Nonparametric deep neural networks for movement planning

Nichtparametrische tiefe neuronale Netze zur Bewegungsplanung
Bachelor-Thesis von Harun Polat aus 58791 Werdohl
März 2017

Nonparametric deep neural networks for movement planning
Nichtparametrische tiefe neuronale Netze zur Bewegungsplanung

Vorgelegte Bachelor-Thesis von Harun Polat aus 58791 Werdohl

1. Gutachten: Dr. Elmar Rueckert
2. Gutachten: Prof. Dr. Jan Peters
3. Gutachten: Msc. Daniel Tanneberg

Tag der Einreichung:

For my parents and my brother

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, den 16. März 2017

(Harun Polat)

Thesis Statement

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, March 16, 2017

(Harun Polat)

Abstract

Robots are an essential part of our industrial production process and in the future robots might accompany us every day.
In all this we need the computing of movement trajectories to solve a task, including considering constraints. This task solving is condensed in the term planning and thus planning is a fundamental knowledge in nearly all robotic tasks. For this application Spiking Neural Networks (SNN) are one of the new state of the art learning algorithms. SNNs incorporate the concept of time, they also include information to be processed asynchrony, event-based and without delay.
SNN's are uniquely equipped to solve planning problems. This demands an increase of synapses, which is (going to be) compensated by neuromorphic hardware. The number of synapses is significantly higher, this also reflects on the increased consumption. In this work, we want to show that models for synapse pruning decrease the synapses needed and in addition result in a significant decrease of memory consumption.

Contents

Figures and Tables

List of Figures

List of Tables

1 Introduction

1.0.1 Motivation and Goals

Robots are an essential part of our industrial production process. There are many examples for robotics in industrial production like the automotive industry where they are used to assemble cars, in computer industries they are used to solder hardware and the transport industry uses them for heavy lifting. Their non-existence would be unthinkable for our times. The visionary approach is to imagine a future, where robots accompany us throughout our daily life in every aspect of it.
Every named example is suitable to show the importance of planning as a fundamental knowledge in nearly all robotic tasks. In the automotive industry the parts assembly only works collision-free by assuring that planning generates the movement trajectories perfectly. This means that the computing of movement trajectories to solve a task, including considering constraints, is condensed in the term planning.
For this application Spiking Neural Networks (SNN) are one of the new state of the art learning algorithms. They are neural networks, which are inspired by the biological structure of the brain and it's amazing abilities in learning, decision-making and action, which sum up to all abilities needed for planning. In contrast to the simpler ANNs, the SNNs incorporate the concept of time, which makes them more energy efficient and faster than ANNs [2]. Besides a concept of time, SNN's also include information to be processed asynchrony, event-based and without delay.
SNN's are uniquely equipped to solve planning problems. They are capable of modeling complex distributions, these are used to encode in case of multi-modal distributions (distributions of more than one task solution).In conclusion, SNN's are capable of solving all tasks just like ANNs and they are simultaneously more powerful and efficient [26]. All this results in a high demand for SNN's in the area of complex task solving.
The downsides of a longer computational time and complexity in vast networks is going to be compensated by neuromorphic hardware. A current example is the IBM TrueNorth, which is able to map spiking neural circuits with up to to 10^6 neurons and $256*10^6$ synapses.
The basis for this work is [1], where SNN's have been used to solve robot planning problems, like target reaching and obstacle avoidance tasks. In [1] the basic setup consists of 225 neurons with 225*225 synapses. Compared to neuromorphic hardware, were the network can consist of 10^6 neurons, the number of synapses are significantly higher, this also reflects on the increased consumption.
In this work, we want to show that models for synapse pruning decrease the synapses needed and in addition results in a significant decrease of memory consumption.

1.0.2 Outlook

In the second Section an overview of biological aspects will be given. These aspects are important to understand the role and motivation of our approach. Firstly operational basics of neurons and synapses will be given, followed by the hebbian theory of learning and the neural cable theory. The hebbian theory explains the change synapses are going through while learning, whereas the neural cable theory explains the expansion of electric isolated synapse input, which works as deep as the center of the neuron. Further the algorithm for model learning, contrastive divergence is given. It builds a basis of the algorithm used in [1], resulting to be a basis in this work as well.
In Section 3 the models of synapse formation and synapse pruning are discussed. Beginning with the relevant parameters of synapse formation and pruning. Going on to the 3 different synapse formations and 2 different synapse pruning models, which are introduced and described. To finalize this our algorithm for learning optimized transition models (LOTM), which includes one of the pruning models, is presented and described.
In Section 4 the experiments for the evaluation of one synapse pruning model is presented. The model is used to reduce the memory consumption through using LOTM. Following the experiments objective, the relevant parameters, like the experimental procedure, evaluation criteria and important model parameters, are explained. After that the results of the pruning process are discussed. This discussion is considering and comparing parameters like memory utilization, computational time and iteration length.
In Section 5 an evaluation of our pruned and by that optimized transition model is given. It has been evaluated by target reaching and obstacle avoidance tasks in simulation.
In Section 6 the summary and future work expectations can be found.

2 Background

To understand the motivation of the selected parameters for the proposed heuristics, this section gives a brief structural and operational overview of the brain. Furthermore, to comprehend the experiment results, a description of the used model learning algorithm is provided.

2.1 Physiological foundations

2.1.1 Neurons and synapses - Operational basics

The human brain consists of $100*10^9$ neurons, which are the central computing units of the human brain. These neurons are organized to fit different needs, for example to tell us how to speak, which is realized by building clusters. To communicate these information neurons receive signals from other neurons on their dendrites. The connection between neurons is called synapse, the sending neuron is called presynaptic - and the receiving neuron is called postsynaptic neuron. Similar to computers the neurons have a computing unit, which is called soma.
The communication works by reaching a certain threshold of received spikes for the neuron to react and generate an action potential, which is also called spike. In line the generated spikes cause the release of presynaptic neuron chemicals, these chemicals are called neurotransmitter. They are used to transfer information. The information is led through the axon over to the synaptic cleft into the postsynaptic neuron.
Subsequently as a result of a learning process synapses are formed. The formation of synapses is called synaptogenesis. The degeneration is called synapse pruning. The usage of synapses is essential to the functionality of the brain. They are a key component of what makes us human.

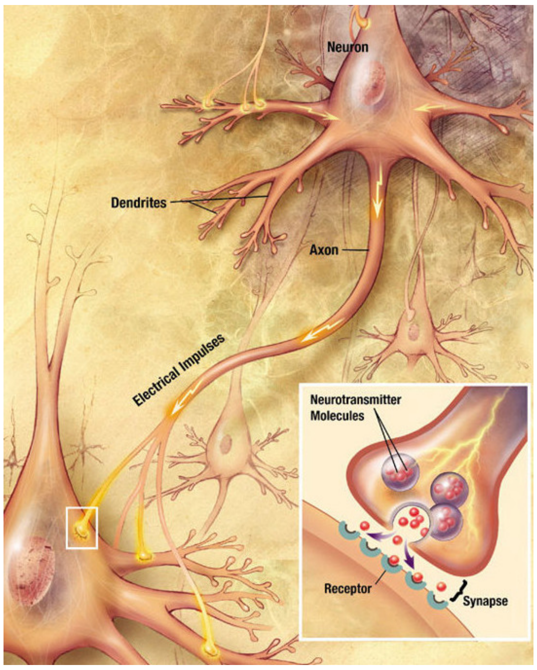

Figure 2.1: Neuron and synapse structure
source: https://upload.wikimedia.org/wikipedia/commons/3/30/Chemical_synapse_schema_cropped.jpg

2.1.2 Neural cable theory

The cable theory is addressing the potentials spread in a dendritic tree. The neurons are receiving synaptic inputs all over the surface. The cable theory predicts that the synaptic inputs that are electrically close are reduced in the ionic driving force, apposed by electrically isolated synaptic inputs, which sum linearly. Apparently this is caused "by a transient decrease of dendritic input resistance" ([3] - p.1) and means that the placement of an input on a dendritic tree affects its functional importance to the neuron. It also gives room for the speculation, that dendritic branches might be used to isolate inputs from one another [3][4].

2.1.3 Hebbian learning - Synpase plasticity

The hebbian theory, coined in 1949 by psychologist Donald Hebb, addresses the basis of learning, which is an improved communication between two cells. This improved communication is a specific physical enhancement, which is called synapse plasticity. It describes the ability of the synapse to improve or decrease depending on usage over time. It works quite simple, with increased usage the connection grows stronger, with decrease usage the connection grows weaker.
Not only a physical change is happening to the synapse, but also a metabolic improvement can happen. This metabolic improvement results in a higher efficiency.

2.2 Machine learning algorithm for model learning

Contrastive divergence for model learning
Contrastive Divergence (CD) is a learning algorithm for approximating the Maximum-Likelihood (ML) gradient. Based on Markov Chains (drawing samples from a proposed distribution), the log-likelihood gradient is obtained.

