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 **1**

1.1. Intro 1

2. Background **3**

2.1. Supervised Machine Learning 3

2.1.1. Fully-connected feedforward ANNs 3

2.1.2. Long Short-Term Memory 4

2.2. Collision model dynamics 5

2.3. Setting 5

3. Experimental Results **9**

3.1. Dependence of ΔT on Work Output 9

3.2. $N = 2$: Learning Single Jump Optimal Control Sequences 11

4. Summary and Outlook **13**

A. Derivations **15**

A.1. Single jump work output 15

A.2. Optimal ? policy for $N = 2$ 16

B. Training protocols **17**

C. Bibliography **19**

1. Introduction

1.1. Intro

Test 123

2. Background

2.1. 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.’ [8] 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 the following sections we review two ANN architectures, the fully-connected feedforward ANN and the Long Short-Term Memory (LSTM) network.

2.1.1. Fully-connected feedforward ANNs

In this section we review ANNs, following the exposition given in [6].

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 2.1). 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. The activations can then be calculated using the following formulae [12]:

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

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 [7, 2] that the rectified linear unit $\mathrm{ReLU}(x) = \max(0, x)$ often provides better results and is used here.

2.1.2. Long Short-Term Memory

While the network architecture introduced in the previous section performs reasonably well on many problems, it destroys
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, where $\{\vec{x}_i\}$ is the input and $\{\vec{y}_i\}$ the output data, and $\vec{a}_{L,i} = \mathfrak{N}(\vec{x}_i)$ is 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 [10, 9].

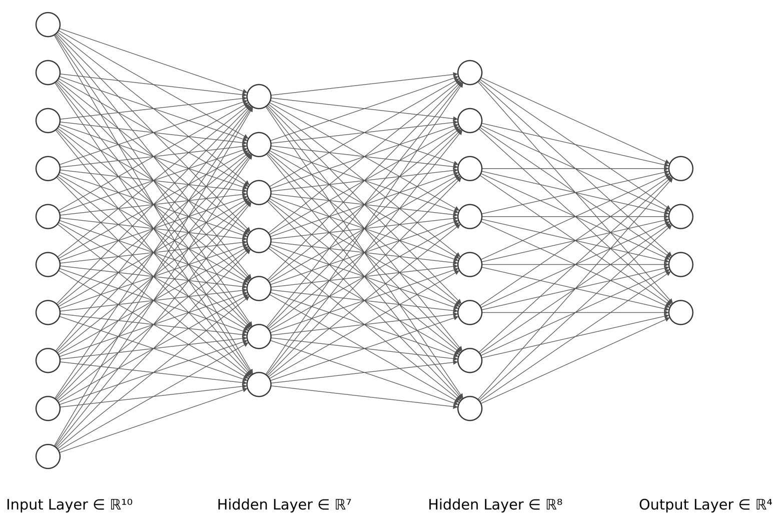


Figure 2.1.: Example fully-connected feedforward ANN with four layers, including input, output and two hidden layers [4]

.

2.2. 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 [5]. 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 ψ_A be 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:

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

2.3. Setting

Our setting 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 2.2), the system qubit is initialised in a pure state. As Drive and Transducer are assumed to be piecewise

constant and therefore pure, we model both qubits as ancillary systems introduced in section 2.2 (see figure 2.3).
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[1]:

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

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

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

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

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.

[1]It is important to make the distinction between ΔT and Δt introduced in section 2.2. Δt is the collision time of a single qubit while ΔT is the time for which the qubits have the same initial state. For each time step i we therefore have $n = \Delta \mathrm{T}/\Delta t$ qubits initialised in the same state.

