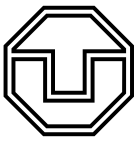


TECHNISCHE
UNIVERSITÄT
DRESDEN

Learning quantum work extraction protocols with recurrent neural networks

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 08. Februar 2021

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

Abstract

An interesting question in the field of quantum thermodynamics is how to extract the maximum amount of work from a given system. We consider a two-state quantum system that is driven by an arbitrary piecewise constant function, from which work can be extracted. We show that for this system a policy exists that is able to extract a non-negative work output independent of the excitation. We investigate the system with respect to randomly sampled drives and find their optimal extraction policies numerically. Using these pairs as training data we employ recurrent neural networks to predict the optimal transducer policy given a drive sequence. We find that our approach performs well for certain model parameters, capable of extracting more than 97 % of optimum. Additionally, we explore the robustness of the model predictions with respect to noise and the ability of our models to generalise to longer drive sequences.

Zusammenfassung

Eine interessante Frage im Gebiet der Quanten-Thermodynamik ist, wie man die maximale Menge an Arbeit aus einem gegebenen System extrahieren kann. Wir betrachten ein zwei-Niveau-Quantensystem, welches durch eine beliebige stückweise konstante Funktion angetrieben wird und aus dem Arbeit extrahiert werden kann. Wir zeigen, dass für dieses System eine Strategie existiert, die in der Lage ist, eine nicht-negative Arbeit unabhängig von der Anregung zu extrahieren. Wir untersuchen das System in Bezug auf zufällig generierte Antriebe und finden deren optimale Extraktionssequenz numerisch. Unter Verwendung dieser Paare als Trainingsdaten verwenden wir rekurrente neuronale Netze, um die optimale Extraktionsstrategie einer gegebenen Antriebssequenz vorherzusagen. Für bestimmte Modellparameter funktioniert dieser Ansatz gut und ist in der Lage, mehr als 97 % des Optimums zu extrahieren. Zusätzlich untersuchen wir die Robustheit der Modellvorhersagen in Bezug auf Rauschen und die Fähigkeit unserer Modelle, auf längere Antriebssequenzen zu generalisieren.

Contents

1. Introduction

Thermodynamics has been a central field of interest in physics ever since its inception in the 19th century [1]. It was in fact a thermodynamic insight which led Einstein to conclude that electromagnetic radiation is quantised [2]. These fields have benn combined to study the thermodynamic properties of quantum systems. We focus on work extraction, which is often modelled as a system coupled to a heat bath, and bounds for the extractable work exist [3]. We take a different approach, following the framework given in [4], as it allows us to model a system explicitly driven by a quantum system. The driving is modelled as a time-dependent system Hamiltonian. The transducer is modelled in a similar fashion and can extract work from the system.

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 [5]. Additionally, these methods are being applied to a broad range of problems in the physical sciences, from statistical physics to quantum computing [6, 7].

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 [8]. In this work we extend this concept to the quantum case. We train recurrent neural networks to predict transducer protocols given a drive sequence, considering both the case where the sequence is known beforehand and the one where it is not.

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. In Chapter 3 we show that for a given drive sequence a lower bound on the expectation value of the extractable work exists. We investigate the system under consideration in Section 4.1 and apply the aforementioned architectures in Sections 4.3 to 4.4.4. The robustness of our solutions to noise as well as the ability of the models to generalise to longer drive sequences are studied thereafter. We summarise our findings and provide an outlook in Chapter 5.

We use units where $\hbar = 1$.

2. Background

2.1. Collision model dynamics

A standard approach to modelling 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 [9]. After a short interaction time Δt, the ancilla systems are traced out to receive the system state. We use this approach to model our setting, which consists of three qubits: the drive ($|\psi_D\rangle$), system and transducer ($|\psi_T\rangle$) qubits. In general, a collision model approach leads to entanglement between system and ancilla. This is undesirable in our model as we assume the drive and transducer to be in definitive, i.e. pure, states. To ensure the ancilla and system states remain pure, we follow the approach used in [4]: let ρ_S be the density operator of the system under consideration and $|\psi_A\rangle = |\psi_D\rangle \otimes |\psi_T\rangle$ the state of the ancilla system. Given $\Delta t \ll 1$ the time evolution of ρ_S under H_{AS} acting on the 3 qubit Hilbert space 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 Eq. (2.1) leads to von-Neumann dynamics on the system for an infinite stream of qubits initialised in the same state $|\psi_A\rangle$:

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

As each ancilla only interacts once with the system and is traced out afterwards, ρ_S remains pure in the limit of $\Delta t \to 0$.
We focus on piece-wise constant (PWC) functions of the drive ($|\psi_D\rangle$) and transducer ($|\psi_T\rangle$) qubits, as it greatly reduces computational complexity. These are implemented by keeping the stream of ancillas $|\psi_A\rangle = |\psi_D\rangle \otimes |\psi_T\rangle$ constant for time ΔT (Figure 2.1).
The drive and transducer qubits can be set in N discrete intervals represented by (θ_D^n, ϕ_D^n) and (θ_T^n, ϕ_T^n) respectively (see Figure 2.2), the system qubit is initialised in a pure state ρ_S.
In the remainder of this work we use the interaction Hamiltonian on the three qubit Hilbert

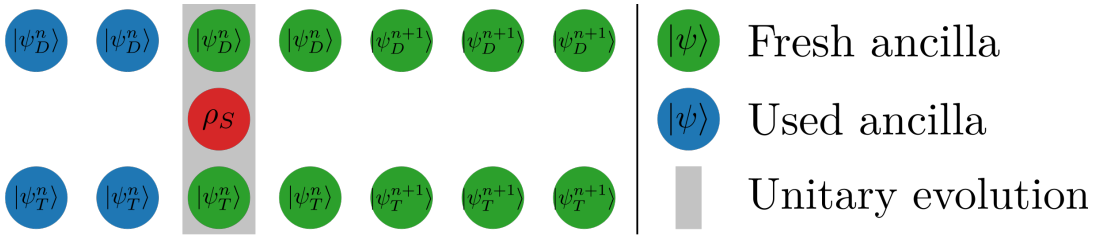


Figure 2.1.: Collision model used in this work: drive and transducer are series of qubits that interact once with the system and evolve the reduced density operator ρ_S. The qubit configuration can be changed in intervals of ΔT.

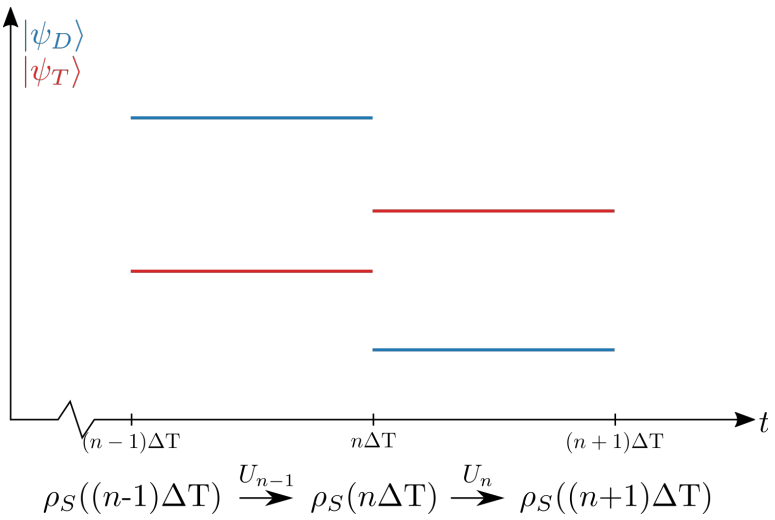


Figure 2.2.: Piecewise constant implementation of drive and transducer qubits: the vertical axis represents an arbitrary parameter of the ancilla states. The qubit states are switched instantaneously and then kept constant for ΔT while ρ_S evolves unitarily. The piece-wise constant drive and transducer settings lead to a piece-wise constant system Hamiltonian.

space

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

Note that the interaction Hamiltonian itself is constant and the time dependence comes from the change in the stream of ancillas only. The energy scale of the Hamiltonian is therefore limited and so are the dynamics of the system state ρ_S. The time evolution and work extraction is then calculated as follows, where ΔT is the time span between qubit switching:

$$H_S^n = \langle\psi_D^n| \langle\psi_T^n| H_{DST} |\psi_D^n\rangle |\psi_T^n\rangle , \tag{2.2}$$
$$\rho_S((n+1)\Delta\text{T}) = U_n \; \rho_S(n\Delta\text{T}) \; U_n^\dagger,$$
$$U_n = e^{-iH_S^n \Delta\text{T}},$$
$$W = \Sigma_n dW_n = -\Sigma_n \, \text{Tr}\{\rho_S(n\Delta\text{T}) \; \Delta H_S^n\}, \tag{2.3}$$
$$\Delta H_S^n = \langle\psi_D^n| \left\langle\psi_T^{n+1}\right| H_{DST} |\psi_D^n\rangle \left|\psi_T^{n+1}\right\rangle - \langle\psi_D^n| \langle\psi_T^n| H_{DST} |\psi_D^n\rangle |\psi_T^n\rangle . \tag{2.4}$$

Here we use the partial Hamiltonian H_S^n on the system at time step $n \in [1, N-1]$, as well as the corresponding system density matrix $\rho_S(n\Delta\mathrm{T})$. An advantage of this framework is that the extracted work is available to the experimenter as a quantum observable [4].
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 Eq. (2.2) to

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

In the following, we will call H_DS the drive and H_ST the transducer Hamiltonian, but note that both Hamiltonians act on the system state ρ_S only. Eq. (2.4) can then be expressed in a simpler form

$$\Delta H_S^n = H_{ST}^{n+1} - H_{ST}^n. \tag{2.5}$$

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.' [10] 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 a drive and its respective optimal transducer policy. Multiple algorithms to find such a mapping exist, however for high dimensional problems artificial neural networks (ANNs) are often used. In general, ANNs are a collection of neurons, usually grouped in layers, which are connected and can transmit signals to each other. The ANN can learn by changing how 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 ANN

In this section we review ANNs, following the exposition given in [11]. 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 $\vec{b}_l \in \mathbb{R}^{n_{l+1}}$ called weights and biases respectively. 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 [13, 14] that the rectified linear unit $\mathrm{ReLU}(x) = \max(0, x)$ often provides better results and is used here.