Contrastive divergence is approximation of the gradient (a good direction of change for the parameters) of the log-likelihood (the basic criterion that most probabilistic learning algorithms try to optimize) based on a short Markov chain (a way to sample from probabilistic models) started at the last example seen. The following description is based on [1] and [5].
We want to model probability of data point x with a function of the form $f(x;\Theta)$, where Θ is a vector of model parameters. To get a valid probability distribution, the probability of a data point $p(x;\Theta)$ over all x has to integrate to 1. Therefore we define the probability of a data point $p(x;\Theta)$ as

$$p(x;\Theta) = \frac{1}{Z(\Theta)} f(x;\Theta) \tag{2.1}$$

with a partition function (special case of normalizing constant) $Z(\Theta)$, which is defined as

$$Z(\Theta) = \int f(x;\Theta)dx \tag{2.2}$$

Given the set of independent and identically distributed (i.i.d.) training data X = $x_{1,\dots,K}$ with K examples, we want to find the model parameters Θ^*, which maximise the probability of X. To find Θ^* we have to maximize the probability of the training data X, given as

$$p(X;\Theta) = \prod_{k=1}^{K} \frac{1}{Z(\Theta)} f(x_k;\Theta) \tag{2.3}$$

or equivalently minimizing the negative log of $p(X;\Theta)$, which is called energy function and denoted as $E(X;\Theta)$:

$$E(X;\Theta) = log(Z(\Theta)) - \frac{1}{K}\sum_{k=1}^{K} log(f(x_k;\Theta)) \tag{2.4}$$

In order to find the optimal parameter Θ^*, the energy function needs to be differentiated w.r.t. Θ and the gradient is given by

$$\frac{\partial E(X;\Theta)}{\partial\Theta} = \frac{\partial log(Z(\Theta))}{\partial\Theta} - \frac{\partial}{\partial\Theta}\left(\frac{1}{K}\sum_{k=1}^{K} log(f(x_k;\Theta))\right) \tag{2.5}$$

For that we determine the partial derivatives of Equation 2.5, which is, for the first term on the right side of the Equation, given by

$$\begin{aligned} &\frac{\partial log(Z(\Theta))}{\partial \Theta} \\ &= \frac{1}{Z(\Theta)} \frac{\partial Z(\Theta)}{\partial \Theta} \\ &= \frac{1}{Z(\Theta)} \frac{\partial}{\partial \Theta} \int f(x;\Theta) dx \\ &= \frac{1}{Z(\Theta)} \int \frac{\partial f(x;\Theta)}{\partial \Theta} dx \\ &= \frac{1}{Z(\Theta)} \int f(x;\Theta) \frac{\partial log(f(x;\Theta))}{\partial \Theta} dx \\ &= \int p(x;\Theta) \frac{\partial log(f(x;\Theta))}{\partial \Theta} dx \\ &= \left\langle \frac{\partial log(f(x;\Theta))}{\partial \Theta} \right\rangle_{p(x;\Theta)} \end{aligned} \tag{2.6}$$

and for the second term given by

$$\begin{aligned} &\frac{\partial}{\partial \Theta} \left(\frac{1}{K} \sum_{k=1}^{K} log(f(x_k;\Theta)) \right) \\ &= \frac{1}{K} \sum_{k=1}^{K} \frac{\partial log(f(x_k;\Theta))}{\partial \Theta} \\ &= \left\langle \frac{\partial log(f(x_k;\Theta))}{\partial \Theta} \right\rangle_{x}, \end{aligned} \tag{2.7}$$

where $\langle \cdot \rangle_X$ denotes the expectation of given data X. With the partial derivatives, the Equation 2.5 can be equivalently written as

$$\frac{\partial E(X;\Theta)}{\partial \Theta} = \left\langle \frac{\partial log(f(x;\Theta))}{\partial \Theta} \right\rangle_{p(x;\Theta)} - \left\langle \frac{\partial log(f(x;\Theta))}{\partial \Theta} \right\rangle_{x} \tag{2.8}$$

Even though the partition function (Equation 2.2) is in general algebraically not intractable and hence samples from a random cumulative distribution curve cannot be drawn, we exploite the last equivalence in Equation 2.6 to approximate the derivative of $log(Z(\Theta))$, which is needed for minimizing the energy function. To do so, the Markov Chain Monte Carlo (MCMC) method is used to draw samples from the model distribution $p(x;\Theta)$. Here we start MCMC sampling with Markov Chains initialized with the training data X as a starting point. After n cycles of MCMC the training data is transformed into samples from the proposed distribution, which is denoted as X^n. With the MCMC method we have everything to continue and refomulate Equation 2.8:

$$\begin{aligned} &\frac{\partial E(X;\Theta)}{\partial \Theta} \\ &= \left\langle \frac{\partial log(f(x;\Theta))}{\partial \Theta} \right\rangle_{p(x;\Theta)} - \left\langle \frac{\partial log(f(x;\Theta))}{\partial \Theta} \right\rangle_{x} \\ &= \left\langle \frac{\partial log(f(x;\Theta))}{\partial \Theta} \right\rangle_{x^\infty} - \left\langle \frac{\partial log(f(x;\Theta))}{\partial \Theta} \right\rangle_{x^0} \end{aligned} \tag{2.9}$$

This solution is still not optimal, since to many MCMC cycles have to be performed, which makes it impractical for computing accurate approximations. Instead, Hinton suggested to use just 1 cycle, which is sufficient to get an idea of the direction the proposed distribution should move to in order to model the training data more clearly. He showed in empirical form that 1 cycle is enough to find a to the maximum likelihood converging solution. With this new insight we are able to formulate the iterative learning algorithm contrastive divergence as the following update rule:

$$\Theta_{t+1} = \Theta_t + \alpha \left(\left\langle \frac{\partial log(f(x;\Theta))}{\partial \Theta} \right\rangle_{x^0} - \left\langle \frac{\partial log(f(x;\Theta))}{\partial \Theta} \right\rangle_{x^1} \right), \tag{2.10}$$

with a learning rate α. Contrastive divergence is a learning algorithm which can be used for complex model learning based on approximations. These approximations are obtained by sampling.

2.2.1 Related Work

Movement planning is essential for robotics. Here we want to name two works based on neural networks.
The work of Rueckert et at. [6] proposes a recurrent spiking neural network for finite and infinite horizon tasks, where planning is implemented as probabilistic inference.
In [7] a deep spiking network architecture for task and joint space planning is presented. To generate smooth trajectories, bidirectional feedback is used.

3 Methods

This section describes the identified influencing parameters for the formation and pruning of synapses. Using these identified parameters, models for synapse formation and pruning are presented. Finally, based on one of the synapse pruning models, an algorithm for learning optimized transition models is formulated. In further sections, this algorithm result, an optimized transition model, is used to evaluate robot planing problems.

3.1 Parameters

To optimize the transition model w.r.t. memory consumption a quality measure for quantitating the importance and impact of synapses for the task is needed. For the construction of such models two influencing factors for the formation and degeneration of synapses have been identified.

Distance: The formation of synapses take place at different locations to the cell body with varying distances. As stated by the cable theroy, the dendritic location of each synapse, i.e., its distance to the cell body, determines the resulting excitatory postsynaptic spike (EPSP) of a neuron [8]. Since a neurons received current travels form the dendrite over the cell body and the axon to the cell body, a reduction of charge happens. Therefore it is expected that synapses with a large distance from the cell body have less influence on the initiation of spikes in the axon [9][10][11][12].

The presynaptic current is crucial for the models and learning in general. The total amount of received spikes need to overcome a certain threshhold to give the neuron an incentive to fire.
Imagine 3 neurons $-A$, B and C. These neurons are connected in line. Further imagine the distance between A and B to be of significantly higher.
Naturally this makes the excitation of C harder. The effect of the significantly higher distance between A and B is a reduced stimulation of neuron B. With a reduced stimulus of B it's spike output is reduced too. This results in even less stimulus for C.
Over time the synapse between neuron B and C experiences a synaptic fatique and therefore can be seen as no longer required in the technical setup. Those insights also apply for the formation of synapses.

Correlated neural activity: As stated in hebbian learning there is a correlation between neurons that fire in the same time window repeatedly or persistently.
For the models, that can be used as a strong parameter in the process of deciding whether a synapse should be formed or degenerated.

3.2 Models for synapse formation and synapse degeneration

The identified factors for synaptogenesis in Section 3.1 provide parameters for constructing models for synapse formation and degeneration. The mathematical foundation of all models is the exponential function with a negative exponent. The models are constructed by modifying the exponential functions shape by stretching and compressing through incoperating those parameters accordingly.

3.2.1 Synapse formation

All synapse formation models are denoted as $P(L=1)$ and incoperate the parameters d_{ij}, $\Delta_{t_{ij}}$, γ_{ij}, η. In addition, the parameters c_1 and c_2 are used as normalizing constants. The denotation describes the probability a synapse should be formed (L = 1).

Table 3.1: Parameter list

Parameter	Init/Default Value	Description
d_{ij}	-	Physical distance between neurons i and j
$\Delta_{t_{ij}}$	-	Temporal difference between the last time neuron i and neuron j have fired
γ_{ij}	-	Activity correlation of neurons i and j
w_{ij}	-	Strength of synapse between neurons i and j
$Var(w_{ij})$	-	Variance of synaptic weight
$\nabla(x)$	-	Direction of change for variable x
η	-	Learning rate

Model one

$$P(L=1) \propto c_1^{-1} e^{-(\frac{d_{ij}}{c_2}\Delta_{t_{ij}} - \eta\gamma_{ij})} \tag{3.1}$$

In this model, all parameters are weighted equally and incoperated as a negative exponent of the exp-function. The distance d_{ij} and the temporal difference $\Delta_{t_{ij}}$ between two neurons i and j mutually weight each other. This means that an increase in d_{ij} decreases the impact of $\Delta_{t_{ij}}$ and vice versa. Finally, the activity correlation γ_{ij} influence the probability of new synapse formation based on the spiking behavior of the neurons. This effect of γ_{ij} on the total probability can be understood by reviewing the way it is used in the model: A high γ_{ij} value, which means an increased spiking activity, the negative exponent gets smaller, which in total results in a higher synapse formation probability.

To get an intuition of this model consider two close neurons that have not fired for a long time but had been active in the same time window in the past. The close distance dampers the exponent, thus increases the probability for synapse formation. Since neurons activity correlate it gets rewarded by decreasing the exponent by η resulting in a probability increase for the creation of a new synapse.

Model two

$$P(L=1) \propto \frac{d_{ij}^{-1}}{c_1} e^{-(\frac{\Delta_{t_{ij}}}{c_2} - \eta\gamma_{ij})} \tag{3.2}$$

As a variation of the first model, this model emphasizes the distance between neurons and keeps the remaining model unchanged. Compared with the other model parameters used in this model, the impact of d_{ij} on the total synapse generation probability is higher. This is due to the fact, that all parameters but d_{ij} are encapsulated as the exp functions exponent. Due to this encapsulation, several parameters are treatet as a single factor in the model formula. As the model is a product of 2 factors, the distance between neurons gets the main factor for decision making. A big d_{ij} value is immediatly reflected in a huge propability drop for new synapse formation.

