# Title of Bachelor thesis

Bachelor-Arbeit
zur Erlangung des Hochschulgrades
Bachelor of Science
im Bachelor-Studiengang Physik

vorgelegt von

Felix Soest
geboren am 16.09.1998 in Düsseldorf

Institut für Theoretische Physik
Fakultät Physik
Bereich Mathematik und Naturwissenschaften
Technische Universität Dresden
2021

Eingereicht am xx. Monat 20xx

1. Gutachter: Prof. Dr. Walter Strunz
2. Gutachter: Prof. Dr. Oscar Dahlsten

# Summary

Abstract
English:
Aufmerksamkeit große Fragen? unsere Frage unsere Antwort

Abstract
Deutsch

# Contents

# 1. Introduction

## 1.1. Intro

Test 123

# 2. Background

## 2.1. Collision model dynamics

A standard approach to modeling open quantum systems is the collision model. The system under consideration interacts with a series of ancilla systems through a unitary transformation on the combined system-ancilla state [10]. After a short interaction time $\Delta t$, the ancilla systems are traced out to receive the system state. This type of interaction will in general lead to entanglement between system and ancilla. To ensure the ancilla and system states remain pure, we follow the approach used in [1]: let $\rho_S$ be the density operator of the system under consideration and $|\psi_A\rangle$ the state of the ancilla system. Given $\Delta t \ll 1$ the time evolution of $\rho_S$ under $H_{AS}$ acting on the combined system is

$$\rho_S' = \mathrm{Tr}_A\{e^{-iH_{AS}\Delta t}(|\psi_A\rangle\langle\psi_A| \otimes \rho_S)e^{iH_{AS}\Delta t}\} \;= \rho_S - i\Delta t[\langle\psi_A|H_{AS}|\psi_A\rangle, \rho_S] + O(\Delta t^2). \tag{2.1}$$

In the continuous limit equation 2.1 leads to von-Neumann dynamics on the system:

$$\dot{\rho}_S = -i[\langle\psi_A|H_{AS}|\psi_A\rangle, \rho_S].$$

As each ancilla only interacts once with the system and is traced out afterwards, $\rho_S$ remains pure in the limit of $\Delta t \to 0$. This framework allows the experimenter to create constant Hamiltonians by initialising multiple ancillas in the same state (see Figure 2.1). In this fashion one can create piece-wise constant Hamiltonians by initialising $n = \frac{\Delta \mathrm{T}}{\Delta t}$ identical ancillas, where $\Delta\mathrm{T}$ is the time for which the Hamiltonian remains constant.

We use this approach as the framework to our setting, which consists of three qubits: the Drive, System and Transducer qubits. The Drive and Transducer qubits can be set by the experimenter in N discrete steps modelled as piecewise constant functions (PWC) of $(\theta_D, \phi_D)$ and $(\theta_T, \phi_T)$ respectively (see Figure 2.1), the system qubit is initialised in a pure state.

In the remainder of this work we use the interaction Hamiltonian on the three qubit Hilbert space

$$H_{DST} = H_I \otimes \mathbb{1}_T + \mathbb{1}_D \otimes H_I, \qquad H_I = \sigma_+ \otimes \sigma_- + \sigma_- \otimes \sigma_+$$

unless otherwise noted. The time evolution and work extraction is then calculated as follows, where $\Delta$T is time span between qubit switching:

$$H_S^i = \left\langle \psi_D^i \right| \left\langle \psi_T^i \right| H_{DST} \left| \psi_D^i \right\rangle \left| \psi_T^i \right\rangle \tag{2.2}$$

$$\rho_S^{i+1} = U_i \rho_S^i U_i^\dagger,\ U_i = e^{-iH_S^i \Delta \mathrm{T}} \tag{2.3}$$

$$W = -\Sigma_i \mathrm{Tr}\ \rho_S^i\ dH_S^i \tag{2.4}$$

$$dH_S^i = \left\langle \psi_D^i \right| \left\langle \psi_T^{i+1} \right| H_{DST} \left| \psi_D^i \right\rangle \left| \psi_T^{i+1} \right\rangle - \left\langle \psi_D^i \right| \left\langle \psi_T^i \right| H_{DST} \left| \psi_D^i \right\rangle \left| \psi_T^i \right\rangle . \tag{2.5}$$

Here we use the partial Hamiltonian $H_S^i$ on S at time step $i \in [1, N-1]$, as well as corresponding system density matrix $\rho_S^i$.

Using the Bloch sphere representation $|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle$ to represent Drive and Transducer qubits reduces equation 2.2 to

$$H_S^i = \frac{1}{2} \left[ \sin\left(\theta_D^i\right) e^{i\phi_D^i} + \sin\left(\theta_T^i\right) e^{i\phi_T^i} \right] \sigma_+ + h.c. \tag{2.6}$$

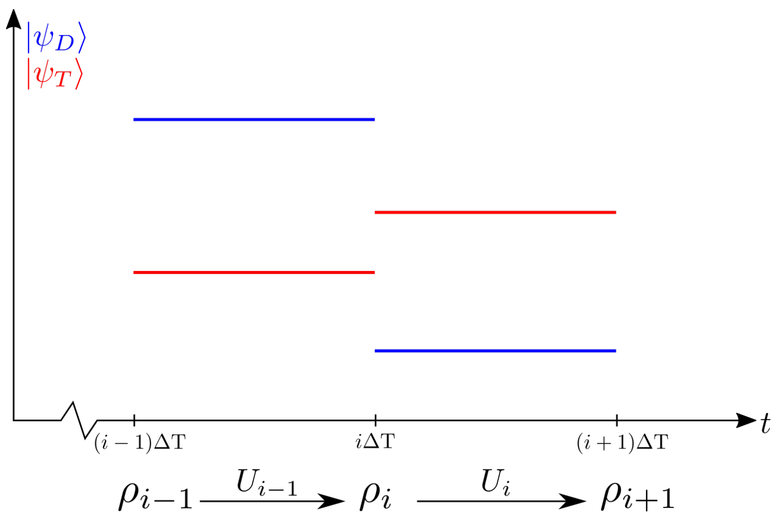



**Figure 2.1.:** Piecewise constant implementation of Drive and Transducer qubits: the vertical axis an arbitrary parameter of the ancilla states. The qubit states are switched instantaneously and then kept constant for $\Delta\mathrm{T}$ while $\rho_S$ evolves unitarily.

