Leveraging a Discrete-Time-Crystal to Solve Classification Problems with a Quantum Extreme Learning Machine

Simon Thomas Mader

Ludwig-Maximilians-Universität München
Department: Physics
Supervisor: PD Dr. Christoph Räth

Munich, April 10th, 2025

Nutzung eines diskreten Zeitkristalls zur Lösung von Klassifikationsproblemen mit einer Quantum Extreme Learning Machine

Simon Thomas Mader

Ludwig-Maximilians-Universität München
Fakultät: Physik
Betreuer: PD Dr. Christoph Räth

München, 10. April 2025

Abstract

This thesis explores the field of Quantum Extreme Learning Machines based on many-body localized discrete time crystals. This approach holds two advantages: Leveraging an exponentially large Hilbert states while not relying on error-corrected quantum gates. Firstly, the concept of discrete time crystals and the theoretical framework behind QELMs are presented. After some introductory results to test the potentials and limitations of unitary evolution, the method's phase dependency is discussed to observe the melting of the discrete time crystal also in the classification accuracies. The remainder of the thesis can be split into two categories: Investigating effects on the classification accuracies when changing the quantum layer and an investigation of amendments in the readout layer. Both parts contain a comparison for thermal and discrete time crystal phase as well as reasonable comparisons with classical results. In the readout layer analysis new readout methods are presented that hold the potential of increasing the classification accuracies based on random shuffling. The work concludes with a combination of all methods and an evaluation of their combined performance. Overall, the thesis provides a comprehensive overview of the exciting field of Quantum Extreme Learning Machines, offering new insights as well as new pathways for further advancements in the field.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Christoph Räth, for his invaluable guidance and support throughout this master thesis. His openness to my ideas and encouragement to pursue independent work have been instrumental in shaping this research. I truly appreciate the balance he provided between offering direction and allowing me the freedom to explore my own approaches. Secondly, I would like to thank Joel Steinegger and Gonzalo Camacho for their patience in answering my questions and their support in any arising problems, whether they were technical or topic related.
I am especially thankful to my parents, Kerstin and Dieter, for their unwavering emotional and financial support. Their constant encouragement has been a pillar of strength throughout this journey.
Finally, I would like to thank my girlfriend, Gaia, for her patience, love, and belief in me, even during the most challenging moments and stressful times.
Lastly, I would like to take the time to thank all the friends I made throughout my studies. I would not have made it here without you and I am deeply thankful for the collaboration that arose from our interactions throughout that time.

EHRENWÖRTLICHE ERKLÄRUNG

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst zu haben und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet zu haben. Die in dieser Arbeit vorgestellten Forschungen und Ergebnisse werden in einen in Vorbereitung befindlichen Forschungsartikel einfließen. Alle Inhalte, Bilder und Analysen sind meine eigene Arbeit und werden in Teilen in dem erwähnten Forschungsartikel veröffentlicht werden.

DECLARATION OF AUTHORSHIP

I hereby declare that this thesis is my own work, and that I have not used any sources and aids other than those stated in the thesis. The research and results presented in this thesis will contribute to a paper in preparation. All the contents, images and analysis are my own work and will partially be published in the nominated paper.

Simon Thomas Mader

Place and Date

Contents

List of Figures

Abbreviations

Abbreviation	Full Term
LinearSVC	Linear Support Vector Classifier
MBL-DTC	Many-Body Localized Discrete Time Crystal
NISQ	Noisy Intermediate-Scale Quantum
NN	Neural Network
ONN	One-Layer Neural Network
QELM	Quantum Extreme Learning Machine
QRC	Quantum Reservoir Computing
RC	Reservoir Computing

Chapter 1

Introduction

Although it still feels like a relatively new and exciting field, the origins of research in Quantum Computing already date back to the 1980s. In 1981 Richard Feynman held a revolutionary keynote speech at a conference [1] outlining the potential that Quantum Mechanics holds for accelerating computations. Since then, the field has of course grown vastly and generated innovative as well as promising solutions and ideas to overcome difficulties associated with it. Especially notable are the introductions of efficient quantum algorithms by Deutsch [2], Shor [3] and Grover [4]. Furthermore, significant leaps forward to enable these algorithms were the theoretical propositions of Quantum Fourier Transforms [4] and Quantum Phase Estimators [3]. A longstanding and intrinsic drawback of Quantum Computers has been their sensitivity to errors. No system is entirely isolated from its environment. Therefore, no real-world quantum system can be fully described by its unitary evolution. Fortunately, decades of work have also equipped us with techniques to describe and mediate arising errors [5].
Simultaneously, the field of Artificial Intelligence, more precisely Machine Learning, has been evolving. Over the past two decades, driven by the exponential growth of computing capabilities, this growth has accelerated and culminated in the recent emergence of Large Language Models [6]. The overwhelming pace and innovative approach [7] with which these models have changed many aspects of our lives exemplify the potential effects Machine Learning is capable of. A relevant factor to consider when discussing advances in Machine Learning is the giant consumption of financial and energetic resources that have been and still are relevant to drive further development [8]. In view of these constraints it is more important than ever to look for efficient computational methods to go beyond classical von Neumann architectures while preserving computational capabilities.
A promising approach in this field is the combined Quantum Computing and Machine Learning to the emergent field of Quantum Machine Learning [9]. This new field can be further split into gate based Quantum Machine Learning [10] and unconventional computing [11]. Given that Quantum Computing is and will be in the near future working with NISQ-devices, unconventional computing is an already implementable approach. In our work, we will be focusing on this branch of Quantum Machine Learning. In the spirit of other works, the Hamiltonian we are using as quantum system is a discrete time crys-

tal [11] [12]. Originating from the study of complex systems, a lot of work has already been done to leverage unconventional computing methods to predict time series of dynamical systems. As this approach is known as Reservoir Computing (RC) [13], the quantum extension is called Quantum Reservoir Computing and has been demonstrated successfully [14]. However, the goal of this work is to leverage the technology for optimization problems instead of time series prediction. Within this class of problems, we will be focusing on classification rather than regression. This algorithm is known as Quantum Extreme Learning Machines, originating from extending classical Extreme Learning Machines with a quantum layer. Due to their rich physical behavior a common choice for the quantum layer of a QELM are discrete time crystals [15] [11]. We will also follow this path. The most suitable and standard data set established in the academic world is the mnist digit data set [16]. The goal of this work is to explore the potential and methods of a QELM based on discrete time crystals when being used for classification problems.
While previous work has focused on the comparison of classical and quantum performance [15] or the comparison of feature reduction techniques as well as encoding methods, our work has a slightly different focus. In chapter 5 we focus on the effects of measuring different observables for our training process. This includes all Pauli matrices and 2-qubit correlations of all qubits present in the 1D-chain. In chapter 6 we shift our focus to a study of different post-processing methods. While temporal multiplexing has been a well established method in the field of RC and QRC [14], its application has not been explored in QELMs yet. Furthermore, chapter 6 will include new suggestions regarding non-linear processing of the quantum system's measurement. So far, in QRC the standard readout methods of RC have been employed. We extend these methods by introducing randomness and comparing the results for shifting the feature vector.

Chapter 2

Quantum Extreme Learning Machine: Theory and Model

Quantum Machine Learning is a relatively new field, characterized by rapid development and a great diversity of promising new approaches. One of these approaches are quantum extreme learning machines, hereinafter referred to as QELM. The concept of QELM builds on the utilization of the Hilbert space of entangled quantum systems, enabling the leverage of non-linear combinations of the original input data. While this is one of the large advantages of leveraging quantum systems, the current status of the field also comes with disadvantages. So far, we are not able to reliable build large quantum computers consisting of several hundreds of mathematical qubits. Given the close link between features to encode and the size of quantum systems, this limits the nature of problems to tackle with a QELM in the NISQ era. Consequently, for more complex problems we need to perform some pre-processing. In particular we reduce the number of features present in mnist by applying a PCA. Afterwards, we need to choose an encoding scheme to encode the classical data into the quantum system. The quantum system is left to evolve for a fixed number of Floquet cycles. In our case, the simulation software yields a measurement after each individual Floquet cycle. Lastly, the measurement output is used in the classical training on the classification labels. In the following sections, all of these steps are explained in depth and presented as we used them.

2.1 Feature Reduction

One crucial problem in noisy and modern quantum computing is that the size of quantum systems is highly limited. Widely used quantum computers rarely consist of systems with more than a few dozen physical qubits. Consequently, processing data with a lot of features, e.g. image data, is not a trivial, straightforward task. The original, most common approach is to use Principal Component Analysis (PCA). We also follow this path, while some other current works already employ more complex schemes like Autoencoders [12]. Although Autoencoders are better at encoding information in the same number of vari-

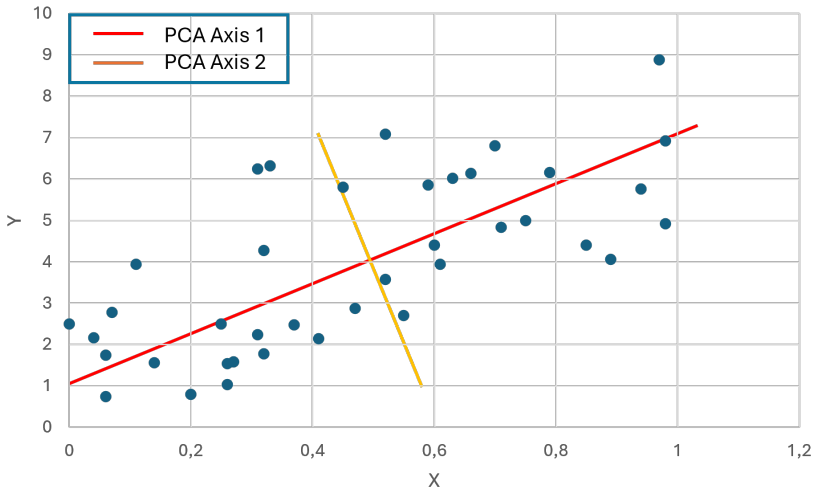


Figure 2.1: Visual representation of PCA applied for 2-dimensional data.

ables [12], they also require a more complex and time-consuming process to create. The inherent linearity and complexity present in deep neural networks lead to a black-box nature of the encoding method. This is not the case of PCA: As we will see, in contrast to Autoencoders it is a mathematically simple, well established and fast method to encode a large share of the variation present in a system in fewer features.
The crucial idea behind PCA is to map the original features to a new set of features that each capture as much variety as possible while keeping orthogonality to all previous PCA components. Naturally, the largest share of the present variety is stored in the first components of the PCA. This aligns well with our goal of reducing a data set with a large number of features to a data set with fewer features while keeping as much information about the variations present as possible.
In our short explanation of PCA we follow the path outlined by Lindsay Smith [17].
While the first step of each feature reduction process is to gather and understand the data, this is fairly straightforward for the mnist-digit data set. For a PCA, it is essential that all features are of the same scale and have a mean equal to 0. Consequently, we start by standard scaling our data. This step is crucial as otherwise we would be comparing the variance of different scales, disguising the real varieties.
Secondly, the covariance matrix must be calculated for all features. Afterwards, the eigenvectors and eigenvalues of the covariance matrix must be calculated. The eigenvectors define the new axes of our transformed data in units of the old axes. For a random 2D data set we have displayed the two eigenvectors as PCA axis 1 and 2 in figure2.1.
The first eigenvector is equivalent to a linear regression. All subsequent eigenvectors must be orthogonal to the previous ones. In the given example, this immediately yields the second new axis. By calculating the eigenvectors, we have found the lines that best characterize the data. Now, we move on to transforming the data itself. First, we sort our eigenvectors by the largest eigenvalues. These characterize the most important relation-

ships observed in the data. Although in principle we could keep all components, our goal is a feature reduction. Consequently, we choose to keep the first $p < n$ eigenvectors to keep. In this case, n is the number of all calculated eigenvectors. Although we lose some information making the choice $p < n$, we ensured that we keep as much information as possible by calculating the eigenvectors and ordering them. The eigenvectors are now appended column-wise to form a matrix. This matrix as well as the standard scaled data are now transposed and multiplied via matrix multiplication. The resulting matrix is the transformed data that is of smaller size than the original data, but still carries a large share of the variation present in the original data set. This can be seen in 4.1 where we present results for applying the PCA on our data set. More information in our application of PCA can be found in section 4.1.

2.2 Encoding

The next parameter to fix when working with QELMs is the choice of an encoding method. After the classical data has been reduced in the previous step, the challenge remains to choose an efficient yet robust way of encoding the data into our quantum system. Various encoding methods have already been developed, each with its advantages and limitations[18].

Here, we will only present angle encoding and dense angle encoding. Dense angle encoding is currently the only encoding method implemented in the software package used in this work. It builds on the well established angle encoding [19] [20] that only encodes one feature per qubit. Our presentation of the encoding methods originates from [18]. In the following, we will encode a general data vector with n components $x_i \in [0, 1]$. The angle encoding is especially suitable for today's NISQ hardware. The state of the entire system is encoded to be:

$$|\mathbf{x}\rangle = \bigotimes_{i=0}^{N} \cos(x_i) |0\rangle + \sin(x_i) |1\rangle \tag{2.1}$$

Obviously, this encoding enables us to encode 1 feature x_i per qubit. Its generalization is known as dense angle encoding. This is the encoding we are using throughout this work. While the regular angle encoding leaves the relative phase of the state aside, the dense angle encoding leverages this parameter as well.

$$|\mathbf{x}\rangle = \bigotimes_{i=0}^{\frac{N}{2}} \cos(\pi x_{2i-1}) |0\rangle + e^{2\pi i x_{2i}} \sin(\pi x_{2i-1}) |1\rangle \tag{2.2}$$

This encoding utilizes the entire Bloch sphere by encoding the values on the entire sphere. Another advantage is that the dense angle encoding is able to identify decision boundaries exhibiting sinusoidal structure instead of only straight lines[18]. This should ensure that the quantum classifier in the end performs better on arbitrary data sets whose decision

boundaries cannot be estimated to be linear. Furthermore, [18] showed that the dense angle encoding is particularly robust when encountering noise. While in this work we are not investigating the effect of noise, it will be a topic for the future as the functionality is already implemented in the simulation software.

2.3 Evolution of the Quantum System

As Hamiltonian we decided to work with an innovative physical system named many-body localized discrete time crystal that can be described with Floquet theory. Our presentation of the physical background follows the one layed out by Ippoliti et al [21]. Using an MBL-DTC is a common choice for Hamiltonians in the field of QRC and QELM[12][15]. The goal of Floquet theory is to find solutions to time-dependent Schrödinger equations with time dependent Hamiltonians. The Floquet unitary also shown in equation 4.2 can be split into three parts:

$$\hat{U}_F = e^{-\frac{i}{2}\sum_i h_i \hat{\sigma}_i^z} e^{-\frac{i}{4}\sum_i J_i \hat{\sigma}_i^z \hat{\sigma}_{i+1}^z} e^{-\frac{i}{2}\pi g \sum_i \hat{\sigma}_i^x} \tag{2.3}$$

Clearly an Ising-like Hamiltonian containing the interaction part between neighboring sites with hopping parameter J_i and the external magnetic field with parameter h_i, we can also see a third part of the operator. This part is the external drive that is responsible for the rich dynamics observed in [22].
In Floquet theory, Hamiltonians can exhibit discrete time-translation symmetry such that $H(t) = H(t+T)$ for a fixed parameter T holds. The discrete time crystal has the property to spontaneously break this symmetry by exhibiting subharmonic responses that can be described as $H(t) = H(t+mT)$ where T is again a constant shift but $m \in \mathbb{Z}$. Consequently, the physical system responds with an oscillating behavior to the external drive, but the oscillation period is an integer multiple of the drive period. An analogy for this peculiar behavior would be that the system 'chooses' to oscillate at a certain frequency like a clock, but it does not sync with the external drive applied to it. The period multiplexing exhibited by discrete time crystals occur vastly for single- or few-body systems [23] but are unexpected in many-body systems like ours. The reason is that many-body systems tend to absorb energy from their drive and to thermalize in the long term. However, in Quantum Mechanics many-body localization can take place to prevent this process[24, 25, 26, 27, 28, 29]. Although equilibrium is not achieved in these systems, the eigenstates of the many-body Hamiltonian can display order, which is known as eigenstate order [30]. This order exists throughout the entire quasi-energy spectrum. Unlike equilibrium systems, who are solely determined by their ground state, the MBL-DTC's order is encoded in its individual eigenstates. Consequently, each state of the system has an inherent order defining its behavior over time. The astonishing result of these dynamics is that, if the many-body localization happens, the DTC's oscillations are infinitely long-lived. Furthermore, the DTC phase is stable against weak perturbations in terms of imperfect rotations (in our case meaning $g \lessapprox 1$), generic perturbations (meaning the interaction part of the Hamiltonian is unequal to 0) or even longitudinal magnetic fields that can explicitly break Ising symmetry, in our case the first part of the Hamiltonian in equation 2.3. A crucial advantage of this system

choice is that, unlike other ongoing research in the field of Quantum Computing, it is already implementable today.

2.4 Measurement

The next layer of a QELM is the so called measurement layer. This means that we need to measure previously fixed observables on our quantum system. The default approach is to measure the observables in the computational basis. For example, Sakurai et al. [15] measure in the single qubit Pauli-Z basis. The choice of observables to measure is crucial as these are the values to build the feature vector with at a later stage. We will see that the training success does depend lightly on the observables that we measure. Within this work, we will work with measurements for all three Pauli matrices. Furthermore, we will work with the correlations between these observables for two qubits. While the software package enables us to measure Pauli-X, Pauli-Y and Pauli-Z on each qubit for every time step, the correlations are measured per pre-defined qubit only. However, when comparing the effects of different observables in section 5.4, we will ensure that the respective feature vectors have the same dimension. Unlike in other works [31], we will also post-process the measurements to investigate the effects of this post-processing in chapter 6.

2.5 Training

For the training process we chose a different method compared to other investigations [12][15]. Inspired by the common approach in the field of classical reservoir computing [32], we decided to use Ridge regression instead of a simple neural network. There are two major advantages of Ridge regression compared to the standard approach: A very low demand for computational resources and therefore faster, more efficient training and secondly the lower level of complexity. This lower level of complexity, originating from the linear nature of ridge regression, makes the results easier to interpret. The fact that there is only one hyperparameter to tune simplifies the training process further. In the interest of completeness we will give a short overview of Ridge regression's mechanism. In our analysis we used sklearn's implementation. Ridge regression is especially suited to problems with a vast amount of parameters that tend to show a high level of collinearity. However, it is an intrinsic features of PCA to remove all collinearity from the data. Since the data are fed into a quantum system and measured after some time evolution, we cannot conclude that this remains valid for the training process. In paragraph 4.2.4 we will see that our model exhibits primarily no dependency on the choice of the Ridge penalty parameter. Consequently, we could probably have also used linear regression in our analysis. However, due to the close relationship to methods and physical models in RC and QRC we decided to stick with Ridge regression.

