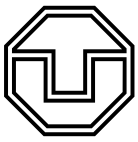


TECHNISCHE
UNIVERSITÄT
DRESDEN

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

Due to the increasing availability of data and computing power, machine learning methods have become a standard approach in many fields such as natural language processing [22]. Additionally, these methods are being applied to a broad range of problems in the physical sciences, from statistical physics to quantum computing [3, 24].
Machine learning has also found use in the field of energy harvesting, concerned with extracting energy from external excitations, e.g. vibrations in human motion [11]. In this work we extend the concept to the quantum case. We examine a system that can be driven by an arbitrary excitation, modelled as a time-dependent system Hamiltonian. The Transducer is modelled in a similar fashion and can extract work from the system following the framework in [1].
The remainder of this work is structured as follows. In Chapter 2 we review the collision model framework used in our approach and introduce two machine learning architectures. We investigate the system under consideration in Section 3.1 and apply the aforementioned architectures in Sections 3.2 to 3.4. We summarise our findings and provide an outlook in Chapter 4.

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 [12]. After a short interaction time Δt, 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 ρ_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 ρ_S under H_{AS} acting on the combined system is

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

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, ρ_S remains pure in the limit of $\Delta t \to 0$. This framework allows the experimenter to create system Hamiltonians with arbitrary time dependence by varying the initial ancilla states. In this fashion one can create piece-wise constant Hamiltonians by initialising $n = \frac{\Delta \mathrm{T}}{\Delta t}$ identical ancillas, where ΔT is the time for which the Hamiltonian remains constant.

We use this approach as the implementation for 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 (θ_D, ϕ_D) and (θ_T, ϕ_T) respectively (see Figure 2.1), the system qubit is initialised in a pure state. The reason behind using PWC functions instead of continuous is that computational complexity is greatly reduced.

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 ΔT 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 ρ_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

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

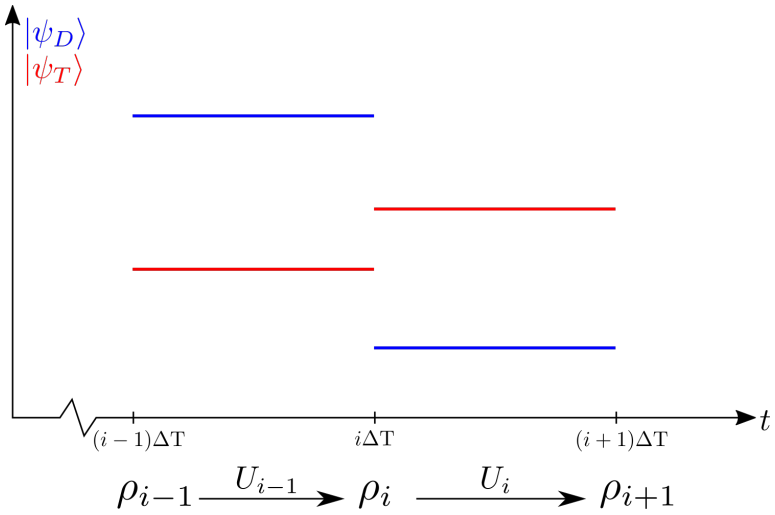


Figure 2.1.: Piecewise constant implementation of Drive and Transducer qubits: the vertical axis an arbitrary parameter of the ancilla states. The qubit states are switched instantaneously and then kept constant for ΔT while ρ_S evolves unitarily.