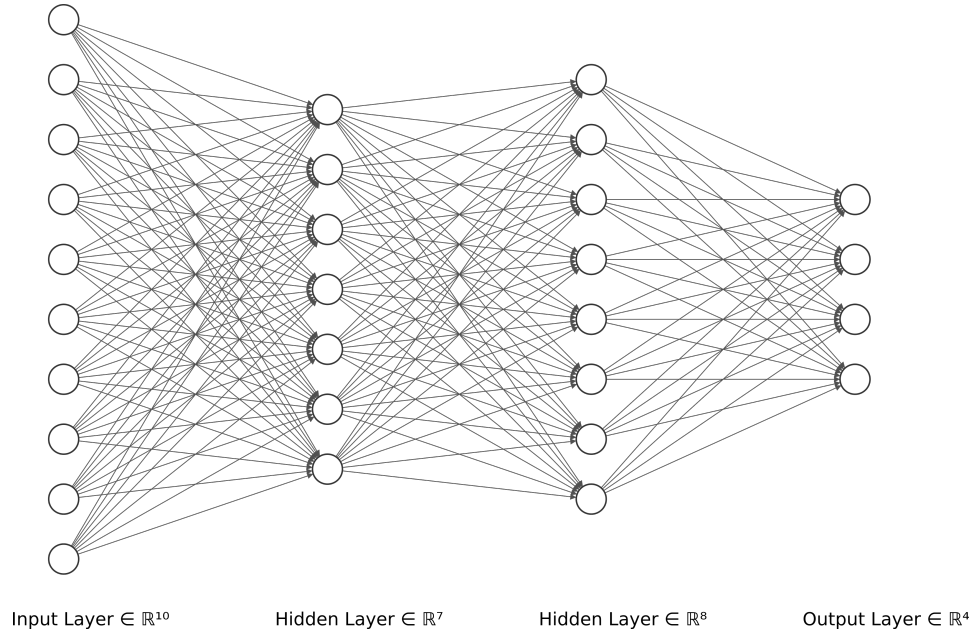


Figure 2.3.: Example fully-connected feedforward ANN with four layers, including input, output and two hidden layers [15].

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. So-called convolutional and recurrent networks are often used for these purposes instead, e.g. in image recognition and time series forecasting [16, 17].

Here we use the LSTM architecture, a type of recurrent neural network (RNN) introduced in [18]. The core idea of RNNs is the usage of loops to store and propagate information through time (Figure 2.4). This is fundamentally different to feedforward networks, where information can only travel in one direction.

The network is made up of one or multiple rows of LSTM cells, each of which share parameters. Figure 2.5 illustrates the internal workings of a single LSTM cell. Besides the current input x_t and the previous output h_{t-1}, the LSTM cell has access to the so called cell state c_{t-1},

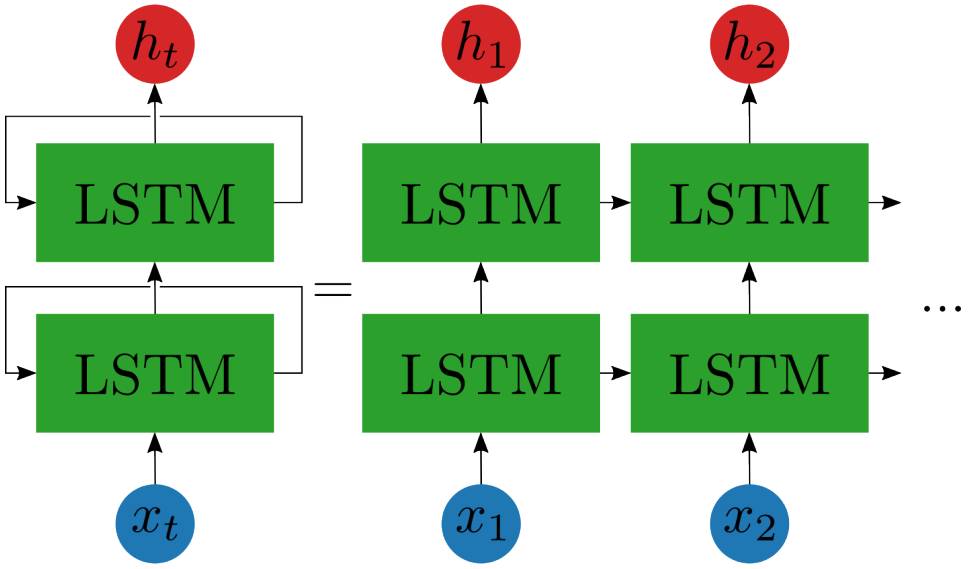


Figure 2.4.: Example two-layer RNN with LSTM architecture. Each row of LSTM blocks has the same parameters and information is fed into the network sequentially.

which acts as the memory and to which the cell can add and delete information, 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 [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 $\odot$ and $\sigma(x)$ are elementwise multiplication and application of the sigmoid function respectively. 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 [20], the network is split into a forward and backward part where the input data is injected in normal and reversed order respectively.

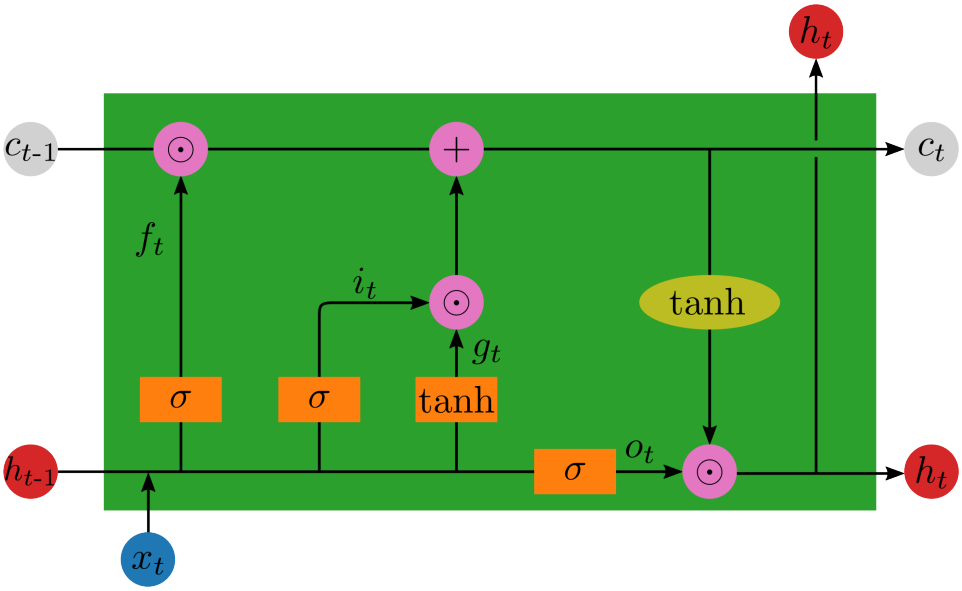


Figure 2.5.: Visualisation of a single LSTM cell. The input x_t enters the cell in the bottom left and is concatenated with the output $h_{t\text{-}1}$ of the previous cell. The combined data then enters multiple single-layer neural networks represented by orange rectangles with their respective activation functions. 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.

2.2.3. Training & Backpropagation

Whereas in the previous sections we considered how information travels through the networks, the question of how one can improve their performance remains. A `loss` or *cost function* is needed to quantify the dissimilarity between the network predictions and correct values. It is important that this function is differentiable, as this allows to calculate the gradient of the cost function with respect to the trainable parameters. This is implemented by the so-called *backpropagation* algorithm [16, 21], which, in simple terms, uses the chain rule to calculate the gradient. It is then used by an optimiser such as Adam [22] or Stochastic Gradient Descent (SGD) to update the trainable parameters and move the predictions closer to the desired values. For *regression* problems such as ours, where one tries to predict a continuous function, the mean squared error (MSE)

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

is often used. 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. Besides the trainable parameters, every machine learning model has one or multiple hyper-parameters. These values are not changed in training and have to be chosen ahead of time. An

example is the amount of hidden layers and neurons per layer in a fully-connected feedforward ANN.