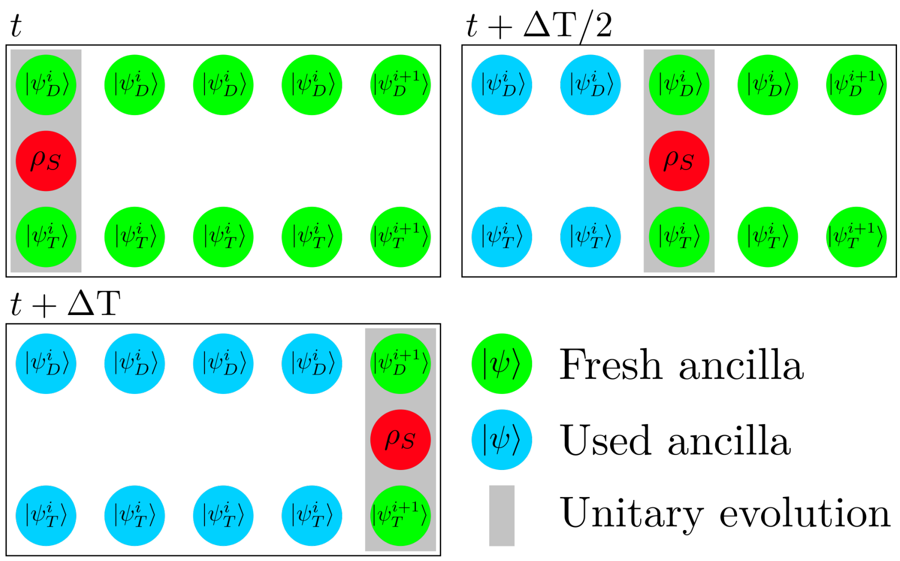



**Figure 2.2.:** Collision model used in this work: Drive and Transducer are series of qubits interact once with the system and evolve the reduced density operator $\rho_S$. The qubit configuration can be changed in intervals of $\Delta\mathrm{T}$.

## 2.2. Supervised Machine Learning

Machine learning is a subfield of artificial intelligence, 'concerned with the question of how to construct computer programs that automatically improve with experience.' [14] Supervised machine learning is one of the three machine learning disciplines, besides unsupervised and reinforcement learning. The goal is to find a mapping between an input and an output, in our case an excitation and its respective optimal harvesting policy. Multiple algorithms to find such a mapping exist, however for high dimensional problems artificial neural networks (ANNs) are usually used. In general, ANNs are a collection of neurons which are connected and can transmit signals to each other. The ANN can learn by changing how different information is passed through the network. In the following sections we review two ANN architectures, the fully-connected feedforward ANN and the Long Short-Term Memory (LSTM) network.

### 2.2.1. Fully-connected feedforward ANNs

In this section we review ANNs, following the exposition given in [11]. Let $\mathfrak{R}$ be a fully-connected feedforward ANN, meaning there are no loops in the neuron connections and all neurons in a layer are connected to every neuron of the next layer, $\mathfrak{R} : \mathbb{R}^{n_1} \to \mathbb{R}^{n_L}$. $n_1$ and $n_L$ denote the dimensionality of the input and output respectively. $\mathfrak{R}$ has $L$ layers, or columns of neurons. The network architecture is given by the amount of neurons $n_l$ in each hidden layer $l \in [2, L-1]$ (see figure 2.3). The neurons in layer $l$ are represented by their activations $\vec{a}_l \in \mathbb{R}^{n_l}$, which represent the matrix multiplication output. Additionally each layer includes trainable parameters $W_l \in \mathbb{R}^{n_{l+1} \times n_l}$ and $\vec{b}_l \in \mathbb{R}^{n_l}$ called weights and biases respectively. The activations can then be calculated using the following formulae [20]:

$$\begin{aligned}
\vec{a}_2 &= W_1 \vec{a}_1 + \vec{b}_1, \\
\vec{a}_l &= W_{l-1} \xi(\vec{a}_{l-1}) + \vec{b}_{l-1}, \ l \in [3, L],
\end{aligned}$$

where $\xi(x)$ is a function called the activation function applied elementwise. Historically, functions such as tanh and sigmoid have been used. However, it has been shown [12, 7] that the rectified linear unit $\mathrm{ReLU}(x) = \max(0, x)$ often provides better results and is used here.

### 2.2.2. Long Short-Term Memory

While the network architecture introduced in the previous section performs reasonably well on many problems, it destroys spatial and temporal correlations present in the data. Instead convolutional and recurrent networks are often used for these purposes, e.g. in image recognition and time series forecasting [18, 4].

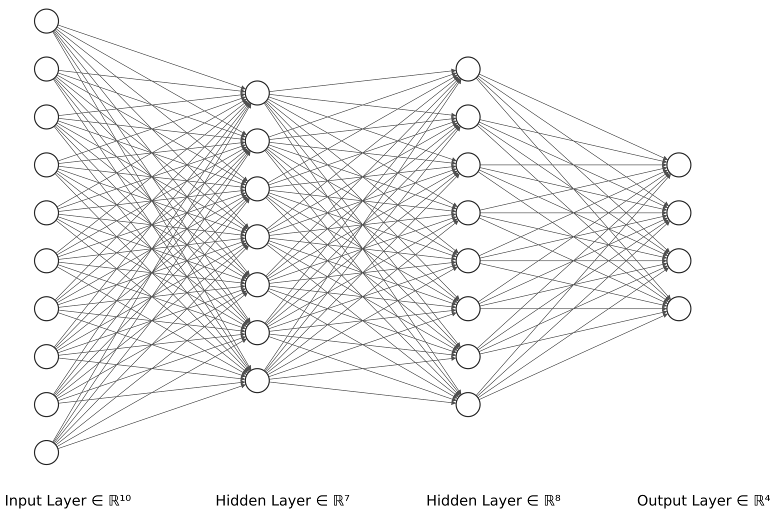



**Figure 2.3.:** Example fully-connected feedforward ANN with four layers, including input, output and two hidden layers [9].