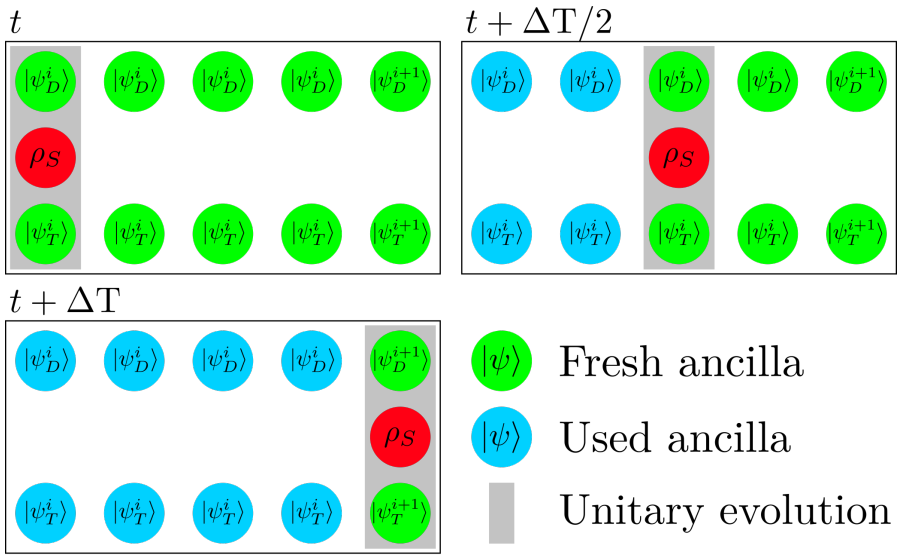


Figure 2.2.: Collision model used in this work: Drive and Transducer are series of qubits that interact once with the system and evolve the reduced density operator ρ_S. The qubit configuration can be changed in intervals of ΔT.

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.' [16] 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 [13]. Let $\mathfrak{N}$ 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, $\mathfrak{N} : \mathbb{R}^{n_1} \to \mathbb{R}^{n_L}$. n_1 and n_L denote the dimensionality of the input and output respectively. $\mathfrak{N}$ 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 [25]:

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

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 [14, 8] 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 [20, 5].

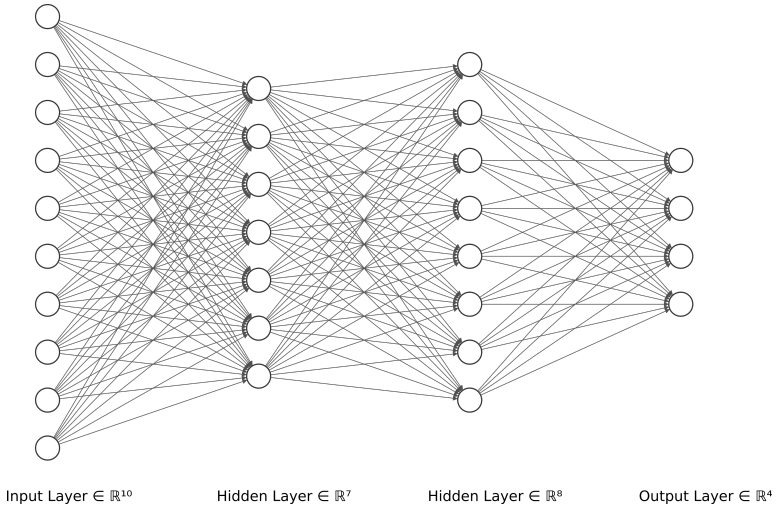


Figure 2.3.: Example fully-connected feedforward ANN with four layers, including input, output and two hidden layers [10].

Here we use the LSTM architecture, a type of recurrent neural network (RNN) introduced in [7]. 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 one or multiple rows of LSTM cells, each of 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 gates of each cell are computed using the following equations

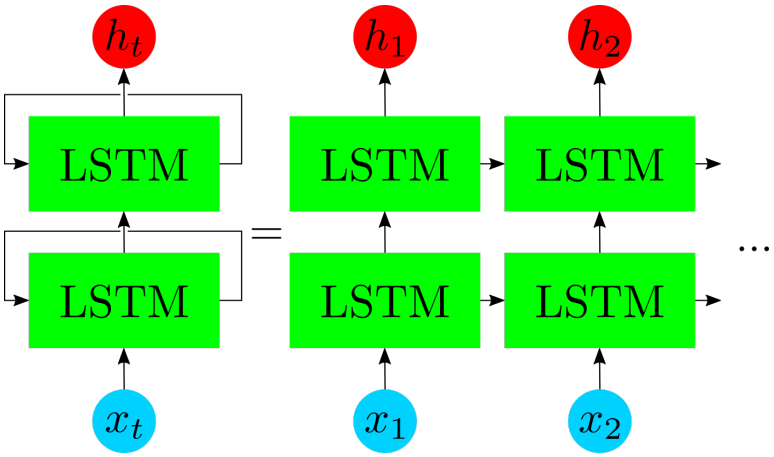


Figure 2.4.: Example two-layer RNN with LSTM architecture. Each row of LSTM blocks has the same parameters and information is fed into the network sequentially.