3. Work lower bound

Before we apply neural networks, we derive a lower bound on the extractable work given a drive sequence $\{|\psi_D^n\rangle\}$. By following a policy where the work output of each step dW is optimised locally, we find that the transducer can be chosen such that $dW \geq 0$ for all drives.
We start in an arbitrary system state $\rho_S(n\Delta\mathrm{T})$. Consider the evolved state $\rho_S((n+1)\Delta\mathrm{T}) = \frac{1}{2}(\mathbb{1} + \vec{r}\cdot\vec{\sigma})$ after time $\Delta\mathrm{T}$, with $\vec{r} = (x, y, z)$, where $\vec{r}$ would be determined by a unitary transformation in our case (note that the following is valid for any completely positive trace preserving (CPT) operation). According to Eqs. (2.3) and (2.5), the step work output dW_n is then given by

$$\begin{aligned} dW_n &= \mathrm{Tr}\big\{\rho_S((n+1)\Delta\mathrm{T})(H_{ST}^n - H_{ST}^{n+1})\big\} \\ &= \mathrm{Tr}\{\rho_S((n+1)\Delta\mathrm{T})\left((\mathrm{Re}\{\tau_n\} - \mathrm{Re}\{\tau_{n+1}\})\sigma_x + (\mathrm{Im}\{\tau_n\} - \mathrm{Im}\{\tau_{n+1}\})\sigma_y\right)\} \\ &= x(\mathrm{Re}\{\tau_n\} - \mathrm{Re}\{\tau_{n+1}\}) + y(\mathrm{Im}\{\tau_n\} - \mathrm{Im}\{\tau_{n+1}\}), \qquad (3.1) \\ \tau_n &= \frac{1}{2}\sin\theta_T^n e^{i\phi_T^n}. \end{aligned}$$

τ_n is determined by the previous step and thus we can only vary τ_{n+1}. Eq. (3.1) has a maximum for $\theta_T^{n+1} = \frac{\pi}{2},\ \phi_T^{n+1} = \arctan\frac{y}{x} + \pi$, giving

$$\begin{aligned} dW_{n,lo} &= \frac{1}{2}\left(\sqrt{x^2+y^2} + \sin(\theta_T^n)\begin{pmatrix}\cos(\phi_T^n)\\ \sin(\phi_T^n)\end{pmatrix}\cdot\begin{pmatrix}x\\ y\end{pmatrix}\right) \\ &\geq \frac{1}{2}\left(\sqrt{x^2+y^2} - \left|\begin{pmatrix}\cos(\phi_T^n)\\ \sin(\phi_T^n)\end{pmatrix}\cdot\begin{pmatrix}x\\ y\end{pmatrix}\right|\right) \\ &\geq \frac{1}{2}\left(\sqrt{x^2+y^2} - \sqrt{x^2+y^2}\right) = 0, \end{aligned}$$

where we used the Cauchy-Schwarz inequality going from the second to the third line.
In our setting, $\vec{r}$ only depends on the current $|\psi_D\rangle$ and ρ_S. For a given drive sequence the sum of the locally optimised work outputs $dW_{n,lo}$ provides a lower bound on the total extractable work as each contribution is non-negative. In the following chapter we will show that protocols with higher outputs exist by optimising globally over all time steps.

4. Experimental results

4.1. 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)

$$W = \frac{1}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{(\tau_2 - \tau_1)\alpha^*\} \tag{4.1}$$

$$\alpha = \frac{1}{2}\left[\sin\left(\theta_D^1\right)e^{i\phi_D^1} + \sin\left(\theta_T^1\right)e^{i\phi_T^1}\right], \qquad \tau_2 - \tau_1 = \frac{1}{2}\left[\sin\left(\theta_T^2\right)e^{i\phi_T^2} - \sin\left(\theta_T^1\right)e^{i\phi_T^1}\right].$$

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 $\Delta\mathrm{T}$, finding their optimal transducer policy numerically using a minimiser algorithm[1] [23]. The average work output over the 500 runs scaled by the number of extraction steps $\overline{W}/(N-1)$ for 20 values of $\Delta\mathrm{T}$ is shown in Figure 4.1a.

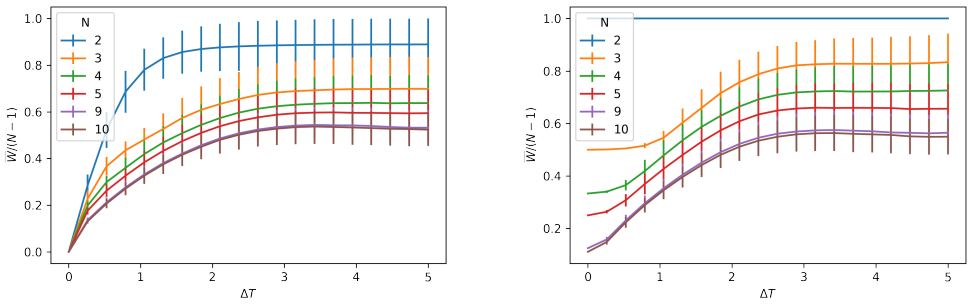
In Figure 4.1b, we plot the average work when the system qubit is initialised in an eigenstate $\rho_0 = |+\rangle\langle +|$ of the partial system Hamiltonian H_{DS}. For $\Delta\mathrm{T} = 0$, the work output is $W = 1$ for all N. This is the amount of work that can be withdrawn from the system by maximising the strength of the extraction Hamiltonians (and therefore their eigenvalues $\lambda_i = \pm\omega$) and switching them such that their eigenvalues change signs. Additionally, they must be chosen so they are diagonal in the basis of ρ_0:

$$\begin{aligned}
H_{ST}^n &= \omega(|+\rangle\langle +| - |-\rangle\langle -|),\\
H_{ST}^{n+1} &= \omega(|-\rangle\langle -| - |+\rangle\langle +|),\\
dW_n &= \mathrm{Tr}\left\{\rho_0(H_{ST}^n - H_{ST}^{n+1})\right\} = 2\omega\,\mathrm{Tr}\{|+\rangle\langle +|\,(|+\rangle\langle +| - |-\rangle\langle -|)\}\\
&= 2\omega.
\end{aligned}$$

This can be done only once, as $\Delta\mathrm{T} = 0$ prevents any evolution of ρ_S.

[1] The work output function is non-convex and therefore difficult to optimise. We try to circumvent this problem by initialising the minimiser in multiple different positions and choosing the minimum of the outputs. It is not guaranteed that these solutions are global minima. In the remainder of this work, by the 'optimal' solution we mean the minimum of the minimiser solutions.

A special case occurs for $N = 2$: the optimal work output 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 $\Delta\mathrm{T} = 0$ (Appendix A.2).
For both initial system states and large N and ΔT, the maximum work per extraction step $\frac{\overline{W}}{N-1} = 0.5$. For smaller ΔT, the work per extraction step is lower, as the optimal system state cannot be reached due to the speed limit in unitary dynamics [24, 25].



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

Figure 4.1.: (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}.

4.2. Data creation, training and evaluation

In the following sections we will apply neural networks to the system outlined before. The training data is created as in the previous section, creating pairs of drive sequences $\{|\psi_D^n\rangle\}$ and their respective optimal transducer protocols $\{|\psi_T^n\rangle\}$. The drive sequences, represented by $\{\theta_D^n, \phi_D^n\}$, are sampled randomly from the Haar measure [26].
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

$$\left\{\begin{pmatrix}\theta^n & \phi^n\end{pmatrix}\right\} \to \left\{\begin{pmatrix}\sin(\theta^n) & \sin(\phi^n) & \cos(\theta^n) & \cos(\phi^n)\end{pmatrix}\right\}. \tag{4.2}$$

The reasons for this operation are twofold: it normalises the data to the interval $[-1, 1]$, which is beneficial to learning [27]. 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.

4.2.1. Choice of cost function for training

Throughout most of this work we use the MSE as introduced in Section 2.2.3. It should be noted that using the MSE for training is not the only choice, and perhaps not the obvious one. Directly using the work extracted is possibly a more intuitive choice, but we refrain from it in the beginning 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 $= O(1)$.

4.2.2. Evaluation of network performance

To compare the accuracy of different models a performance indicator is required. Naturally one might use the MSE. Instead we define the *efficiency* of a model $\mathfrak{N}$ on a dataset $\{(\vec{x}_i, \vec{y}_i)\} = \{(\{|\psi_D^n\rangle\}_i, \{|\psi_T^n\rangle\}_i)\}$ as

$$\eta = \frac{1}{N}\sum_{i=1}^{N}\frac{W(\vec{x}_i, \mathfrak{N}(\vec{x}_i))}{W(\vec{x}_i, \vec{y}_i)}, \tag{4.3}$$

i.e. the arithmetic mean of the ratios of work output predicted by the model to optimal work output. The function $W(\vec{x}_i, \vec{y}_i) = W(\{|\psi_D^n\rangle\}_i, \{|\psi_T^n\rangle\}_i)$ returns the work given a drive and transducer sequence.
See Appendix B for a detailed description of the training implemented as well as the hyperparameters used in this work.