Model three

$$P(L=1) \propto c_1^{-1}\gamma_{ij}\,\eta\, e^{-(\frac{d_{ij}}{c_2}\Delta_{t_{ij}})} (3.3)$$

Like the second model, this probability only differs in one single parameter compared to the first model. To focus certain parameters, again it is given, that all parameters are encapsulated in a single factor except the neural activity correlation (times η). Therefore the probability for synapse formation is primary (negative) proportional to γ_{ij}.

3.2.2 Synapse degeneration

All synapse degeneration models are denoted as $P(L=0)$ and incoperate the parameters d_{ij}, w_{ij}, γ_{ij}, η. In addition, the parameters c_1 is used as normalizing constants. The denotation describes the probability a synapse should be degenerated (L = 0).

Both dieSyn heuristics are comparable in structure as the second model extends the first one. With the exception of d_{ij} all parameters are substituted into one single factor to focus the impact of the distance parameter, as it was done accordingly to the synapse generation models.

Model one

$$P(L=0) \propto \frac{d_{ij}}{c_1}\, e^{-(w_{ij} + \eta\, \gamma_{ij})} \tag{3.4}$$

This model for synapse degeneration is based on the exponential function. To weight the parameters influce on the probability for synapse degeneration differently, the model is constructed as a produkt with only two factors: The first factor is the distance between two neurons d_{ij}. The second factor is the exp-function, which encapsulates the synaptic weight and the neural activity correlation multiplied by a learning rate in its negative exponent. The idea to focus d_{ij} is, that neurons with a large distance in between are probably not going to (intensively) work together, which can be used as a strong indicator for the degeneration of that synapse.
To get an idea of the way the other parameters effect the model, let us consider 2 neurons which are close to each other. In this case the probabily for degeneration depends mainly on the synaptic weight w_{ij} and the neural activity correlation γ_{ij}. The synaptic weight depends on the involved neurons activitiy levels. A weak synaptic connection, i.e. a low value in w_{ij} is a strong indicator, that the synapse is less important and that it can be degenerated. The same holds for γ_{ij}. If two neurons have not been active in a certain time window and therefore their activies do not correlate, the synapse is probably not relevant anymore.

Model two

$$P(L=0) \propto \frac{d_{ij}}{c_1}\, e^{-(w_{ij} + \eta\, \gamma_{ij}\, (1+\nabla(w_{ij}) Var(w_{ij})))} \tag{3.5}$$

The foundation of this model is equal to the first synapse degeneration models structure. This model only differs by including the additional parameter $Var(w_{ij})$ to determine whether and how strong the synaptic weight between two neurons has changed. Since the variance is always positive, the synaptic weights direction of change $\nabla(w_{ij})$ is determined. With $\nabla(w_{ij})$ it is possible to indicate whether a degenerating or generating change of the synapse is happening. In case of no variance, the activity parameter would be multiplied by zero. To avoid that, an additional 1 has been added. To make the effects of the additional parameter clear, we consider two neurons, which form a synapse. In case this connection is infrequently used, the connection is growing weaker and an additional negative factor in the exp-functions exponent, which means an increase in the probability of degenerating this synapse, is present.

Many possible variations of those models, each with a different parameter focus, are possible. The proposed models for synapse degeneration incoperate d_{ij} such that maps the idea, that only close neurons are relevant for solving a task. Motivated by that, distal synapses are degenerated by the models.
Beside the implicit parameter weighting, parameters can be incoperated with a different meaning, e.g. d_{ij}: Neurons, that have ma naged to form a synapse over a large distance are probably highly specialized and relevant for the task. Contrary to the proposed synapse degeneration models, such synapses should not be degenerated.

3.3 Algorithm for learning optimized transition models

The algorithm for learning optimized transition models (LOTM) (Algorithm 1) is inspired by self-organizing recurrent neural networks (particularly SORN [13]). Those networks incorporate two major kinds of activity-dependent plasticity.
(I) Hebbian plasticity, which leads excitatory synapses that effectively take part in exciting postsynaptic cells to grow stronger.
(II) Homeostatic plasticity, which dynamically changes synaptic strength to oppose hebbian learning induced changes in activity levels to maintain them within a dynamic range [14]. Since hebbian plasticity tends to destabilize neural circuits, homeostatic plasticity works as a compensatory and provides stability.
The transition model is learned through a spike dependent version of contrastive divergence. This type of model learning can be seen as a hebbian like learning rule (line 6). Here the introduced models for synapse degeneration have been applied (lines 8-13), rather than modelling and incoperating different types of homeostatic plasticity as it is shown in SORN.
To achieve plausible results, the spike trains analysis and processing needs to be peformed on sufficiently long sub intervals. The parameter for controlling the sub interval length is called timeToOptimize (line 7).
The analysis on too small sub intervals (relative to the length of the processed spike train (line 4) would not adequatly capture correlated neural activity, since a neuron needs to repeatedly or persistently excite an other neuron to detect a correlation in the neural activity (hebbian theory). Depending on the processed spike train (in terms of total length), the sub interval length, has to be set appropriately. In addition, the sub interval length need to be considered while implementing the activity correlation γ in the synapse degeneration model (line 9).
In order to determine γ, a decision limit has to be set. The decision limit is used to finally determine whether there is a correlation between neurons activity or not. After potential correlations have been detected. Which is calculated based on the number of neurons spiking within a symmetric interval around the observerd time step - the x-quiantile is used to decide whether two neurons total spiking amount meets the threshold requirements for beeing classified as correlation.

After calculating the synapse degeneration probability in line 7, the x-quantile q_x is used to determine whether a synapse should maintained or degenerated, which is why q_x acts as a pivot point in LOTM. In function of this processed spike train sub interval, it should be pre-processed in terms of the total neural activity level to set q_x properly.

Algorithm 1 Algorithm for learning an optimized transition model

```
1  function optLearn ($X \in \mathbb{R}^{N \times D}_{NaN}, W \in \mathbb{R}^{D \times D}, \gamma \in \mathbb{R}^{D \times D}, d \in \mathbb{R}^{D \times D}, \eta \in \mathbb{R}^1$)
2  Input  : trainData $X[\#traj, 1 : duration]$,
           initial synapctical weight matrix W, model parameter $X, W, \gamma, d, \eta$ detailed in subsection 3.2
3  Output: learned and optimized transition model encoded in $\tilde{W}$
4  for each trajectory in $X[*, 1 : end]$ do
5      for t in trajectory[1:duration] do
6          $\tilde{W}$ = modelLearn(W) // Learn transition model
7          if $t$ mod $timeToOptimize$ == 0 then
8              for each active synapse $\tilde{w}_{ij}$ do
9                  $p_{\tilde{w}_{ij}} = \frac{d_{ij}}{c_1} e^{-(\tilde{w}_{ij} + \eta \gamma_{ij})}$ // Calc synapses die probability
10             for each probability $p_{\tilde{w}_{ij}}$ in $P_{\tilde{W}}$ do
11                 if $p_{\tilde{w}_{ij}} >= q_x$ then
12                     $\tilde{W}(i, j) = NaN$ // Degenerate synapse between neurons i and j
13                     $W = \tilde{W}$
14 return $\tilde{W}$;
```

4 Experiments

We used our model to learn a state transition model of the form s' = f(s), where s denotes the current step and s' is the next step. The model is encoded in a recurrent neural network presented in [1]. In the following Subsections we motivate the experiment and detail the experimental procedure for collecting the real robot data, the evaluation criteria to verify our model, the important parameters in our model and the results.

4.1 Model complexity and memory requirements

Due to the small number of neurons that have been used (225 in total) in [1], the resulted transition models memory utilization is between 0,08 MB and 8MB (see table 4.1) and managable. However, modern brain inspired chips like IBMs TrueNorth are able to map neural circuits with up to 10^6 neurons and 256x10^6 synapses, respectively. A transition model with 10^6 neurons would result in a memory utilization of 8 TB (table 4.1). Fruthermore, the limitation on the number of storable synapses would make it even harder to reproduce neural circuits as presented in [1] on similiar chips.

Table 4.1: Synapse memory consumption

No. of neurons	Memory consumption
10^0	0 byte
10^1	0.8 kB
10^2	0.08 MB
10^3	8 MB
10^4	800 MB
10^5	80 GB
10^6	8 TB
10^7	800 TB
10^8	80 PB

4.2 Experimental procedure

The experiments have been performed on a transition model that has been generated using real robot data. Through this real data, a realistic transition model of the robot and its task space could be obtained. For that purpose, about 15 minutes of arbitrary robot arm movements at a sampling rate of 1 ms have been recorded, which resulted in a state transition model of the form s' = f(x) with x$\in [0,1]^{15x15}$ and s'$\in [0,1]^{15x15}$ Finally, contrastive divergence has been used to learn from the obtained 900.000 state transitions a realistic transition model.
To use the robot arm movements as training data, the continous movement trajectories were transformed into binary spike pattern. Further information on the experiment setup and using the robot arm movements as training data can be found in [1], and is detailed in Sections 4.1 and 4.2.

4.3 Evaluation criteria

To compare the results, different types of error metrics have been applied. Mean absolute error (MAE or Error), Mean squared error (MSE) and Root mean squared error (RMSE), where N denotes the total number of entries in

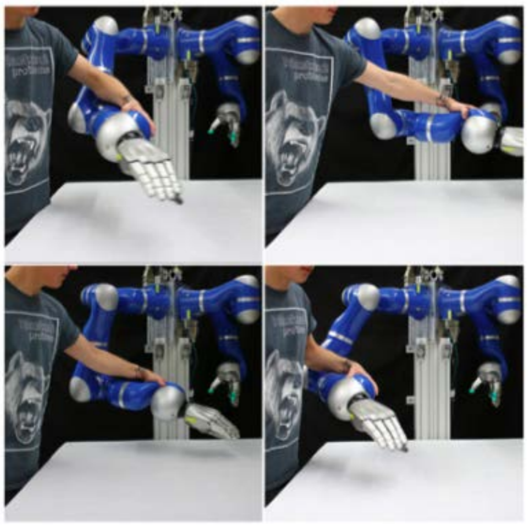

Figure 4.1: Kuka lightweight robot arm - Kinesthetic teaching

the not pruned transition model W, w_i synaptic weight in W and $\bar{w}_i$ synaptic weight in pruned transition model.

$$Error = \frac{1}{N}\sum_{i=1}^{N}|w_i - \bar{w}_i| \tag{4.1}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(w_i - \bar{w}_i)^2 \tag{4.2}$$

$$RMSE = \sqrt{MSE} \tag{4.3}$$

By applying various error metrics, different viewpoints on the model accuracy can be obtained. Taking the mean of the absolute errors, MAE tells how big the expected error on the proned network on avarage is. While it weights all errors equally and is easier to interpret than the other metrics, the errors relative size is not always obvious. By comparision, (R)MSE gives extra weight to large errors. Therefore MAE and (R)MSE can be compared to determine whether the proned network contains large errors.