Here we use the LSTM architecture, a type of recurrent neural network (RNN) introduced in [6]. The core idea of RNNs is the usage of loops to store and propagate information through time (figure 2.4).

The network is made up of a row of LSTM cells which share parameters. The cell uses the current input $x_t$ as well as the previous cell state $c_{t-1}$ and output $h_{t-1}$ to compute the output $h_t$. Internally, the cell is comprised of multiple gates which control the storage of information, $i_t, f_t, g_t, o_t$, which are the input, forget, cell and output gates respectively. The output and

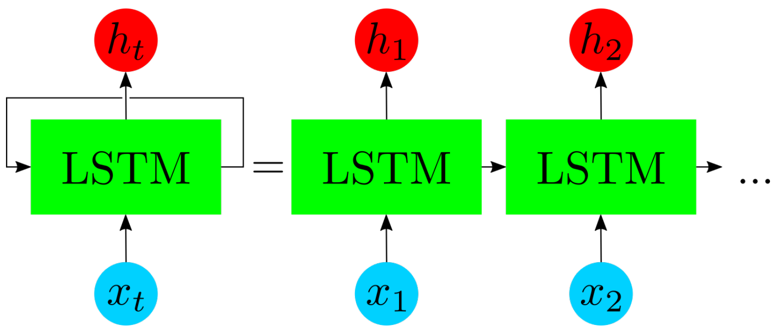



**Figure 2.4.:** Example RNN with LSTM architecture. Each LSTM block has the same parameters and information is fed into the network sequentially.

gates of each cell are computed using the following equations [17]:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}),\\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}),\\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}),\\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}),\\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t,\\
h_t &= o_t \odot \tanh(c_t),
\end{aligned}
$$

where $\odot$ is the Hadamard product and $\sigma(x)$ is the sigmoid function. The input and forget gates $i_t, f_t$ express to what extent new data should be incorporated or deleted from the current cell state $c_t$ respectively.

### 2.2.3. Training & Backpropagation

To train an ANN a cost function is defined, often the mean squared error

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\vec{a}_{L,i} - \vec{y_i})^2,$$

where the summation is performed over the training data $\{(\vec{x}_i, \vec{y}_i)\}$ with $N$ samples. $\{\vec{x_i}\}$ is the input, $\{\vec{y_i}\}$ the output data and $\vec{a}_{L,i} = \mathfrak{N}(\vec{x}_i)$ the output of the neural network. The so-called backpropagation algorithm is used to calculate the gradient of the cost function with respect to the trainable parameters and improve the performance of the ANN [18, 15].

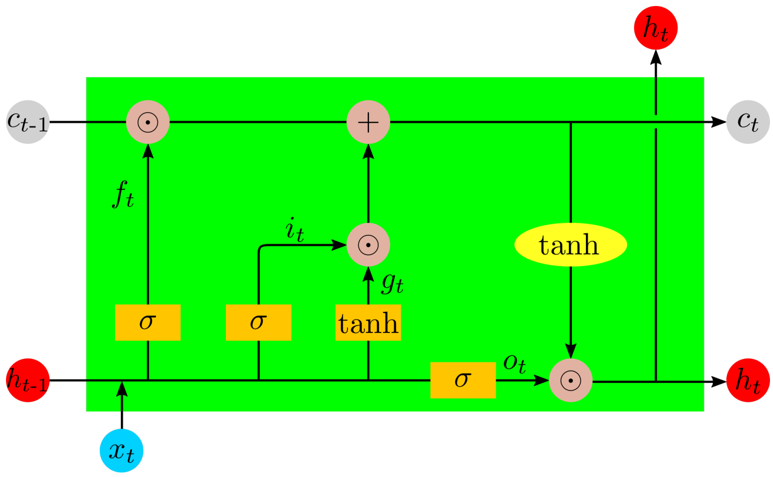



**Figure 2.5.:** Visualisation of a single LSTM cell. The input $x_t$ enters the cell in the bottom left and is concatenated with the output $h_{t\text{-}1}$ of the previous cell. The combined data then enters multiple single-layer neural networks represented by orange rectangles with their respective activation function. The forget gate $f_t$ removes data from the previous cell state $c_{t\text{-}1}$ while $i_t$ and $g_t$ control how new data is added to the memory. tanh is applied elementwise over the cell state (yellow ellipse) to determine the cell output $h_t$ together with the output gate $o_t$.

# 3. Experimental Results

The training data is created using a minimisation algorithm [19], which finds the optimal Transducer protocol $\{|\psi_T^i\rangle\}$ given a Drive sequence $\{|\psi_D^i\rangle\}$. The networks are trained to learn the mapping $\{|\psi_D^i\rangle\} \to \{|\psi_T^i\rangle\}$. Both the input (Drive) and output (Transducer) are transformed by the embedding

$$\left\{\begin{pmatrix}\theta^i & \phi^i\end{pmatrix}\right\} \to \left\{\begin{pmatrix}\sin(\theta^i) & \sin(\phi^i) & \cos(\theta^i) & \cos(\phi^i)\end{pmatrix}\right\}.$$

The reasons for this operation are twofold: it normalises the data to the interval $[-1, 1]$, which is beneficial to learning [8]. Additionally it adds information regarding the periodicity of the qubit angle representation.

To compare the accuracy of different models a performance indicator is required. Naturally one might use the MSE as introduced in section 2.2. Instead we define the *efficiency* of a model $\mathfrak{N}$ on a dataset $\{(\vec{x}_i, \vec{y}_i)\}$ as

$$\eta = \frac{1}{N}\sum_{i=1}^{N} \frac{W(\vec{x}_i, \mathfrak{N}(\vec{x}_i))}{W(\vec{x}_i, \vec{y}_i)}, \tag{3.1}$$

i.e. the arithmetic mean of the ratios of work output predicted by the model to optimal work output. The function $W(\vec{x}_i, \vec{y}_i) = W(\{|\psi_D\rangle\}_i, \{|\psi_T\rangle\}_i)$ returns the work given a Drive and Transducer protocol.