Our theoretical foundation of Ridge regression originates from [33]. Let us consider a regression problem with a data matrix $\mathbf{X}$. We would like to fit this data set to match a vector of labels $\mathbf{Y}$. For multiple linear regression, this leads to the equation

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon \tag{2.4}$$

with errors ϵ. Assuming that our errors are distributed normally, unbiased linear estimation with minimum variance yields the standard equation to estimate β:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{2.5}$$

This estimator is the regular choice for multiple linear regression problems. However, in problems with high collinearity or data sets with a larger number of features than observations, it has proven insufficient. Consequently, ridge regression has been developed. It adds a penalty to the calculation of the best estimator. This penalty ensures lower sensitivity to overfitting and reduces the instability arising from linear regression in the described cases. The formula to calculate the Ridge estimator is as follows:

$$\hat{\beta} = [\mathbf{X}^T\mathbf{X} + k\mathbb{I}]^{-1}\mathbf{X}^T\mathbf{Y} \tag{2.6}$$

Obviously, the equation now includes a variable k known as the penalty parameter. In general, the penalty parameter should be chosen carefully, depending on the respective problem.

Chapter 3

Current Status of Research

As Quantum Computing and in particular Quantum Machine Learning are relatively new fields, not too much work has already been done to build on. However, of utmost importance to us are the previously mentioned paper of Sakurai [15], the paper of Peña et al. [11] and the paper mentioning the potentials and limitations of Quantum Extreme Learning Machines [31]. Furthermore, only in September 2024 a paper of De Lorenzis et al. [12] has been published that investigated QELM's performance on the mnist data set. We will give a short overview of these crucial papers that our work is building on.

The work of Peña a et al. differs from the others mentioned in that it investigated Quantum Reservoir Computing rather than Quantum Extreme Learning Machines. Although the prediction task and the algorithm to achieve the prediction are not equal, the quantum system used in the respective framework can be. Even though we are not using the same Floquet cycles, both systems exhibit similar phase. Consequently, their final result that the QRC performs well in the ergodic phase and at the phase transition is a first hint to our own results. We will later see that for our classification problem we did not reproduce these results entirely. Another similarity is that both their and our Hamiltonian rely on Parameter draws for the site and magnetic parameters from a uniform distribution. This aspect distinguishes our work from the related work of Sakurai et al. [15] also investigating classification performance on the mnist-digit data set with a QELM.

In their article 'Quantum Extreme Reservoir Computation Utilizing Scale-Free Networks' Sakurai et al. fixed their site parameter at a constant value and did not include a magnetic field. Although the paper's name indicates that a time series problem is solved, the term reservoir only refers to the quantum reservoir employed in the QELM. We generally refer to this 'quantum reservoir' as the 'quantum system' or 'quantum layer' of a QELM. This differentiation is performed due to the inherently different tasks that both frameworks are supposed to solve. Sakurai et al. leverage an artificial neural network with only one layer (ONN) as classical training algorithm. Their results are astonishing, as they achieved classification accuracies on the test data of more than 96% while using chain sizes of maximally 11. However, no details are provided on the numbers of observables measured, potential

post-processing methods and therefore the size of the feature vector used in the training process. It is one of the goals of this work to shed some lights on these aspects while trying to partially reproduce their results. However, our numerical model of the Hamiltonian worked in a slightly different way. While Sakurai et al. calculated all $\frac{1}{n}$ interactions within the system, our simulation software only calculated the nearest neighbor interactions. The differences in the numerical simulations of the respective systems as well as the not entirely identical Floquet cycles of course limit the comparability.

In contrast, Innocenti et al. published a more abstract analysis of the general behaviors of QELMs. They started by comparing the classical ELMs, the basis of QELM. Furthermore, they discussed the performance of QELMs injecting the input data only once or several times during the evolution process. This interesting approach of feeding data into the quantum system in several steps is usually performed when working with time series data in QRCs. However, here a copy of the input data was fed into the system marking a promising idea of merging the two fields. As expected, they were able to lower the error during training significantly by injecting the data multiple times. Another crucial finding of their work was that, when working with quantum devices, noise inevitably severely damages the performance of a QELM. As we are working with digitally simulated data of unitary evolutions, this does not impact us at this time. However, when implementing quantum simulators on physical hardware, these aspects will play a key role.

Lastly, we will discuss the article most similar to our own work. De Lorenzis et al. worked with a similar Hamiltonian structure as we did and also used the QELM to classify the mnist digit data set. Apart from that, they focused on the investigation of different approaches for the QELM layers. More precisely, they compared several approaches for feature reduction, data encoding and also varied the Hamiltonian. As expected, they found that an autoencoder is more suited to capture the variety in a data set than a simple PCA. Their results suggest that the best encoding methods for a QELM are dense angle encoding and uniform Bloch sphere encoding. Given that we are also using dense angle encoding this is a satisfying result. They did not find any dependency on the Hamiltonian used for the quantum system. This is not too surprising given that the three Hamiltonians used were very similar and all exhibit complex dynamics that should be sufficient to explore a lot of states in Hilbert space.

Chapter 4

Our Model: Pre-Processing, Quantum System and Post-Processing

In this chapter we explore the QELM we used in our investigation. We discuss the application of all steps outlined in 2. First, we talk about the data set we used and the PCA applied to it. Then, we delve deeper into the simulation software we used and get into more detail of the physical model. We give a short overview of the default configuration used by us. Lastly, we shortly discuss the post-processing of the measurements.

4.1 MNIST-Digit Data and PCA

The data set used in our analysis is fairly standard: mnist's handwritten digit data set[16]. This data set contains 70,000 black and white images with 28x28 pixels displaying handwritten single digits. Although these images are not predestined for QELMs, they have evolved into a standard data set to compare QELM configurations [12] [15]. It should be noted that QELM research investigating real-life data has already been performed as well [34].Within this work, our standard approach is to split the data set into three smaller sets: a training, a validation and a testing set. The training data set contains 50,000 observations and the validation as well as the testing data set contain 10,000 observations each. Although we do not need the validation data set when applying Ridge regression, we still keep this splitting for the comparison with other training methods in chapter 6. The loss of accuracy caused by this smaller training data set should be negligible given that we are not trying to beat any accuracy records. We are mostly performing relative comparisons between configurations that all experienced the same data split. The splitting is performed randomly, but reproducible with the sklearn seed set to 100.
Before splitting, the PCA discussed in section 2.1 is applied to perform the feature reduction. As we are losing some information by choosing fewer PCA components than the number of original features in the data, we display the remaining explained variance in the

model for various PCA components in 4.1

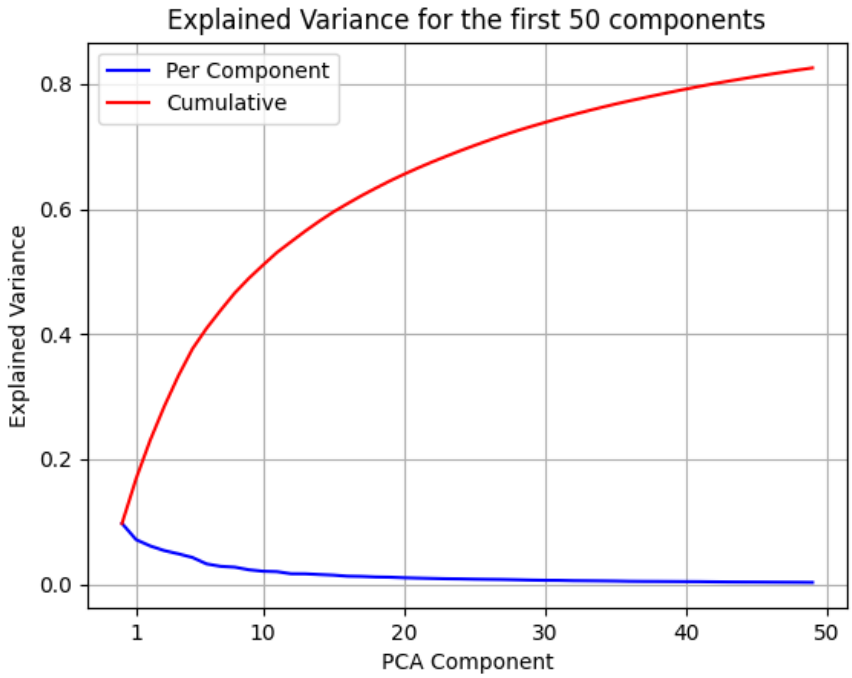


Figure 4.1: (Cumulatively) explained variance for up to 50 PCA components of mnist digit data set.

In the interest of completeness, we show the explained variance cumulatively and individually for up to 50 PCA components. However, due to simulation and time constraints, we limit ourselves to a maximum of 40 PCA components in this work. Using dense angle encoding, we feed 40 PCA components into 20 physical qubits. Although the first 20 PCA components already contain 60% of the variation present in the original data set, the inclusion of further PCA components is obviously worth it, as the cumulative explained variance increases to about 80% for the first 40 PCA components. Even though further PCA components would push this value even higher, we will settle for 40 components as a compromise between information fed into the system and runtime for our classical simulations of a quantum system.

In section 2.2 we already discussed that we use dense angle encoding to feed our data into the quantum system. However, as displayed in figure 4.2, we scale the PCA components to the interval $[0, \pi]$. Consequently, we adjust the mapping that encodes the data a bit. It matches the mapping from Sakurai et al. [15]. After calculating the first $2N$ PCA components, we split them in the middle to yield the values for the angles θ_i and ϕ_i as can be seen in figure 4.2. These vectors are read into the system per qubit i to be in the

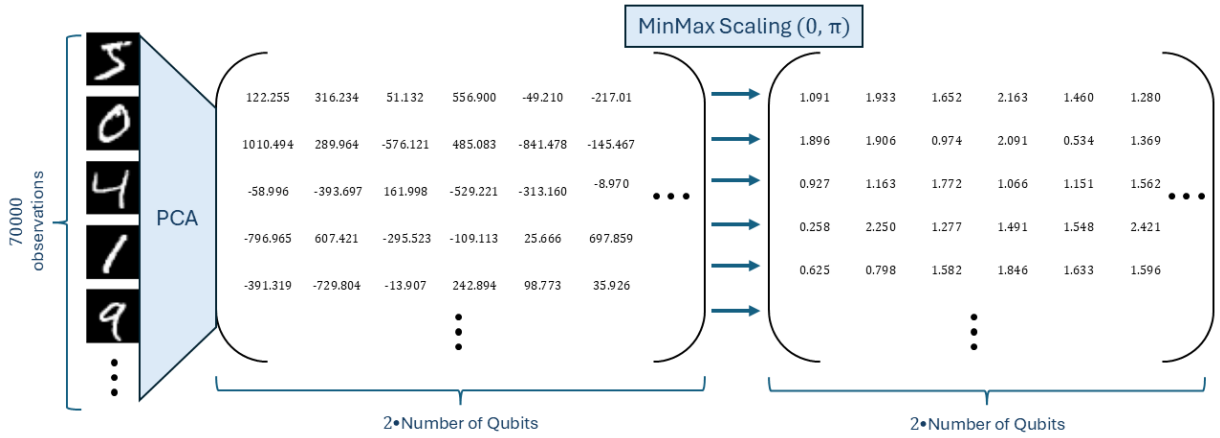


Figure 4.2: Process of encoding mnist data in QElM via PCA and MinMax scaling to $[0, \pi]$.

following input state:

$$|\mathbf{x}_i\rangle = \cos\left(\frac{\theta_i}{2}\right)|0\rangle + e^{i\phi_i}\sin\left(\frac{\theta_i}{2}\right)|1\rangle \tag{4.1}$$

The entire data set is now encoded in a matrix with $2N$ columns where N is the number of qubits. The next step is to split these columns into two and use each resulting part for one of the angles of dense angle encoding as displayed in figure 4.3. This column vector represents one encoded image and is equivalent to one row of the final matrix in figure 4.2.

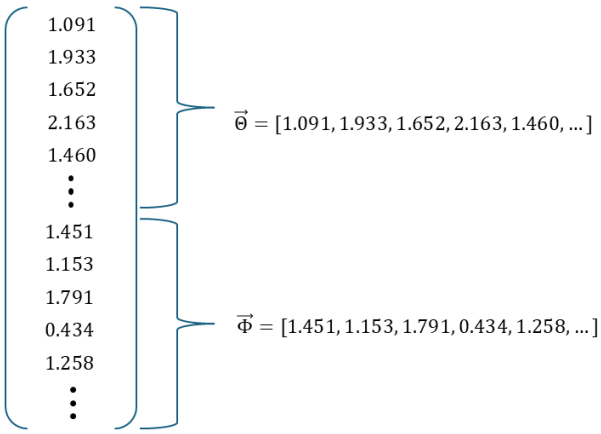


Figure 4.3: Encoding of one image with $2 \cdot N$ PCA components into Φ and Θ with dimension N

This state serves as the initial state of our quantum system. The evolution of the system and how we simulate it are discussed in the following section.

4.2 Quantum System and Simulation Software

4.2.1 Simulation Software

In section 2.3 we discussed that we decided to use an MBL-DTC as quantum system of our QELM. As we use the simulation software of Camacho and Fauseweh [35], the operator presented in their paper 'Prolonging a discrete time crystal by quantum-classical feedback' is the same operator we are using.
However, the simulation software can do many more things that were of relevance to our work. While we focus on the closed system simulations, the original background of the creation of the software was to implement a correction scheme based on a quantum-classical combination to account for potential noise effects on the real world hardware implementation. As can already be seen from this, apart from the closed unitary evolution the software can be used to perform noise evolutions and noise evolutions with corrections (in [35] denoted as protocol (ii) and (iii)). In theory, one could of course use also both these protocols to conduct the simulations. We limit ourselves to the unitary case and do not introduce noise. The impressive property of introducing corrections to simulated noise is also left aside by us due to time constraints. However, for further research an investigation of the validity of our results given noise as well as correcting for it is definitely worth a deeper analysis.

Another key aspect of the simulation software is the availability of unusually large systems. While simulating small quantum system with a one-digit number of qubits is possible on a regular local machine, quantum simulations by their nature scale exponentially [1]. Tackling the problem of simulating larger quantum systems requires super computers as well as proper simulation methods. Camacho and Fauseweh in this case employed tensor networks in combination with representing the system as a matrix product density operator (MPDO). Fortunately, this enabled us to conduct simulations with up to $L = 20$ qubits in a reasonable amount of time.

4.2.2 Physical Model

The physical model is a periodically kicked 1D-Ising model, in accordance with [36]. This model exhibits an extraordinary physical behavior when choosing the phase parameter appropriately, leading to a many body localized discrete time crystal as described in 2.3. This MBL-DTC phase has already been observed on real NISQ-quantum hardware [22] [37]. Consequently, making progress in this field and generating new insights can have immediate implications for real-life NISQ-quantum devices already existing today.
The mathematical operator defining this system is presented in equation 4.2. This operator is used for all time evolutions and only varies depending on the choices for each variable. It should be noted that our work differs from the work conducted by Sakurai et al [15] as their Hamiltonian did not contain a magnetic field part. Consequently, the physical results

are not directly transferrable.

$$U_F = e^{-i\frac{T}{4}\sum_{j=1}^{L-1} J_j \sigma_j^z \sigma_{j+1}^z} e^{-i\frac{T}{2}\sum_{j=1}^{L} h_j \sigma_j^z} e^{-i\frac{\pi g}{2} T \sum_{j=1}^{L} \sigma_j^x} \tag{4.2}$$

The physical system originates from [21] and exhibits three different physical phases: paramagnetic, thermal and MBL-DTC, depending on the pulse parameter g. In this work, we will call the parameter g the phase parameter since it is the parameter responsible for the system's physical phase. As it is our goal to investigate the performance of a QELM for such physical systems, we will focus on the thermal and the MBL-DTC phases as well as the transition region between the two.
The interval to choose g from is $[0, 1]$ with $g = 1$ denoting an entire π-pulse on the Bloch sphere and $g = 0$ removing the external drive entirely. A value of $g = 1$ would mean an external drive of an entire π-pulse flipping all qubits in one step and returning them to their original state after two periods. Consequently, the parameter interval of interest lies in between. According to [22], a value of $g = 0.60$ is already deep in the thermal phase. Therefore, in this work we will focus on values for $g \in [0.70, 0.99]$. For $g \sim 1$ we expect the MBL-DTC phase, while around $g = 0.84$ the transition from discrete time crystal to the thermal phase should take place. These partial π-pulses are what we vary throughout our analysis.
The Pauli-matrices σ_j^x and σ_j^z are applied on the qubit on site j respectively. The parameter T is set to 1 covering one Floquet period. Therefore, the time steps playing a role in all simulations listed below are equivalent to the Floquet cycles that the quantum system experiences. Since L is the length of the 1D-system it also is the number of qubits used in each simulation. The two remaining variables are the coupling parameters J_j and the magnetic field strengths h_j. These are drawn randomly from uniform distributions with $J_j \in [-1.5\pi, -0.5\pi]$ and $h_j \in [-\pi, \pi]$. These random and independent choices introduce stochastical dependence into our simulations. When interpreting our results we need to be more careful than when interpreting purely analytical results as there is a chance of us observing results that are purely based on a fortunate choice of random parameters. This is the reasoning behind us performing 10 realizations for each simulation we conduct.

4.2.3 Standard Settings

While it would have been possible to modify the intervals of which to choose J_j and h_j, we did not make use of this feature. Previous reserach [12] suggests that small deviations from the Hamiltonian defined in 4.2 do not affect the results strongly. Furthermore, the goal of this work is not to investigate different DTC-Hamiltonians, but to focus on one physical system and gain insights about the potential of using this system for QELMs.
The number of values encoded in the quantum system of a QELM is one of the key aspects of the method. It determines how much information enters the system in the first place. The PCA components described in 4.1 are the values fed into into the system with the presented dense angle encoding. Consequently, changing the chain size L in the quantum

system can have a strong impact on the overall results. Generally, we use 8 or 10 qubits in our simulations. However, as we also explore the effect of the system size, the parameter is of course varied for this investigation. Furthermore, when we have the goal to achieve very strong results, a large feature vector for training is necessary. To achieve this we might also use larger systems like $L = 15$ or $L = 20$.

The Floquet unitary given in equation 2.3 is the operator applied in each time step. When speaking of time steps or Floquet cycles in this work we are referring to the number of iterative applications of this operator. Of great use for us is the property of the simulation software to measure every requested observable in every time step. This yields great flexibility when investigating which aspects of the quantum system affected the results strongest without having to perform too many simulations. More precisely, when we request a simulation of size $L = 10$ with $t = 20$ time steps (i.e. Floquet cycles) and the observables s_x, s_y and s_z, the software returns *numpy*-arrays with 10 measurements for each observables per time step summing up to 210 values per observable in total. The first 10 measurements are performed in the 0'th time step, before any evolution has been applied. This vast supply of simulated measured observables is one of the crucial aspects enabling us to conduct this work.

One parameter we have full control of is the number of time steps to simulate. We vary this parameter depending on the goal of our analysis from $t = 4$ up to $t = 100$. The default choice remains $t = 20$. This ensures that the system evolved for some time to experience the desired physical effects. At the same time, the evolution is not too long to lose the encoded information entirely.

For the observables measured by the simulation software the situation is more complex. Every simulation will contain the three Pauli matrices X, Y and Z on every qubit in every time step. However, a recent extension of the software package enabled us to also measure correlations of the observables between different qubits. Due to the nature of the simulation software, technically specifying the measurement of all 2-qubit observables is cumbersome. Consequently, we conducted one simulation that included the measurement of 30 measurement 2-qubit observables, while all other simulations yielded only a subset or left the 2-qubit observables aside. As the system size varies, this subset is of course not constant over all simulations. However, if two simulations have the same chain size L, they are also equal in the measured 2-qubit observables. The 2-qubit observables are chosen such that they cover as many aspects of the chain as possible, some spanning over the entire chain while others were measured on neighboring qubits or on qubits with an intermediate distance.