4.4 Important model parameters under test

The LOTM-algorithm optLearn, as described in Subsection 3.0.3, was executed with a learning rate of 0.3. The activity correlaton has been calculated with the .75 quantile as the decision limit. Fruther, optLearn has been executed 5 times and each time with an increased iteration length, so that synaptic pruning did take place every 2000ms, 20000ms, 40000ms, 60000ms and 120000ms, respectively. The iteration length is definded as the points in time when pruning is applied.

4.5 Pruning results

The following Subsections will show the effects of applying our synaptic pruning model to the network. We discuss the impact of synaptic pruning on memory consumption and computation time and compare them with the applied error metrics's total error relative to increasing intervals of pruning (iteration lengths).

4.5.1 Total achieved error after pruning

Pruning every 20 seconds let the Error grew 16 times higher than pruning every 2 seconds (from 0,003 to 0,048). On average the biggest error could be archived by pruning once a demonstration (0,173) and reducing the computation time more than half (from 54,08 to 19,78) at the same time.
Since most of the training data contain 120000 milliseconds of robot arm movements, pruning every 120000ms has been chosen as the maximum iteration length (see Table 4.2).

Table 4.2: Optimized learning total error - Summary

Iteration length	RMSE	MSE	Error	Comp. duration
2000 ms	0.055	0.003	0.003	54,08 min
20000 ms	0.216	0.047	0.048	23,28 min
40000 ms	0.231	0.053	0.055	20,25 min
60000 ms	0.285	0.081	0.084	19,59 min
120000 ms	0.410	0.168	0.173	19.78 min

At all error metrics a strong correlation between iteration length and total error could be observed (see Figure 4.2). This is due to the iteration lengths direct influence on the number of used spikes during pruning.

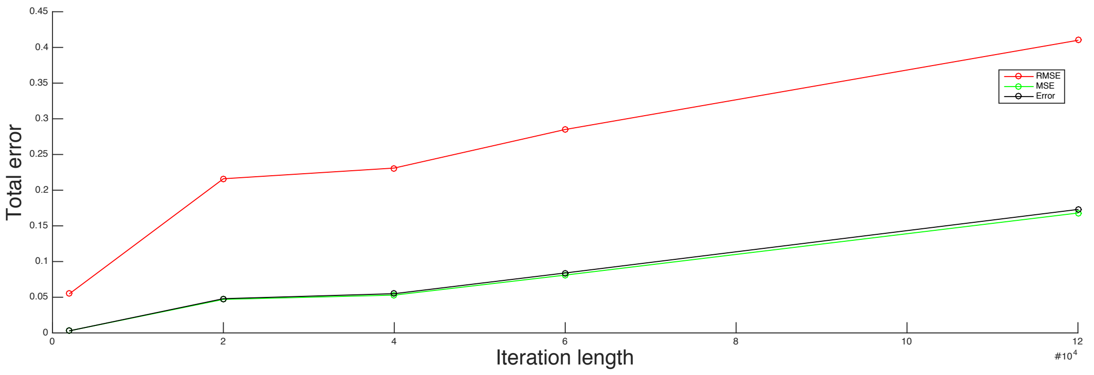

Figure 4.2: Error metrics change magnitude with increasing iteration length

The biggest reduction of memory utilization has been observed by pruning once a demonstration (120000 ms). A clear correlation between total error and computation time coud not be detected (see fig. 4.3).

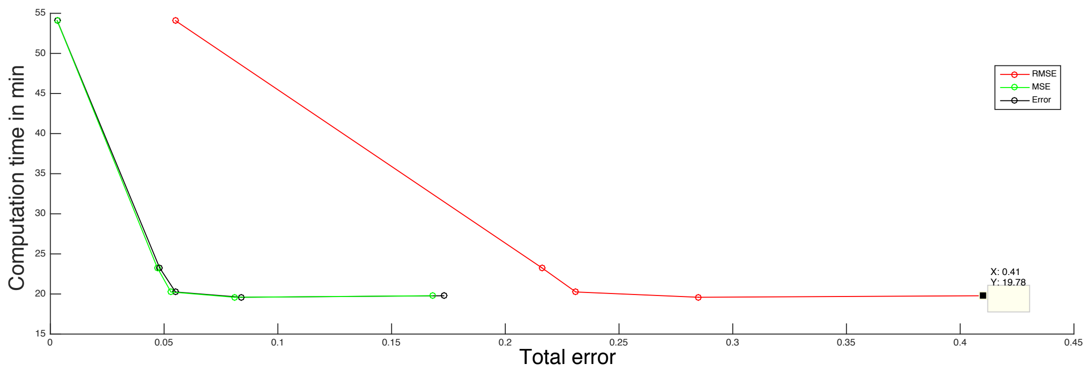

Figure 4.3: Computation duration magnitude with increasing total error

All plots in Figure 4.3 have in common that they have similar curve characteristics. The Error and MSE are in the same magnitude and the RMSE is clearly higher than both at every time step. At all iteration lengths and at every time step, there is a big difference between the (mean absolute) Error and RMSE. That indicates big network errors, which means the degeneration of many synapses. The effect of an increase in iteration length can directly be seen on all error metrics (Figure 4.3) such that the larger the iteration length is, the bigger the total error is.

Pruning every 2 seconds (Fig. 4.4) shows a dynamic error development throughout every metric. When reviewing the error metrics a continuing tendency is visible. MSE and Error concentrate around 0,003 and RMSE concentrates around 0,055.

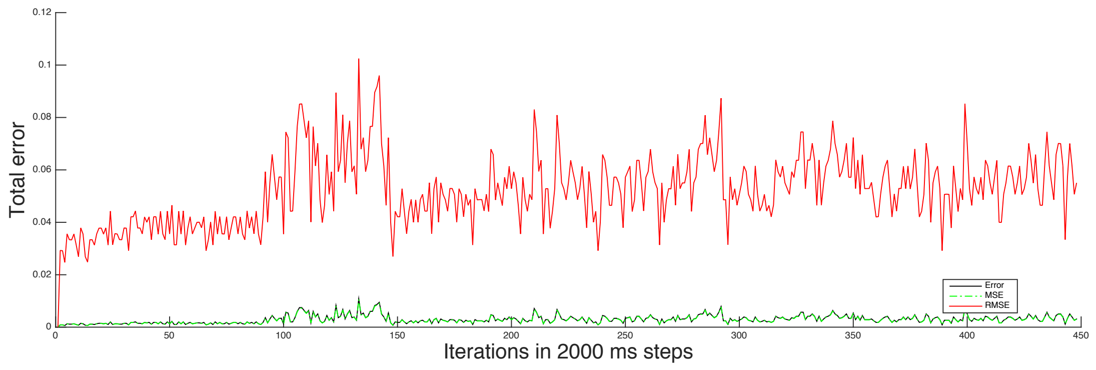

Figure 4.4: Original and proned network error metrics at different iteration lengths

Obviously pruning less frequent leads to a downfall of the error metrics dymanic that has been shown before. The error development is directly connected to the pruning time, which is shown in error progression; 20 seconds pruning time (Fig. 4.5) shows light progression, 40 seconds (Fig. 4.6) shows a clearly visible error increase.

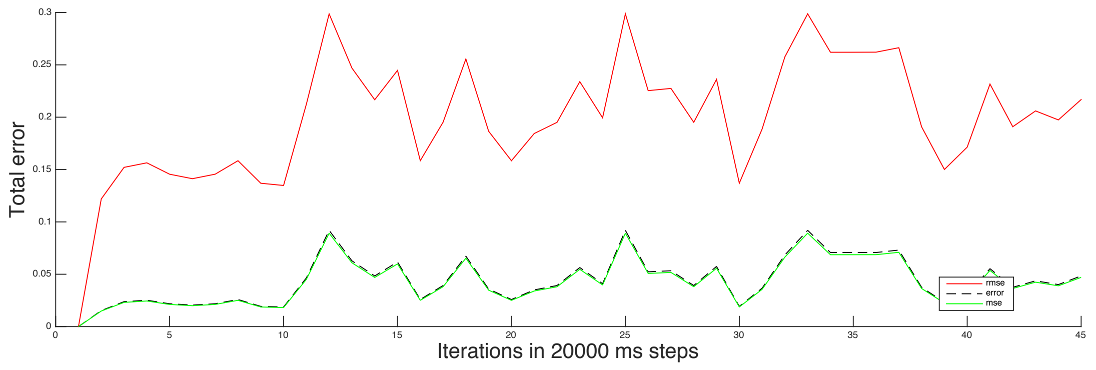

Figure 4.5: Original and proned network error metrics at different iteration lengths

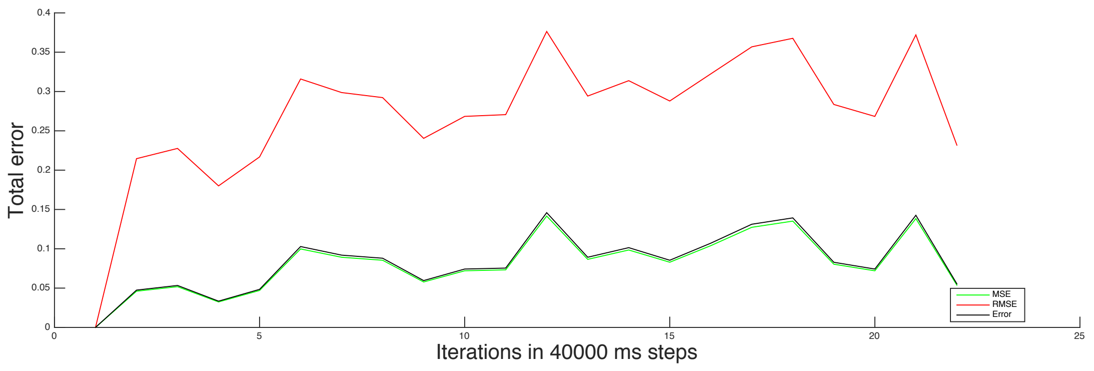

Figure 4.6: Original and proned network error metrics at different iteration lengths

In addition, 60 seconds of pruning time (Fig. 4.7) shows a similar error development as pruning every 40 seconds (Fig. 4.6), it shows a clear tendency upwards, although the trend is less dynamic.

