 and 218.7 points in figure 4.6.

Looking at the subplots a)-c) for figures 4.3, 4.6 and 4.7. It becomes clear that there is in fact a threshold for the robustness of the CartPole parameters, where the scores converge to zero when the CartPole parameters tend towards ten. Looking at figure 4.7, those limits seem to lie around 0.1-6 kg for both the pole and cart masses. While the pole length lies around 0.1-4 m and appears to have the largest impact on the resulting score. This corresponds to the network being able to manage CartPole parameters three to four times larger then what it has been trained for.

Some observations that can be drawn is that if the pole was short, higher scores were generally received. But once the pole got too long and heavy it became more difficult to control, which may be because of the poles centre of gravity moving too far away in relation to the cart's mass.

Another possible reason why the points decline with increasing pole and cart masses, is that since the total mass of the system increases, the forces necessary to swing up and move the CartPole in the manner it has trained for, becomes too small. Having a force vector that is scaled up according to the CartPole variables might be a suitable solution to compensate this.

Finally, one possible reason is that it might just come a point where the network no longer can make ”good” predictions since it just do not have enough ”experience” of that particular situation.

Although, being able to balance a CartPole pendulum ranging from 0.1 m all the way up to 7 m is quite an impressive feature in itself.

5.3 Observations

Since the cart and pole had a total mass of around 1-4 kg. Applying a force of 300 N might seem a bit much, corresponding to an acceleration of sometimes 100 times the force of gravity. This is an extreme amount for any physical system. It seems reasonable that smaller forces should be needed once the pendulum is swung up and balanced. When trying a force vector with smaller values, the training data got drastically lower points. Resulting in a needed increase in total number of runs in order to get a sufficiently large set of good training data. It was however possible to use a force vector with smaller values for the simulation then was used to create the training data and still be able to swing up and balance the CartPole. However, in order to be consistent we choose to use the same force vector for both the training and simulation.

Attempts were made in retraining the network on the saved training data created using the predicted actions of the networks, however this did not result in a better network. Getting lower scores and performance than before. Also, creating a new network on this training data did result in a worse network than before. This is most likely a result of the training data being saved was too small to train a new network on. E.g. looking at the medium sized network, the training data size for the simulation was 3 753 runs, instead of 33 081 runs as for the randomly created training data. Doing longer simulations would be a good starting point in order to investigate this further. However, we choose to limit the report and not investigate this further, training the networks on only one set of data that was created randomly.

5.4 Improvements

For future testing there are a couple of things that could be changed. The first thing is to apply a scaling force vector like mentioned above. The reason to this is making the relation between total mass of the cart and force pushing the CartPole smaller. This will prevent the cart having a small force and a large mass or vice versa. This could have been the reason of the low scores in figure 4.3c for small pole and cart mass.

Changing the parameters like learning rate, keep probability, epochs, activation-and optimizer functions can result in achieving a better network. As well as testing other ANNs lying close to the size of the medium sized ANN.

Doing longer simulations would in turn provide larger sets of even better training data that could be used to further train a better ANN. However this takes exceptionally long time since almost all of the runs get an extremely high score, meaning that they also go on for almost all time steps per run, which makes the training data being saved incredibly large.

To further test the robustness, a random nudge can be implemented on some random time when generating the training data and later in the simulation. This would make the network learn to handle small errors or interferences on the system, which is something very useful for real life systems.

Another improvement for the physical system is to account for frictional forces that might occur.

5.5 Summary

To summarize, an artificial neural network is a functional and suitable controller to stabilize the unstable, non-linear inverted CartPole system. Both in the phase of swinging up the pole as well as balancing it in its upright position for a longer period of time. The medium sized network performed the best and showed to be a robust network able to be stable for CartPole parameters that went 3-4 times beyond the range of what the network had been trained for. It also showed that a larger sized ANN does not equal a better controller. There are a multitude of parameters one can vary and test in the CartPole simulation, and testing all of them was beyond the scope of this report. But doing so could result in an even better ANN being achieved.

References

[1] Wikipedia. *Nonholonomic system*. URL: https://en.wikipedia.org/wiki/Nonholonomic_system (visited on 04/10/2019).

[2] Wikipedia. *Stabilization and Tracking Control of Inverted Pendulum Using Fractional Order PID Controllers*. URL: http://dx.doi.org/10.1155/2014/752918 (visited on 04/16/2019).

[3] Callinan, Tim. *Artificial Neural Network identification and control of the inverted pendulum*. URL: https://pdfs.semanticscholar.org/3f3b/5a043e7154b5dc0277b5cf38c8d35c2e9649.pdf (visited on 04/29/2019).

[4] Zulkifli, Hafidz. *Understanding Learning Rates and How It Improves Performance in Deep Learning*. URL: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 (visited on 04/29/2019).

[5] Budhiraja, Amar. *Dropout in (Deep) Machine learning*. URL: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 (visited on 04/29/2019).

[6] SHARMA, SAGAR. *Epoch vs Batch Size vs Iterations*. URL: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9 (visited on 04/29/2019).

[7] Walia, Anish Singh. *Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent*. URL: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f (visited on 04/29/2019).

[8] SHARMA, SAGAR. *Activation Functions in Neural Networks*. URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (visited on 04/29/2019).

[9] CTMS. *Inverted Pendulum: System Modeling*. URL: http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling (visited on 04/29/2019).

[10] TFLearn. *TFLearn: Deep learning library featuring a higher-level API for TensorFlow*. URL: http://tflearn.org/ (visited on 04/16/2019).

[11] OpenAI. *CartPole-v0*. URL: https://gym.openai.com/envs/CartPole-v0/ (visited on 04/16/2019).

Appendices

Our GitHub to the CartPole

https://github.com/Monras/CartPole