A disadvantage for our work a disadvantage is the pre-defined encoding setting. While in related works the aspect of the encoding has been investigated and proven to be of great importance for the quality of the results [12] [18], our encoding method must remain fixed. While this lowers the complexity of finding the right configuration and turns our focus to the quantum system as well as the post-processing, this also limits the generalizability of our results. In our case, the data points to read into the quantum system are fixed to be twice the size of the 1D chain. Fortunately, other works have shown that the encoding used in our work is a favorable choice [18].

4.2.4 Post-Processing

As we are comparing results created by many training processes, it is necessary to discuss and, as far as possible, fix the potential parameters of the training layer of the QELM. Before running any training process, it is necessary to determine the data set to train on. If not specified otherwise, we train on the measurement results after the 20^{th} time step. When applying temporal multiplexing, we usually use all available time steps from 1 to 20. As observables, we most often leverage the expectation values of Pauli-X and Pauli-Z measurements. This is the most simple choice possible, as using only one of the two matrices would neglect one axis that we considered when encoding our data.

The next step is to determine which of the various existing training methods to use as a default. As already mentioned in 2.5, due to the close relationship to Quantum Reservoir Computing we decided to work with Ridge Regression. Compared to Sakurai's choice of a one-layer neural network we have fewer hyperparameters to tune. This results in a simpler and more comparable training process as we do not need to choose the number of epochs, the batch size, the optimizer, or the learning rate.
This simpler configuration also lies in the interest of achieving more Explainable AI systems. Another advantage that grows more crucial with increasingly bigger models in Machine Learning is the faster training process, which corresponds to lower resource consumption. As a short study of other choices for training methods we investigate the usage of linear support vector machines or a one-layer network in analogy to Sakurai et al. in chapter 6.

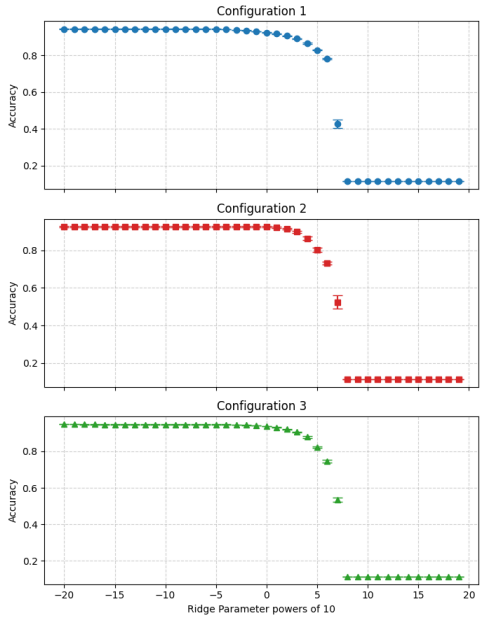


Figure 4.4: Different training configurations to showcase low sensitivity to Ridge regression.

Now that we established Ridge regression as the default training algorithm, it is necessary to discuss the choice of Ridge's only hyperparameter: the Ridge penalty, in RC literature known as either α or β. Our work shows that our training processes are primarily insensitive to the order of magnitude of the Ridge penalty. Conducting a hyperparameter scan to support this claim for every analysis we present in this work would have taken too

much time. Consequently, we chose three different configurations that are very similar to various parts of this work and tested the performance for $\alpha \in [10^{-20}, 10^{20}]$. The results are presented in Figure 4.4. As can be seen, all three configurations exhibit the same behavior. They perform well for exponents below 0 and rapidly deteriorate for penalties larger than 10^5. Obviously, ridge penalties larger than 10^8 turn the training process into a random model that does not depend on input or training configuration. It predicts the classes with an accuracy of about 10%, exactly the results we would expect for a random classification. Given these results, we are free to choose our default ridge penalty from the plateau shown for all exponents from -20 to 0. In the interest of computational stability, we settled for 10^{-5} as our standard choice.

A useful approach in the field of RC and QRC is to create nonlinear combinations of the feature vector with itself [38]. These combinations are appended to the original feature vector, multiplying its size by the degree of nonlinearity. This part of the QELM is called readout and we will also make use of it in chapter 6. However, when investigating the nature of the quantum layer of the QELM, we will focus on a so-called 'linear' readout, meaning that we will not create nonlinear combinations of the feature vector. In contrast to that, chapter/section 6.3 will be an in-depth analysis of the effect of readout and its potential extensions.
Lastly, when tackling classification problems it is necessary to decide on an appropriate metric. Fortunately, the mnist-digit data set is sufficiently large and totally balanced for each digit to settle for the most basic choice. Consequently, it makes sense to stick to purely using accuracy as a metric. In case of imbalance one might consider using recall, precision or F1-score, in case of the given data set this is not necessary.

Chapter 5

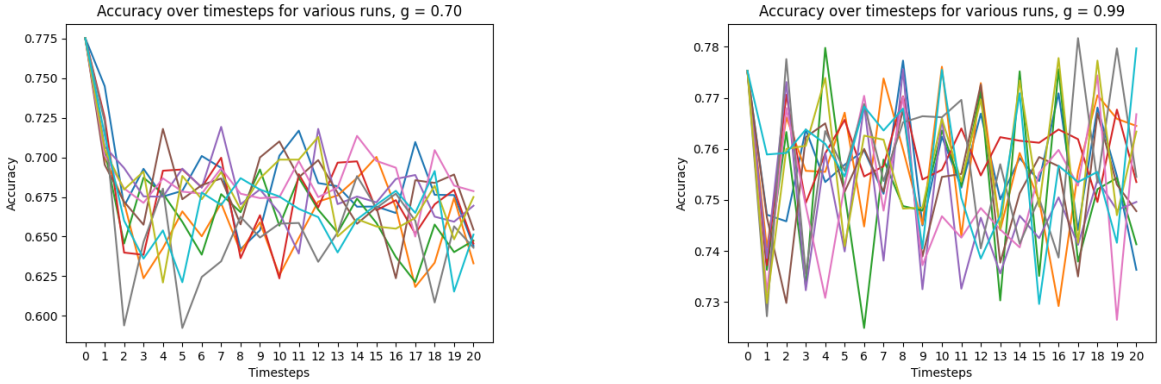
Investigating the Quantum System's Effects

In this chapter, we have a look at the effect of the quantum layer on the classification accuracy. In the last chapter we mentioned that the random draw of numbers introduces a stochastic dependency. This is discussed in 5.1. As our Hamiltonian exhibits a thermal as well as an MBL-DTC phase, the phase will be the next aspect investigated in 5.2. We conclude this chapter with a presentation of the effect of different chain sizes compared to classical training in 5.3 and by discussing results for using different sets of observables in the training process in 5.4.

5.1 Dependency of Results on Drawn Numbers

One crucial aspect of the software package we use to simulate the quantum system is that the parameters J_j and h_j are drawn at random for all sites j. Therefore, one run might yield outstanding results that are purely based on a fortunate choice of parameters that is experimentally difficult to achieve. Producing reliable results with random choices of parameters is more attractive in the long term considering real world difficulties with reliable quantum computers. The divergence of different runs can be observed in Figure 5.1.

While the data points only exist for each integer time step, the lines in the two graphs are connected to ensure that trends along the simulations are more visible. The first and at this point crucial observation is that the dependency on random draws truly must be considered. While all simulations from both plots show a certain level of fluctuation over the entire time period, some simulations perform consistently better than others. For example, in the thermal phase the blue line achieves results above average for almost all time steps. In contrast, the gray line in the same phase achieves bad results for almost all time steps. Keeping in mind that the classification accuracies after evolving the system range from 60% to 72.5% these differences must be considered. A similar conclusion can be drawn from the graph displaying the accuracies for the MBL-DTC phase. Consequently, it makes



(a) Time-crystal phase: g = 0.70

(b) Thermal phase: g = 0.70

Figure 5.1: Accuracy results per individual simulations for different phases

sense to perform several simulations for each configuration and create a statistic for each analysis we conduct. Otherwise, we might risk choosing a simulation with an unusually good or bad performance with non-generalizable results. If not otherwise stated, from now on we will always use ten simulations and present the arithmetic mean of their results. To account for the statistical dependency, the standard deviation belonging to each mean calculation will be used as an error bar around it.

Apart from the statistical dependency, we can already gain further interesting insights from the two graphs. First, we focus on the thermal phase graph. One aspect that immediately catches the eye is that the performance is best at the 0'th time step. As there is no 0'th time step, it corresponds to the training process on the encoded data. The accuracy at this point is calculated purely on the encoded PCA data without any evolution. After any number of Floquet cycles that we simulated the classification accuracy does not return to the original value. The physical explanation in this case would be that in the deep thermal phase of the system the information spreads in the system sufficiently quickly for a significant part of the link between input data and label to get lost without being regained later. This is a suboptimal result. It motivates our focus on different phases of the system when trying to achieve high classification accuracy.

Turning to the time-crystal phase of the system the situation is completely different. In this localized phase the information read into the system appears to be preserved. While most of the simulations clearly fluctuate, there are strong patterns and the amplitudes of the fluctuations are lower. Notably, some of the simulations achieve better performances than the training on the encoding in some time steps. This is remarkable considering the low dimension of the feature vector originating from only two observables that were measured on all eight qubits. Another interesting aspect that should not get lost is that not all simulations show the clear fluctuations described earlier. Some patterns are more

similar to the ones observed in the thermal phase, although the classification accuracy is still significantly higher. The reason for these diverging behaviors can only be found in the random choice of the parameters h_j and J_j, however, investigating them lies beyond the scope of this work. We consider this observation as further support to always build our insights and statements on a statistic of simulations rather than a single one. The two graphs yield the impression that the system is undergoing an evolution also in terms of classification capabilities when changing the phase parameter that leads to less random fluctuations. This impression is also supported by the same evaluation for a phase parameter of $g = 0.85$, whose results we added in the appendix A.1.
As we presented the necessity to conduct statistics as well as shown first results for the system's dependency on the physical phase, we will now move on to a deeper discussion of the phase effects.

5.2 Effects of Phase

When using a discrete time crystal Hamiltonian for a QELM, one of the key aspects to explore is the effect of phase on the training success. We have already gained some first impressions but now want to take a deeper, more detailed look. In our case, according to Camacho and Fauseweh [35], we should distinguish three different phase regions: the MBL-DTC phase, the thermalized phase and the transition from one phase to another. As discussed earlier, the state of the phase depends on the set of parameters h_i, J_i and g. However, as J_i and h_i are drawn randomly, g is the only parameter under our control. To investigate the effect of different phases, it makes sense to keep the training process as simple as possible. This should enable us to see the impact of different phases as clearly as possible. Consequently, we use the standard basic set-up with identity readout, observables Pauli-X and Pauli-Z, and a chain length of 8 qubits. First, we discuss the performance when measuring only once after 20 time steps. Then we will turn to a more complicated training process that makes use of combining measurements after 10 or 20 Floquet cycles. Although we will have a more in-depth discussion of potential effects of temporal multiplexing later, it makes sense to get a first impression which of the phases benefits the most from this technique.

The data points in graph 5.2 show the mean performance over ten runs, while the error bars represent one standard deviation around this mean. The considered parameter interval spans all regions of interest to us with g close to 1 creating the discrete time crystal phase and g lower than 0.85 being the thermal phase. Setting $g = 1$ would remove the external drive part of the Hamiltonian which is why we do not include it in this analysis. In steps of 0.01 we consider all parameter levels down to $g = 0.70$. At this point, the system should be deep in the thermalized region which is why we did not lower it any further. Although previous results from Peña et al [11] performed well at the transition, results from Sakurai et al. [15] showed a better performance for the thermalized region encouraging us to expect similar results. As our only observables to measure are Pauli-x

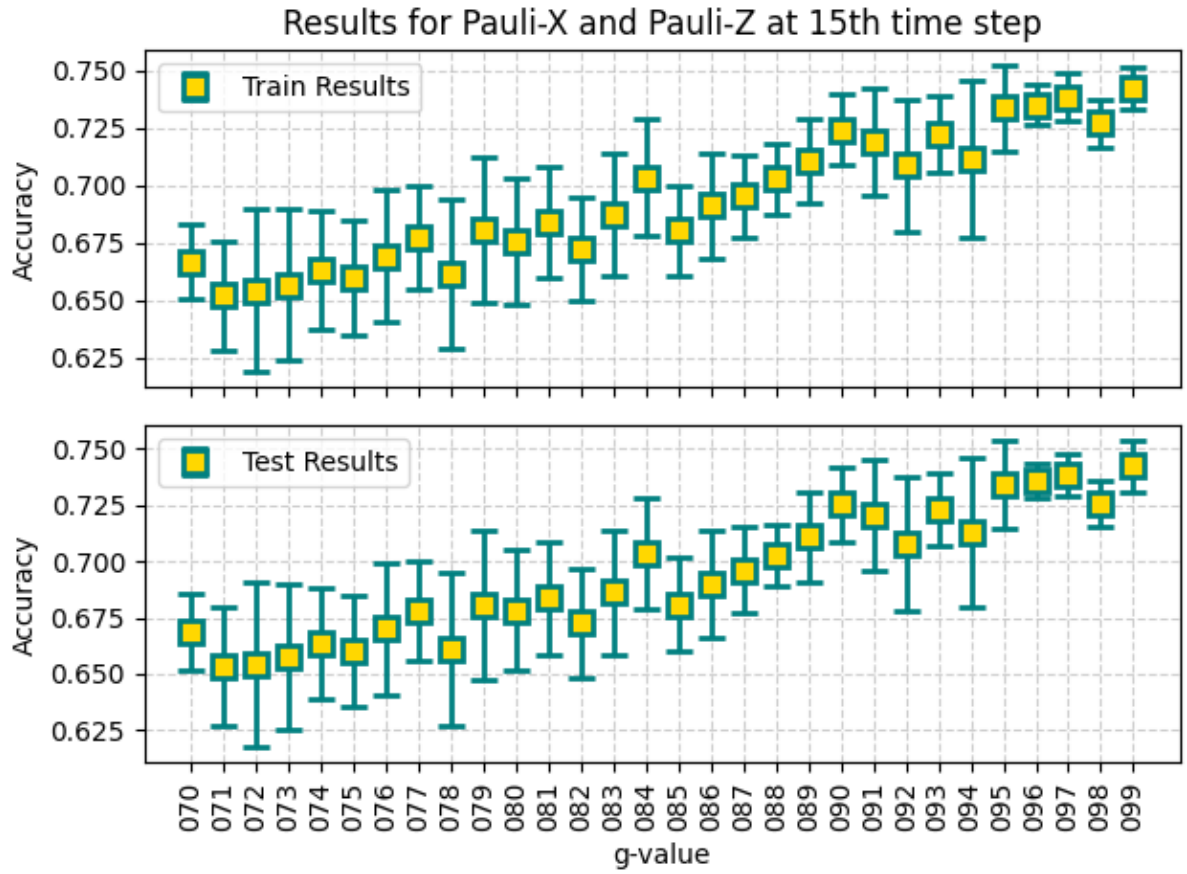


Figure 5.2: Train and Test results after 15 time steps without temporal multiplexing over all phase parameters.

and Pauli-z on each qubit, the feature vector used during training is 16-dimensional. We performed the analysis for 0 to 20 time steps. Of course, before the evolution, all phase parameters across all individual simulations yield the same classification accuracy as we are only training on the encoded data. Here, we present the results for measurements after 15 Floquet cycles. Given that the results are extremely similar for all possible choices of time steps, these work as proxies for the phase behavior. As comparison, we have added the results after 20 time steps in appendix A.3

We observe a clear positive correlation with increasing phase parameter. This aligns with our results from 5.1. The impression that performance is on average better for a higher phase value is now supported by a consistent upward trend over all choices for $g \in [0.70, 0.99]$. Furthermore, the lowest performance seems to be bound from below as it saturates in the region of $g \in [0.70, 0.77]$. At the other end of the considered interval, the performance also appears to have an upper bound as it saturates for values approaching $g = 0.97$. This behavior is not as clear for all time steps we consider, but does occur e.g. for a measurement after 15 time steps as shown in 5.2.

Consequently, we can conclude that, when training only on data originating from measurements obtained after a fixed number of time steps, the discrete time crystal performs significantly better than the thermal phase and the transition region. Thus, we find our assumption from 5.1 confirmed. Our results appear to yield contradictory results to Sakurai et al. [15]. However, they did not explain the configuration of their training process. We cannot conclude that they did not use temporal multiplexing, such that our results are not necessarily comparable.

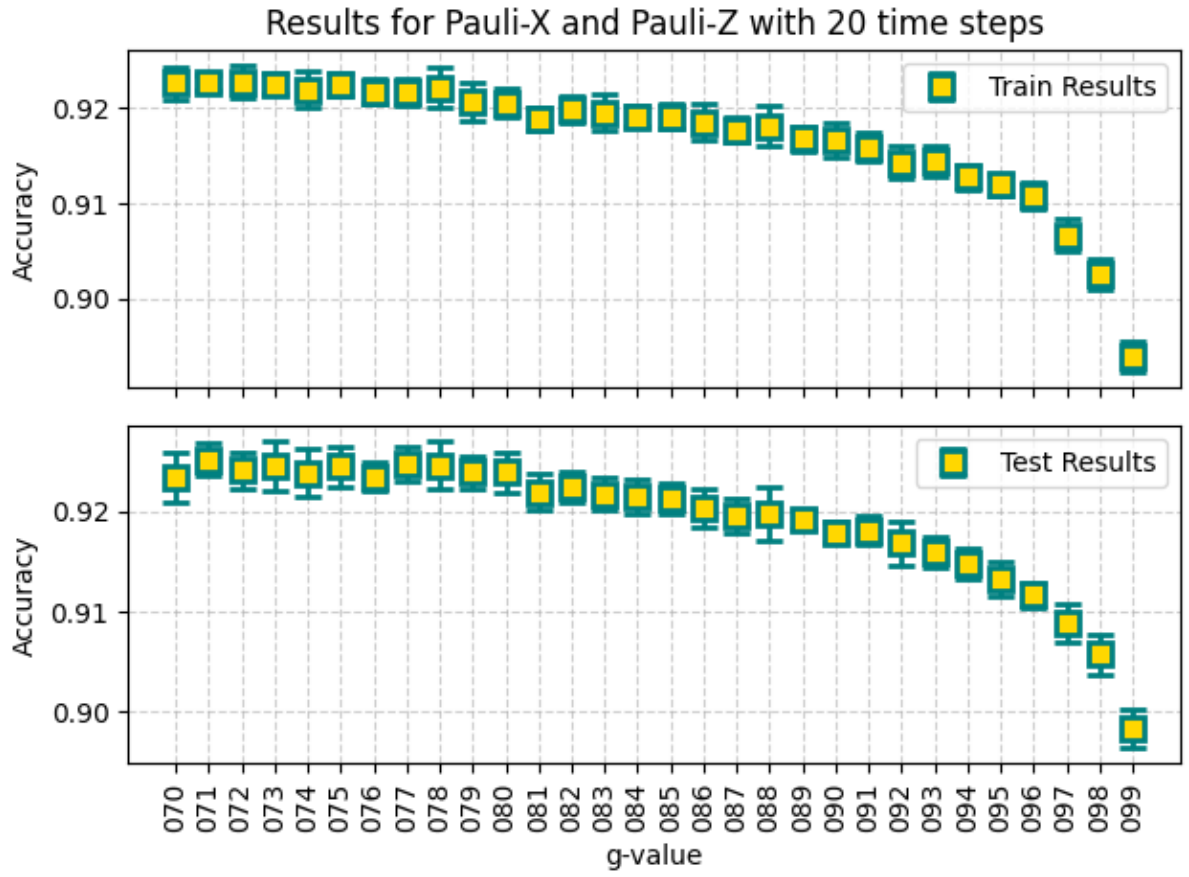


Figure 5.3: Classification accuracy for all 20 time steps combined.