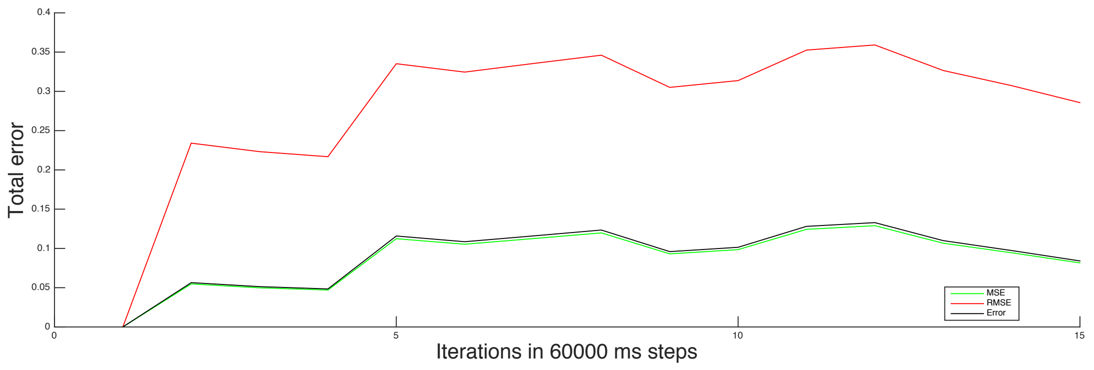

Figure 4.7: Original and proned network error metrics at different iteration lengths

Stressing synapse degeneration as aim, the best results could be achieved by pruning once a demonstration (120000ms; Fig. 4.8). It shows the biggest errors development and therefore the biggest synapse degeneration. A clear upward trend in error development is noticable, but is stabilizing; the MSE and Error concentrate around 0,15 and RMSE concentrates around 0,4.

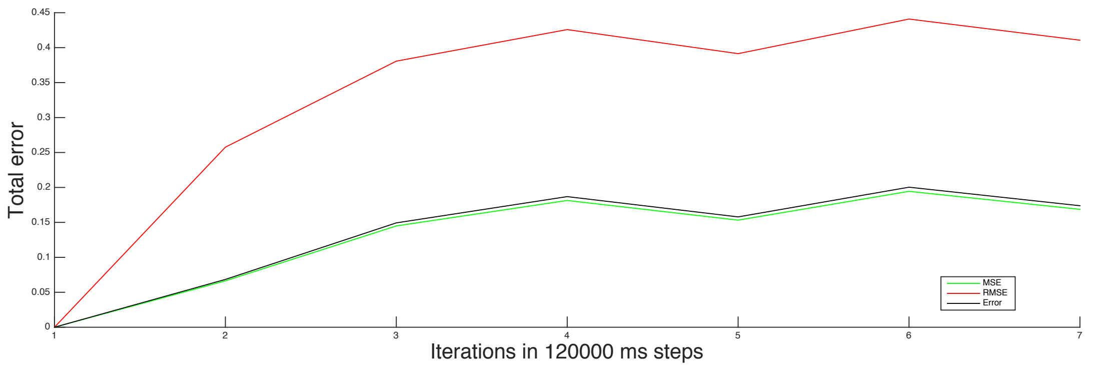

Figure 4.8: Original and proned network error metrics at different iteration lengths

Here we compare the memory requirements in all models presented so far. Note that the models' memory in the last iteration is used for the comparison. Everything but the last iteration is used to examine the effects of an increased number of neural activity data (spikes) and is essential to show the magnitude of pruned synapses.
The initial consumption of neural circuits memory was 0.405 MB. By increasing the time between two prunings (pruning time) and decreasing the number of iterations (effort), the memory requirements could be reduced (see Table 4.3). For example, pruning 14 times every 60 seconds lead to a 9% decreased memory utilization. The biggest achieved reduction of memory usage is 18% and has been achieved by using twice the pruning time (120 seconds) and approximately cutting the effort by half (to 6).

Table 4.3: Memory utilization vs. pruning time/effort - Memory consumption before pruning: 0.405 MB

Total memory	Pruning time	Effort	Memory reduction
0.4037 MB	2000	447	0.3%
0.3848 MB	20000	44	5%
0.3819 MB	40000	21	6%
0.3699 MB	60000	14	9%
0.3324 MB	120000	6	18%

When reviewing pruning time of 2 seconds a nearly uniformed memory utilization is recognizable, this is showing in small amplitudes (that are concerning the memory utilization relative to its initial memory consumption). The memory consumption is reduced by 0.3%, which you can see in Figure 4.9.

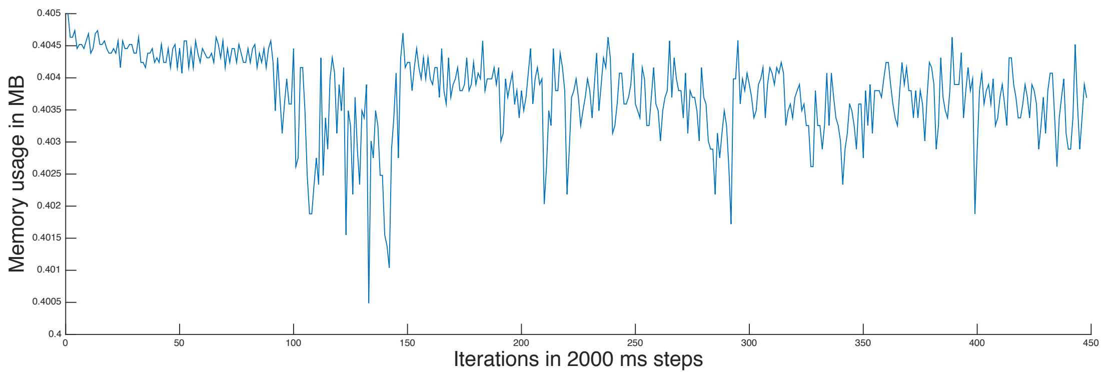

Figure 4.9: Memory usage - Pruning time: 2000 ms, total effort: 447 iterations

When using a pruning time of 20 and 40 seconds the memory utilization is reduced by 6%. In both intervalls, the memory consumption shows reduced dynamics in amplitudes. This is due to the reduced pruning points in time (iterations). As mentioned, this was also done with 40 seconds pruning time. This doubling leads to a falling tendency of decreased memory usage, as seen in Figure 4.11.

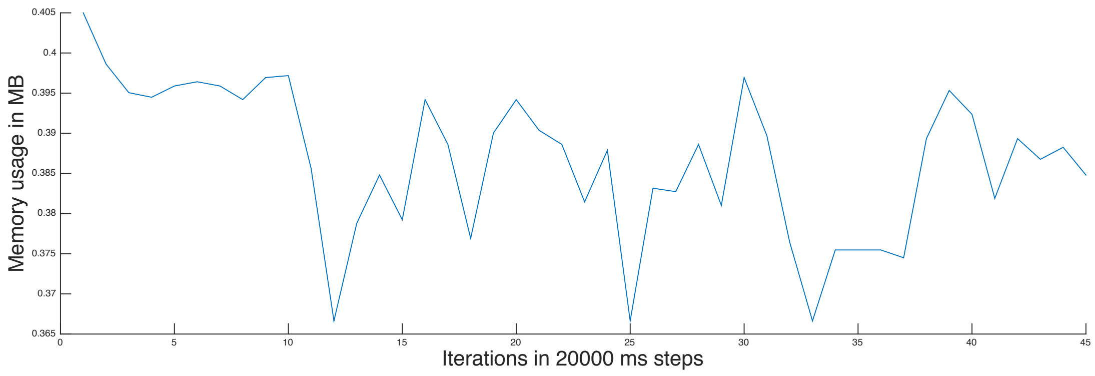

Figure 4.10: Memory usage - Pruning time: 20000 ms, total effort: 44 iterations

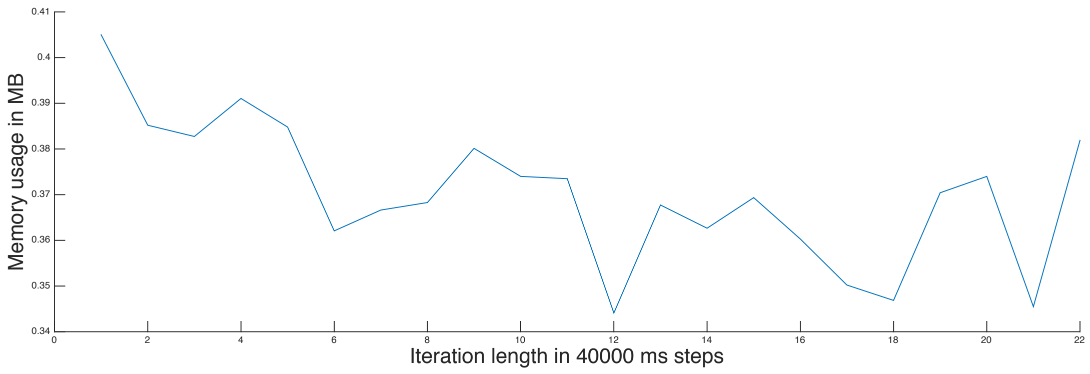

Figure 4.11: Memory usage - Pruning time: 40000 ms, total effort: 21 iterations

To attest to the decreasing tendency of memory usage an increased pruning time of 60 seconds was used. On one hand this increase shows a 9% lower memory utilization and on the other hand it confirms the decreasing dynamic even further.