[19]:

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

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.
In some time-series problems data from a future time step might be required for the current prediction. In these cases, bidirectional RNNs can provide a better performance. First introduced in [21], the network is split into a forward and backward part where the input data is injected in normal and reversed order 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 [20, 17].

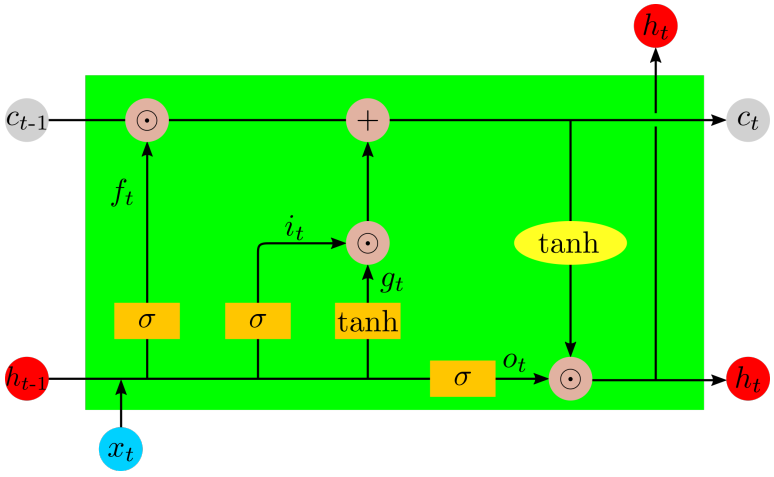


Figure 2.5.: Visualisation of a single LSTM cell. The input x_t enters the cell in the bottom left and is concatenated with the output $h_{t\text{-}1}$ of the previous cell. The combined data then enters multiple single-layer neural networks represented by orange rectangles with their respective activation 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 [23], 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 [9]. Additionally it encodes information regarding the periodicity of the qubit angle representation. We add either a zero or a one as an extra input parameter for LSTM training depending on whether or not the input is the last in a sequence.
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 the only choice, and perhaps not the obvious. Directly using the work extracted is possibly a more intuitive choice, but we refrain from it in the first section 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, as calculating the work is $O(N-1)$ while MSE $= O(1)$.

3.1. Influence of ΔT on Work Output

We start our investigation by determining the work output W when varying the time between qubit switching ΔT. 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)

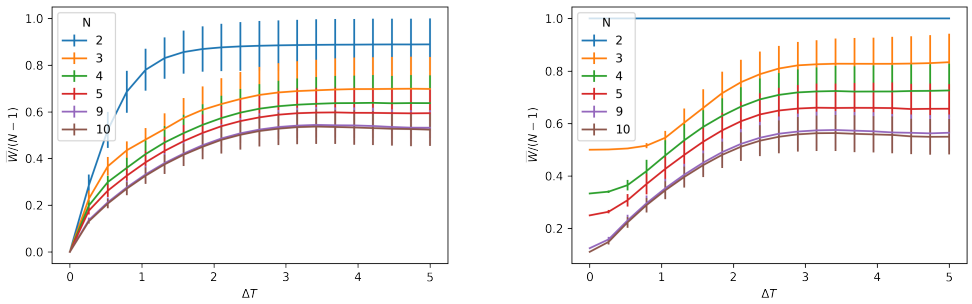
$$W = \frac{1}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{(\tau' - \tau)\alpha^*\} \tag{3.2}$$

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

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 ΔT, 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 ΔT 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 ΔT. 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}. ρ_S remains in the eigenstate and after time ΔT, $W = 1$ can be extracted from the system as with $\Delta\mathrm{T} = 0$.

For both initial system states and large N and ΔT, 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 ΔT 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 [4, 6].



(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 [15]. 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 ρ_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 $\rho_0 = |+\rangle\langle +|$.