It should be noted that using the MSE for training is not an optimal choice, as we will see later. Directly using the work extracted would be a better choice, but we refrain from it here for two main reasons. Firstly, the interaction Hamiltonian must be known to the experimenter for use as a cost function. Secondly, the computation of the extracted work becomes costly for larger $N$.

## 3.1. Influence of $\Delta\mathrm{T}$ on Work Output

We start our investigation by determining the work output $W$ when varying the time between qubit switching $\Delta\mathrm{T}$. If the System qubit is initialised in the pure state $\rho_S = |0\rangle\langle 0|$, the work output for a single jump is given by (see appendix A.1 for a derivation)

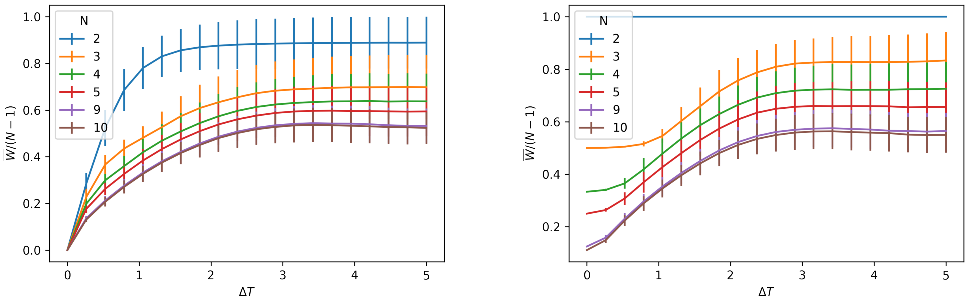
$$W = \frac{1}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{(\tau' - \tau)\alpha^*\} \tag{3.2}$$

$$\alpha = \frac{1}{2}\left[\sin\left(\theta_D^1\right)e^{i\phi_D^1} + \sin\left(\theta_T^1\right)e^{i\phi_T^1}\right], \qquad \tau' - \tau = \frac{1}{2}\left[\sin\left(\theta_T^2\right)e^{i\phi_T^2} - \sin\left(\theta_T^1\right)e^{i\phi_T^1}\right].$$

We note that for $\Delta\mathrm{T} \to 0, W \to 0$ as $\mathrm{Tr}\{\rho_S H_S\} = 0$ for all configurations of $|\psi_D\rangle$ and $|\psi_T\rangle$. We simulate 500 random Drive functions for multiple values of $N$ and each $\Delta$T, finding their optimal Transducer policy. The average work output over the 500 runs scaled by the amount of work extractions $\overline{W}/(N-1)$ for 20 values of $\Delta$T is shown in Figure 3.1a.

In 3.1b, we plot the average work when the system qubit is initialised in an eigenstate $\rho_0 = |+\rangle\langle+|$ of the partial system Hamiltonian acting only on drive and system $H_{DS} = \langle\psi_D|\langle\psi_T| H_I \otimes \mathbb{1}_T |\psi_D\rangle|\psi_T\rangle$. For $\Delta\mathrm{T} = 0$, the work output is $W = 1$ for all $N$. This is the amount of work that can be extracted by switching the Hamiltonian in such a way that the eigenvalues change signs. A special case occurs for $N = 2$: the optimal case is independent of $\Delta$T. Here, the maximum work output per step of $W = 1$ can be achieved by setting the transducer such that the total system Hamiltonian commutes with $H_{DS}$. $\rho_S$ remains in the eigenstate and after time $\Delta$T, $W = 1$ can be extracted from the system as with $\Delta\mathrm{T} = 0$.

For both initial system states and large $N$ and $\Delta$T, the maximum work per extraction step $\frac{\overline{W}}{N-1} = 0.5$ as it takes two steps to evolve the system into a favourable state and then be able to extract the maximum work per step $W = 1$. For smaller $\Delta$T the maximum extractable work per extraction step is not reached. In these cases the optimal system state cannot be reached due to the speed limit in unitary dynamics [3, 5].




**(a)** $\rho_0 = |0\rangle\langle 0|$ **(b)** $\rho_0 = |+\rangle\langle +|$

**Figure 3.1.:** (a) We plot the average work $\overline{W}$ over $n = 500$ runs of random excitations divided by amount of qubit changes $N-1$, with $\rho_0 = |0\rangle\langle 0|$, for multiple $N$. The error bars correspond to the standard deviation $\sigma_W = \sqrt{\frac{1}{n-1}\Sigma_i^n(\overline{W} - W_i)^2}$. (b) We plot $\overline{W}/(N-1)$ for multiple $N$ where the system state is initialised in an eigenstate of the Drive Hamiltonian $H_{DS}$.

## 3.2. $N = 2$: Learning Single Jump Optimal Control Sequences

For the simplest case of $N = 2$, we generate data sets of size $N_{\text{data}} = 20000$ for $\rho_0 = |0\rangle\langle 0|, |+\rangle\langle +|$ and random pure states. The drive qubits are sampled randomly from the Haar measure [13]. We train each data set on a fully-connected feedforward ANN with a single hidden layer with 10 neurons. The efficiency of the models is presented in table 3.1. For $N = 2$, starting in an eigenstate of the drive Hamiltonian $H_{DS}$ gives the highest model efficiency, as the optimal Transducer policy is trivial to learn and implement (see appendix A.2).