4.3. $N = 2$: Learning single jump optimal control sequences

For the simplest case of $N = 2$, we generate data sets 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 4.1. 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.[2] For comparison, we train a network with the same hyperparameters using the random initial state as additional inputs. This increases the test data efficiency, but is still far below the efficiency for $\rho_0 = |+\rangle\langle +|$.

[2]The deviation from zero is a relic of the finite sample size. For $N_{\text{data}} \to \infty$ it would disappear.

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 4.1.: 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.

4.4. $N = 5$

4.4.1. $\Delta T = 5$

In this section we examine the higher-dimensional 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 (see Figure 4.2). This hyperparameter distinguishes the two cases of whether or not the complete drive sequence is known in advance.
Before entering the LSTM cell, each embedded $|\psi_D^n\rangle$ passes through two fully-connected layers to increase the input dimensionality and add non-linearities to the input. After passing the LSTM a single fully-connected layer is applied to the output to produce the embedding size to recover $|\psi_T^n\rangle$.

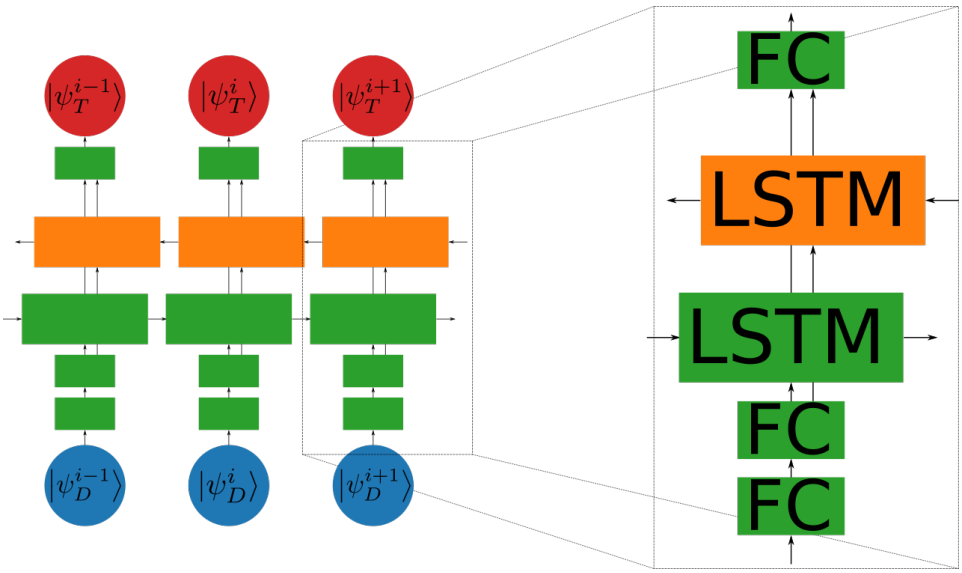


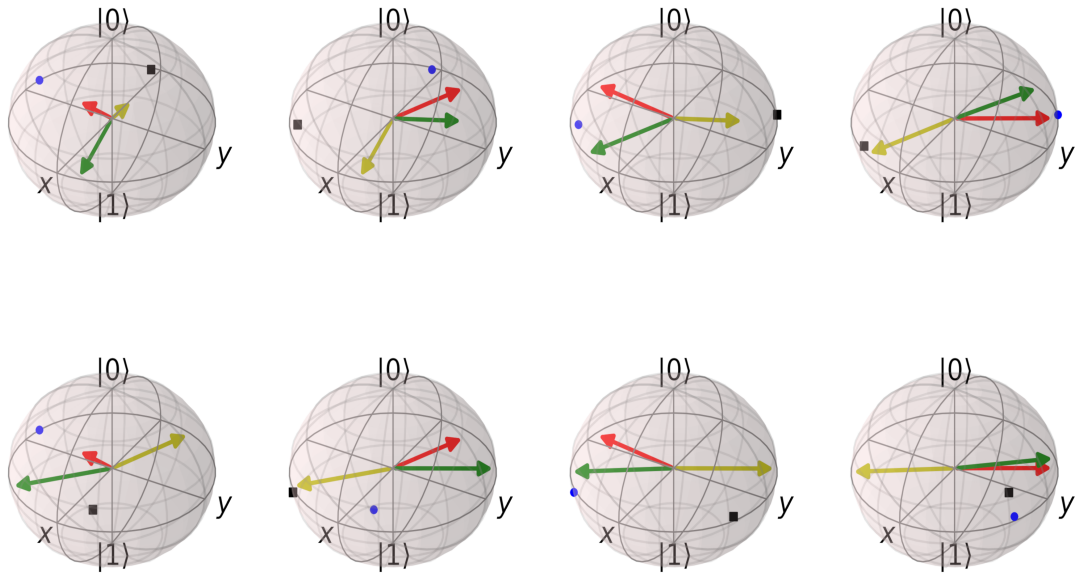
Figure 4.2.: Architecture of the LSTM networks. The green blocks represent the unidirectional case. For the bidirectional case, a second LSTM block (orange) is included with separate trainable parameters. Each LSTM block includes multiple layers of LSTM cells. The blocks labeled 'FC' represent fully-connected layers.

The FCANN is made up of three fully connected hidden layers, the size of which are selected to approximately match the amount of trainable parameters of the bidirectional LSTM.
The test data efficiency as well as the amount of trainable parameters are presented in Table 4.2.

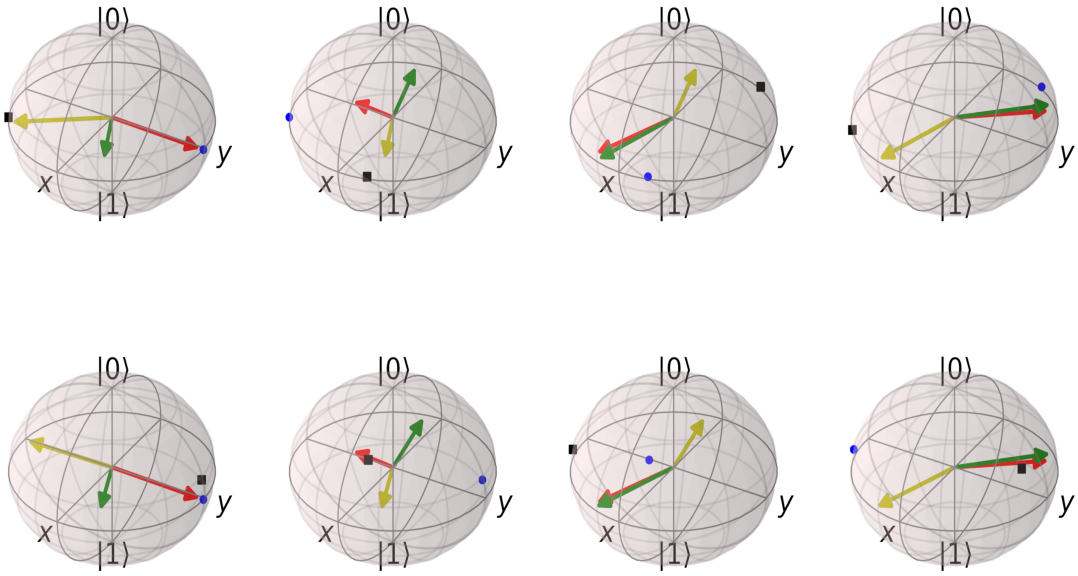
Network Architecture	η_{test} [%]	MSE_{test}	W_{test}	# Parameters
FCANN	19.3	0.1083	0.53	8,086,020
Bidir. LSTM	33.1	0.0948	0.91	7,700,222
Unidir. LSTM	19.5	0.1707	0.52	3,206,990

Table 4.2.: Efficiencies η on the test data for model architectures with given number of trainable parameters for $N = 5,\ \Delta\mathrm{T} = 5$.

The efficiency of the best model for $N = 5,\ \Delta\mathrm{T} = 5$ is drastically lower than that for $N = 2$. Crucially, the models are unable to extract the test set average of the lower bound calculated by local optimisation $W_{lo} = 1.4$.
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 $dW = -\mathrm{Tr}\{\rho_S\ \Delta H\} = \mathrm{Tr}\big\{\rho_S(H_{ST}^n - H_{ST}^{n+1})\big\}$, finding the optimal solution is a compromise of choosing ΔH so as to maximise the expectation value dW while controlling the unitary evolution of ρ_S such that $\mathrm{Tr}\{\sigma_z\rho_S\}$ is small. This is beneficial as the system Hamiltonians and thus ΔH only have σ_x- and σ_y-, but no σ_z-components. We plot the optimal and predicted trajectory of a random data point in the test in Figure 4.3a set to illustrate this compromise. In Figure 4.3b we plot the trajectories for the worst-performing sample in the test set. In this case, the prediction for $|\psi_T^n\rangle$ deviates only slightly from the optimum for $n \in [2, N]$ but deviates significantly for $n = 1$. 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. However, the deviation in the first qubit causes 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 4.4 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 switching, i.e. setting $\theta_T^N = \frac{\pi}{2}$, to maximise its work output. Additionally ϕ_T^N should be set so that H_{ST}^N is antiparallel to the projection of ρ_S^N onto the x-y-plane of the Bloch sphere.