ρ_0	η_{test} [%]
$\|0\rangle\langle 0\|$	72.7
$\|+\rangle\langle +\|$	100.0
Random	0.5
Random, ρ_0 as input	43.0

Table 3.1.: Efficiencies η 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 ρ_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. $N = 5$

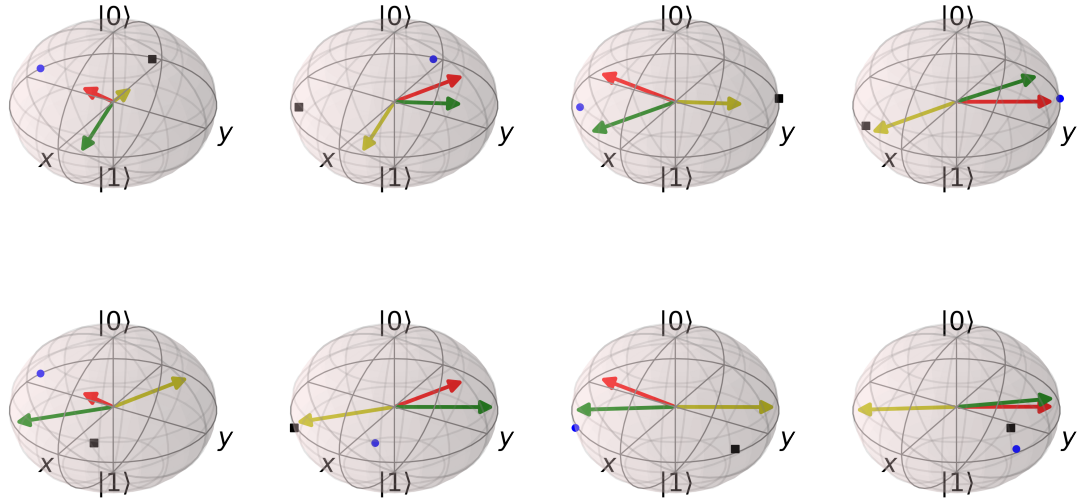
3.3.1. $\Delta T = 5$

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 hidden 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.

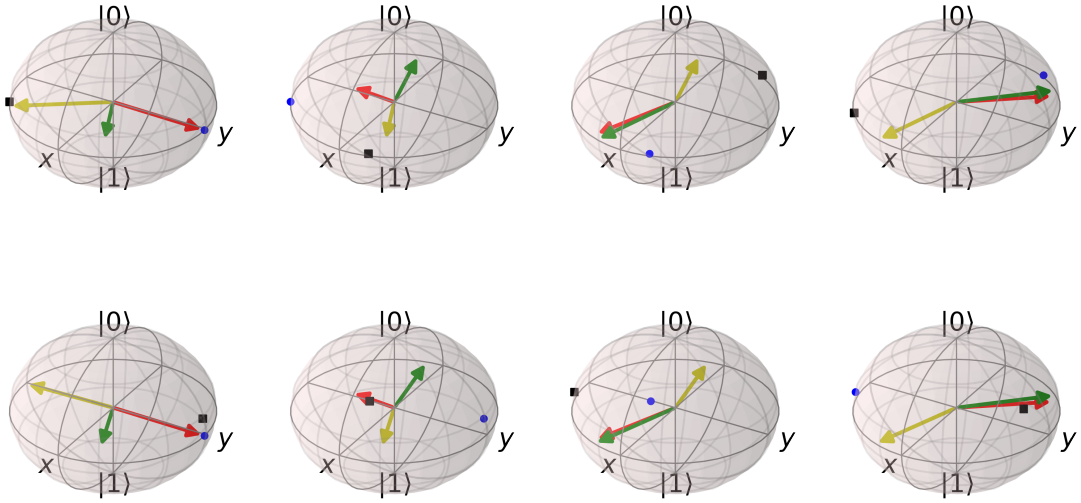
Network Architecture	η_{test} [%]	MSE_{test}	# Parameters
FCANN	19.3	0.1083	8,086,020
Bidir. LSTM	33.1	0.0948	7,700,222
Unidir. LSTM	19.5	0.1707	3,206,990