For random initial states the efficiency is close to zero. This is to be expected, as without knowledge of the system state $\rho_0$ the optimal Transducer policy cannot be determined.[1] We therefore train the same network with the random initial state as additional inputs using the same embedding as the Drive sequence. This increases the test data efficiency, but is still far below the efficiency for $\rho_0 = |+\rangle\langle +|$.

| $\rho_0$ | $\eta_{test}$ [%] |
|---|---|
| $\|0\rangle\langle 0\|$ | 72.7 |
| $\|+\rangle\langle +\|$ | 100.0 |
| Random | 0.5 |
| Random, $\rho_0$ as input | 43.0 |

**Table 3.1.:** Efficiencies $\eta$ on the test data for models with a single hidden layer with 10 neurons trained on drive protocols with $N = 2$ and differing initial states $\rho_0$.

[1] The deviation from zero is a relic of the way the test data is shuffled. For $N_{\text{data}} \to \infty$ it would disappear.

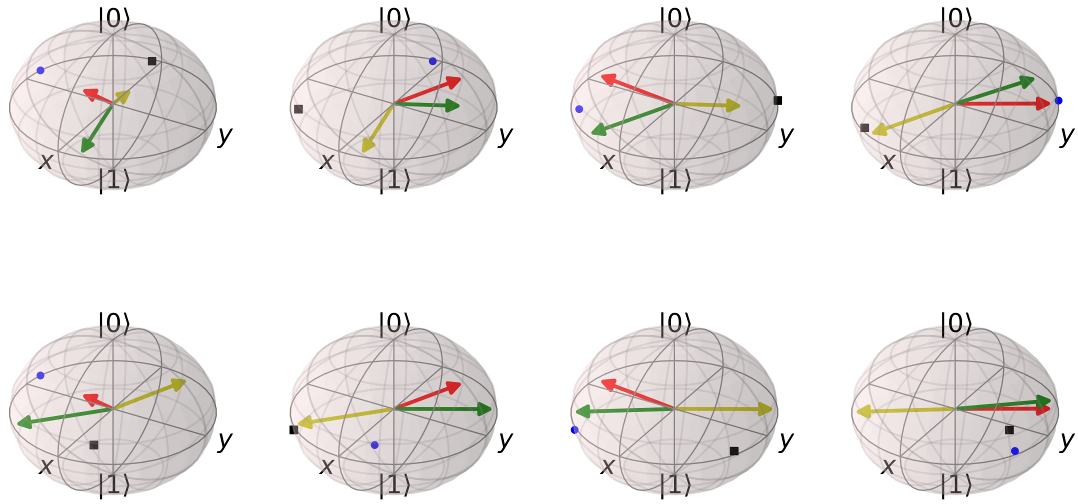
## 3.3. $N = 5, \Delta\mathrm{T} = 5$

In this section we examine the higherdimensional case of $N = 5$. As the data set with $\rho_0 = |+\rangle\langle+|$, the eigenstate of the drive Hamiltonian, performed best in the previous section, we focus our attention on this case. We compare the efficiency of a fully connected ANN (FCANN) to a unidirectional and bidirectional LSTM to analyse the effect of different architectures on the predictive power in our setting. Both LSTM networks have the same architecture, bar the bi-directionality. Before entering the LSTM cell, each embedded $|\psi_D\rangle$ passes through a two layer FCANN to increase the input dimensionality. After passing the LSTM a single layer FCANN is applied to the output to produce the embedding size to recover $|\psi_T\rangle$. The third network uses three fully connected layers, the size of which is selected to approximately match the amount of trainable parameters of the bidirectional LSTM.
We use Bayesian optimisation implemented by [2] to tune model hyperparameters. The test data efficiency as well as the amount of trainable parameters are presented in table 3.2.

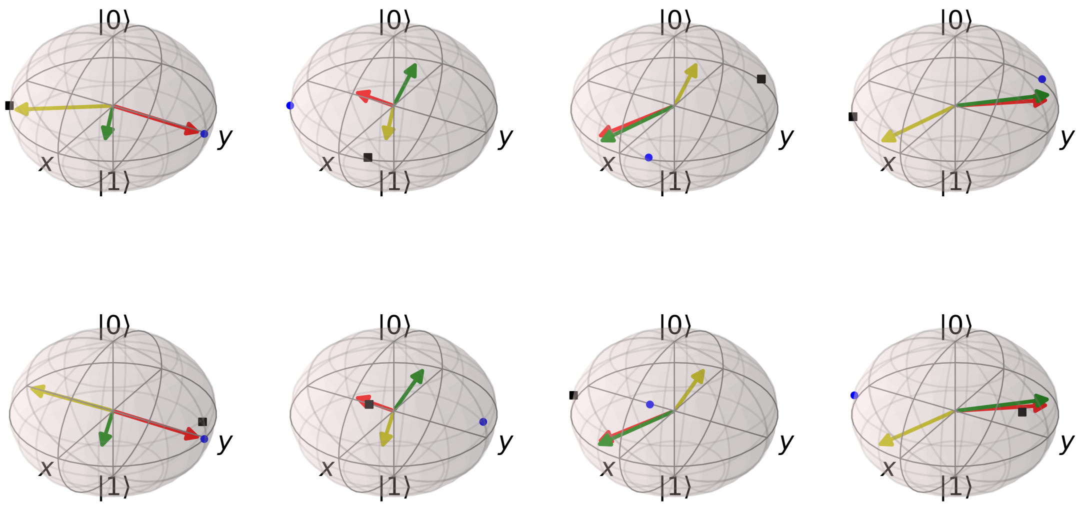
| Network Architecture | $\eta_{test}$ [%] | # Parameters |
|---|---|---|
| FCANN | 19.3 | 8,086,020 |
| Bidirectional LSTM | 33.1 | 7,700,222 |
| Unidirectional LSTM | 19.5 | 3,206,990 |