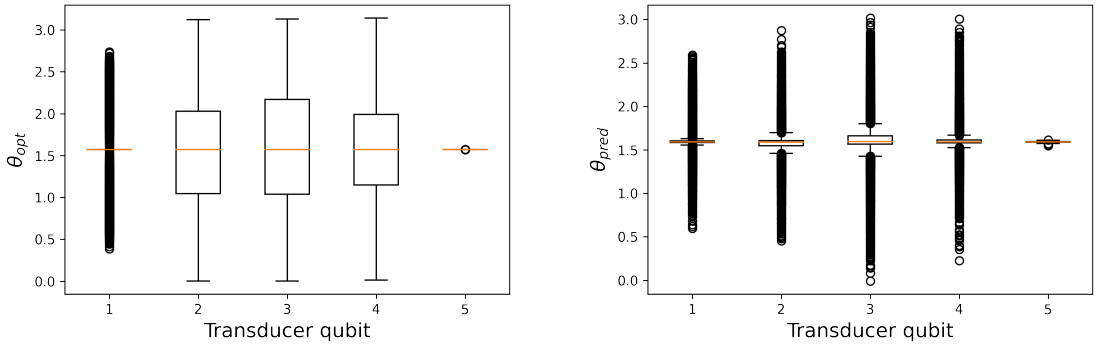


(a) $W_{opt} = 3.03, W_{pred} = 1.11$



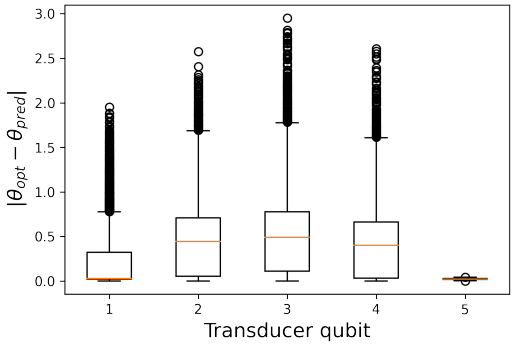
(b) $W_{opt} = 3.06, W_{pred} = -2.65$

Figure 4.3.: (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^n (blue dot), ρ_S^{n+1} (black square), the partial system Hamiltonian acting only between system and drive H_{DS}^n (red vector) as well as H_{ST}^n (yellow vector) and H_{ST}^{n+1} (green vector). **Top row:** we plot the system dynamics for the transducer series generated by the optimiser for $n \in [1, N-1]$. In the optimal case, H_{ST}^n is chosen such that ρ_S remains near the x-y-plane for all times and $\mathrm{Tr}\{\rho_S^{n+1} H_{DS}^n\}$ 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 4.4, 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 $n \in [2, N]$ are very close to the optimal solutions (top row of (b)). However, the first transducer prediction is wrong, leading to a deviation from the optimal system dynamics.

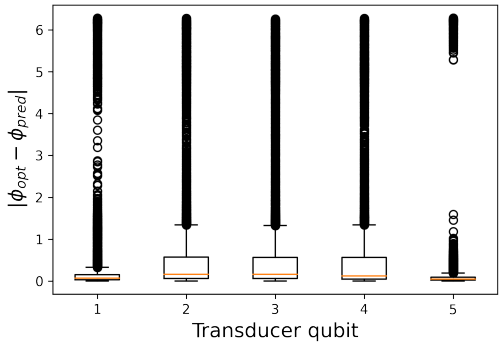


(a) θ^n_{opt}

(b) θ^n_{pred}



(c) $|\theta^n_{\text{opt}} - \theta^n_{\text{pred}}|$



(d) $|\phi^n_{\text{opt}} - \phi^n_{\text{pred}}|$

Figure 4.4.: For the bidirectional LSTM network and $\Delta T = 5$, we plot boxplots (inside the box are values from the lower to the upper quartile (25th to 75th percentile), the orange line represents the median and the vertical lines represent the values that lie in 1.5 times the interquartile range (height of the box)) of the optimal **(a)**, predicted **(b)** and absolute differences **(c)** of θ^n_T for the five qubits of each trajectory in the test set. The prediction for θ^N_T 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^0_T is reasonably good too, 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. **(d)**: we plot the absolute difference between optimal and predicted ϕ^n_T. We refrain from plotting the optimal and predicted ϕ^n_T themselves as these are distributed equally.

4.4.2. $\Delta T = 1$

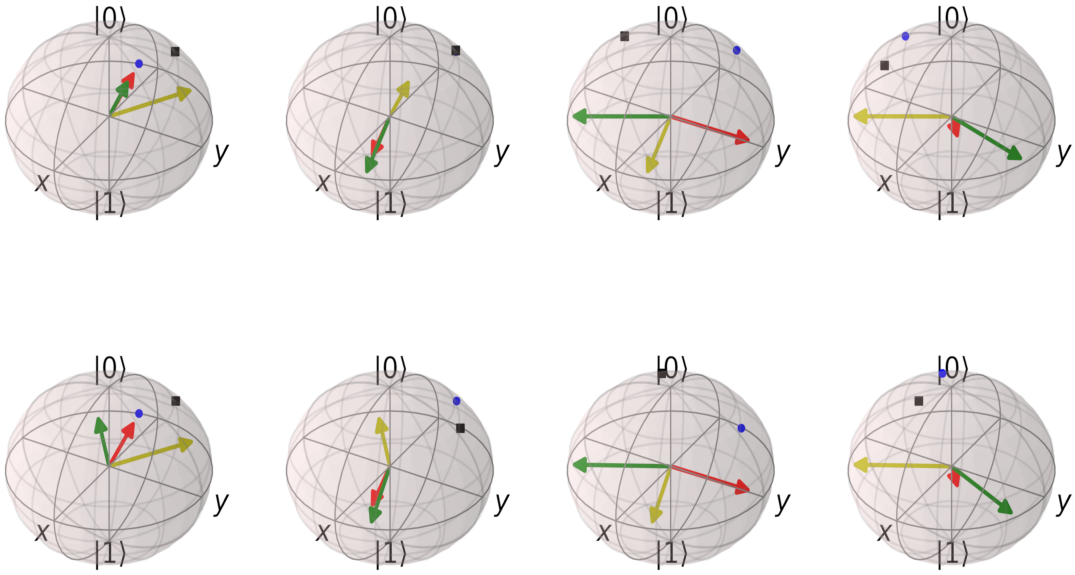
We apply the methodology of Section 4.4.1 to the case of $\Delta T = 1$ with $N = 5$ drive and transducer qubits. Contrary to $\Delta T = 5$, the optimal system states[3] cannot be reached from every point on the Bloch sphere, leading to a lower total work output following the protocol determined by the optimiser (Figure 4.1).
We train neural networks with the same hyperparameters as in Section 4.4.1. The performance of each network is listed in Table 4.3. We find that all networks outperform their $\Delta T = 5$ counterparts, both in terms of efficiency and average work output. In addition, the MSE score on the test data is lower, indicating that the predictions themselves are better. There are two reasons for this occurrence: firstly, the predictive power of the networks is greater for $\Delta T = 1$ than $\Delta T = 5$, illustrated in Figure 4.6. Secondly, deviations between optimal and predicted transducer settings lead to a smaller difference between the respective system states as the evolution time is shorter, leading to a higher work output. This can be seen in Figure 4.5, where we plot optimal and predicted transducer policies of the same drive for the two switching times.