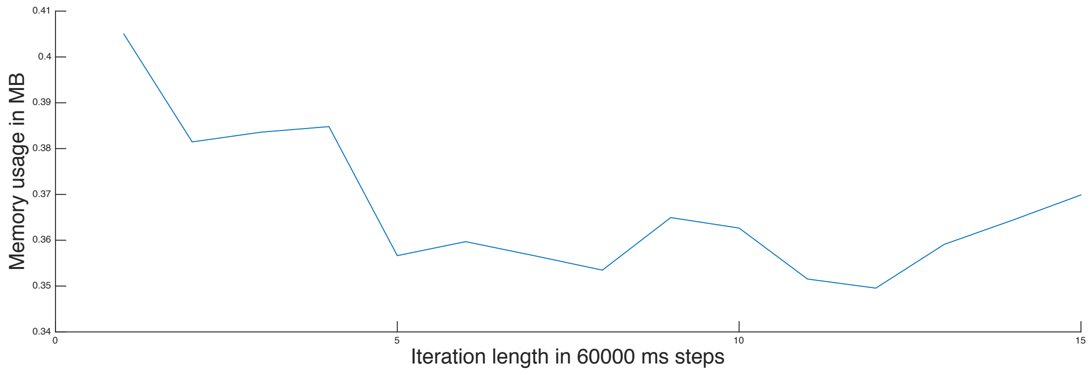

Figure 4.12: Memory usage - Pruning time: 60000 ms, total effort: 14 iterations

Further more pruning only once a demonstration (every 120 seconds) shows a clear tendency in decreased memory utilization (fig. 4.13). The memory consumption could be reduced by around a fifth (18%).

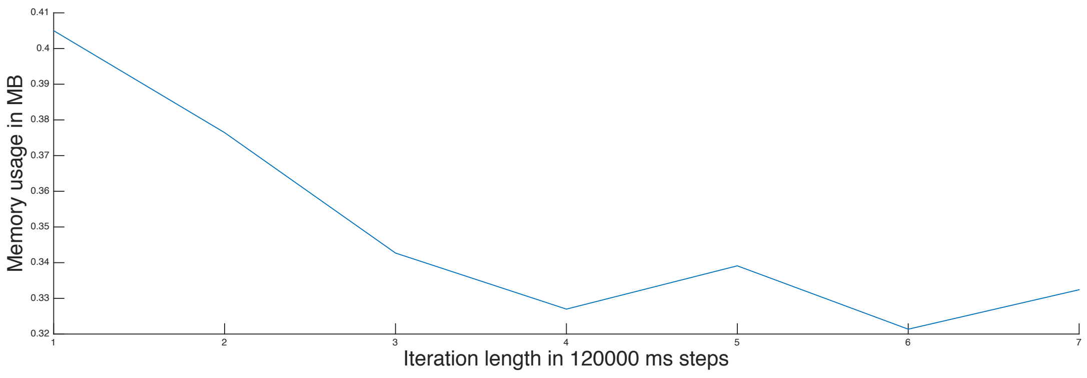

Figure 4.13: Memory usage - Pruning time: 120000 ms, total effort: 6 iterations

As shown in Figure 4.14, the memory utilization decreases linear with increasing Error and MSE. In addition RMSE is behaving similarly, the decrease is close to linear.

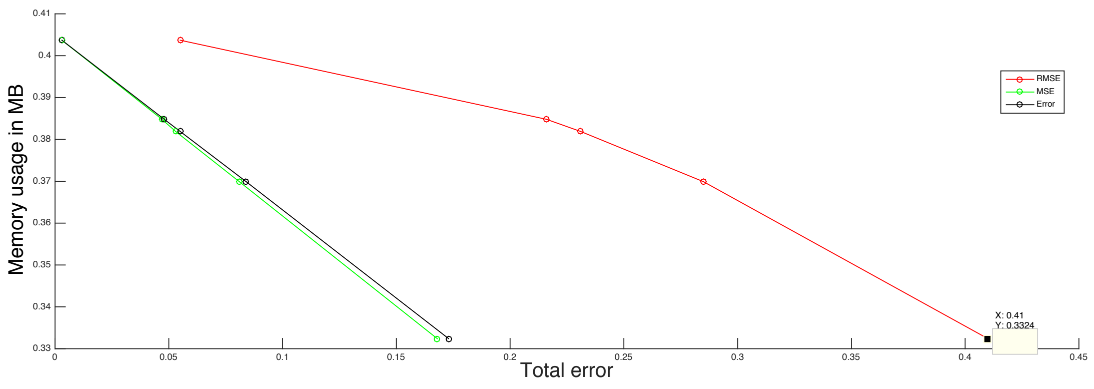

Figure 4.14: Memory usage magnitude with increasing total error

Summarized through the experiment it can be said that with increasing iteration lengths the computational expense reduces (see Figure 4.15).
That is due to the fact, that in less occasions the data set has to be pruned which reduces the number of applications (Table 4.4 - column effort) of the computationally intensive pruning method.
When comparing the original time needed to build learn synapse connections, which lasts 0,56 seconds, one can see the raise of computational effort by 950%; when pruning every 2000 ms. This effect is reversible when reducing the total number of applications of pruning; when pruning every 20000 ms the computational effort is cut in half to 403%. This is also true for pruning times 40000 ms, 60000 ms and 120000ms. The computational effort could be reduced to 350% on average.

Table 4.4: Computation time vs. error - Computation time without pruning: 0,56 minutes

Computation time	Pruning time	Effort	Comp. expense
54,08 min	2000	447	+ 950%
23,28 min	20000	44	+ 403%
20,25 min	40000	21	+ 359%
19,59 min	60000	14	+ 347%
19,78 min	120000	6	+ 351%

As you can see in Figure 4.15, the interval of 2000ms to 40000ms was the most efficient reduction regarding computational expense.
The main reason should be named the neural activity parameters γ poorly performing implementation, which is in the worst case even worse than cubic in the number of neurons (n).

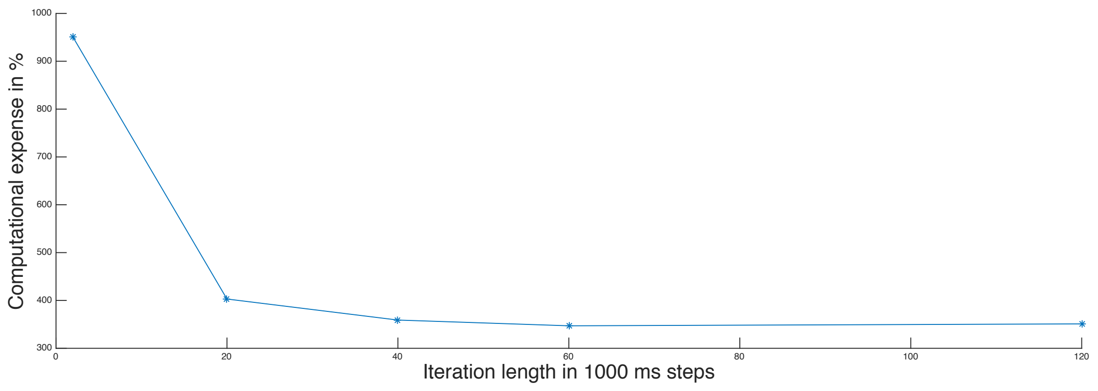

Figure 4.15: Computational time reduction magnitude

5 Results

5.1 Task space description

In this Section description of the task space will be given.
The two dimensional task space, which equals a cartesian xy plane, spanns the area, the robot endeffector can reach. In this area two types of neurons are current: State and Task neurons (see Fig. 5.1). To model the task space, state neurons encode the state transition model in synaptic connections, task neurons encode initial position and target position and obstacles. These neurons stretch across the whole area in a consistent pattern and are fully connected.

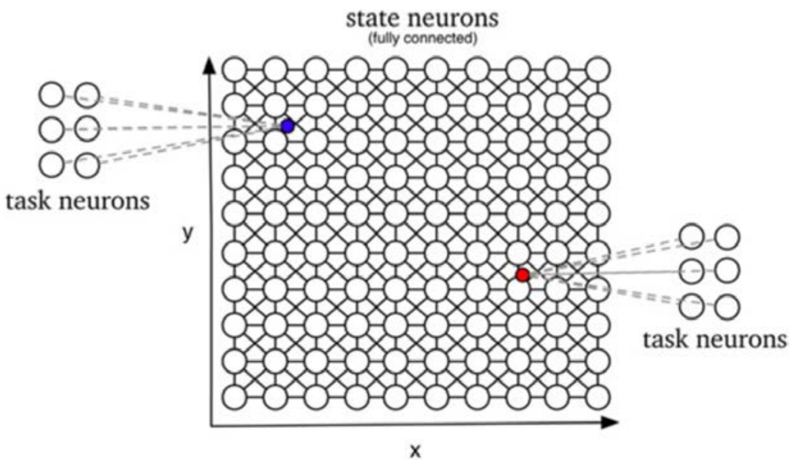

Figure 5.1: Two dimensional task spaces transition model - Fully connected state neurons; Task related information encoded task neurons; Source [1]

This Section is about showing the changes induced by the pruning of the synapses.
To emphasise these changes the original and the pruned transition model is analysed in further detail.
The visualisation of the two dimensional task spaces transition model is shown as a heatmap and built as follows: Both on X and Y axis the all neuron IDs is pictured. A point on the xy plane stands for the synaptic weight between two neurons. The synaptic weights are between the value of - 1 and 1. They are also colour coded: Blue for -1 (equals strong inhibition), white for 0 (equals degeneration) degeneriert and red for 1 (strong excitation).
In Figure 5.2 the current state of the synaptic weights prior to learning is shown. Synapses between the same neurons have been set to 0, all other have been distributed uniformly $\mathscr{U}$(-0.99,-0.95) and inhibitory initialized.