So far, we have worked with very small feature vectors with only 16 entries. This will not be sufficient when trying to achieve competitive results. Throughout this work, we consider three ways of enlarging the feature vector: adding more observables measured in the same time step, adding measured data of the same observables from different time steps, or appending a non-linear combination of the existing feature vector to the feature vectors. As we regularly use the addition of measurements from multiple time steps later on, it makes sense to investigate the performance per phase parameter now.

The results for training on all 20 time steps are presented in graph 5.3. Obviously, the results are much better than previous analyses. This is based on the vast feature vector that was used in training. As we combine measurement data from 20 time steps, our feature vector grew from 16 dimensions to 336 dimensions. This explains the jump in classification accuracy from approximately $70 - 75\%$ to more than 90%.
Throughout this work, we generally present the classification results on the test data set. The foundation for this shortened presentation can already be seen in 5.3. This graph contains 30 different physical configurations with 10 simulations each and a large feature vector with 336 dimensions. In none of these training processes we can observe any relevant gap between training and test results. Consequently, from now on we will only present the results on the test data set as no additional insight can be gained from the train data results.
The second aspect immediately catching the eye is that the shape of the 'curve' has changed drastically compared to the same evaluation without temporal multiplexing. The previously best performing discrete time crystal phase now performs significantly worse than the thermal region and the transition region between the two. Clearly, the training process of the thermal region benefits much more from the temporal multiplexing than the training process using the discrete time crystal measurements. An explanation for this could be that the thermal phase is better at mixing the input data than the localized discrete time crystal. However, when training on a feature vector that is small due to only measuring at one time step, the advantage of the mixing turns into a disadvantage as for the training algorithm it is not possible to recognize the label from the low-dimensional feature vector. For the localized discrete time crystal the consideration of more time steps does not improve the performance just as much because the input data is mixed much less during the system's evolution. The localization that preserves the information in the state also prevents the beneficial mixing that occurs in the thermal phase.
To conclude the phase investigation we note that both the thermal and the MBL-DTC phase have their benefits. While the discrete time crystal phase was superior for trainings without temporal multiplexing, the same holds for the thermal phase when temporal multiplexing is leveraged during training. Furthermore, as the results look very similar for all choices of 0 to 20, the number of Floquet cycles that the system evolves before measurement appears to not be of great importance.

5.3 Effects of System Size

Another aspect that so far we considered only indirectly is the effect of the physical system's size. This should be considered mainly for two separate reasons. Firstly, increasing the system's size enables us to encode more information into the system. Every additional qubit can be used to encode two additional values. Secondly, we need a sufficiently large system to trigger the physical effects described in the paper of Camacho and Fauseweh [35]. A system with a chain length of two or three qubits will not yield the same richness in

dynamics as a system with 10 or more qubits. Consequently, in this chapter we have a look at this aspect of the QELM. We compare the performance of three different feature vectors created with QELM observables with classical performance of $2N$ PCA components. In this work, we used a feature vector built from Pauli-X and Pauli-Z observables as default configuration. To justify this, we have also shown the results for a configuration built from Pauli-X or Pauli-Z only in figure 5.4. It should be noted that the feature vectors built from one observable only are also only half of the size of the classical or QELM standard configuration. However, it is obvious that a training process containing only information from one Pauli matrix is not sufficient for good classification results.

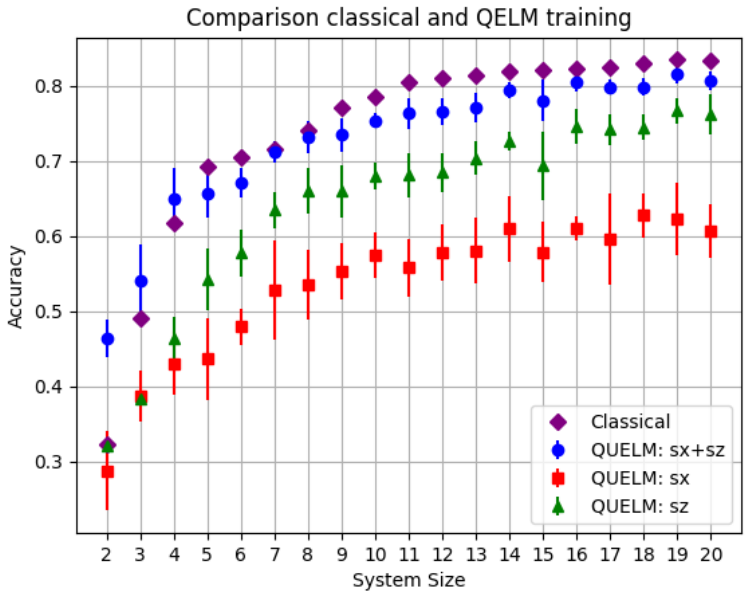


Figure 5.4: Comparison of classification accuracy for classical system and 1-qubit observables over system size.

As an input for the classical training we use the PCA components that were also calculated as input for the QELM. As the feature vector based on the QELM is twice the chain length (based on Pauli-X and Pauli-Z), we also need twice as many PCA inputs for the classical training. Hence, we train on feature vectors of dimensions 4 to 40 to get the results for 'Classical' and 'QELM: sx+sz'. Of course, as the classical result do not have any statistical dependency they also do not have any error bars. We will discuss the impact of different observable sets further in section 5.4.
Leaving the obvious underperformance of the configurations with one Pauli matrix aside, we observe a few trends. For very small physical systems, the QELM manages to beat the classical training process. However, for systems of with 5-8 qubits, the classical training performs at least equally well. For all larger systems, the classical results consistently exceed those of the QELM.

When trying to explain these results, it is worthwhile to keep in mind the nature of the data on which we are training. Although the first PCA components contain the largest share of the overall variation, training on their values when the entire feature vector is small could limit the results. The mnist digit data set contains 10 different categories. Ignoring variations within these categories, we quickly see that while all digits will have some pixels in common, they of course vary drastically in different regions of the image. Encoding this information in less than 10 PCA components appears to overstrain the method of PCA given that the feature vector's dimension is smaller than the numbers of potential categories to sort the data in. Therefore, it makes sense that the results are very low but quickly improve when adding more PCA components as they appear to contain the bit of information missing that the algorithm needs to distinguish properly. Keeping all this in mind, it is as surprising as satisfying that the QELM manages to achieve better results for such small system sizes. We measure our system after an evolution of 20 Floquet cycles. Due to this comparably long evolution, it appears that despite the MBL-DTC phase the information encoded in the system has had time to spread and mix within the chain. Consequently, the measurements appear to yield a more distinguishable data set than the pure PCA components could. Given that both training processes started with the same input data and were trained with the exact same algorithm, the dynamics within the physical system are the sole possible reason for the QELM results exceeding those achieved from classical training.

As mentioned above, with an increasing number of PCA components, the classical system manages to catch up to the results of the QELM. The QELM appears to be unable to take advantage of the additional information as much as the classical training. To us, it seems that the previously managed mixing that happens during the evolution period might be harming the performance.

Obviously, both the classical training and the physical system are benefiting from more PCA components and larger physical system sizes respectively. However, the classification accuracy reaches a plateau around 80% for both methods. While the additional information margin lowers with each PCA component, we suspect that the main reason for this plateau is the limited potential of ridge regression as a training algorithm. An insight to take from this chapter is that system sizes with 10 or more qubits already achieve a classification accuracy that is sufficient to compare different configurations. The increase in accuracy achieved by using more than 15 qubits should also be weighted against the increased difficulty and complexity of physically building or digitally simulating such a large quantum system. Although it would have been nicer if the QELM results consistently exceeded the classical results, we can conclude that this is not the case. We will need additional tools and methods from the quantum system itself as well as during the post-processing of the results. Then we can hope to achieve results that exceed those of the classical training also with ridge regression.

5.4 Training on Different Observables

5.4.1 Standard System

When training on the measurement output of a quantum system, another key topic is the list of observables that one should measure. A first result has already been presented in 5.4. Now we will have a closer look at this question by also including correlations between qubits in our analysis. As the choice of observables of course depends on the physical system one is using, we would like to especially consider whether it makes a difference to use observables that were measured on each qubit individually or correlation observables. In other works we have not found any comments regarding the observables that were measured for training except Sakurai's logical statement that in principal for systems with N qubits 2^N observables can be used for training [15]. As we are in the fortunate position of having a software package at hand that can measure a plethora of physical observables, we will delve deeper into the effects of using different observables configurations in the training process.
Firstly, we need to set the framework that we want to use for our investigation. Most importantly, when comparing the performance of different settings, the feature vector used in the training process must have the same size. Not taking this into account would result in our analysis suffering from a large level of uncertainty, given the strong dependency on the feature vector's size displayed in 5.3. Secondly, we would like to ensure that our system is large enough to experience complex physical dynamics. In this interest, we choose a system with 10 qubits. Also, we set the phase parameter to $g = 0.99$ such that our system exhibits the MBL-DTC behavior. This choice is made based on the results from section 5.1 where we showed that without temporal multiplexing the MBL-DTC phase yields better results. Lastly, we let the system evolve for 10 time steps. This is a compromise between long evolution times for quantum dynamics and the computational resources necessary to simulate the measurement of more observables. To conclude, we conduct the same analysis for the thermal phase with $g = 0.70$. The reason for this is to investigate the potential generalizability of our results.

5.4.2 Comparison 1-Qubit and 2-Qubit Observables

As a starting point, we investigate the potential difference in suitability of 1-qubit and 2-qubit correlation observables. In the first case, the feature vector is composed of all three Pauli matrices measurements on all 10 qubits. This results in a feature vector of size 30. To reach a similar dimension for a feature vector built from 2-qubit observables, we need 30 different correlation measurements. In this case, we choose 30 random correlations of qubits spanning the length of the entire chain with randomly assigned correlations to measure.

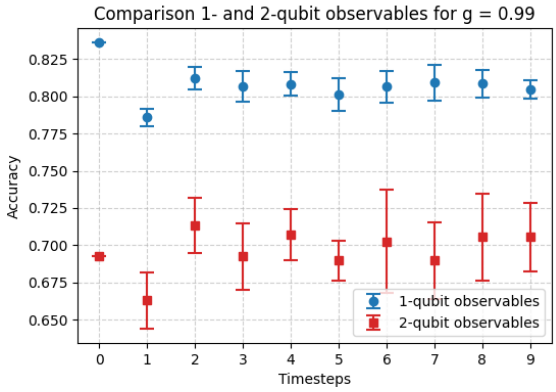


Figure 5.5: Comparing results of 1-qubit and 2-qubit observables in the thermal phase, both feature vectors with 30 dimensions.

The results for this comparison over the process of the entire simulation can be found in Figure 5.5. Let us first have a look at the results for the three Pauli matrices. In 5.4 the system with 10 qubits and Pauli-X and Pauli-Z achieved an accuracy of approximately 75%. Consequently, adding also Pauli-Y yielded a satisfying increase in accuracy of another 5% on average. We also observe again that the results after the time evolution are worse than right after the encoding.
However, this does not hold for the 2-qubit observables. Although the overall results are clearly worse than the ones achieved with the three Pauli matrices, they are on average better than the ones measured from the encoding. Sadly, both configurations experience a serious drop in accuracy when measuring after the first timestep. While the 2-qubit configuration appears to overcome this drop by returning to accuracies above the starting point, the Pauli matrices configuration performs consistently worse than the starting point. However, due to the decreased accuracy, we can conclude that for equivalent feature vector sizes using only 2-qubit correlations in the training process is not sufficient. Consequently, the next logical step is to combine both 1-qubit and 2-qubit observables in an analysis to investigate their potential.

5.4.3 Combination of 1-Qubit and 2-Qubit Observables

The first question to arise is which of the Pauli matrices to use in such a combination, given that they yield such varying accuracies when analyzed over different system sizes 5.4. To shed light on this question, we conduct an analysis whose results can be seen in figure 5.6. The approach is to use all possible choices of 2 Pauli matrices as starting points and then add the measurements of 2-qubit observables step by step. We apply this process iteratively until the feature vector reaches a dimension of 50, which logically happens after adding the 30 2-qubit correlations whose single results are displayed in 5.5. All measurements are taken after 10 time steps.

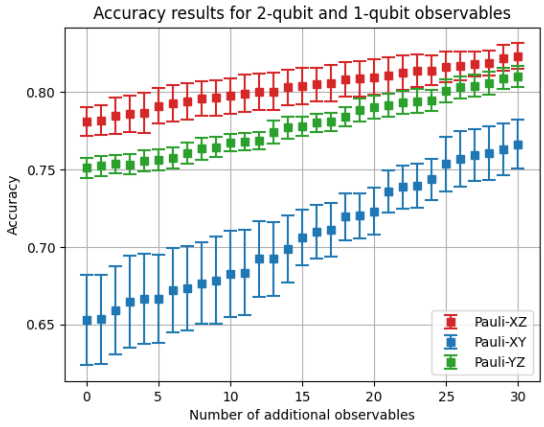


Figure 5.6: Comparison of classification accuracy for different 1-qubit starting observables in the MBL-DTC phase.

Given that we are working with a system of size 10, we can immediately compare the first data point to the results for Pauli-X and Pauli-Z in Figure 5.4 and observe that they match. Secondly, we notice that this is also the consistently best starting choice. All three configurations improve their accuracy when adding more 2-qubit observables, but the Pauli-X and Pauli-Z performs best in each step. Given the encoding it is unsurprising that the two configurations containing Pauli-Z perform better than Pauli-X and Pauli-Y. However, this 'bad' encoding also relatively benefits the most from adding more observables. This makes sense given that the unseen information entering this training process has a stronger effect as it contains more Pauli-Z parts. Overall, the trends observed suggest that a converging process is taking place that will result in all configurations achieving similar results for the same number of observables. Although this suggests that the choice of observables to measure is arbitrary for our results, in our cases this does not hold. We never measure sufficiently many or even all observables to approximate this equality. Consequently, for choosing lower numbers of observables as we are doing in this work, it makes sense to work with Pauli-X and Pauli-Z as a basis to achieve the best results with a fixed number of observables.

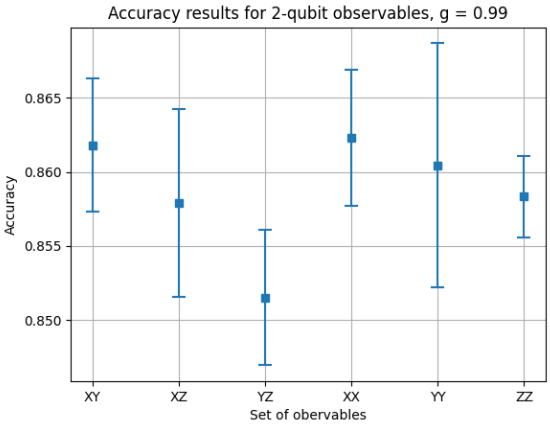


Figure 5.7: Classification accuracy results for different categories of 2-qubit observables in the MBL-DTC phase.

5.4.4 Investigating the best 2-Qubit Observables

Due to the limits imposed on us by the simulation software, we can only work with a small set of measurements of 2-qubit observables. So far, we used 30 different observables in our analysis. Now, we expand this to the currently maximum possible number of 60 observables. These observables fall into 6 different categories, depending on the Pauli correlations between different qubits: XY, XZ, YZ, XX, YY and ZZ. Given that we have access to 60 observables in total, each of these categories contains six observables. As a feature vector with only six entries is too small for efficient training, we also apply temporal multiplexing for the first ten steps. The results per observable category can be found in Figure 5.7.

The results displayed are difficult to interpret. Given the small size of the feature vectors used in the training process, the large error bars are not surprising. However, considering the relatively large range covered by the y-axis we can conclude that it appears to make a difference, which observables are measured and used in training. While it is easy to identify that 'YZ' seems to be a bad candidate, all other categories yield similar results. Consequently, while we suspect that some correlation is present, a more thorough investigation would be necessary to identify clear and statistically significant behaviors. Due to time constraints, this is left for further research.

5.4.5 Results for $g = 0.70$

Lastly, we want to shortly compare the results presented in this section for the thermal phase. We chose to work with the MBL-DTC phase as default. In Figure 5.8 one can immediately see why: The classification accuracy drastically reduces with the first few time steps for the 1-qubit observables when using the thermal phase. Furthermore, we will see that the results are less interesting, since they show a lower level of variety.

The results for the three Pauli matrices exhibit a clear negative trend that reaches a plateau

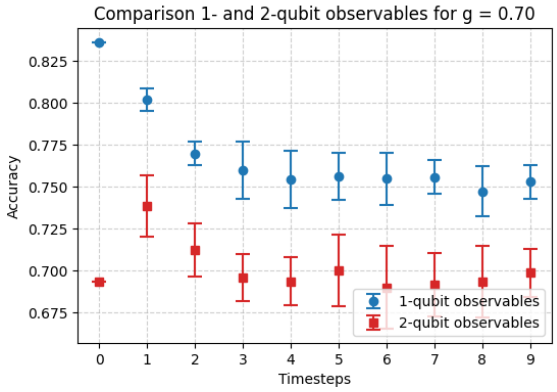


Figure 5.8: Comparing results of 1-qubit and 2-qubit observables in MBL-DTC phase, both feature vectors with 30 dimensions

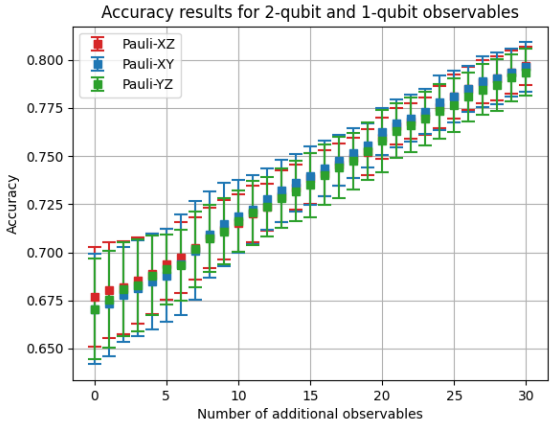


Figure 5.9: Comparison of classification accuracy for different 1-qubit starting observables in the thermal phase.

at about 75%. Consequently, even though we use more observables than we did in 5.4, the accuracy is the same. This already justified our choice to focus on the configuration in the thermal phase.

In contrast, the 2-qubit results are more interesting. In terms of thermal evaluation, the classification accuracy is consistently lower. However, the results are not stable or decreasing. Although the accuracy reaches a plateau after three time steps on the level of the encoding accuracy, after one and two time steps the results are clearly better. Apparently, for a short time the evolution of the quantum system was able to increase the results for the 2-qubit observables.