Network Architecture	η_{test} [%]	MSE_{test}	W_{test}	# Parameters
FCANN	96.4	0.0265	1.61	8,086,020
Bidir. LSTM	97.4	0.0238	1.62	7,700,222
Unidir. LSTM	70.1	0.0638	1.18	3,206,990

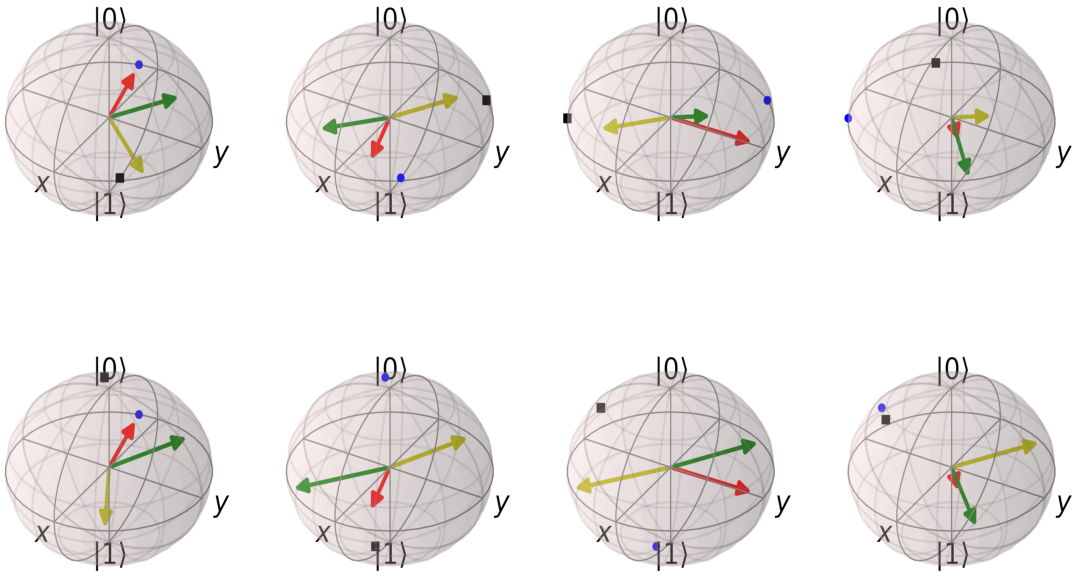
Table 4.3.: Efficiencies η and MSE loss on the test set for differing model architectures for $N = 5, \Delta T = 1$.

For $\Delta T = 1$, all models beat the locally optimised average $W_{lo} = 1.14$. It is therefore beneficial to follow the model predictions instead of the policy governed by local optimisation even if the drive sequence is unknown.

[3] $\text{Tr}\{\rho_S(n\Delta T)\sigma_z\} = 0$

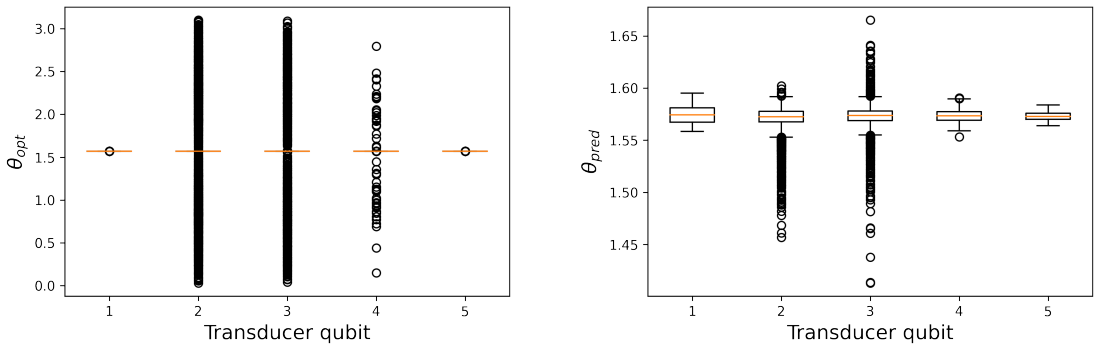


(a) $\Delta T = 1 : W_{opt} = 1.40, W_{pred} = 1.32$

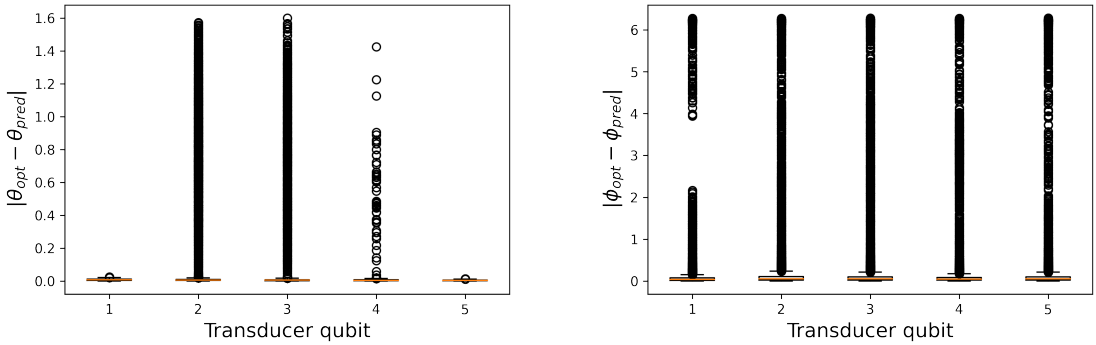


(b) $\Delta T = 5 : W_{opt} = 2.42, W_{pred} = 0.57$

Figure 4.5.: We plot the optimal (top row) and predicted (bottom row) system state and transducer settings for the bidirectional LSTM, using the same representation as in Figure 4.3, for $\Delta T = 1$ (a) and $\Delta T = 5$ (b). The drive sequence is the same in all rows and is a sample from the test set. For $\Delta T = 1$, the optimal system state, where $\mathrm{Tr}\{\rho_S(n\Delta T)\sigma_z\} = 0$, cannot be reached. The discrepancy of the system states (blue/black dots) between optimal (top row) and predicted (bottom row) policies are smaller than for $\Delta T = 5$.



(a) θ^n_{opt} **(b)** θ^n_{pred}



(c) $|\theta^n_{\text{opt}} - \theta^n_{\text{pred}}|$ **(d)** $|\phi^n_{\text{opt}} - \phi^n_{\text{pred}}|$

Figure 4.6.: We plot the performance of the bidirectional LSTM for $\Delta\text{T} = 1$. Unlike the optimum for the longer switching time, $\theta^n_T = \frac{\pi}{2}$ for a majority of the central three qubits as well. Again, predicted and optimal ϕ^n_T are distributed equally and therefore we do not plot them here.

4.4.3. Noise resistance

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

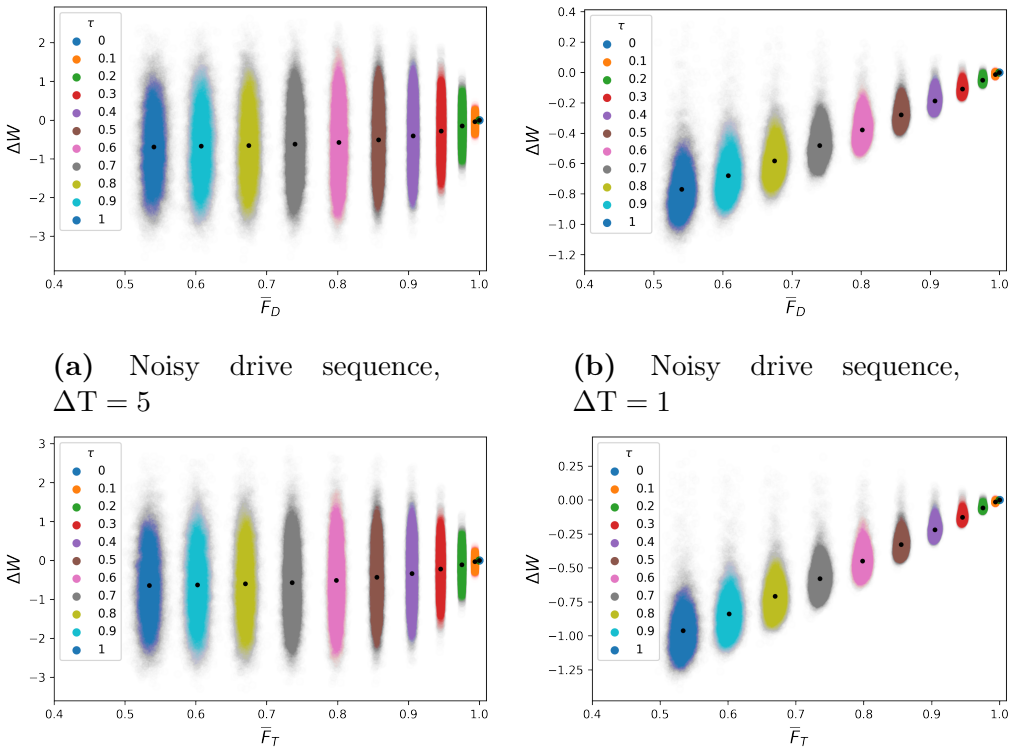
$$|\phi_n\rangle = e^{-iH_n\tau}\,|\psi_n\rangle \;\; \forall n \in [1, N].$$

H_n are randomly generated Hermitian matrices with $\|H_n\| = 1$ and τ is a real parameter used to control the strength of the noise. To quantify the dissimilarity between $\{|\phi_n\rangle\}$ and $\{|\psi_n\rangle\}$ we use the fidelity F as defined in [28]:

$$F_{\text{run}}(\{|\phi_n\rangle\}, \{|\psi_n\rangle\}) = \prod_n F(|\phi_n\rangle, |\psi_n\rangle) = \prod_n |\langle\phi_n|\psi_n\rangle|.$$

The performance of the bidirectional LSTM for $\Delta\text{T} = 1$ and $\Delta\text{T} = 5$ on noisy sequences is presented in Figure 4.7. For $\Delta\text{T} = 1$, the performance degrades among the test set, as the noiseless predictions are near their optima. The variance in the deviation of work output

between noisy and noiseless data $\Delta W = \overline{W}_{noise} - W_{pred}$ is larger for $\Delta T = 5$ as the longer evolution time leads to a larger difference between the system states of the noisy and noiseless case. As the network is less likely to predict a solution near the optimum, the noisy data will sometimes be closer to the optimum and thus a significant amount of data points with $\Delta W > 0$ exist as well.