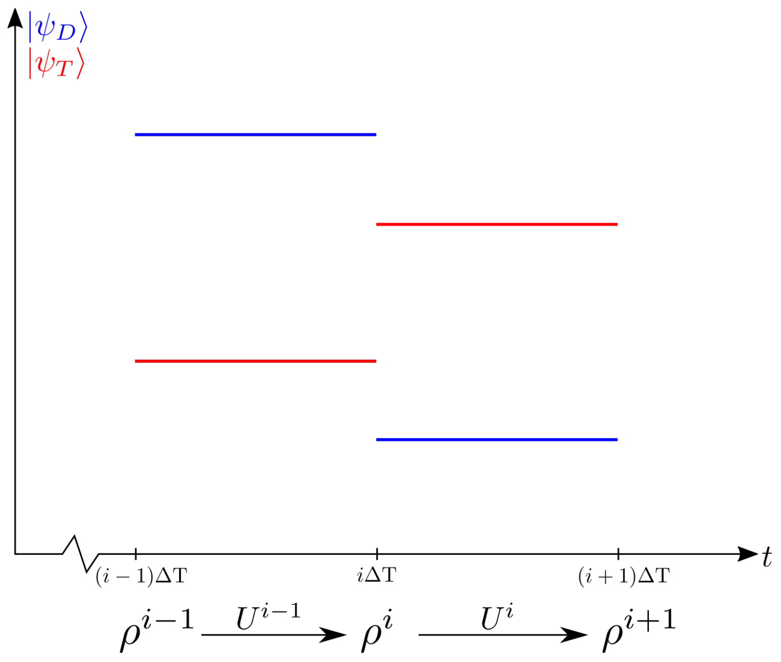


Figure 2.2.: Piecewise constant implementation of Drive and Transducer qubits: the vertical axis shows qubit state in arbitrary units. The qubit states are switched instantaneously and then kept constant for ΔT while ρ_S evolves unitarily.

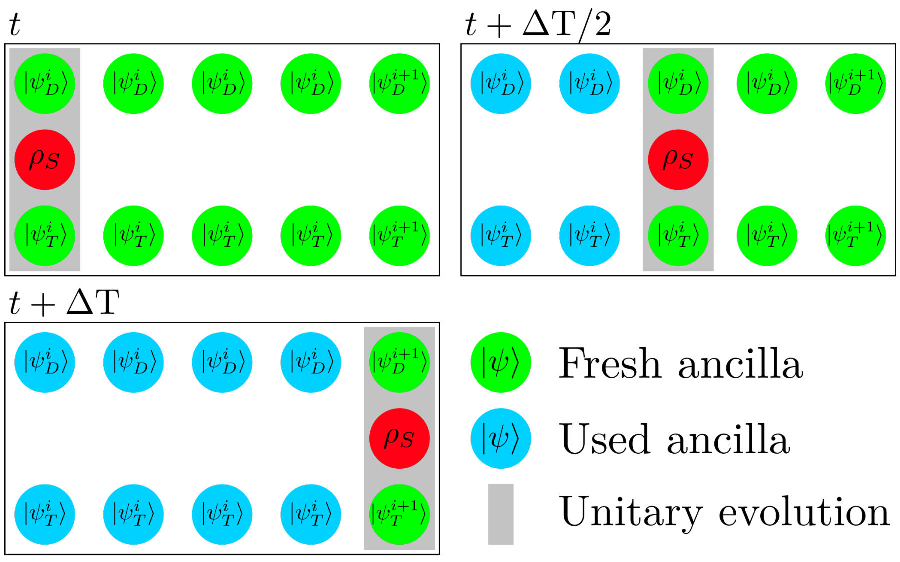


Figure 2.3.: Collision model used in this work: Drive and Transducer are series of qubits interact once with the system and evolve the reduced density operator ρ_S. The qubit configuration can be changed in intervals of ΔT.

3. Experimental Results

The training data is created using a minimisation algorithm [11], 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 [3]. 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 2.1. 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.

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

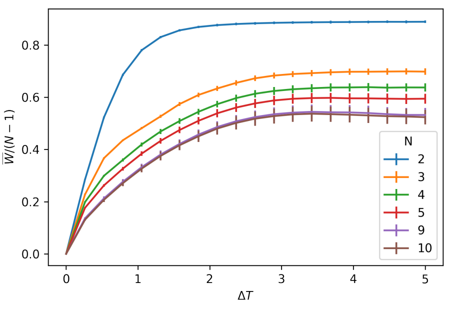
We start our investigation by determining the work output W when varying the time between qubit switching $\Delta\mathrm{T}$. If the System qubit is initialised in the pure state $\rho_S = |0\rangle\langle 0|$, the work output for a single jump is given by (see appendix A.1 for a derivation)