**Table 3.2.:** Efficiencies $\eta$ on the test data for model architectures with given number of trainable parameters.

The efficiency of the best model for $N = 5$ is drastically lower than that for $N = 2$. Contrary to the case of $N = 2$, the evolution of the system state becomes relevant over multiple time steps. As the work per time step is determined by $\mathrm{dW} = -\mathrm{Tr}\{\rho_S \, dH\} = \mathrm{Tr}\big\{\rho_S(H_{ST}^i - H_{ST}^{i+1})\big\}$, where $H_{ST} = \langle\psi_D|\,\langle\psi_T|\,\mathbb{1}_D \otimes H_I\,|\psi_D\rangle\,|\psi_T\rangle$ is the partial system Hamiltonian acting only between system and transducer, finding the optimal solution is a compromise of choosing $dH$ so as to maximise the expectation value while controlling the unitary evolution of $\rho_S$ such that $\mathrm{Tr}\{\sigma_z\rho_S\}$ is small. We plot the optimal and predicted trajectory of a random data point in the test in figure 3.2a set to illustrate this compromise. In 3.2b we plot the trajectories for the worst-performing sample in the test set. In this case, the prediction for $|\psi_T\rangle$ deviates only slightly from the optimum for $i \in [2, N]$ but deviates significantly for $i = 1$. This illustrates the problem of using the MSE as a cost function for training - to the network, this is a good prediction as most transducer qubits are near their optimal setting. The deviation in the first qubit leads to a completely different evolution on the system, especially as $\Delta\mathrm{T} = 5$ is large. This leads to a negative work output when following the predicted protocol.
Figure 3.4 shows the prediction of the bidirectional LSTM and optimal values for the trajectories of each data point in the test set. There is a notable difference in the predictive quality between the first four qubits and the last one: for the final qubit the dynamics after the work extraction step are inconsequential. It is therefore beneficial to maximise the strength during the final

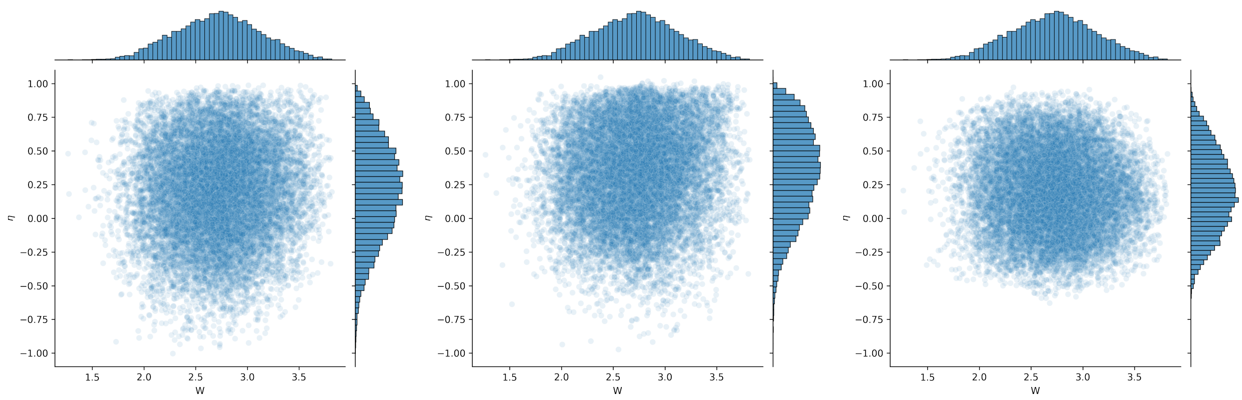



**(a)** $W_{opt} = 3.03, W_{pred} = 1.11$




**(b)** $W_{opt} = 3.06, W_{pred} = -2.65$

**Figure 3.2.: (a)** We plot the evolution of a single sample from the test set for $N = 5$ and $\Delta\mathrm{T} = 5$. For this sample we have $W_{opt} = 3.03, W_{pred} = 1.11$. Each Bloch sphere shows $\rho_S^i$ (blue dot), $\rho_S^{i+1}$ (black square), the partial system Hamiltonian acting only between system and drive $H_{DS}^i$ (red vector) as well as $H_{ST}^i$ (yellow vector) and $H_{ST}^{i+1}$ (green vector). **Top row:** we plot the system dynamics for the transducer series generated by the optimiser for $i \in [1, N-1]$. In the optimal case, $H_{ST}^i$ is chosen such that $\rho_S$ remains near the x-y-plane for all times and $\mathrm{Tr}\{\rho_S^{i+1} H_{DS}^i\}$ is large. **Bottom row:** we plot the dynamics for the same drive protocol with transducer qubits predicted by the bidirectional LSTM. The overall difference of $\rho_S$ to the x-y-plane is larger. As shown in figure 3.4, $\theta_T$ is often set to $\frac{\pi}{2}$ which maximises the strength of $H_{ST}$ as can be seen in all Bloch spheres in the bottom row. In this case, the $H_{DS}$ are chosen by the network to be antiparallel, irrespective of the current system state. **(b)** We show the same plot as in (a) for the worst performing sample from the test set to illustrate a shortcoming of the model. As can be seen from the yellow and green vectors, the predictions for $j \in [2, N]$ are very close to the optimal solutions. However, the first transducer prediction is wrong, leading to a deviation from the optimal system dynamics.




**(a)** FCANN **(b)** Bidir. LSTM **(c)** Unidir. LSTM

**Figure 3.3.:** We plot the efficiency $\eta$ of each data point in the test set over its optimal work output $W$ for the three network architectures and $\rho_0 = |+\rangle\langle+|$. The top and right plots on each graph are histograms for $W$ and $\eta$ respectively.

switching, i.e. setting $\theta_T^N = \frac{\pi}{2}$, to maximise its work output. Additionally $\phi_T^N$ is set so that $H_{ST}^N$ is antiparallel to $\rho_S^N$.