(a) Noisy drive sequence, $\Delta T = 5$

(b) Noisy drive sequence, $\Delta T = 1$

(c) Noisy transducer sequence, $\Delta T = 5$

(d) Noisy transducer sequence, $\Delta T = 1$

Figure 4.7.: We plot the difference $\Delta W = \overline{W}_{noise} - W_{pred}$ of each element of the test set for the bidirectional LSTM. W_{pred} is the work output following the model prediction. **(a), (b)**: we create 100 noisy drive sequences and calculate the average $\overline{W}_{noise}$ of their work output following the predicted transducer protocol. **(c), (d)**: we create 100 noisy transducer sequences and calculate the average of their work output with a given drive sequence from the test set. In both plots, the black dots indicate the average fidelities and ΔW for a given τ.

4.4.4. Extracted work as cost function

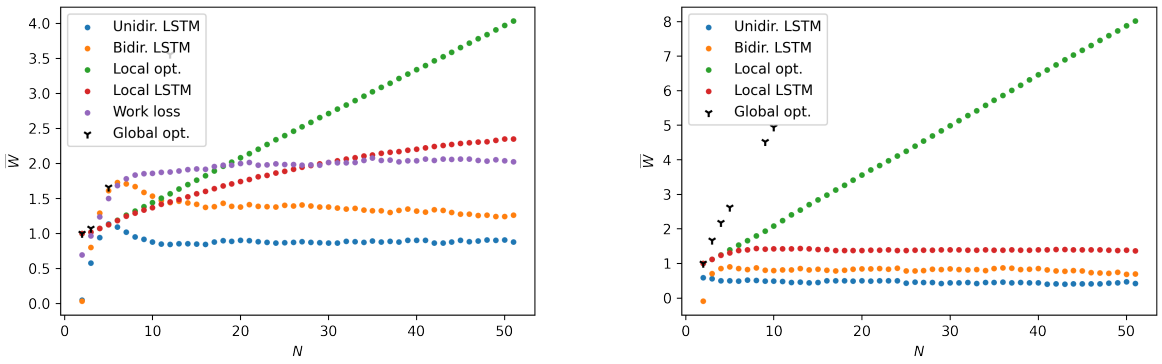
In Section 4.4.1 we encountered a conceptional problem regarding using the MSE as a loss function, where a low MSE score does not necessarily imply a large work output. In the following we aim to investigate whether directly using extracted work W as the network cost function can improve performance.

For $\Delta T = 1$, we train a bidirectional LSTM using this approach, giving a work output of $W_{test} = 1.51$ and efficiency $\eta_{test} = 91.0\%$. This is below the performance of the model trained

on the MSE loss. A possible reason for this difference is that while each training sample for the MSE loss is optimised individually, the work loss maximises the output of multiple drive samples at the same time because of the batch training used in this work (Appendix B).

4.5. Generalisation to other N

The modular structure of the LSTM networks used in this work allows us to generalise the approach to values of N other than the one the networks were trained on. In this section we investigate the performance of the networks trained with optimal sequences for $N = 5$ when varying the amount of extraction steps. We compare the performance of the uni- and bidirectional LSTM networks trained on the MSE loss to a unidirectional LSTM model trained on the locally optimised protocol as well as the work output when following the policy of local optimisation directly. We plot the average work output of the aforementioned in Figure 4.8. For $\Delta T = 1$, the LSTM networks trained using the MSE loss perform best for values close to $N = 5$, which they were trained on, and are unable to generalise to larger N. This is expected as the training set consists of policies optimal only for $N = 5$. $\overline{W}$ drops for $N > 6$ while no such drop can be seen in the LSTM model trained using the work as loss. The LSTM trained on the locally optimised policies is able to generalise to larger N for $\Delta T = 1$, although the extracted work per step decreases with N. Learning the locally optimised policy requires learning a representation of the system state ρ_S, the errors of which will propagate if the representation is not exact. For large N, this representation will be no better than the fully mixed state, leading to $dW = 0$ on average.



(a) $\Delta T = 1$ **(b)** $\Delta T = 5$

Figure 4.8.: Generalisability of MSE and local optimisation models. For each N, we generate 1000 random drives and predict their transducer policies. We plot the mean of their work output $\overline{W}$ for varying N. Where available, we plot the optimal average work output. For $\Delta T = 1$, we additionally plot $\overline{W}$ for the bidirectional LSTM trained using the extracted work as the loss function. The average output of the local optimisation protocol is plotted as a benchmark.

For $\Delta\mathrm{T} = 5$, all models fail to generalise to larger N. A possible reason can be found in the poor performance of the unidirectional LSTM trained on the locally optimised protocol. Here, it becomes evident that the model is unable to learn a precise representation of the system state which remains valid for larger N. Two possible reasons come to mind, firstly the larger evolution time leads to a faster propagation of errors in the representation. Additionally, the mapping $U = e^{-iH_s\Delta\mathrm{T}}$ from initial to evolved state, which depends non-linearly on $\Delta\mathrm{T}$ and the model inputs and outputs, is more difficult to represent for the LSTM cell. The quality of any polynomial approximation of U learned by the model will degrade more quickly for large $\Delta\mathrm{T}$.

5. Summary and outlook

In this work we used Long Short-Term Memory networks to predict transducer policies given an excitation driving a two-level quantum system. The performance was compared to the lower bound derived through local optimisation. We demonstrated that our approach is capable of predicting transducer policies that will produce a positive work output given a drive data set. We found that the efficiency greatly depends on the model parameters N and $\Delta\mathrm{T}$. For $N = 2$ and $\rho_0 = |+\rangle\langle+|$, where an analytic optimum exists, a simple linear network is able to reproduce the optimal policy. For the higher-dimensional case $N = 5$, we examined two switching times $\Delta\mathrm{T}$, finding that the network performs better on the shorter than the longer time. For $\Delta\mathrm{T} = 5$, the models were unable to outperform the lower bound and were only able to extract 33.1 % of the optimum. The reason for the lower performance is likely a combination of two factors. The models struggle to find a representation of the system state, which is necessary to predict the optimal next step. Additionally, we found an inherent difference in the optimal solutions between the two switching times: for $\Delta\mathrm{T} = 1$, the transducer Hamiltonian strength was maximised for most samples and thus θ_T easy to predict. For $\Delta\mathrm{T} = 5$, where the amount of reachable states on the Bloch sphere is greater, the optimal transducer strength is chosen such that the evolved state is in a favourable position. This dependence on both the initial and evolved state adds difficulty. In all cases the bidirectional outperformed the unidirectional LSTM, as the former has access to the complete drive sequence while the latter can only make predictions from the previous and current drive qubits.

We identified the use of the mean squared error as a possible disadvantage in calculating the loss during training, especially for $\Delta\mathrm{T} = 5$. For $\Delta\mathrm{T} = 1$ we trained a bidirectional model on the extracted work directly which performed slightly worse than the model trained on MSE loss but has better generalisability. Overall, the main difficulty in all models seems to have been representing the current system state correctly. In our approach, this representation had to be found implicitly, as ρ_S was not included in the training data. An alternative would be following an approach similar to those in [29, 30], where recurrent neural networks are used to directly model the system dynamics.

An extension to the model is to add dissipative effects to the system state ρ_S. Depending on its realisation, this can lead to a defined steady-state of the system which might be easier for the models to learn.

In all machine learning based approaches, the performance of such models highly depends on

the choice of hyperparameters. We employed Bayesian optimisation (see Appendix B for more details) to find hyperparameters that maximise performance. However, the approach is limited in the amount of computing power available and the maximum over the total hyperparameter space may not be found in finite time.

It has to be noted that the approach followed in this work depends on maximising expectation values of work outputs. While this is certainly justified in the limit of infinite experimental realisations, the question of how the results might differ in the single shot case remains. A reinforcement learning based approach, where problems are usually formulated as Markov Decision Problems and probabilistic reward functions are common [31], would be a suitable alternative. Additionally, this approach could be extended to the case of continuous drive and transducer functions.

We calculated the work output as the negative expectation value of the internal energy of the system, which is extracted through the use of transducer qubits. The question of how this output could be used in practice, e.g. to charge a battery, remains open.

6. Bibliography

[1] William Thomson. *On an absolute thermometric scale founded on Carnot's theory of the motive power of hear, and calculated from Regnault's observations*, volume 1 of *Cambridge Library Collection - Physical Sciences*, page 100–106. Cambridge University Press, 2011.

[2] A. Einstein. Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt. *Annalen der Physik*, 322(6):132–148, January 1905.