When adding 2-qubit observables in the thermal phase, the results differ a lot from those obtained in the MBL-DTC phase in Figre 5.6. In Figure 5.9 we can immediately see that the starting observables do not have a statistically valid effect on the overall results. The

addition of 2-qubit observables has the same effect on the results as it did for the Pauli-X and Pauli-Y combination in the MBL-DTC phase. The results increase steadily by about 12% in total. Given that the system exhibits a much stronger interaction between the different qubits compared to the MBL-DTC phase, these results were expected. As we measured after 10 time steps the system had enough time to evolve and the information is well spread across observables and sites. Unfortunately, a relevant part of the information appears to get lost from this thermalization as we start at an accuracy of about 67.5% for no 2-qubit observables compared to the 77.5% for Pauli-X and Pauli-Z in the MBL-DTC phase.

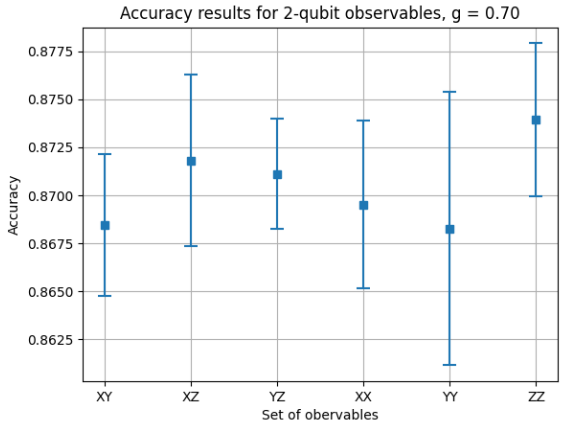


Figure 5.10: Classification accuracy results for different categories of 2-qubit observables in the thermal phase.

The previous results suggested that for the thermal phase it does not make a difference which observables to measure. We find this to be confirmed by taking a look at the performance of the different categories of 2-qubit observables. From the interval covered by the y-axis we find that the overall spread of all data points is lower than the spread for the MBL-DTC phase. Furthermore, the data points are closer together and, except for one outlier, all data error bars cover the means of all other data points. While we could not draw any reliable conclusions for the 2-qubit observables in the MBL-DTC phase, it appears that for the thermal phase, the performance is mostly insensitive to the choice of observables for the training process.

Chapter 6

Investigating Post-Processing Effects

6.1 Comparing Different Training Methods

In the spirit of Reservoir Computing, Quantum Reservoir Computing and the success Ridge has delivered in these fields when predicting time series in complex systems [14], we originally chose Ridge regression as our default training algorithm. In Figure 4.4 we investigated Ridge's sensitivity to varying default parameters. This analysis already suggested that Ridge might not be the best approach when training our QELM. Consequently, we decided to conduct a short study to compare Ridge's performance with other training algorithms that are also well established in either QELMs (ONN) or Machine Learning (SVC). In subsection 6.1.1 we present two competing approaches and how we applied them. These approaches are a One-Layer Neural Network that has also been used by Sakurai et al. [15] and a standard support vector classifier. In subsection 6.1.2 we present some results for these training methods and Ridge regression on four different sample configurations that we drew from the simulations conducted throughout this work.

6.1.1 Considered Training Methods

Before we move on to evaluate the performance of varying training algorithms on sample configurations, it is necessary to present the training algorithms we will consider in more detail. So far, we worked with Ridge regression and presented its mechanism in section 2.5. Given that there are a plethora of other algorithms to leverage for classification problems, we would like to take a moment to motivate our choices here.
In view of the close relationship between our work and the research performed by Sakurai et al. [15], we decided to implement their training algorithm and investigate its performance in our configuration. As it is not our goal to achieve an improvement in the classical training layer, we do not expand the implementation to a discussion of overfitting and dropout. As this algorithm is a tuning of a one layer neural network, we also consider the results that are achieved for a simpler implementation that does not require a vast exploration of the hyperparameter space.
Lastly, we conduct the training with a very simple implementation of a Linear Support

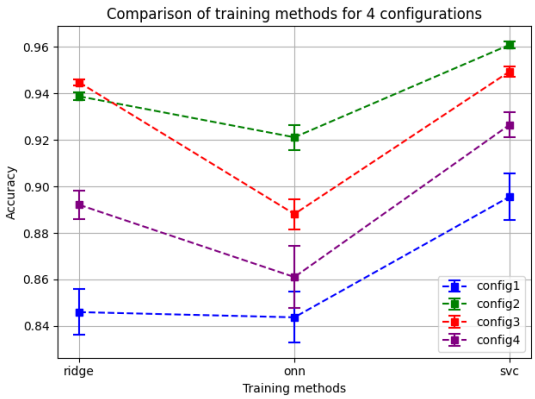
Vector Classifier. During our research, we found that this algorithm already performs well without any tuning while not taking too long to train. Consequently, we decided to shortly present it and evaluate its performance on several configurations.

The simplest implementation we chose is the one layer neural network without any tuning. This is equivalent to a matrix mapping the feature vector linearly on the classes to distinguish for. As we do no apply any activation function, no further non-linearity is introduced after the quantum system. Given the simplicity of this approach, we only trained the neural network for 10 epochs. After these epochs we choose the one hot encoded class with the maximum value in the predicted dimension.
The implementation chosen by Sakurai et al. [15] follows a more sophisticated path. Apart from training the weights for 200 epochs, an activation function is applied to calculate the probabilities of each class. With both algorithms consisting of the same number of nodes, the main difference to the previous method is the number of training epochs. This should enable the algorithm to learn more difficult patterns and achieve a higher classification accuracy.
Lastly, we use a linear support vector classifier. Instead of tuning the hyperparameters, we choose to use the default configuration set by sklearn's implementation. While the neural network is optimized to approximate the respective class probabilities, the support vector classifier is trained to find the optimal hyperplane separating all classes.

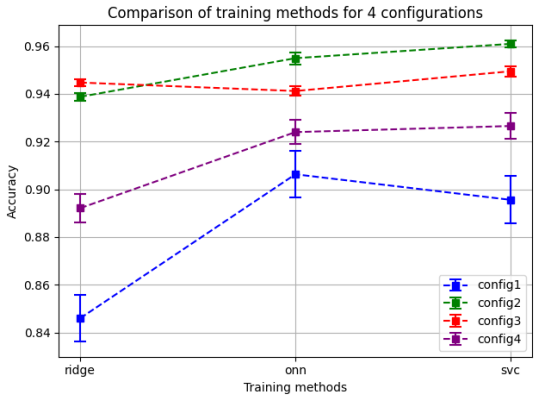
6.1.2 Performance on Sample Configurations

As it is not our plan to establish a different training algorithm for this work we did not apply it to every analysis performed so far. This would have only increased the number of dimensions open for exploration and added further complexity to the analysis. To avoid this, we have chosen four sample configurations that have also been used in the same or a similar fashion at some point in this work. Their technical configurations can be found in the appendix B.1. The results for training on these four configurations can be found in Figure 6.1a.

Considering the varying parameters assigned to each configuration it is not reasonable to compare the performances across different configurations. However, we can draw some conclusions as the comparison within the configurations exhibit clear trends. It should be kept in mind that during this training process no activation function is applied to the feature vector while training the ONN. Furthermore, it is only trained for 10 epochs. Without an activation function, the ONN is equivalent to a multiple linear regression. In that case, the only relevant difference to Ridge regression with a very small penalty is the number of epochs that the ONN had the time to find the best possible mapping. Considering these settings and their results, it appears that the ONN does not have enough training time as the Ridge model performs better for 3 out of 4 configurations and equally well for the last one. Because of these results, we conduct another analysis in Figure 6.1b with the same ONN as used in Sakurai et al. [15]. Returning to our current comparison, we



(a) Comparison of training methods for 4 representative QELM configurations.



(b) Comparison of training methods for 4 representative QELM configurations. ONN equivalent to Sakurai et al. [15].

Figure 6.1: Performance of different training methods on 4 representative configurations.

can also conclude that the Support Vector Classifier would have probably been the better choice for all trainings throughout this work. It manages to consistently achieve higher classification accuracies than both the Ridge model and the ONN. While the training process for the SVC takes more time than the training process for Ridge, it might be worth the additional time given the clear improvement of classification accuracy.

6.2 Effect of Temporal Multiplexing

So far we mostly worked with measurement results obtained after a fixed number of Floquet cycles. To improve our results, we would like to increase the size of our feature vector. We have already done this by adding further observables measured at the same time or by adding the best performing measurements from other time steps. Now, we would like to perform a deeper study of the effect of temporal multiplexing. The goal is to find an optimal number of steps to add to the training process and to figure out whether it makes a difference which time steps we choose to combine. Furthermore, we would like to understand whether it is better to combine data originating from the MBL-DTC phase or from the thermal phase of the physical system.

6.2.1 Temporal Multiplexing Depending on Physical System

As a first step, we want to compare the training with temporal multiplexing for different physical phases. To develop an intuition for the necessary range of temporal multiplexing, we start with an analysis of the first 100 Floquet cycles. In chapter 5 we already discovered that the phases have different effects on the results depending on whether we

use temporal multiplexing or not. Here, we find this result confirmed. The discrete time crystal performed consistently better than the thermal phase in the case of training on observables originating from only one time step. However, when combining measurements from several time steps, the stronger mixing of information in the thermal phase resulted in an advantage for the training process. This also holds for long evolution times as depicted in 6.2.

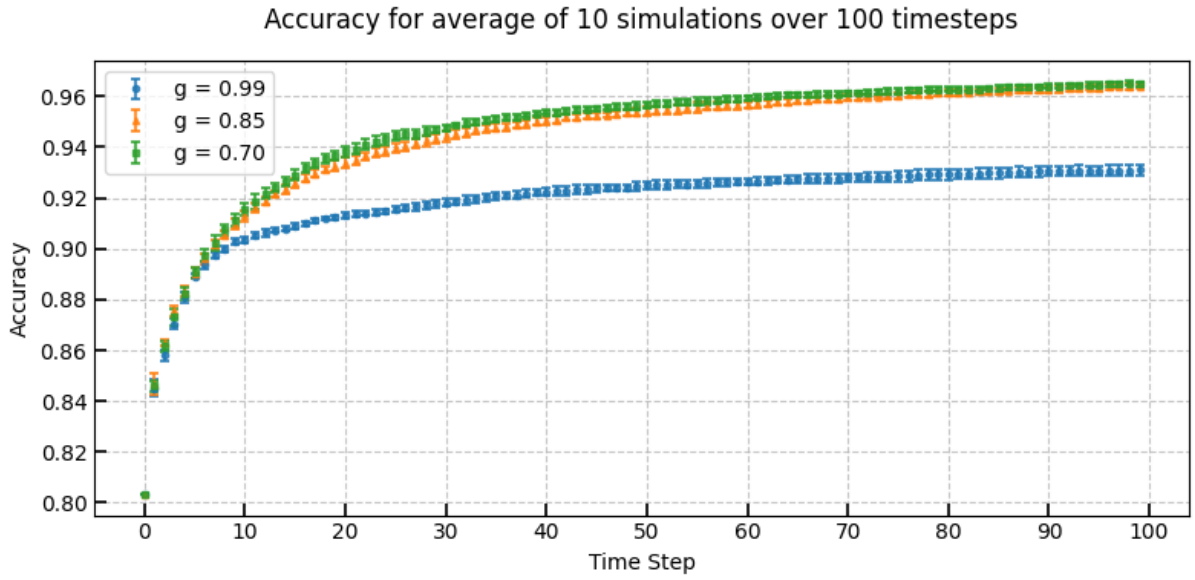


Figure 6.2: Temporal Multiplexing results for 100 steps and different phases

It should be taken into account that the growth of the feature vector, while linear, results in a really large vector after 100 time steps. We are operating on a system with 10 qubits and measure two observables on every qubit and for each time step. This results in an addition of a 20-dimensional vector to the feature vector per time step. Consequently, in the end we train on a vector with 2020 entries, while at the starting point of the time evolution we trained on a vector with only 20 dimensions.
Overall, all data points exhibit small standard deviations around their mean. This is a reassuring result that the 10 random draws per simulation are sufficient to account for the randomness in the quantum reservoir. Throughout this work we repeatedly find that the application of temporal multiplexing, in addition to increasing the accuracy, is in general successful in lowering the deviations of the predictions.
We can also see that until eight time steps the training success is relatively similar for all phases. After this, the performance of the discrete time crystal phase deviates as the thermal and transition phase results grow at higher rates. However, the classification accuracy still increased from approximately 90% to 93% in the remaining 90 time steps. The results of the thermal and the transition regime are more impressive. The logarithmic

growth observed for both results in more than 96% accuracy after 100 time steps. There is no sign that the accuracy might worsen for more time steps, but it also does not seem likely that the system manages to correctly classify all images after any number of time steps.
Furthermore, the overall small difference between the thermal and the transition regime is notable. While the transition regime seems to underperform the thermal phase slightly for step numbers lower than 80, the two curves converge in the end. One can conclude that the transition regime takes more time to achieve the same classification accuracy but that it managed to catch up after some time. Concluding, one gains the impression that as long as we are not deep in the discrete time crystal phase, the system is performing really well when using temporal multiplexing.
However, as the further gain in accuracy is relatively small, it does not seem appropriate to create incredibly large feature vectors from temporal multiplexing. The method is invaluable to increase the classification results but it should not be stretched too far. Combining a lower level of temporal multiplexing with other methods appears to be the more promising approach. Fortunately, we find our original choice of using 20 time steps as the default simulation length confirmed. After 20 time steps all three phases have already reached around 80% of the entire accuracy gain achieved after all 100 time steps. Therefore, to us this seems like an appropriate choice for temporal multiplexing. Of course, this choice must be reevaluated once actual physical quantum hardware is available to us and we do not have to rely on classical digital simulations anymore.

6.2.2 Partial Studies

After having had a look at the entire time evolution we now move on to discuss smaller intervals of this evolution. In chapter 5 we have already discussed that one of the best performing steps of the entire measurement interval is the measurement before any evolution. As now we would like to compare different time intervals of the evolution, we exclude the 0th time step.
As the thermal phase achieved the best results while also exhibiting a smooth logarithmic growth of classification accuracy, we focus on this simulation in our discussion. To get an impression from different perspectives, we investigate the performances for time intervals of length 10, 20, and 50. Therefore, in the first case we compare 10 intervals with 10 time steps that add up to a feature vector of size 200. In the second case, we compare 5 intervals with twice this size. When comparing for 50 time steps, we only have 2 intervals. Their feature vectors' have 1000 dimensions each.

Before delving into the comparison of the results, we notice that for all 3 splits of the evolution process the data points exhibit a remarkably low spread. While for 10 time steps all accuracies scatter within a 1% interval, the accuracies for 20 time steps vary even less. The mean accuracies as well as their error bars are all covered by the interval $[93.5\%, 93.9\%]$. Although we only have two data points, the trend of decreasing dispersion continues as the error bars are within $[95.50\%, 95.75\%]$. This makes sense given the increasing sizes of the

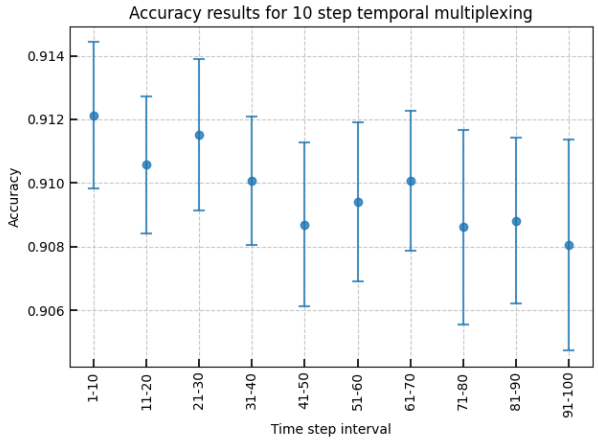
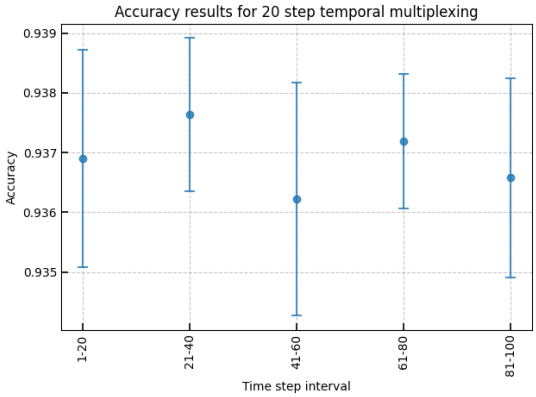


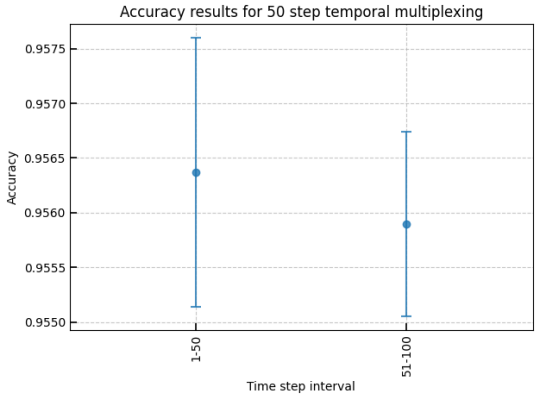
Figure 6.3: Accuracy results for training on different slices of 10 steps with temporal multiplexing

feature vector. However, it puts our other comments on potential trends in the analysis in a different perspective. Although some trends can be observed, we conclude that the overall impact of choosing different intervals is low.

That being said, we notice that the analysis of the 10 time steps exhibits a negative trend



(a) Accuracies for 20 time steps of temporal multiplexing.



(b) Accuracies for 50 time steps of temporal multiplexing.

Figure 6.4: Accuracies for varying numbers of time step intervals

over the entire interval considered. Unlike for QRC, when using a QELM the data is read into the system only once. Therefore, it is expected that some information present in the beginning of the time evolution gets lost. As this might be countered by the opposite effect of beneficial mixing of information, it is difficult to assess how much this is the case. Overall the size of the error bars should be kept in mind. While the negative trend is clearly there, some measurements in the last bracket of 91-100 time steps still achieve comparable

results to some bad performing measurements from 1-10 time steps.
When turning to the analysis of size 20 intervals we immediately notice a different picture. Although we have fewer data points, we cannot confirm the negative trend observed for size 10 intervals. The most surprising observation is that the first 20 time steps are not the best performing time steps interval. The result based on time steps 21-40 does not only achieve a higher mean but also smaller error bars. This is even more stunning when considering the size 10 intervals. Here, the average result for time steps 1-20 is higher than the one for time steps 21-40. Apparently, we cannot draw immediate conclusions for larger intervals when training on smaller time intervals. Given this non-trivial relationship between different time interval sizes, it might be worthwhile to combine other intervals with one another. However, at the current state this goes beyond the scope of this work and is left for further research. The other results of size 20 align more or less with those for size 10 as they show similar fluctuations.
The results for 50 time steps do not keep any reliable further insights. The accuracy increases again to more than 95%. Despite the large error bars we observe a small tendency for the later time steps to be worse. This aligns with the results for the size 10 intervals.

In conclusion, we can state that the effect of choosing different intervals for the training does not have a strong impact on the result. If it has any reliable impact, it will make sense to stick to the first 50 time steps or a subset of these. Investigating less trivial time step configurations such as size 10 intervals from different parts of the evolution remains to be explored. The lack of strong dependency on the time of measurement is promising for future research. These results support the conservation of the initial data in the physical system also for up to 100 time steps.