### 3.3.1. Noise resistance

Besides the efficiency of the networks the resistance of their predictions to noise is also of interest. We create a noisy sequence $\{|\phi_j\rangle\}$ of N qubits from $\{|\psi_j\rangle\}$ using

$$|\phi_j\rangle = e^{-iH_j\tau}|\psi_j\rangle, \; \forall j \in [1, N].$$

$H_j$ are randomly generated Hermitian matrices and $\tau$ is a real parameter used to control the strength of the noise. To quantify the dissimilarity between $\{|\phi_j\rangle\}$ and $\{|\psi_j\rangle\}$ we use the fidelity $F$ as defined in [16]:

$$F_{\text{run}}(\{|\phi_j\rangle\}, \{|\psi_j\rangle\}) = \prod_j F(|\phi_j\rangle, |\psi_j\rangle) = \prod_j |\langle\phi_j|\psi_j\rangle|.$$

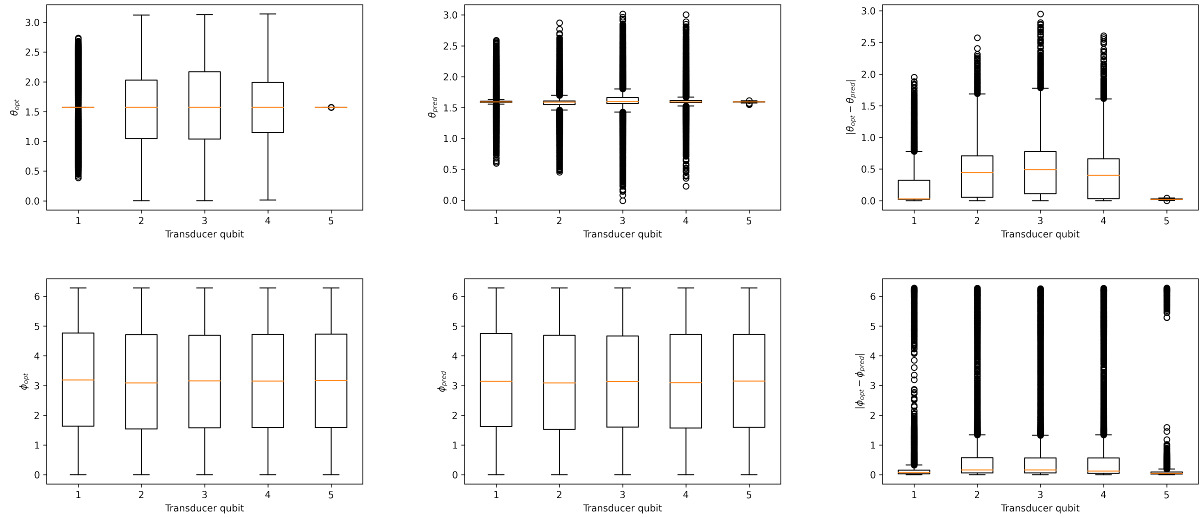



**Figure 3.4.: Top row:** for the bidirectional LSTM network, we plot boxplots of the optimal, predicted and absolute differences of $\theta_T$ for the five qubits of each trajectory in the test set. The prediction for $\theta_T^N$ is very good as the optimal solution is to set it to $\frac{\pi}{2}$ in all cases to maximise the work output of the final step. The prediction for $\theta_T^0$ is reasonably good as well, as the optimal solution is again $\frac{\pi}{2}$ in a majority of trajectories. The network is unable to predict the three central qubits, with median absolute differences of approximately 0.5. **Bottom row:** we plot the same quantities as above for the transducer azimuth $\phi_T$.

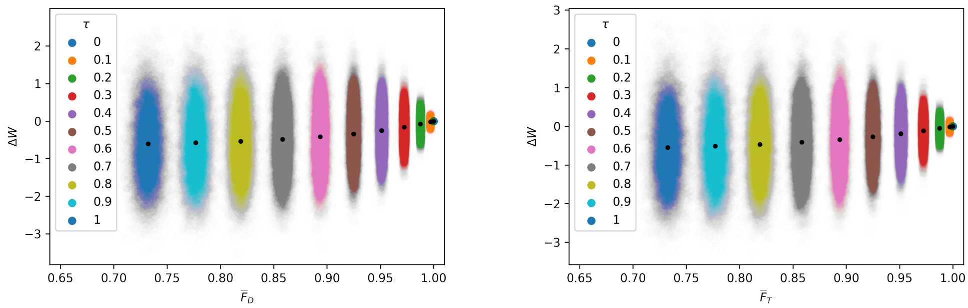



**Figure 3.5.:** $\Delta W = \overline{W}_{noise} - W_{pred}$

# 4. Summary and Outlook

# A. Derivations

## A.1. Single jump work output

$$
\begin{aligned}
H_S(\theta_D^t, \phi_D^t, \theta_T^t, \phi_T^t) &= \frac{1}{2}\left[\sin\!\left(\theta_D^t\right)e^{i\phi_D^t} + \sin\!\left(\theta_T^t\right)e^{i\phi_T^t}\right]\sigma_+ + h.c.\\
&= \alpha\sigma_+ + h.c.
\end{aligned}
$$

$$
\begin{aligned}
dH &= H_S(\theta_D^t, \phi_D^t, \theta_T^{t+1}, \phi_T^{t+1}) - H_S(\theta_D^t, \phi_D^t, \theta_T^t, \phi_T^t)\\
&= \frac{1}{2}(\sin\!\left(\theta_T^{t+1}\right)e^{i\phi_T^{t+1}} - \sin\!\left(\theta_T^t\right)e^{i\phi_T^t})\sigma_+ + h.c. =: (\tau' - \tau)\sigma_+ + h.c.
\end{aligned}
$$

$$
\begin{aligned}
U = e^{-iH_S\Delta\mathrm{T}} &= \exp\begin{pmatrix} 0 & -i\alpha^*\Delta\mathrm{T} \\ -i\alpha\Delta\mathrm{T} & 0 \end{pmatrix}\\
&= \frac{1}{2}\begin{pmatrix} \frac{\alpha^*}{|\alpha|} & -\frac{\alpha^*}{|\alpha|} \\ 1 & 1 \end{pmatrix}\begin{pmatrix} e^{-i|\alpha|\Delta\mathrm{T}} & 0 \\ 0 & e^{i|\alpha|\Delta\mathrm{T}} \end{pmatrix}\begin{pmatrix} \frac{|\alpha|}{\alpha^*} & 1 \\ -\frac{|\alpha|}{\alpha^*} & 1 \end{pmatrix}\\
&= \begin{pmatrix} \cos(|\alpha|\Delta\mathrm{T}) & -i\frac{\alpha^*}{|\alpha|}\sin(|\alpha|\Delta\mathrm{T}) \\ -i\frac{|\alpha|}{\alpha^*}\sin(|\alpha|\Delta\mathrm{T}) & \cos(|\alpha|\Delta\mathrm{T}) \end{pmatrix}
\end{aligned}
$$