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

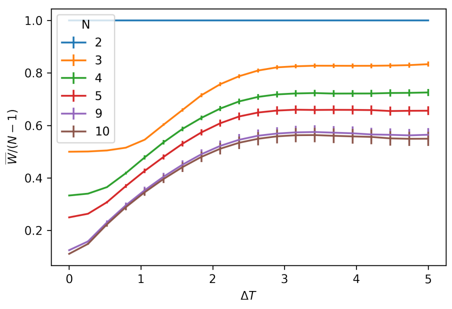
$$\alpha = \frac{1}{2}\left[\sin\!\left(\theta_D^1\right)e^{i\phi_D^1} + \sin\!\left(\theta_T^1\right)e^{i\phi_T^1}\right], \qquad \tau' - \tau = \frac{1}{2}\left[\sin\!\left(\theta_T^2\right)e^{i\phi_T^2} - \sin\!\left(\theta_T^1\right)e^{i\phi_T^1}\right].$$

We note that for $\Delta\mathrm{T} \to 0, W \to 0$ as ρ_S is orthogonal to H_S for all configurations of $|\psi_D\rangle$ and

$|\psi_T\rangle$. We simulate 500 random Drive functions for multiple values of N and each $\Delta\mathrm{T}$, 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 $\Delta\mathrm{T}$ is shown in figure 3.1a.
In 3.1b, we plot the average work when the system qubit is initialised in an eigenstate of partial system Hamiltonian acting only between drive and system $H_{DS} = \langle\psi_D| \langle\psi_T| H_I \otimes \mathbb{1}_T |\psi_D\rangle |\psi_T\rangle$, $\rho_0 = |+\rangle \langle+|$. 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 $\Delta\mathrm{T}$. 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}. ρ_S remains in the eigenstate and after time $\Delta\mathrm{T}$, $W = 1$ can be extracted from the system as with $\Delta\mathrm{T} = 0$.
For both initial system states, the



(a) $\rho_0 = |0\rangle\langle 0|$



(b) $\rho_0 = |+\rangle\langle +|$

Figure 3.1.: (a) We plot the average work $\overline{W}$ over 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 $\sigma_{\overline{W}}$. (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. 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.

4. Summary and Outlook

A. Derivations

A.1. Single jump work output

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

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

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

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

$$\begin{aligned}
\rho_0 &= \begin{pmatrix} |a|^2 & ab^* \\ a^*b & |b|^2 \end{pmatrix}, \ \rho = U\rho_0 U^\dagger = \begin{pmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{pmatrix}, \\
\rho_{00} &= |a|^2 \ \cos^2(|\alpha|\Delta\mathrm{T}) + |b|^2 \ \sin^2(|\alpha|\Delta\mathrm{T}) - \frac{1}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{ab^*\alpha\}, \\
\rho_{01} &= \frac{\alpha^*}{2|\alpha|} i \ \sin(2|\alpha|\Delta\mathrm{T})(|a|^2 - |b|^2) + \frac{\alpha^*}{\alpha} a^* b \sin^2(|\alpha|\Delta\mathrm{T}) + ab^* \ \cos^2(|\alpha|\Delta\mathrm{T}), \\
\rho_{10} &= \frac{|\alpha|}{2\alpha^*} i \ \sin(2|\alpha|\Delta\mathrm{T})(|b|^2 - |a|^2) + \frac{\alpha}{\alpha^*} ab^* \ \sin^2(|\alpha|\Delta\mathrm{T}) + a^* b \ \cos^2(|\alpha|\Delta\mathrm{T}), \\
\rho_{11} &= |a|^2 \ \sin^2(|\alpha|\Delta\mathrm{T}) + |b|^2 \ \cos^2(|\alpha|\Delta\mathrm{T}) + \frac{1}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{ab^*\alpha\}.
\end{aligned}$$

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

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 $N = 2$ is equivalent to solving $ab^* = a^* b \frac{\alpha^*}{\alpha}$. This leads to the optimality condition

$$\arg\alpha = \phi_D, \tag{A.1}$$

considering only the principal branch. This ensures $[H_{DS}, H_S] = 0$ and leads to the independence of the work output 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] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

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

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

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

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

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

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

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

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

[11] 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.

[12] 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