Table 3.2.: Efficiencies η 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 $\text{dW} = -\text{Tr}\{\rho_S\, dH\} = \text{Tr}\{\rho_S(H^i_{ST} - H^{i+1}_{ST})\}$, 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 ρ_S such that $\text{Tr}\{\sigma_z\rho_S\}$ 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 causes a completely different evolution on the system, especially as $\Delta T = 5$ 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

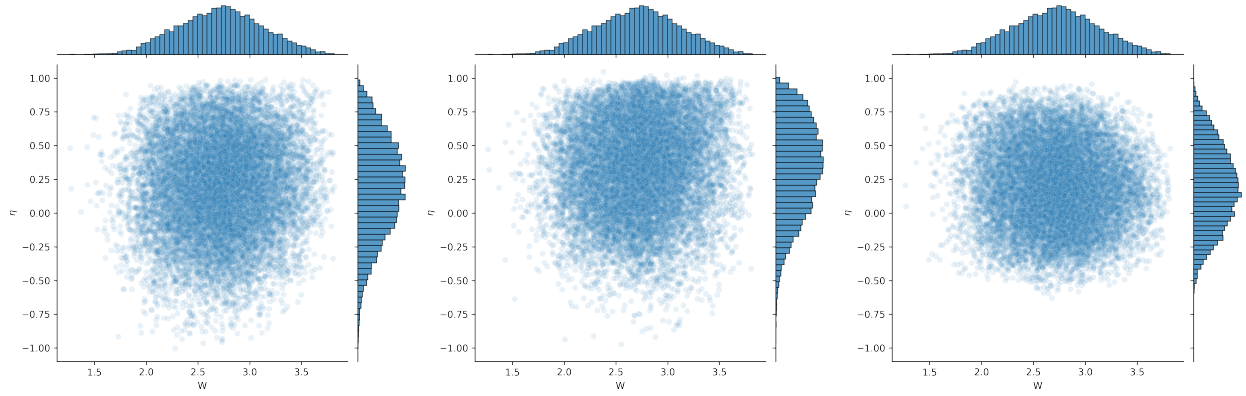


(a) $W_{opt} = 3.03, W_{pred} = 1.11$



(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 $\Delta T = 5$. For this sample we have $W_{opt} = 3.03, W_{pred} = 1.11$. Each Bloch sphere shows ρ_S^i (blue dot), ρ_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 ρ_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 ρ_S to the x-y-plane is larger. As shown in Figure 3.4, θ_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 η 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 η respectively.

step are inconsequential. It is therefore beneficial to maximise the strength during the final switching, i.e. setting $\theta_T^N = \frac{\pi}{2}$, to maximise its work output. Additionally ϕ_T^N is should be set so that H_{ST}^N is antiparallel to ρ_S^N on the Bloch sphere. The LSTM performance on this parameter indicates that the network does learn a representation of the system state ρ_S.

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 τ 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 [18]:

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

3.3.2. $\Delta T = 1$

We apply the methodology of Section 3.3.1 to the case of $\Delta T = 1$ with 5 Drive and Transducer qubits. Contrary to $\Delta T = 5$, the optimal System states cannot be reached, leading to a lower total work output following the protocol determined by the optimiser (Figure 3.1).
We train the LSTM networks from Section 3.3.1 as well a network with a larger cell state dimension to match the amount of trainable parameters of the Bidirectional LSTM. The