6.3 Effect of Readout

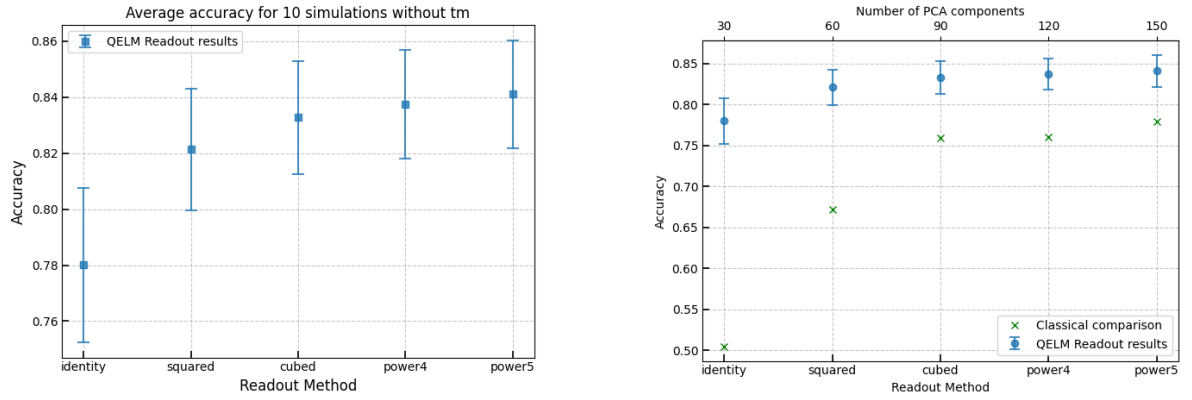
A well established method to improve results in the field of QRC and RC is to apply a function to the already existing feature vector [38]. The most common approach is to apply the Schur multiplication to the feature vector by multiplying the feature vector element-wise with itself:

$$\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} \circ \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} x_0 x_0 \\ x_1 x_1 \\ \vdots \\ x_{n-1} x_{n-1} \\ x_n x_n \end{pmatrix} \xrightarrow{\text{append to existing vector}} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \\ x_n \\ x_0 x_0 \\ x_1 x_1 \\ \vdots \\ x_{n-1} x_{n-1} \\ x_n x_n \end{pmatrix} \tag{6.1}$$

In equation 6.1 we can see an example for squared readout as the feature vector is multiplied once with itself, creating a vector with squared entries of the original entries. In the same fashion, we could create cubed or quadrupled feature vectors. Of course, this could be continued to infinitely large exponents. However, as the feature vector entries are averaged measurements in the interval of -1 and 1, the reasonable application of this method requires to use low integer exponents. Multiplying the vector with itself too often would create input data for our training that is too close to 0. This could result in the training being more difficult and unstable due to the increased orders of magnitude that our data would cover. For example, when creating a quadrupled feature vector, on one hand data points with an original value of $x_j = -0.9$ would result in $O(10^{-1})$. On the other hand, data points with an original absolute value of less than 0.1 would result in a smaller value than $O(10^{-4})$. When trying to map both these points to a label with one regression, the weights must span a wide range to map properly to the labels. Weights spanning a larger order of magnitude would result in a less stable process. Therefore, we will only consider multiplications of up to power 5.
Before starting with the analysis of different readout methods, we need to fix the simulation parameters to use for this part of the work. As we are not yet applying any other improving methods, it is not necessary to choose the best possible performance. We stick to a system with $L = 15$ qubits and a phase parameter $g = 0.95$. As observables, we use Pauli-X and Pauli-Z. We measure after 20 time steps and temporal multiplexing is not applied as the readout methods themselves already improve the size of the feature vector sufficiently.
To evaluate our results we compare them to a classical training based on a feature vector with the same dimension. To achieve this, we randomly choose n variables from the original mnist-digit data and train on this data.

6.3.1 Standard Readout-Methods

As standard and well established readout methods we consider all multiplications up to the power of 5. Their results are displayed in Figure 6.5. As a comparison, we added the results for no further function applied to the feature vector. This is labeled as 'identity'. We will stick to this descriptive notation for the entire chapter.
We can immediately observe an improvement of the classification accuracy for the applied readouts. However, the positive effect seems to exhibit logarithmic growth. If one follows the diminishing returns of the readout methods and takes our intuition about increased orders of magnitude for the feature vector into account, we expect that the performance starts worsening quickly for larger readout methods. Another advantageous effect is that the application of the readout methods lowers the standard deviation of the results as can be observed by the shortened error bars. Given that this effect can also be observed when adding more observables to the feature vector, we suspect that this decreased variety is generally caused by increasing the feature vector size.
As a further comparison we added the classification accuracy for a classical training algorithm with a feature vector of the same size in the right image. The classical training is performed by using the dimension of the feature vector of the classical training as the



(a) Readout results without a classical comparison.

(b) Readout results with a classical comparison.

Figure 6.5: Main caption for both figures.

number of random variables that were to be drawn from the original mnist black and white image. Obviously, the results should be interpreted with caution, as of course much more information was fed into the QELM than is contained in the classical feature vectors. However, it is still nice to see that a feature vector with less than 100 dimensions can already achieve accuracies above 80%. Also promising is the fact that the results for 'power5' are still outperforming the classical results. As the classical result is trained on 20% of the available variables, an identification of the digit is possible.
Before advancing to our suggestions for further readout methods, we have a look at the results for temporal multiplexing here. The results for the same configuration as in Figure 6.5 but with all 20 times temps are presented in Figure 6.6. For temporal multiplexing we already start with a feature vector of dimension 630 that increases up to 3150. Consequently, we expect significantly better results than we had for the evaluation without temporal multiplexing. Given these large dimensions, it is not purposeful to compare the results to any trainings based on classical mnist-digit data.

While the overall performance of all readout methods improves drastically, the difference between the individual readouts decreases. Without temporal multiplexing, applying squared readout already increased the results by 4%. Now, the increase is only about 1.5%. However, when considering that the identity readout already yields great results, the lower increase is expectable. There is a remarkable similarity in the increase of classification accuracy, as the application of 'squared' leads to a correct classification of about $\frac{1}{6}^{th}$ of the previously wrongly classified data points. While the temporal multiplexing shifts the overall performance significantly to more than 90%, one can argue that the improvement achieved by the readout method seems to remain constant based on the remaining shares of wrongly classified data.
Concluding our analysis of the already in use standard readout methods, we note that while

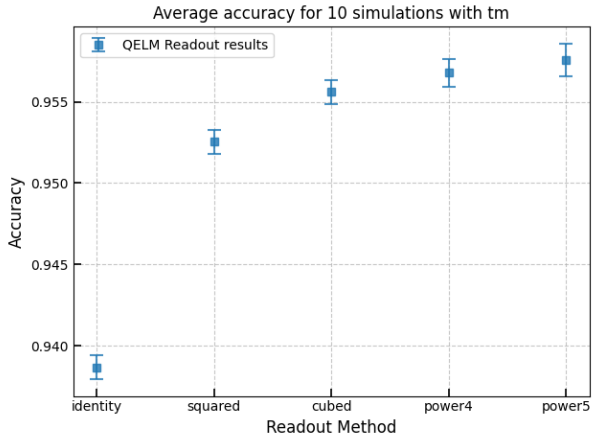


Figure 6.6: Readout results for temporal multiplexing with 20 steps

the application of readout methods of increasing degree improves the results significantly, the marginal gain in accuracy decreases. These effects are observed independently of the application of temporal multiplexing. Now that we developed an understanding of the readout methods' potentials and limitations, we turn to some newly developed strategies for applying readout functions.

6.3.2 Newly Proposed Readout-Methods

When discussing candidate functions beyond the previously discussed as readout methods, we are in the fortunate yet challenging situation of choosing appropriate but simple functions that could yield an improvement. Firstly, we consider shuffling different parts of the readout vector. Secondly, in the same spirit but less dependent on randomness, we multiply each component of the feature vector with a later component of the feature vector, depending on a pre-defined shift parameter. Lastly, we introduce a few common functions in machine learning and investigate their performance. While there is an infinite number of functions to choose from, we wanted to keep the complexity sufficiently low not to create a black box whose behavior and potential success are too challenging to explain.

Shuffled Readout

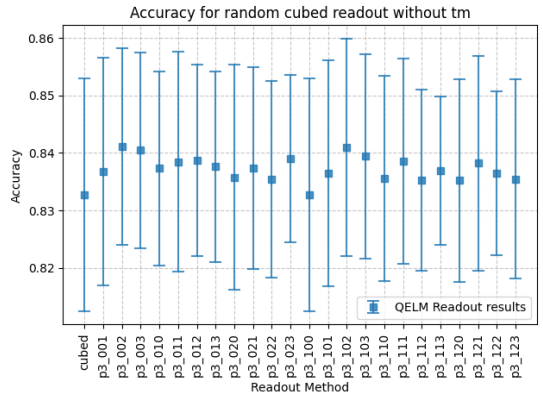
The most prominent part in our discussion is straightforward: we introduce randomness into the functions discussed in 6.3.1. As an example, we work with readout vectors of third power. This is a compromise between complexity and computational efficiency. Within this third power vector, we introduce randomness at every possible moment. The original structure of the cubed readout vector can be found in equation 6.2.
The system represented is the same we used in our training process. While the quantum system is made of a chain containing 15 qubits, we use Pauli-X and Pauli-Z in training. Thus, we create an original feature vector of dimension 30. The measurements are con-

ducted after 20 time steps and we do not use temporal multiplexing such that the actual size of the identity feature vector was 30. However, due to the 'cubed' readout and its shuffled variations the size of the feature vector to train on grew to 90 dimensions.

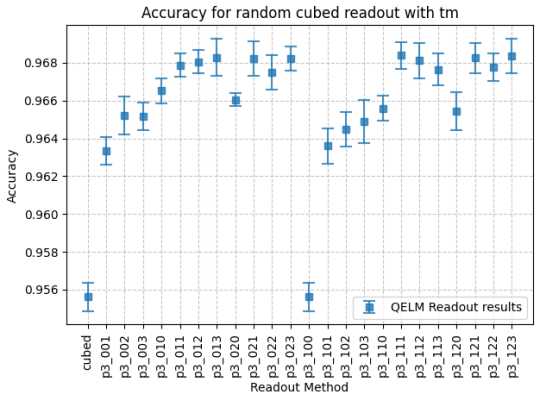
$$\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{28} \\ x_{29} \\ x_0x_0 \\ x_1x_1 \\ \vdots \\ x_{28}x_{28} \\ x_{29}x_{29} \\ x_0x_0x_0 \\ x_1x_1x_1 \\ \vdots \\ x_{28}x_{28}x_{28} \\ x_{29}x_{29}x_{29} \end{pmatrix} \xrightarrow{p3\_102} \begin{pmatrix} x_{17} \\ x_4 \\ \vdots \\ x_{25} \\ x_{11} \\ x_0x_0 \\ x_1x_1 \\ \vdots \\ x_{28}x_{28} \\ x_{29}x_{29} \\ x_0x_{14}x_8 \\ x_1x_{27}x_{13} \\ \vdots \\ x_{28}x_6x_{25} \\ x_{29}x_{21}x_3 \end{pmatrix} \tag{6.2}$$

To visualize the meaning of the x-axis ticks in Figure 6.7 we display an example of the shuffling on the right side of equation 6.2. The example 'p3_102' represents a cubed vector containing one shuffling for the 'identity' part of the readout vector, no shuffling for the 'squared' part, and two shufflings for the 'cubed' part. In total, when counting the original cubed version, we calculate the results for 24 different configurations. As in this case we shuffle all possible components, the data point for 'p3_123' exhibits the highest level of randomness. As the results without temporal multiplexing do not show any trend or clear dependency on the shuffling, we also perform the analysis for temporal multiplexing on all 20 steps in the right-hand side of Figure 6.7. Without temporal multiplexing, the error bars span almost the entire range of the results. Although the first result for the method 'cubed' is on average lower than many other results, each data point's error bars cover all other data points. Consequently, it is not yet possible to draw any conclusions about the success of our newly suggested method. However, we note that, since all mean results scatter between 83-84% compared to approximately 83% without shuffling, our method does not appear to worsen the prediction quality.

The next logical step is to follow the path already outlined in the previous paragraph. An increase in the feature vector's size might yield a clearer look and will most probably improve the overall results. When considering the classical readout methods, we observed a quantitatively similar result for temporal multiplexing whose predictions were shifted. Here, we also manage to discover a stronger distinction between the tested methods. All readout methods in combination with temporal multiplexing succeed in improving the results to around 96%. At the same time, the standard deviations decrease vastly to less than 0.1% for all data points. When taking a closer look at the performances, two data



(a) Readout results without a classical comparison.



(b) Readout results with a classical comparison.

Figure 6.7: Readout effects for random readout 'cubed' with and without temporal multiplexing.

points immediately stand out. They perform significantly worse than all other tested configurations. These two are the only data points that do not include any shuffling in the appended 'squared' and 'cubed' parts of the feature vector. We included the regular 'cubed' configuration as a benchmark for all other configurations and the 'p3_100' in the spirit of completeness as well as a sanity check. For shuffling the 'identity' part of the feature vector we do not expect any performance change compared to 'cubed'. This expectation is confirmed by the analysis.
Of course, of superior interest is which configurations reliably perform better than the remaining ones. Our results exhibit a remarkably consistent behavior. All configurations that are shuffled at least once in the 'squared' and 'cubed' part of the feature vector belong to the best performing group. This group scatters around 96.8%. Performance in the interval $[96.4\%, 96.6\%]$ is achieved in configurations that are not shuffled in the 'squared' part, 'cubed' part, or in both of them. Another clear result that aligns with the previously mentioned bad performance of 'p3_100' is that shuffling the 'identity' part of any well performing configuration does not alter the result beyond statistical fluctuations. Drawing a final conclusion about the overall best performing configuration is not possible. Shuffling all three constituents of the 'cubed' tends to yield very strong results. But there are also other configurations like 'p3_111' or 'p3_021' or 'p3_121' that achieve comparable performance.

Although the improvement of the results is roughly 1% for most configurations, this is still a remarkable result given that we already started at a classification accuracy of approximately 95.6%. Naturally the question arises why these improvements occur. Unlike the case of temporal multiplexing or the comparison of already established readout methods, we cannot find the reason in an improved size of the feature vector. Given that we

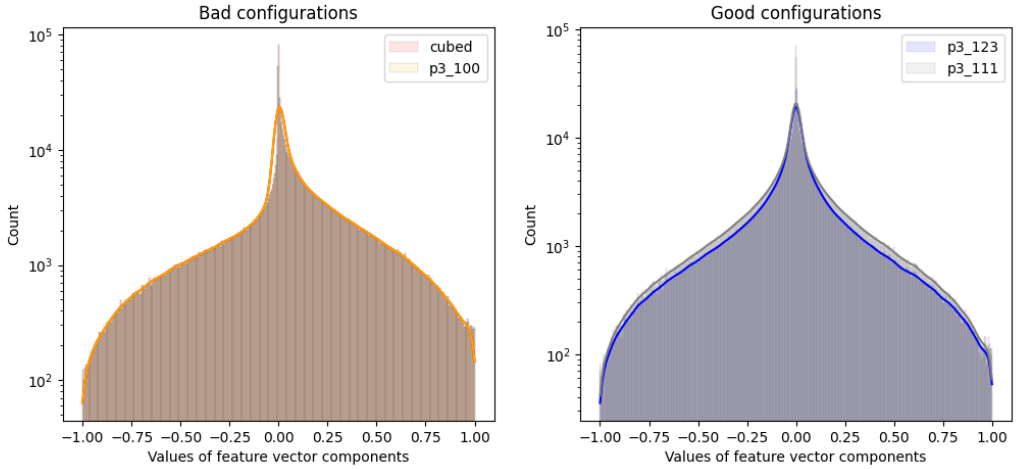


Figure 6.8: Log-distribution of feature vector components for best and worst performing shuffling configurations.

only shuffle parts of the 'cubed' feature vector, they all have the same dimension as the 'cubed' benchmark.

When observing these results our first intuition is that the shuffling solves the earlier discussed problems of some feature values approaching 0 too quickly. However, we do not find this intuition confirmed when investigating the distribution of the values in the feature vectors. In Figure 6.8 we display the first 10^6 components of the feature vectors yielding the accuracy results of the given labels in Figure 6.7. In the left part of the Figure we see the histograms for bad performing configurations, while the right part displays two samples from the best performing group. Apparently, the increased shuffling leads to a more symmetric distribution of our feature vector components around the origin. For Ridge regression to perform well on a given data set, a Gaussian distribution of the features is advantageous. Consequently, given the different distributions of our configurations' feature vector values, it is logical that one group of configurations outperforms the other group.

In the following subsection we investigate whether the improvements achieved from shuffling the feature vector can also be achieved from a simpler mathematical operation of just shifting the part of the feature vector that each component is multiplied with by a fixed number. Afterwards, we have a look at common activation functions from Machine Learning to investigate whether these are able to improve the results. Given that a more symmetrical distribution of feature vector components around the origin significantly improved the results, it is worth investigating if other symmetrical mappings are able to achieve similar results.