[3] D Egloff, O C O Dahlsten, R Renner, and V Vedral. A measure of majorization emerging from single-shot statistical mechanics. *New Journal of Physics*, 17(7):073001, jul 2015.

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

[5] 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.

[6] 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.

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

[8] 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.

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

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

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

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

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

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

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

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

[17] 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.

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

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

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

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 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] 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.

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

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

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

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

[29] Leonardo Banchi, Edward Grant, Andrea Rocchetto, and Simone Severini. Modelling non-markovian quantum processes with recurrent neural networks. *New Journal of Physics*, 20(12):123030, Dec 2018.

[30] E. Flurin, L. S. Martin, S. Hacohen-Gourgy, and I. Siddiqi. Using a recurrent neural network to reconstruct quantum dynamics of a superconducting qubit from physical observations. *Phys. Rev. X*, 10:011006, Jan 2020.

[31] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[32] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[33] Peter I. Frazier. A tutorial on bayesian optimization, 2018.

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

A. Derivations

A.1. Single jump work output

$$
\begin{aligned}
H_S(\theta_D^t, \phi_D^t, \theta_T^t, \phi_T^t) &= \frac{1}{2}\left[\sin\big(\theta_D^t\big)e^{i\phi_D^t} + \sin\big(\theta_T^t\big)e^{i\phi_T^t}\right]\sigma_+ + h.c. \\
&= \alpha\sigma_+ + h.c. \\
dH &= H_S(\theta_D^t, \phi_D^t, \theta_T^{t+1}, \phi_T^{t+1}) - H_S(\theta_D^t, \phi_D^t, \theta_T^t, \phi_T^t) \\
&= \frac{1}{2}(\sin\big(\theta_T^{t+1}\big)e^{i\phi_T^{t+1}} - \sin\big(\theta_T^t\big)e^{i\phi_T^t})\sigma_+ + h.c. =: (\tau' - \tau)\sigma_+ + h.c. \\
U &= e^{-iH_S\Delta\mathrm{T}} = \exp\begin{pmatrix} 0 & -i\alpha^*\Delta\mathrm{T} \\ -i\alpha\Delta\mathrm{T} & 0 \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} \frac{\alpha^*}{|\alpha|} & -\frac{\alpha^*}{|\alpha|} \\ 1 & 1 \end{pmatrix}\begin{pmatrix} e^{-i|\alpha|\Delta\mathrm{T}} & 0 \\ 0 & e^{i|\alpha|\Delta\mathrm{T}} \end{pmatrix}\begin{pmatrix} \frac{|\alpha|}{\alpha^*} & 1 \\ -\frac{|\alpha|}{\alpha^*} & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos(|\alpha|\Delta\mathrm{T}) & -i\frac{\alpha^*}{|\alpha|}\sin(|\alpha|\Delta\mathrm{T}) \\ -i\frac{|\alpha|}{\alpha^*}\sin(|\alpha|\Delta\mathrm{T}) & \cos(|\alpha|\Delta\mathrm{T}) \end{pmatrix}
\end{aligned}
$$

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

$$
\begin{aligned}
\rho_0 &= \begin{pmatrix} |a|^2 & ab^* \\ a^*b & |b|^2 \end{pmatrix},\ \rho = U\rho_0 U^\dagger = \begin{pmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{pmatrix} \\
\rho_{00} &= |a|^2\ \cos^2(|\alpha|\Delta\mathrm{T}) + |b|^2\ \sin^2(|\alpha|\Delta\mathrm{T}) - \frac{1}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{ab^*\alpha\} \\
\rho_{01} &= \frac{\alpha^*}{2|\alpha|}i\ \sin(2|\alpha|\Delta\mathrm{T})(|a|^2 - |b|^2) + \frac{\alpha^*}{\alpha}a^*b\sin^2(|\alpha|\Delta\mathrm{T}) + ab^*\ \cos^2(|\alpha|\Delta\mathrm{T}) \\
\rho_{10} &= \frac{|\alpha|}{2\alpha^*}i\ \sin(2|\alpha|\Delta\mathrm{T})(|b|^2 - |a|^2) + \frac{\alpha}{\alpha^*}ab^*\ \sin^2(|\alpha|\Delta\mathrm{T}) + a^*b\ \cos^2(|\alpha|\Delta\mathrm{T}) \\
\rho_{11} &= |a|^2\ \sin^2(|\alpha|\Delta\mathrm{T}) + |b|^2\ \cos^2(|\alpha|\Delta\mathrm{T}) + \frac{1}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{ab^*\alpha\} \\
\mathrm{dW} &= -\,\mathrm{Tr}\ \rho\ \mathrm{dH} = \frac{|a|^2 - |b|^2}{|\alpha|}\sin(2|\alpha|\Delta\mathrm{T})\,\mathrm{Im}\{(\tau' - \tau)\alpha^*\} \\
&\quad - 2\ [\cos^2(|\alpha|\Delta\mathrm{T})\,\mathrm{Re}\{(\tau' - \tau)ab^*\} + \sin^2(|\alpha|\Delta\mathrm{T})\,\mathrm{Re}\left\{(\tau' - \tau)a^*b\frac{\alpha^*}{\alpha}\right\}]
\end{aligned}
\tag{A.1}
$$

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

$$[H_{DS}, H_S] = [\delta\sigma_+ + \delta^*\sigma_-, \alpha\sigma_+ + \alpha^*\sigma_-] = 0 \tag{A.2}$$

with $\delta = \frac{1}{2}\sin\theta_D e^{i\phi_D}$. Considering only the principal branch, Eq. (A.2) is equivalent to

$$\mathrm{Im}\{\alpha\delta^*\} = 0 \implies \arg\alpha = \phi_D \implies ab^* = a^*b\frac{\alpha^*}{\alpha} \implies \phi_T^{opt} = \phi_D, \tag{A.3}$$

which leads to the independence of $dW = W$ (for $N = 2$, the sum collapses to a single term) with regard to ΔT, as Eq. (A.1) reduces to

$$W = -\,\mathrm{Re}\big\{(\tau' - \tau)e^{-i\phi_D}\big\} = -\frac{1}{2}\,\mathrm{Re}\Big\{(\sin\theta'_T e^{i\phi'_T} - \sin\theta_T e^{i\phi_T})e^{-i\phi_D}\Big\} \tag{A.4}$$

$$\overset{A.3}{=} -\frac{1}{2}\,\mathrm{Re}\Big\{(\sin\theta'_T e^{i\phi'_T} - \sin\theta_T e^{i\phi_D})e^{-i\phi_D}\Big\}. \tag{A.5}$$

Eq. (A.5) is maximised by $\theta'_T = \theta_T = \frac{\pi}{2}$ and $\phi'_T = \phi_D + \pi$, giving

$$\begin{aligned} W &= -\frac{1}{2}\sin\frac{\pi}{2}\,\mathrm{Re}\big\{(e^{i(\phi_D+\pi)} - e^{i\phi_D})e^{-i\phi_D}\big\} \\ &= -\frac{1}{2}\,\mathrm{Re}\big\{-2e^{i\phi_D}e^{-i\phi_D}\big\} = 1. \end{aligned} \tag{A.6}$$

B. Training protocols

In all models we split the data into training, validation and test data sets 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 data 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 and the parameters updated.
A common problem in machine learning is *overfitting*, where the model learns noise in the data instead of the underlying functional correlation. It is indicated by a good performance on the training data but poor performance on unseen data. We employ two techniques to prevent overfitting. Firstly we use early stopping to stop training when the cost function on the validation set (unseen data) does not decrease for *patience* epochs. Additionally, we apply dropout after all layers except for input and output. During training, each neuron is deactivated with probability *dropout*. This has been proven to prevent overfitting [32].
All performance data is calculated on the test set, which the model has seen neither during training or has been used for early stopping.

B.1. Hyperparameters Section 4.3

For $N = 2$, we use the Adam [22] 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 4.3.

B.2. Hyperparameters Section 4.4

The learning rate (LR) is determined by the 'Reduce LR on Plateau' schedule. The learning rate is multiplied by the hyperparameter *factor* if the cost function decreased for the past

patience / *pat drop* epochs. The hyperparameters are found using Bayesian optimisation [33], which uses multiple training runs with different hyperparameters to create a statistical model of the model performance with respect to the hyperparameters, implemented by [34]. The values can be found in Table B.2. The FCANN in Section 4.4.1 has three hidden layers with 2000 neurons each. The hyperparameters for the bidirectional LSTM from Section 4.4.4 are presented in Table B.3.

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 Sections 4.4.1 to 4.4.3.

Hyperparameter	Value
Optimiser	SGD
Batch size	23
Dropout	0.1343610845377841
Input layer 1 size	58
Input layer 2 size	1115
Output layer size	738
LSTM hidden size	528
Initial learning rate	0.0484126686015994
LSTM layers	1
Pat drop	1.5006100847416177
Patience	78
Factor	0.5967308787201352

Table B.3.: Hyperparameters used for the models in Section 4.4.4.

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, Februar 2021