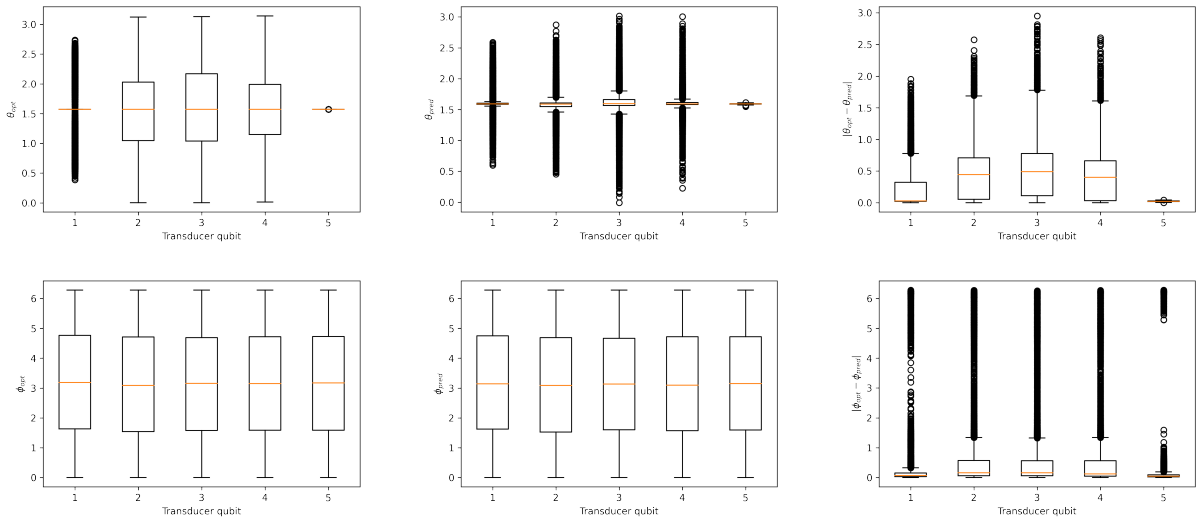
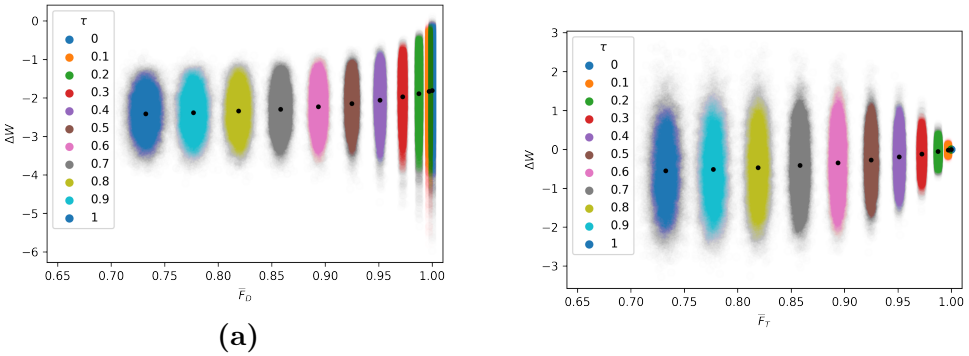


Figure 3.4.: Top row: for the bidirectional LSTM network, we plot boxplots of the optimal, predicted and absolute differences of θ_T for the five qubits of each trajectory in the test set. The prediction for θ_T^N is very good as the optimal solution is to set it to $\frac{\pi}{2}$ in all cases to maximise the work output of the final step. The prediction for θ_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 ϕ_T.



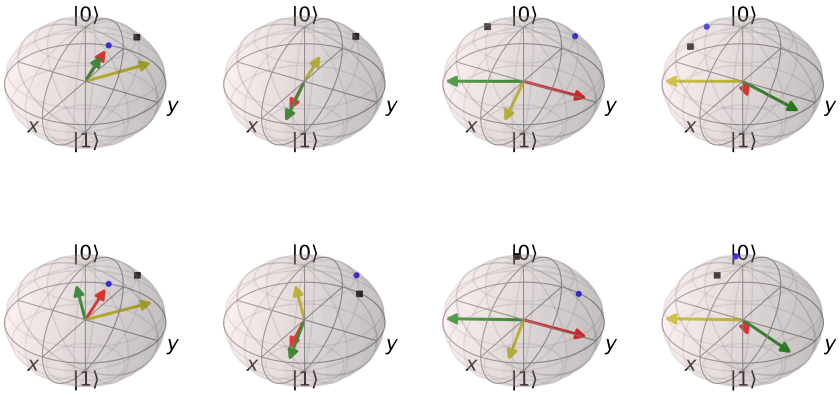
(a)