Squared Shifted Readout

The next investigation we conduct follows a similar approach as the shuffling discussed previously. Instead of shuffling, we now multiply each singular component with one of the following components. In the interest of developing a reliable understanding, we investigate shifts of several ranges. To keep matters simpler, we only conduct this analysis for the squared readout instead of the cubed readout used previously. A visual representation of shifting by one step can be found in equation 6.3.

$$\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{28} \\ x_{29} \\ x_0x_0 \\ x_1x_1 \\ \vdots \\ x_{28}x_{28} \\ x_{29}x_{29} \end{pmatrix} \xrightarrow{shift} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{28} \\ x_{29} \\ x_0x_1 \\ x_1x_2 \\ \vdots \\ x_{28}x_{29} \\ x_{29}x_0 \end{pmatrix} \tag{6.3}$$

It is worth keeping in mind that $x_1...x_{15}$ are measured values of Pauli-X while $x_{15}...x_{29}$ are the measurements of Pauli-Z. This visualization is valid for a feature vector without temporal multiplexing. In the case of temporal multiplexing, all time steps for Pauli-X are appended before the results for Pauli-Z are appended. This is important when interpreting the results displayed in 6.9. The readout methods on the x-axis are squared readout shifted by the respective integer.
As a comparison we added the performance for the 'squared' and the 'squared_shuffled' readout methods.
Although the first impression of the results might be that they fluctuate without any pattern between two main levels, this assumption does not stand up to closer scrutiny. Inspecting the data points that result in an accuracy similar to the accuracy without shuffling, we realize that all shifts are exact multiples of 15. Given that our chain size is 15, in these cases each original feature vector component has been multiplied with its measurement from a later time step. In contrast, a lot of shifts deviating from the multiples of 15 are able to almost reproduce the accuracy achieved by random shuffling. However, none of them are able to achieve the same level of accuracy or even surpass it, making random shuffling still the top performer for squared readouts. Furthermore, shifts of only 1 or 2 appear to be too small to leverage the effect that the shuffling and larger shifts benefit from. The results are increasingly better than without shifting, but are the only ones clearly in between no shuffling or shifting and only shuffling. The last aspect to discuss is the apparent decrease of accuracy for shifting by 315. Given that our feature vectors were created by temporal multiplexing for a system with 20 steps and Pauli-X and Pauli-Z as observables, the feature vector has a total size of 630. Consequently, the multiplication

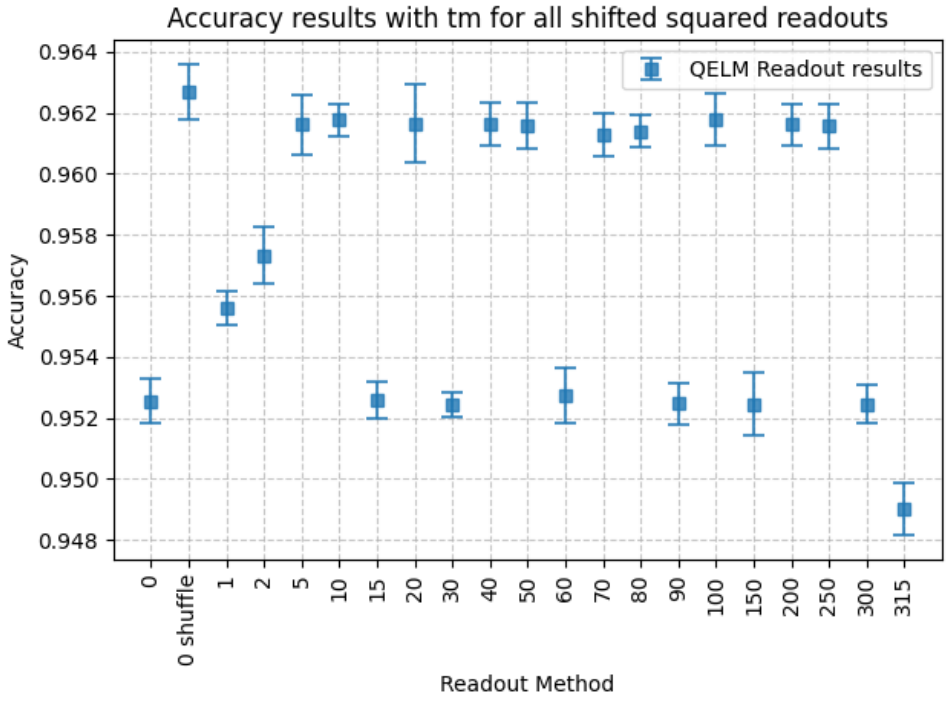


Figure 6.9: Accuracy results for different shifts of feature vector in 'squared' readout.

after shifting by 315 is a multiplication with the same qubit at the same time step but the measurement of Pauli-Z instead of Pauli-X. Apparently, this configuration is even worse than multiplying each component with itself. While all of these are interesting findings that lay a foundation for further research, the key takeaway remains that the shuffling's performance is unbeaten.

Simple Functions

Lastly, we consider simple functions from the area of Machine Learning. These are also used in neural networks to introduce non-linearity. Similarly as for the previous methods we apply the respective function on each component of the feature vector and append the results to the original vector. Plots of all used functions can be found in the appendix A.2.

- Leaky ReLu (Rectified Linear Unit): $max(0.1x, x)$ Originating from ReLU, all positive values are linearly mapped onto themselves. While in ReLu all negative values are mapped to 0, we do not want to lose all of this information. Consequently, we used the Leaky ReLu that multiplies negative values with 0.1.

- Extreme Sigmoid: $\frac{1}{1+e^{-7x}}$ Maps all values to the interval $[0, 1]$. While the normal sigmoid exhibits an almost linear behavior in the interval $[-1, 1]$, using e^{-7x} instead of e^{-x} yields a highly non-linear mapping.

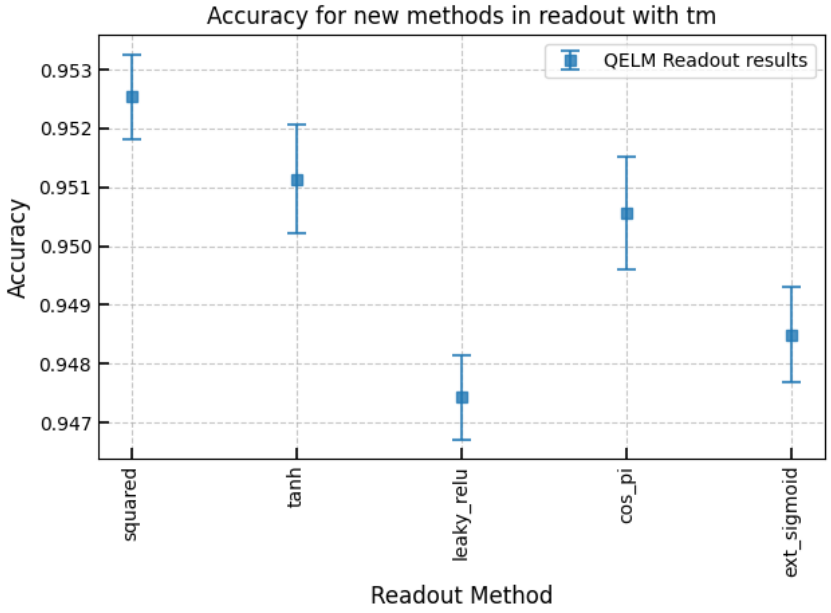


Figure 6.10: Accuracy results for suggested readout functions compared to standard squared readout.

- Tangens Hyperbolicus: $\frac{sinh(x)}{cosh(x)}$ Does not map values to entire range of $[-1, 1]$. Decreasing slope for x-values close to -1 or 1.
- Cosine: $\cos(\pi x)$ While not a standard activation function, we considered it due to the interesting aspect of values close to 0 and close -1 and 1 yielding similar results.

From our previous investigation we learned that without temporal multiplexing the effect of readout functions is negligible compared to the error bars of the statistic. Consequently, in this paragraph we only discuss the results for temporal multiplexing. The four functions are chosen based on their diverse behavior in the interval $[-1, 1]$. While the curves of $tanh(x)$ and $ext\_sigmoid$ mostly differ in their curvature, $LeakyReLu$ and $cos(\pi x)$ exhibit completely different behaviors in the given interval. Keeping these aspects in mind, the results displayed in Figure 6.10 are disappointing. Due to the large feature vectors the variety across readout methods is naturally small. As the variation between methods is still greater than the respective error bars it is large enough to draw conclusions. We can immediately see that all new suggestions worsen the results compared to the original 'squared' readout. The performances of $tanh(x)$ and $cos(\pi x)$ are only slightly worse than the 'squared' results. In contrast, the results obtained from $LeakyReLu$ and $ext\_sigmoid$ are significantly worse. Hence, a first result is that none of these functions are worth using for readout. They might still improve the results when using a feature vector with higher order non-linear combinations of the data. Due to time constraints this is left for further research.

Apart from this result, why do the *LeakyReLu* and *ext_sigmoid* yield such bad results compared to the other two functions? While $tanh(x)$ is almost a linear mapping of the data, $cos(\pi x)$ fluctuates an entire cosine oscillation around the origin. Both functions have in common that there is a high level of diversity in the mapping's output. In contrast, the bad performers *LeakyReLu* and *ext_sigmoid* map a large share of the values to very similar regions either close to 0 (*LeakyReLu*) or 0 and 1 (*ext_sigmoid*). Consequently, a high level of dispersion in the mapping's output appears to be advantageous for the classification accuracy. This result of course raises the question which other functions might yield better results than those presented here. An idea could be a cosine wave oscillating with higher frequency around the origin. While some share of the data would be mapped to the same values, they are originally not very close. Therefore, this mapping should not harm the overall results similarly and still hold interesting potential. This investigation is left for further research.

6.4 Performance after Combination of these Tools

6.4.1 Fixing Methods and Parameters

So far, the focus of this thesis has mostly been investigating the effect of several approaches to improve the results of a QELM by changing an isolated aspect of it. In this section we want to combine all of our findings to create a QELM with maximum classification accuracy. Before looking at the results of this training process, we have a short recap of the aspects we investigated and which conlusion they led to.
In section 5.3 we presented that the achievable results of course depend on the amount of information fed into the system. Keeping this in mind, we want to use a large system to encode as much variety of the input images as possible. However, as the PCA components are ordered by size we experience diminishing returns as the added variety per component decreases steadily. Given the cumulatively explained variance in figure 4.1, we settle for 20 PCA components.
Secondly, we learned in section 5.2 that the system's physical size strongly influences the classification. When we are not using temporal multiplexing, the discrete time crystal yields the best results. In the case of temporal multiplexing, the thermal phase managed to improve its results sufficiently to overtake the discrete time crystal phase. Within this work, we have usually compared both these phases. In the interest of consistency we apply the same approach here. However, as we are also applying temporal multiplexing we expect the thermal phase to achieve better results than the discrete time crystal phase.
Combining our results from Sections 5.3 and 5.4 we learned that the best performing observables are the three Pauli matrices. While the addition of 2-qubit observables led to an improvement in accuracy, the by far best results were achieved by using Pauli-X and Pauli-Z. As also the combination of Pauli-Y and Pauli-Z performed well, we choose to use all three matrices in our current case. For now, we leave the 2-qubit observables aside.
Regarding the training, we have to decide on the number of time steps to use, a readout method, a training method, and the hyperparameter(s) of the chosen training method.
In Figure 6.2 we discussed that the largest increase per additional time step is by far achieved within the first additional time steps. Consequently, we settle for a choice that we applied repeatedly throughout this work and include the first 20 time steps as well as the measurement before any time evolution.
Our extensive treatment of readout methods in Section 6.3 yielded clear results on the most appropriate choices for the readout method. Although the application of the four typical activation functions in Section 6.3.2 appeared to be a promising approach, the results were unable to live up to those expectations and clearly lacked classification accuracy compared to the application of random readouts. Building on these findings, we use the cubed readout that is shuffled once in every appended part. Considering the results in Figure 6.7 we could have also used other configurations, but the similarity of the results suggests that this choice does not have a significant impact. While we could have additionally shifted the readout before creating the non-linear parts of the feature vector, our results from Figure 6.9 do not suggest that any improved classification accuracy could be gained from

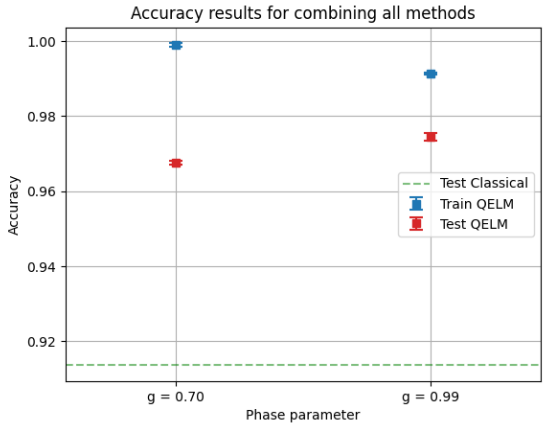


Figure 6.11: Best results for combining several methods. Trained with LinearSVC.

this.
Lastly, the choice of the training method is relatively simple. Throughout this work we used Ridge regression due to its simplicity and speed. However, in Section 6.1.2 we showed that on average, Ridge regression is slightly inferior to the one-layer neural network used by Sakurai et al. [15] as well as a standard linear support vector classifier. The results for these two are similar, although the LinearSVC performs slightly better than the ONN in three out of four cases. Consequently, we choose LinearSVC as training algorithm. It should be noted that this choice is not crucial, most likely the ONN would yield similar results.

The results of the training are displayed in Figure 6.11. As expected, the results are really good in the sense that the classification accuracy is very high. The performance of the QELM is better than the performance for feeding in only the classical information of the image. However, one should keep in mind that the feature vectors do not have the same sizes and, therefore, the performance is only of limited comparability. The classical training is based on 784 dimensions, originating from $28 \cdot 28$ pixels. The QELM works with a feature vector with 1890 dimensions. Measuring 3 observables on 10 qubits leads to 30 measurements per time step. As we include the measurement before any evolution in addition to 20 time steps, we arrive at 630 dimensions. Due to the cubed shuffled readout this feature vector is extended by its own size twice, resulting in 1890 dimensions. Given this large feature vector, we expected an improved performance. However, there are a few drawbacks to be noted. Throughout this work, we only showed the test accuracy for classification as it was not significantly different from the training accuracy. Here, for the first time we experience overfitting. While the training for the thermal phase is sufficiently successful to classify with almost 100% accuracy, the test accuracy does not manage to achieve this level and is significantly lower with less than 97%. Obviously, the model learned some patterns in the training data that are not generalizable to the entire data set. In this case, to improve the results we would have to apply some further techniques

to reduce overfitting during the training process. Considering the results for the discrete time crystal phase we experience the same issue, albeit to a lesser extent. This training process appears to be less prone to overfitting given that a lower training accuracy actually led to a higher test accuracy. Common strategies to reduce overfitting are increasing the regularization strength or the implementation of early stopping. The goal of this training process is to showcase the potential of a combination of all discussed techniques in this work. Given this focus we do not implement the overfitting reduction mechanisms as the tuning of this algorithm is not a focus of our work.
However, the question arises why the discrete time crystal phase leads to less overfitting than the thermal phase. The key difference between the two phases is that the discrete time crystal phase is better at preserving the information fed into the system. In contrast, the thermal phase leads to a fast mixing of the information. The difference in performance is remarkable especially given our previous results from Section 5.2. We learned that the discrete time crystal is superior to the thermal phase without temporal multiplexing. When applying temporal multiplexing, the roles clearly changed and the classification accuracy of the thermal phase is significantly higher. Although both phases perform really well when applying every method, we can identify another reversal of roles. Both systems are trained with the same algorithm and both systems are based on 10 simulations to ensure that one-time effects cannot play a role in the results. Consequently, we can conclude that there must be a fundamental difference in the phases that results in the thermal phase being more prone to overfitting.

Chapter 7

Summary

Throughout this work we have explored and analyzed a QELM based on a discrete time crystal Hamiltonian. While we have not changed the Hamiltonian, we have varied other parameters of the method to gain insights about its behavior as well as potentials to improve their performance. Our analysis was focused on the mnist digits data set which is arguably a simple data set for an algorithm to classify. However, we do not see a reason why the principles investigated in this thesis should not hold for more complicated data sets that are prepared appropriately.
We started the analysis part of this thesis by discussing our implementation of QELMs in chapter 4. This was followed by a presentation and explanation of the standard parameters and settings we used. In chapter 5 we displayed that it is of utmost importance to run several classical simulations with randomly drawn numbers to get statistically reliable results. Furthermore, we investigated the importance of the physical phase for the classification results. The discrete time crystal phase performed well without temporal multiplexing, preserving the inserted information well. In contrast, as expected the information spread quicker in the thermal phase. However, when applying temporal multiplexing the thermal phase clearly yielded better results than the discrete time crystal phase. Apparently, the spread of information is beneficial when constructing the feature vector from multiple time steps.
We also investigated the effect of changing the chain length of the physical model. As this increase requires more PCA components as inputs that also contain more information we expected an improved performance. This could be observed in Figure 5.4. For any choice of 1-qubit observables the performance increased with growing chain length. However, none of the large configurations consistently outperformed the classical training based on the same number of PCA components as input features. Furthermore, we were able to observe an effect of marginal returns as the classification accuracy improvement saturated for systems with chain sizes larger than 10.
The last aspect explored by us in chapter 5 were the observables used in the training process. We conducted this analysis for both physical phases and also included correlation between different qubits as observables which we called 2-qubit observables. Feeding only these observables into the training process in either phase resulted in a similar performance

to the usage of the pure Pauli matrices. However, leveraging both kinds of observables for the feature vector led to different results based on the Pauli matrices used and the phase the system was in. While the discrete time crystal phase exhibited varying behavior for every combination with Pauli-X and Pauli-Z as best performing starting observables, the thermal phase did not show any dependence on the observables used. In both cases, the addition of 2-qubit observables led to an increased performance. In case of the discrete time crystal, the performance exhibited some converging behavior. As last part of chapter 5 we investigated whether the subgroups of 2-qubit observables showed any differentiation in terms of classification accuracy. While we could not identify clear behaviors, there is a trend consistent with the previous results that the thermal phase does not exhibit differentiation whereas the discrete time crystal phase does.

With chapter 5 having been focused on varying aspects of the quantum system, in chapter 6 we turned our focus to the post-processing layer of the QELM. Firstly, we displayed several possible training methods beyond the well-established Ridge regression and compared their performance on four sample configurations. As expected, more complex training algorithms like neural networks and linear support vector machines were able to consistently improve the results. Unfortunately, they also come with a trade-off in computational complexity with the training process taking significantly longer. Given the focus of this work, we continued working with Ridge regression with only one exception.

Next, we discussed a well-established method from RC: Temporal Multiplexing. This method is almost invaluable to achieve results in QELM that are comparable to classical results. Firstly, we showed that, while temporal multiplexing drastically improves the results, it is only of limited usefulness to add further time steps. The growth of classification accuracy per added time step reduces drastically after the first 10-40 time steps, depending on the phase of the system. For every subsequent analysis leveraging temporal multiplexing, we worked with these results by feeding only 20 time steps into the feature vector. Our study of several parts of the evolution process revealed that with a sufficient number of steps included for temporal multiplexing, it does not matter anymore at which point of the evolution one starts measuring. Furthermore, the variations between different segments of the evolution process decreases for increasing feature vector dimensions. Feature vectors built from 50 time steps exhibit almost the same classification accuracy.

The last aspect of the post-processing layer we analyzed was the readout mechanism deployed in the method. As default, we only used the pure observables from one or more time steps and called this method of readout 'identity'. The established variants of readout built on non-linear combinations of this method. We presented these results for nonlinear combinations of up to power 5. The classification accuracy grew logarithmically with growing feature vector size both with and without temporal multiplexing. However, given that the feature vectors with temporal multiplexing are already large without the readout method, the absolute differences of the classification accuracies were smaller. Still, the functional behavior of both cases were similar.

Focusing on the cubed readout, we introduced randomness into the nonlinear combinations by shuffling the feature vector randomly. According to our knowledge, this is an approach nobody else ever followed. To our surprise, it had a strong impact on the classification ac-

curacy. While the results without temporal multiplexing were not affected by this change, the results for temporal multiplexing exhibited rich behaviors depending on which part of the cubed feature vector was shuffled. Our key insight was that the shuffling of the identity part does not make a difference, whereas the shuffling of the 'squared' and 'cubed' segments of the feature vector led to a clear improvement from the regular cubed results. For both, it does not matter whether all factors or only a subset of the multiplication are shuffled. Investigating the cause for these pleasing results we displayed in Figure 6.8 that it might be caused by a more symmetric distribution of the feature vector components.
Building on these insights, we tried two further approaches of nonlinearly processing the observables measured from the physical system. The first approach was to reduce logical complexity of the process by shifting one of the feature vectors involved in the multiplication instead of shuffling it. These shifts were not able to outperform the random shuffling, but some worked similarly well. Apart from that, the best results were achieved by those shifts that led to a strong mixing of qubit and time step. In case of a measurement being combined in one time step with the same observable on the same qubit in later time steps, the classification accuracy dropped to the same level as the original cubed readout without shuffling.
Instead of shuffling and shifting the feature vector components, we also applied four widely used activation functions to the feature vector. While all four functions decreased the classification accuracy compared to the original method without shuffling, the more symmetrical functions led to better results, consistent with the previous results from random processing. Especially notable is the clear decrease for the hyperbolic tangent that almost has the same functional shape as the 'idenity' readout. Consequently, one should not make use of any of the discussed function as the classification accuracy appears very sensitive to this change and the functions were not able to improve the results without shuffling, far away from reaching the shuffled results.
Lastly, we wanted to run a test of what the maximally possible test accuracy with our discussed methods was. Without tuning the hyperparameters, we did not expect perfect results and actually ran into overfitting. However, the results were still better than any previous training process, suggesting that a combination of all methods appears meaningful.