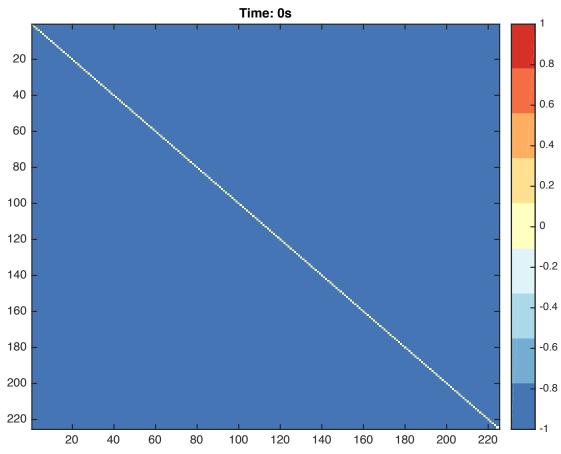

Figure 5.2: Two dimensional task spaces transition model, after strong randomly inhibited (-1) initialization, before learning of 225*225 transitions

Here, in Figure 5.3, the resulting transition model of unoptimized learning of the task space is visualized, where it shows that synaptic weights equal their initial inhibitionary weight (blue areas), some synapses have been degenerated (weight around 0, yellow spots), including a minor quantity of weights that have been learned excitatory (around 1, red spots).
A highlight in the optimized (in terms of pruning) transitions model (Figure 5.4) is the number of inhibitory initialised synapsed which have died off (to be seen as large yellow areas) and no (strong) excitatory synapses to be found.

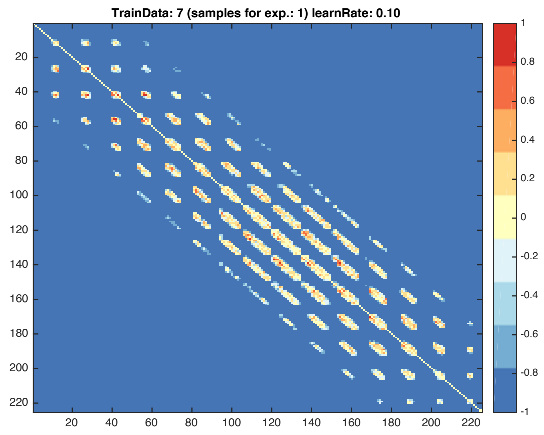

Figure 5.3: Final model of two dimensional task spaces unpruned transition model with 225 neurons

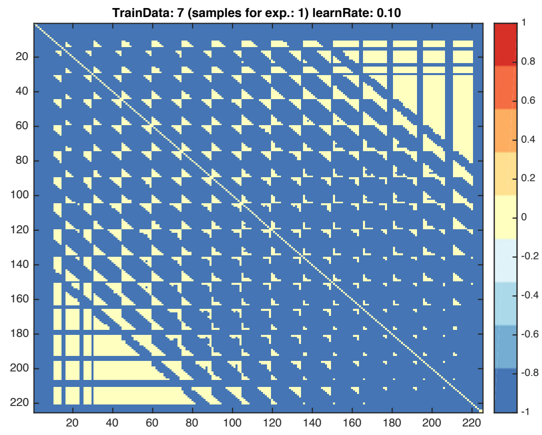

Figure 5.4: Final model of two dimensional task spaces pruned transition model with 225 neurons

In this Section the target reaching task is visualized.
The focal point lies in the target reaching task, given start position and end position. The goal is to find a robot arm movement (trajectory) from start- to end position based on the learned task space planning model.

The following 4 figures (Fig. 5.5 - Fig. 5.8) present movement trajectories in 2 dimensional task space. 50 sampled trajectories each are visualized from the original and pruned learned model. The figures are build on an xy axis. Each neuron was given a number starting top left to bottom right which means every neuron is accounted for. To visualize the experiment the starting position is marked by black circle and the end position is marked by a black square. Grey squares are obstacles, that cannot be crossed. The colored lines show the movement trajectories.

Original network - Trajectory overview
In the Fig. 5.5 - 5.6 movement trajectories generated by the original network are shown. Out of the 50 sampled trajectories, 40 sampled trajectories have been accepted (Fig. 5.5). Inspecting the rejected trajectories (Fig. 5.6), it can be seen, that many of the trajectories do not provide smooth robot arm movements, as indicated by the movement between neuron 130 and 131 in Fig. 5.6.

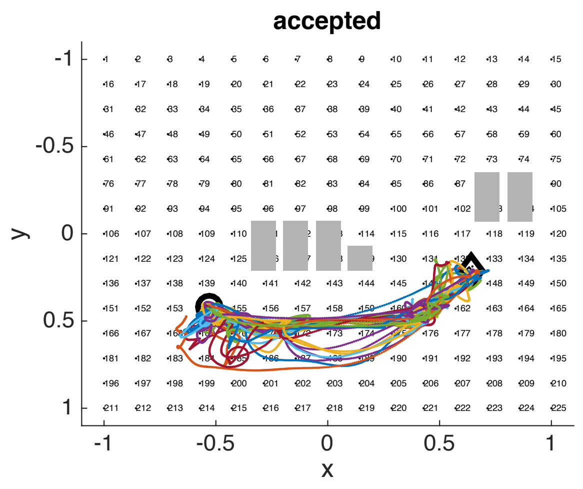

Figure 5.5: Target reaching experiment; Originial network; 40 accepted trajectories from start to end position found

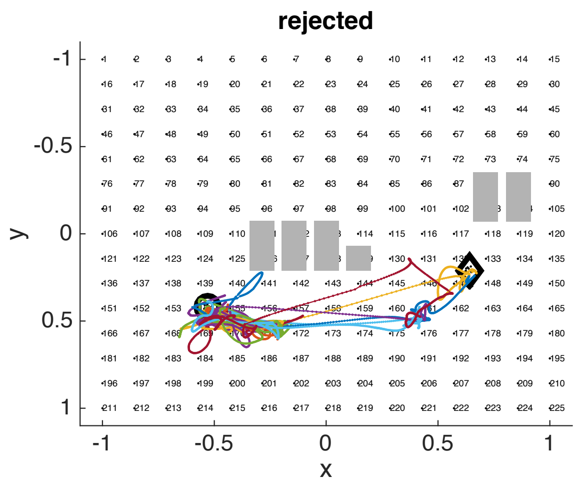

Figure 5.6: Target reaching experiment; Original network; 10 rejected trajectories from start to end position

Pruned network - Trajectory overview
In Figures 5.7 and 5.8 the generated trajectories are shown, based on the pruned network. In Fig. 5.7 it is visible that 2 of the 50 trajectories have been accepted (and 48 declined). The declined trajectories (could) have been declined, because either the jerk has been to explosive (when maximum rate blub blub blub) or the trajectories have hit an obstacle (see Fig. 5.8).

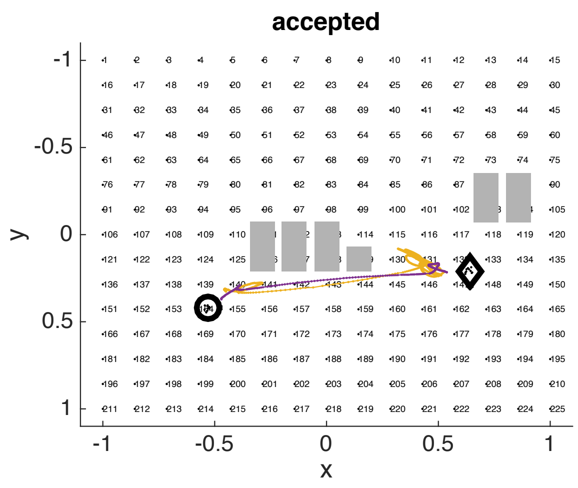

Figure 5.7: Target reaching experiment; Pruned network; 2 accepted trajectories from start to end position found

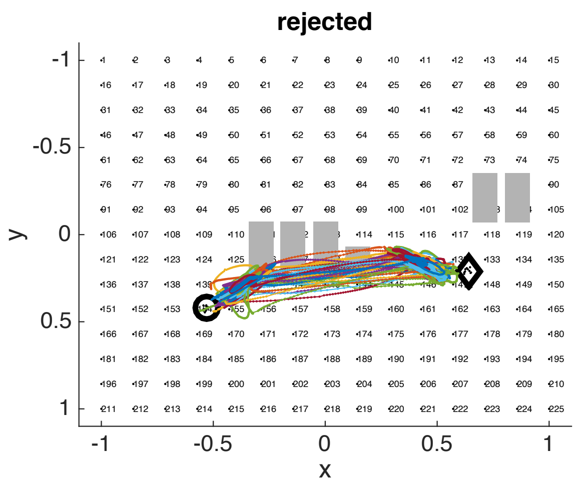

Figure 5.8: Target reaching experiment; Pruned network; 48 rejected trajectories from start to end position

Pruned network - Target reaching movement, accepted trajectories
In Figures 5.9 and 5.10 the movement of the robotarm is shown, with main focus on the 2 accepted trajectories - from circle to square. It differs slightly from the previous Figures 5.5 - 5.8 by inserting a robot arm and x and y axis are slightly adjusted.