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

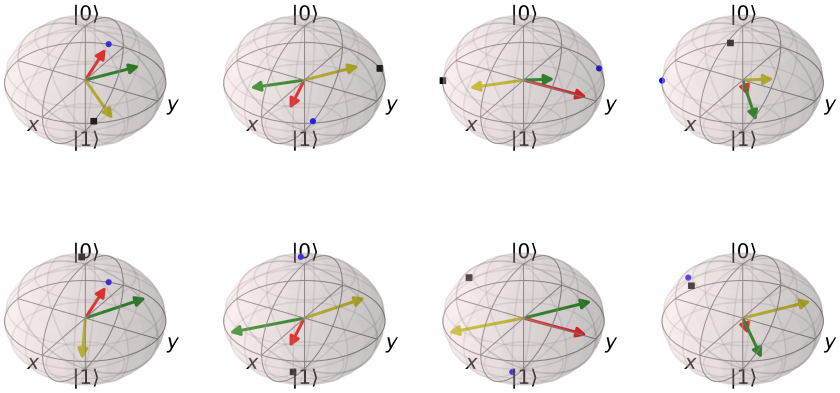
performance of each network is listed in Table 3.3. We find that all networks perform better than their $\Delta T = 5$ counterparts.

Network Architecture	η_{test} [%]	MSE_{test}	# Parameters
Bidir. LSTM	97.4	0.0238	7,700,222
Unidir. LSTM	70.1	0.0638	3,206,990
Unidir. LSTM, higher cell state dimension	69.9	0.0637	7,704,062

Table 3.3.: Efficiencies η and MSE loss on the test set for differing model architectures for $N = 5, \Delta T = 1$.



(a) $\Delta T = 1 : W_{opt} = 1.40, W_{pred} = 1.32$



(b) $\Delta T = 5 : W_{opt} = 2.42, W_{pred} = 0.57$

3.4. Extracted work as cost function

In Section 3.3.1 we encountered a conceptional problem regarding using the MSE as a loss function, where a low MSE score does not necessarily indicate a predicted protocol will extract a large amount of work. In the following we aim to investigate whether directly using extracted work W as the network cost function can improve performance.

4. Summary and Outlook

A. Derivations

A.1. Single jump work output

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

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

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

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

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

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

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

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

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

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

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 ΔT.

B. Training protocols

In all models we split the data into training, validation and test data with $p_{test} = 18\%$ and $p_{valid} = 8.2\%$. The models are implemented using the PyTorch library [19]. Training is performed over n epochs, in each of which the complete training set is used once in batches of *batch size*. In each batch, the gradient of the loss function with regard to the model parameters is calculated. The gradient average over the data points in a batch are used to update the trainable parameters. We use early stopping to stop training when the value of the cost function on the validation set does not improve for *patience* epochs.

B.1. Hyperparameters section 3.2

For $N = 2$, we use the Adam optimiser with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. The learning rate (LR) is determined by an exponential decay schedule

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

where i denotes the optimiser step.

Patience	Initial LR	Decay Rate
100	10^{-2}	0.995

Table B.1.: Hyperparameters used in training for the models in section 3.2.

B.2. Hyperparameters 3.3.1

The learning rate (LR) is determined by the 'Reduce LR on Plateau' schedule. The learning rate is multiplied by the hyperparameter *factor* when the cost function does not improve for *patience / pat drop* epochs. The hyperparameters are optimised using Bayesian optimisation and can be found in Table B.2.

Hyperparameter	Value
Optimiser	SGD
Batch size	44
Dropout	0.3179732914255167
Input layer 1 size	793
Input layer 2 size	488
Output layer size	228
LSTM hidden size	324
Initial learning rate	0.02847560288866327
LSTM layers	3
Pat drop	3.698498258242224
Patience	55
Factor	0.24509238889070978

Table B.2.: Hyperparameters used for the models in section 3.3.1.

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] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4), Dec 2019.

[4] 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.

[5] 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.

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

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

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

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

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

[11] Feiyang Liu, Yulong Zhang, Oscar Dahlsten, and Fei Wang. Intelligently chosen interventions have potential to outperform the diode bridge in power conditioning. *Scientific Reports*, 9(1):8994, Jun 2019.

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

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

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

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

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

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

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

[19] 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.

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

[21] Mike Schuster and Kuldip Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45:2673 – 2681, 12 1997.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[23] 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.

[24] David F. Wise, John J. L. Morton, and Siddharth Dhomkar. Using deep learning to understand and mitigate the qubit noise environment, 2021.

[25] 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