With $|\psi_0\rangle = a\,|0\rangle + b\,|1\rangle$, $|a|^2 + |b|^2 = 1$, we have

$$\rho_0 = \begin{pmatrix} |a|^2 & ab^* \\ a^*b & |b|^2 \end{pmatrix},\ \rho = U\rho_0 U^\dagger = \begin{pmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{pmatrix}$$

$$\rho_{00} = |a|^2\ \cos^2(|\alpha|\Delta\mathrm{T}) + |b|^2\ \sin^2(|\alpha|\Delta\mathrm{T}) - \frac{1}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{ab^*\alpha\}$$

$$\rho_{01} = \frac{\alpha^*}{2|\alpha|}i\ \sin(2|\alpha|\Delta\mathrm{T})(|a|^2 - |b|^2) + \frac{\alpha^*}{\alpha}a^*b\sin^2(|\alpha|\Delta\mathrm{T}) + ab^*\ \cos^2(|\alpha|\Delta\mathrm{T})$$

$$\rho_{10} = \frac{|\alpha|}{2\alpha^*}i\ \sin(2|\alpha|\Delta\mathrm{T})(|b|^2 - |a|^2) + \frac{\alpha}{\alpha^*}ab^*\ \sin^2(|\alpha|\Delta\mathrm{T}) + a^*b\ \cos^2(|\alpha|\Delta\mathrm{T})$$

$$\rho_{11} = |a|^2\ \sin^2(|\alpha|\Delta\mathrm{T}) + |b|^2\ \cos^2(|\alpha|\Delta\mathrm{T}) + \frac{1}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{ab^*\alpha\}$$

$$\begin{aligned}\mathrm{dW} = -\ \mathrm{Tr}\ \rho\ \mathrm{dH} = &\frac{|a|^2 - |b|^2}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{(\tau' - \tau)\alpha^*\} \\ &- 2\ [\cos^2(|\alpha|\Delta\mathrm{T})\,\mathrm{Re}\{(\tau' - \tau)ab^*\} + \sin^2(|\alpha|\Delta\mathrm{T})\,\mathrm{Re}\left\{(\tau' - \tau)a^*b\frac{\alpha^*}{\alpha}\right\}]\end{aligned}$$

## A.2. Optimal policy for $N = 2$

For $\rho_0 = |+\rangle\langle+|$, or $a = e^{-i\phi_D}/\sqrt{2}, b = 1/\sqrt{2}$ following the notation used in appendix A.1, finding the optimal solution of dW for $N = 2$ requires finding the transducer setting that ensures

$$[H_{DS}, H_S] = [\delta\sigma_+ + \delta^*\sigma_-, \alpha\sigma_+ + \alpha^*\sigma_-] = 0 \tag{A.1}$$

with $\delta = \frac{1}{2}\sin\theta_D e^{i\phi_D}$. A.1 is equivalent to

$$\mathrm{Im}\{\alpha\delta^*\} = 0 \implies \arg\alpha = \phi_D \implies ab^* = a^*b\frac{\alpha^*}{\alpha},$$

which leads to the independence of dW with regard to $\Delta\mathrm{T}$.

# B. Training protocols

In all models we split the data into training and test data, with a test size of 18 %. The models are trained using the Adam optimiser implementation from TensorFlow, with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. The learning rate (LR) is determined by an exponential decay schedule

$$\text{LR}\,(i) = \text{initial LR} * \text{decay rate}^{i/\text{decay steps}}, \tag{B.1}$$

where $i$ denotes the optimiser step. The parameters for equation B.1 as well as the patience for the early stopping routine are listed in table B.1.

| Section | Architecture | Patience | Initial LR | Decay Rate | Decay Steps |
|---|---|---|---|---|---|
| 3.2 | [10] | 100 | $10^{-2}$ | 0.96 | 6000 |

**Table B.1.:** Hyperparameters used in training for the models in this work.

# C. Bibliography


[1] Konstantin Beyer, Kimmo Luoma, and Walter T. Strunz. Work as an external quantum observable and an operational quantum work fluctuation theorem.

[2] Lukas Biewald. Experiment tracking with weights and biases, 2020.

[3] Sebastian Deffner and Steve Campbell. Quantum speed limits: from Heisenberg's uncertainty principle to optimal quantum control. *Journal of Physics A: Mathematical and Theoretical*, 50(45):453001, Oct 2017.

[4] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In Shun-ichi Amari and Michael A. Arbib, editors, *Competition and Cooperation in Neural Nets*, pages 267–285, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

[5] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum limits to dynamical evolution. *Phys. Rev. A*, 67:052109, May 2003.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[8] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[9] Alexander LeNail. Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019.

[10] Salvatore Lorenzo, Francesco Ciccarello, and G. Massimo Palma. Composite quantum collision models. *Physical Review A*, 96(3), Sep 2017.

[11] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples, 2020.

[12] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[13] Francesco Mezzadri. How to generate random matrices from the classical compact groups. *Notices of the American Mathematical Society*, 54(5):592 – 604, May 2007.

[14] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[15] Michael A. Nielsen. *Neural Networks and Deep Learning.* Determination Press, 2015.

[16] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, USA, 10th edition, 2011.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[19] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[20] Andreas Zell. *Simulation neuronaler Netze.* Oldenbourg, München, 2., unveränd. nachdr. edition, 1997.

## Erklärung

Hiermit erkläre ich, dass ich diese Arbeit im Rahmen der Betreuung am Institut für Theoretische Physik ohne unzulässige Hilfe Dritter verfasst und alle Quellen als solche gekennzeichnet habe.

Felix Soest
Dresden, Monat 2021