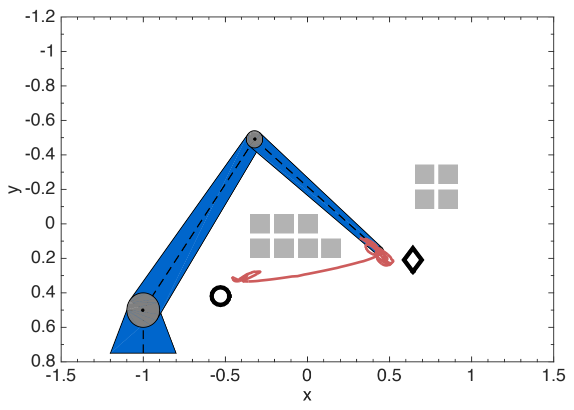

Figure 5.9: Target reaching robotarm; Pruned network; First accepted trajectory

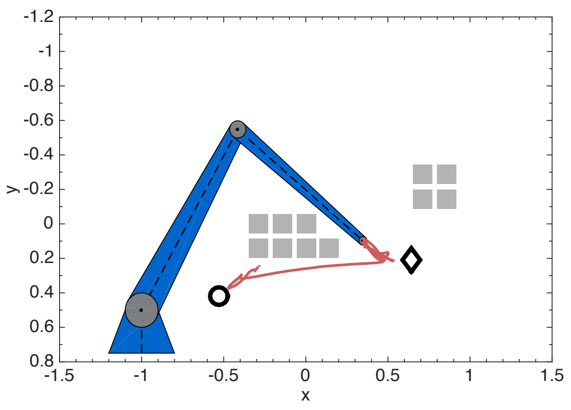

Figure 5.10: Target reaching robotarm; Pruned network; Second accepted trajectory

In this section a visualization of the pruned networks application in an obstacle avoidance task will be shown. The obstacle avoidance task is about finding a valid (in terms of acceptance) trajectory from a given start- to end position with arbitrary placed obstacles, that cannot be crossed. Here the same visualization is used as in Section 5.3 Figures 5.5 - 5.8.
What is the most important to mention here: Obstacles are encoded as repelling forces through synaptic inhibition. Trajectories that cross an obstacle equals the robots arm hitting an obstacle.

Original network
As it is shown in Fig. 5.11, 5 accepted trajectories could be found, where each respresent a robot arm movement from start to end position by avoiding the obstacles at the same time. Of 50 sampled trajectories 45 have been declined. On one hand they have been declined because the arm hit an obstacle (see Fig 5.12) in its course. On the other hand a few have been declined because they haven't reached the end point or because the jerk was too explosive.

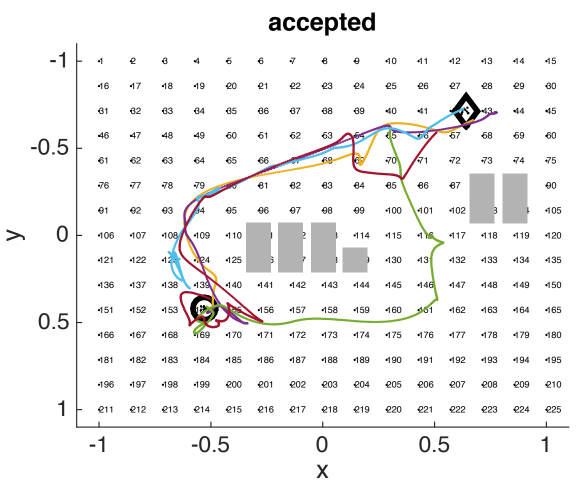

Figure 5.11: 5 accepted trajectories

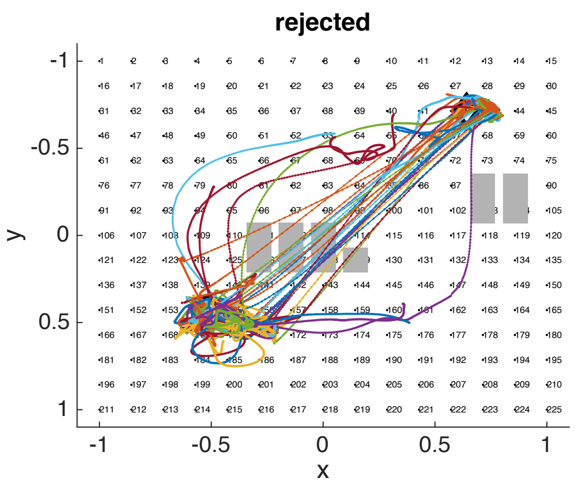

Figure 5.12: 45 rejected trajectories

Pruned network
With the pruned network all sampled trajectories got rejected (compare with Fig. 5.13 - no accepted trajectories). As shown in Figure 5.14, all trajectories reach nearly the end posotion, but cannot avoid crossing the obstacles.

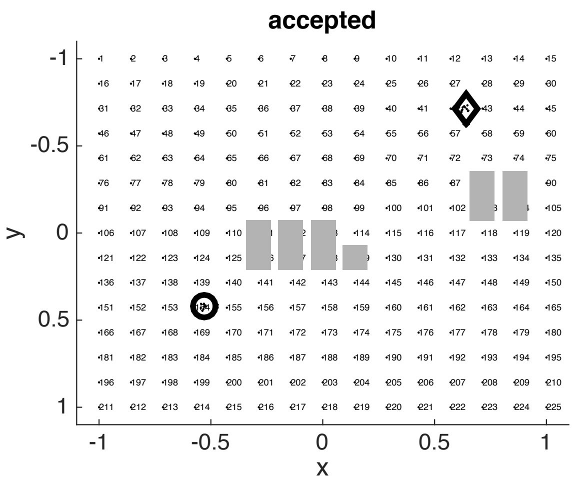

Figure 5.13: No trajectories from start position to end position found

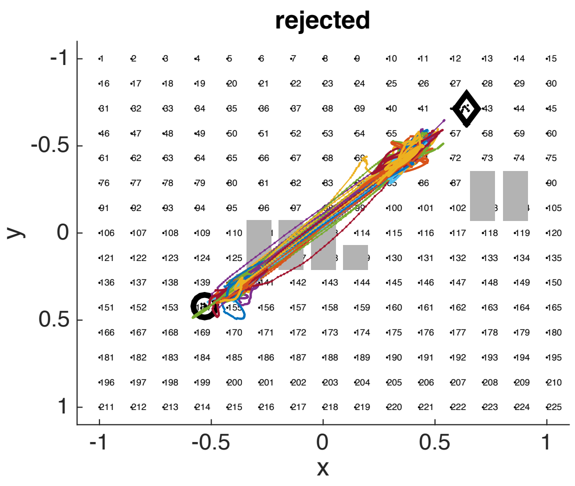

Figure 5.14: (All) 50 trajectories rejected

6 Conclusion & Future work

6.1 Conclusion

The goal of this thesis was to demonstrate the usability of models for synapse formation and pruning to reduce the memory requirements while maintaining the task perfermance. We were able to reduce the memory requirement of a spiking neural network (SSN) by 18%, without destroying the objective of the network to generate accurate movement plans.
To solve such a demanding problem, we firstly investigated relevant neurobiological parameters, which influence the generation and degeneration of synapses. This lead to an interdisciplinary work, that was inspiring and taxing by itself. Secondly, after identifying the main parameters, we have built models that simulate the synaptogenesis and synapse pruning. The fundament for all models is the exponential function with negative exponent.
For testing the developed strategies, we used the network in [1] for learning a transition model for robot planning problems. The algorithm is based on contrastive divergence and shows the complex learning process of SNN robot planning during problem solving. We have extended the learning process by including a synapse pruning model and the resulting two dimensional task space transition model, both have been evaluated by a target reaching and obstacle avoidance task.
The goal was to optimize the memory requirement of the SSN encoding the transition model. For the target reaching task 225 fully connected neurons were used. We reduce the memory requirement by approximately 1/5, while still being able to solve the target reaching task.

6.2 Future work

The optimized transition model failed the obstacle avoidance task, although it reached a significant reduction of memory requirement. This failure and success is reason to investigate further:
Further consideration should be given to include task performance as pruning criteria. By including task performance the death of synapses can be done task dependet and therefore more targeted.
In addition the model parameter learning rate, has not been changed throughout the experiment and the equidistant pruning points in time have been determined manually. This leads to the conclusion that using algorithms or fixed methods like cross validation for hyper parameter tuning is going to optimize this experiment to a more accurate level.

Bibliography

[1] D. Tanneberg, “Spiking neural networks solve robot planning problems,” no. 1, 2015.

[2] “Spiking neural networks for vision tasks.” `https://www.nst.ei.tum.de/fileadmin/w00bqs/www/publications/as/2015WS-HS-SpikingVision.pdf`. Accessed: 2017-03-14 21:07.

[3] S. Cash and R. Yuste", “Linear summation of excitatory inputs by ca1 pyramidal neurons,” 1999.

[4] “580.439 course notes: Cable theory.” `http://pages.jh.edu/motn/coursenotes/cable.pdf`. Accessed: 2017-03-14 22:24.

[5] “Notes on contrastive divergence.” `http://www.robots.ox.ac.uk/~ojw/files/NotesOnCD.pdf`. Accessed: 2017-03-14 21:24.

[6] E. Rueckert, D. Kappel, D. Tanneberg, D. Pecevski, and J. Peters", “Recurrent spiking networks solve planning tasks,” 2016.

[7] D. Tanneberg, A. Paraschos, J. Peters, and E. Rueckert", “Deep spiking networks for model-based planning in humanoids,” 2016.

[8] D. A. N. et al., “Distance-dependent differences in synapse number and ampa receptor expression in hippocampal ca1 pyramidal neurons,” 2006.

[9] G. Stuart, N. Spruston, B. Sakmann, and M. Hausser", “Action potential initiation and backpropagation in neurons of the mammalian cns,” 1997.

[10] N. Spruston, D. B. Jaffe, and D. Johnston", “Dendritic attenuation of synaptic potentials and currents: the role of passive membrane properties,” 1994.

[11] N. L. Golding, J. T. Mickus, Y. Katz, W. L. Kath, W. L., and N. Spruston", “Factors mediating powerful voltage attenuation along ca1 pyramidal neuron dendrites,” 2005.

[12] G. Stuart and N. Spruston", “Determinants of voltage attenuation in neocortical pyramidal neuron dendrites,” 1998.

[13] A. Lazar, G. Pipa, and J. Triesch", “Sorn: A self-organizing recurrent neural network,” 2009.

[14] F. Faghihi and A. A. Moustafa", “The dependence of neuronal encoding efficiency on hebbian plasticity and homeostatic regulation of neurotransmitter release,” 2015.