Chapter 8

Outlook

Lastly, we would like to present potential next steps building on our work. In the spirit of this thesis we split these steps into general remarks, further work focused on the quantum layer of the QELM, and suggestions for the readout layer.
In this work we only considered unitary evolutions of quantum systems. Although mathematically simple and easily simulatable, of course, this treatment does not account for real-world complexities of quantum hardware. Consequently, a reasonable and necessary extension would be to repeat the key steps of our analysis with classical simulators taking noise into account. Fortunately, the simulation software used by us already contains this functionality. Additionally, one could test our findings for the third protocol introduced by Camacho and Fauseweh [35]. Given the difficulties arising from noisy quantum channels, their suggested protocol of restoring the quantum state by quantum-classical feedback is also worth investigating.
Apart from this, it is important to note that all results in this work are validated by 10 simulations that depend on draws of random numbers. In consideration of our previous simulations not producing any outliers we do not expect that the number of simulations run by us were too low. However, given that some of the error bars in figures like 6.7 were large enough to easily cover neighboring data points and, therefore, hindering the interpretability of the respective results, it seems reasonable to run further simulations. This would ensure stronger statistical validity and reliability. A potential shrinking of the error bars might also reveal further insights, especially in 5.10.
One of the large beneficial aspects of using discrete time crystals as quantum reservoirs is that they have already been implemented experimentally [37]. Although they are fun, we are not doing Theoretical Physics and numerical simulations for their own sake. In contrast, we are hoping to implement the scheme presented by Camacho abd Fauseweh [35] in an experiment in the near future. In light of this plan, a key question will be how experimentally feasible the results of this work are. Especially the complexity of creating many representations of the same system and the ability to measure a plethora of observables in a reasonable amount of time are factors to investigate. A deeper study of these aspects is necessary and requires close collaboration between theoretical and experimental Physicists. Fortunately, our work supports previous findings that the particular Hamiltonian as well

as the parameters J_j and h_j are not affecting the results strongly. As mentioned earlier, discrete time crystals have already been implemented. Leveraging these experimental possibilities to prove that our results do not only hold for digital simulators is invaluable.
As we demonstrated throughout our entire research, the QELM is not successful in achieving better results than classical machine learning in case of equally sized feature vectors. This also holds true for the same amount of information being fed into both methods. Although we cannot be sure that this search will be successful, it seems necessary to delve further into this topic with a more systematic approach. Some combinations of readout method, temporal multiplexing and observables to train on might still be able to outperform the classical approach if chosen carefully. Any configurations that achieve this or even those that get close can lead to further improvements of QELMs and deepen our understanding of their functionality.
Another general factor to consider for further research is the problem that the QELM is trying to solve. Given that our work is of an exploratory nature, with mnist digits we settled for one of the simplest, standard data sets. Of course, it is the long term goal to develop QELMs that are able to tackle real life goals beyond standardized data sets. While this has already been achieved by de Lorenzis et al. [12], an interesting approach to us would be to use QELMs to solve optimization problems in Physics. One key question that remains in all of these tasks is the dimension of the input data and, if it is too large, how it can be efficiently compressed beyond PCA.

In chapter 5 we discovered a strong correlation between the phase of the physical system and the performance of the QELM that leverages the system as quantum reservoir. Although we have already presented reasonable approaches to physically explain the diverging behavior, a clear theoretical explanation or a proof considering the spread of information over multiple Floquet cycles are crucial next steps. Apart from supporting our hypotheses, leveraging these explanations would yield a more explainable scheme of QELMs.
The latter part of chapter 5 is dedicated to the study of observables measured for our training process. The results found in this section of course only hold for the Hamiltonian that we investigated. As we found differing results for the phases considered, a strong dependency on the Hamiltonian for the performance of individual observables is probable. Also, theoretically the spread of the information throughout the system of course depends on the form of the Hamiltonian used. However, in our example all Pauli matrices as well as all 2-qubit observables were able to achieve results that exceeded random classifications. This raises the question, how strong the relationship between Hamiltonian and favorable observables is. In a subsequent step, the results of numerical simulations, theoretical predictions and experimental implementation should be checked for consistency.

Although we covered multiple important aspects, the investigation of readout effects is far from complete.
The most relevant of our results to investigate further is the improved performance discovered for shuffled readouts. In figure 6.8 we displayed that the improvement may be caused by a more symmetric distribution of the feature vector's components. This insight can be

used to implement more symmetric distribution as an assessment whether the result holds or can even be improved. Additionally, when shifting the components of the vector we did not investigate more complicated mixing methods than just shifting by a fixed number. More sophisticated shifts building on the number of observables or time steps can be imagined and are worth an examination.
An issue that arises from creating non-linear readouts of increasing power is that on average features tend to 0. This could harm training and may be a reason for the lack of improvement examined by higher order combinations. We suspect that intermediate scaling to an appropriate range while keeping the more symmetric distribution could improve the results further.

Appendix A

Additional Plots

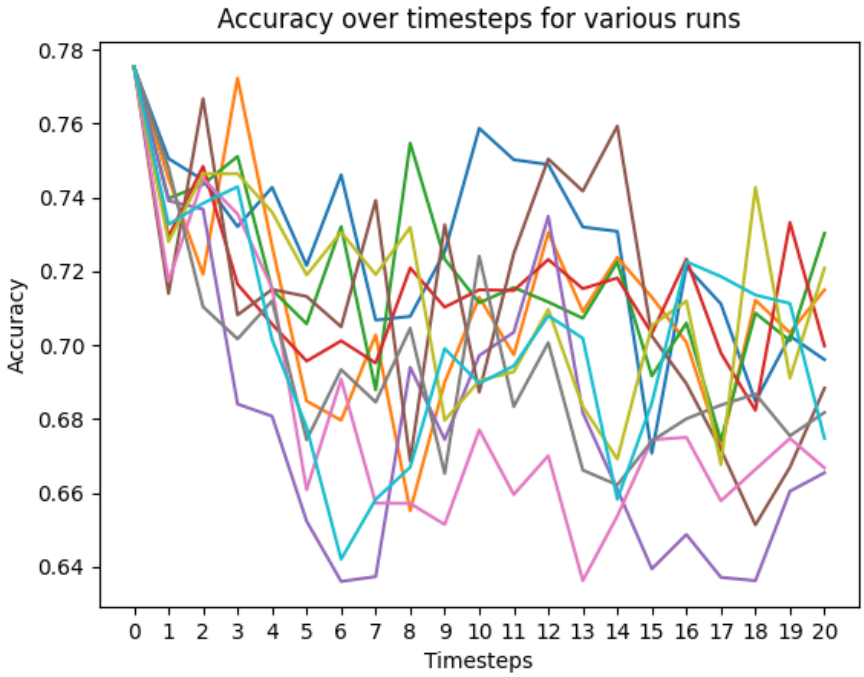


Figure A.1: Single run performance for transition region g = 0.85.

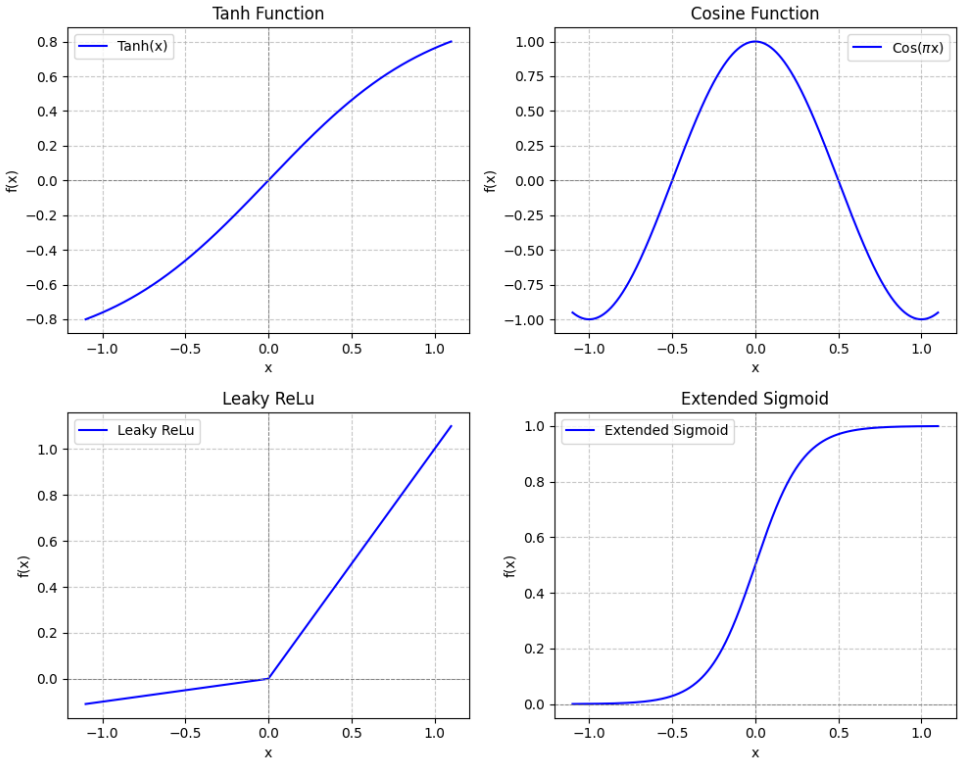


Figure A.2: Graphical representation of suggested readout functions in the relevant regions.

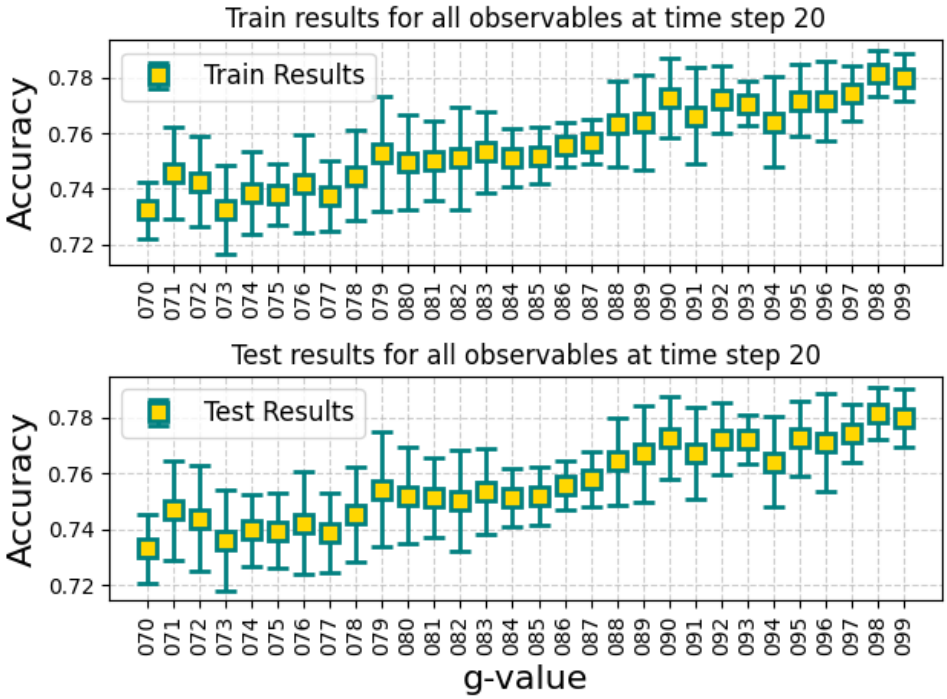


Figure A.3: Train and Test results after 20 time steps without temporal multiplexing over all phase parameters.

Appendix B

Parameters of Sample Configurations for Training Method Comparison

Configuration	1	2	3	4
phase parameter g	0.95	0.70	0.99	0.70
chain size L	20	10	10	10
time steps	[20]	[0:20]	[0:40]	[10]
observables	$[s_x, s_z]$	$[s_x, s_y, s_z]$	$[s_x, s_z]$	$[s_x, s_y, s_z$ + 60 correlations]
readout	squared	identity	squared	cubed

Table B.1: Sample configurations for the comparison of training methods.

Bibliography

[1] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, 1981.

[2] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proceedings of the Royal Society A*, vol. 400, pp. 97–117, 1985.

[3] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," pp. 124–134, 1994.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," p. 212–219, 1996.

[5] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, p. 226–245, July 2019.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[7] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu,

X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025.

[8] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," 2021.

[9] V. Dunjko, J. M. Taylor, and H. J. Briegel, "Quantum-enhanced machine learning," *Phys. Rev. Lett.*, vol. 117, p. 130501, Sep 2016.

[10] L. Gyongyosi and S. Imre, "Training optimization for gate-model quantum neural networks," *Scientific Reports*, vol. 9, Sept. 2019.

[11] R. Martínez-Peña, G. L. Giorgi, J. Nokkala, M. C. Soriano, and R. Zambrini, "Dynamical phase transitions in quantum reservoir computing," *Phys. Rev. Lett.*, vol. 127, p. 100502, Aug 2021.

[12] A. D. Lorenzis, M. P. Casado, M. P. Estarellas, N. L. Gullo, T. Lux, F. Plastina, A. Riera, and J. Settino, "Harnessing quantum extreme learning machines for image classification," 2024.

[13] H. Jaeger, "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, 01 2001.

[14] J. Steinegger and C. Räth, "Predicting three-dimensional chaotic systems with four qubit quantum systems," *Nature Scientic Reports*, vol. 15, 6201, 2025.

[15] A. Sakurai, M. P. Estarellas, W. J. Munro, and K. Nemoto, "Quantum extreme reservoir computation utilizing scale-free networks," *Phys. Rev. Appl.*, vol. 17, p. 064044, Jun 2022.

[16] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[17] L. Smith, "A tutorial on principal component analysis." Université de Montréal, 2002.

[18] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," *Physical Review A*, vol. 102, Sept. 2020.

[19] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, "Hierarchical quantum classifiers," *npj Quantum Information*, vol. 4, Dec. 2018.

[20] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*. 01 2018.

[21] M. Ippoliti, K. Kechedzhi, R. Moessner, S. Sondhi, and V. Khemani, "Many-body physics in the nisq era: Quantum programming a discrete time crystal," *PRX Quantum*, vol. 2, p. 030346, Sep 2021.

[22] X. Mi, M. Ippoliti, C. Quintana, A. Greene, Z. Chen, J. Gross, F. Arute, K. Arya, J. Atalaya, R. Babbush, J. C. Bardin, J. Basso, A. Bengtsson, A. Bilmes, A. Bourassa, L. Brill, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, B. Chiaro, R. Collins, W. Courtney, D. Debroy, S. Demura, A. R. Derk, A. Dunsworth, D. Eppens, C. Erickson, E. Farhi, A. G. Fowler, B. Foxen, C. Gidney, M. Giustina, M. P. Harrigan, S. D. Harrington, J. Hilton, A. Ho, S. Hong, T. Huang, A. Huff, W. J. Huggins, L. B. Ioffe, S. V. Isakov, J. Iveland, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, T. Khattar, S. Kim, A. Kitaev, P. V. Klimov, A. N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, J. Lee, K. Lee, A. Locharla, E. Lucero, O. Martin, J. R. McClean, T. McCourt, M. McEwen, K. C. Miao, M. Mohseni, S. Montazeri, W. Mruczkiewicz, O. Naaman, M. Neeley, C. Neill, M. Newman, M. Y. Niu, T. E. O'Brien, A. Opremcak, E. Ostby, B. Pato, A. Petukhov, N. C. Rubin, D. Sank, K. J. Satzinger, V. Shvarts, Y. Su, D. Strain, M. Szalay, M. D. Trevithick, B. Villalonga, T. White, Z. J. Yao, P. Yeh, J. Yoo, A. Zalcman, H. Neven, S. Boixo, V. Smelyanskiy, A. Megrant, J. Kelly, Y. Chen, S. L. Sondhi, R. Moessner, K. Kechedzhi, V. Khemani, and P. Roushan, "Time-crystalline eigenstate order on a quantum processor," *Nature*, vol. 601, p. 531–536, Nov. 2021.

[23] V. Khemani, R. Moessner, and S. L. Sondhi, "A brief history of time crystals," 2019.

[24] D. A. Abanin, E. Altman, I. Bloch, and M. Serbyn, "Colloquium: Many-body localization, thermalization, and entanglement," *Rev. Mod. Phys.*, vol. 91, p. 021001, May 2019.

[25] D. Basko, I. Aleiner, and B. Altshuler, "Metal–insulator transition in a weakly interacting many-electron system with localized single-particle states," *Annals of Physics*, vol. 321, no. 5, pp. 1126–1205, 2006.

[26] J. Imbrie, "On many-body localization for quantum spin chains," *Journal of Statistical Physics*, vol. 163, pp. 998–1048, 2016.

[27] A. Lazarides, A. Das, and R. Moessner, "Fate of many-body localization under periodic driving," *Phys. Rev. Lett.*, vol. 115, p. 030402, Jul 2015.

[28] R. Nandkishore and D. A. Huse, "Many-body localization and thermalization in quantum statistical mechanics," *Annual Review of Condensed Matter Physics*, vol. 6, p. 15–38, Mar. 2015.

[29] P. Ponte, Z. Papić, F. m. c. Huveneers, and D. A. Abanin, "Many-body localization in periodically driven systems," *Phys. Rev. Lett.*, vol. 114, p. 140401, Apr 2015.

[30] D. A. Huse, R. Nandkishore, V. Oganesyan, A. Pal, and S. L. Sondhi, "Localization-protected quantum order," *Phys. Rev. B*, vol. 88, p. 014206, Jul 2013.

[31] L. Innocenti, S. Lorenzo, I. Palmisano, A. Ferraro, M. Paternostro, and G. M. Palma, "Potential and limitations of quantum extreme learning machines," *Communications Physics*, vol. 6, May 2023.

[32] J. H. . S. B. Lukoševičius, M., "Reservoir computing trends," *Künstliche Intelligenz*, vol. 26, pp. 365–371, 2012.

[33] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[34] X. Wang, S. Ali, A. Arrieta, P. Arcaini, and M. Arratibel, "Application of quantum extreme learning machines for qos prediction of elevators' software in an industrial context," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, FSE 2024, (New York, NY, USA), p. 399–410, Association for Computing Machinery, 2024.

[35] G. Camacho and B. Fauseweh, "Prolonging a discrete time crystal by quantum-classical feedback," *Phys. Rev. Res.*, vol. 6, p. 033092, Jul 2024.

[36] V. Khemani, A. Lazarides, R. Moessner, and S. Sondhi, "Phase structure of driven quantum systems," *Physical Review Letters*, vol. 116, June 2016.

[37] P. Frey and S. Rachel, "Realization of a discrete time crystal on 57 qubits of a quantum computer," *Science Advances*, vol. 8, no. 9, p. eabm7652, 2022.

[38] H. Ma, D. Prosperino, and C. Räth, "A novel approach to minimal reservoir computing," *Scientific Reports*, vol. 13, 2023.