A Machine Learning Model for Trapped-ion State Classification

by

Severyn Balaniuk

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Science
in
Physics (Quantum Information)

Waterloo, Ontario, Canada, 2025

© Severyn Balaniuk 2025

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapter 4 – Raw fluorescence dataset

The underlying data were collected by the Group of Dr. Rajibul Islam (University of Waterloo). *My contribution:* data-set specification, preprocessing, feature extraction, model training, and analysis.

Chapters 4 and 5 – Machine-learning pipeline

Yi Hong Teoh assisted with problem-solving and debugging during development of the data-handling and training scripts. *My contribution:* wrote the final code base, executed all experiments, generated figures, and drafted the text.

All other chapters and analyses were solely authored by me. Unless explicitly credited in its caption, every figure was created by me.

Abstract

Academia and industry have been working to build a quantum computer that is able to perform certain tasks significantly better than classical computers. This thesis focuses on improving a trapped-ion-based approach to quantum computing. This platform has advanced significantly over the last 10 years, but there are numerous issues we need to resolve to make this architecture scalable. We consider experiments that use a high-sensitivity photon detection module as the readout tool. This setup lets us see qubit measurement outcomes as a fluorescent signal on a digital camera. This thesis addresses the problem of classifying fluorescence states for experimental systems of up to four ions, representing them as binary sequences. Our model hopefully will be able to classify systems with 8, 16, and 32 ions, allowing for further application of this methodology as quantum computers grow in scale. Our datasets are mainly unlabelled, which is the biggest challenge for training an accurate machine learning (ML) model. Nevertheless, we showed a significant improvement in classification accuracy with reduced bias over legacy models on unlabelled data containing mixed states of a four-ion system. We tested different machine learning architectures like feed-forward neural network (FFNN), convolutional neural networks (CNN), and semi-supervised learning to evaluate their efficiency for our specific dataset and tested their performance. In addition, we also developed and improved our own FFNN architecture with custom loss functions.

Acknowledgements

I would like to thank my supervisor, Roger Melko, for his support and guidance. Teoh Yi-Hong, for his ideas and assistance during project development. My family for believing in my research and goals. My friends for keeping my spirit high during the challenging times.

Dedication

This is dedicated to the pursuit of knowledge, the unstoppable curiosity that drives innovation, and the individuals who inspire and support us along the way to a universal quantum computer.

Table of Contents

Author's Declaration ii

Statement of Contributions iii

Abstract iv

Acknowledgements v

Dedication vi

List of Figures ix

List of Tables xiv

1 Introduction 1

1.1 Background 1

1.2 Key Principles 2

1.2.1 Quantum State Fundamentals 3

1.2.2 Machine Learning Framework 7

1.3 Previous Work 9

1.4 Thesis Goal 12

2 The Experimental Trapped-ion Platform 13

2.1 Properties and Choice of Atoms 14

2.2 Readout Details 18

3 Machine Learning Theory 23

3.1 Feedforward Neural Network 24

3.2 Convolutional Neural Network 27

3.3 Variational Autoencoders 29

3.3.1 Theoretical Framework . . . 29
3.3.2 Optimization Objective . . . 30
3.4 Semi-Supervised Learning . . . 30
3.4.1 The Mean Teacher . . . 31
3.4.2 ReMixMatch . . . 32

4 Data and Methodology **37**
4.1 Data Collection . . . 38
4.1.1 Initial Approach and Challenges . . . 40
4.1.2 Improved Approach: Classifying Individual Ions . . . 41
4.1.3 Cropping of Ion Images . . . 41
4.1.4 Data Preparation . . . 44
4.2 Other Architectures . . . 45
4.2.1 Convolutional Neural Network results . . . 45
4.2.2 Autoencoder-Based Semi-Supervised Learning Approach . . . 47
4.3 FFNN development . . . 51

5 Results **57**
5.1 Final Model Development and Results . . . 61
5.2 Final Comparison . . . 70

6 Conclusions **72**
6.1 Outlook . . . 73

References **74**

List of Figures

1.1 Adopted from Ref. [51]. Ions confined in a trap. Trapped ions are located inside the electro-magnetic field and their fluorescence is measured by a camera. 2
1.2 Depicting 4 states of a qubit using Bloch sphere representation. 4
1.3 Generic three-level system. (a) Ions have three energy levels, each corresponding to a different electronic state of the ion. On the bottom, we have our ground state, which can be transferred to the fluorescent cycling state $|s\rangle$ via laser excitation, which we need for measurement. However, transitions from $|1\rangle$ to $|s\rangle$ are deliberately excluded by tuning the excitation laser frequency to be resonant only with the $|0\rangle$ to $|s\rangle$ transition, exploiting the different energy gaps between states. (b) We can observe the result of the fluorescence, noticing how as we grow a system we encounter more merging between ion signals on camera. 6

1.4 Adopted from Ref. [35]. Imaging of Quantum States using Electron Multiplying Charge-Coupled Device (EMCCD) detection. This figure shows two important imaging representations of four-ion configurations prepared in $^{171}Yb^{+}$. Panel (a) shows the time-averaged acquisition, where the original 36×36 pixel frame is binned into a much smaller 16×16 pixel region to enhance clarity. Two regions of interest are segregated, specifically, the thresholding method region enclosed by red borders and the maximum likelihood estimation (MLE) method region represented in white. Panel (b) shows sample single exposures for each of the sixteen possible quantum states in our four-ion system. The single exposures have complex intensity distributions because of the shot-noise-limited conditions. The intensities in a single exposure are inhomogeneous and point to detailed features of the signal that could not be correctly captured by standard thresholding and maximum likelihood methods due to crosstalk (an optical interference between neighbouring ions where fluorescence from one ion affects the measurement of adjacent ions) and other effects of time-varying background signals. 10

2.1 Qubits encoded in the internal energy levels of an ion. (a) Optical qubit; (b) Hyperfine qubit. 15

2.2 Energy levels of $^{171}Yb^{+}$. 17

2.3 Adopted from Ref. [65]. The EMCCD camera used a 0.9 N.A. spherical mirror to capture a single Ba^{+} ion picture, which was then compared to a 0.25 N.A. multi-element lens with a total magnification of 70 (see inset). The exposure duration was 0.8 seconds for both images. The pixel size is 10 μm. The picture intensity is expressed in the same (arbitrary) units. Mirrors have a much greater integrated intensity of the ion picture compared to multi-element lenses. 19

2.4 Sources of noise impacting ion measurements visualized: (1) Environmental noise, i.e., external vibrations and electromagnetic interference, which disrupt ion positioning and cause fluctuations in fluorescence collection; (2) Electrical noise from RF electrodes used for trapping ions, which can modulate the trapping potential and cause ion motion that blurs fluorescence images; (3) Intensity and frequency fluctuations from laser sources, which directly alter fluorescence rates and detection efficiency; and (4) Cross-talk noise from neighbouring ions, which can lead to spatial overlap of fluorescence signals. All these noise sources directly influence the fluorescence signal captured by our camera, affecting both the spatial and intensity characteristics of the detected photons, thereby impacting the accuracy of quantum state detection. 21

3.1 The FFNN architecture can be represented by a diagram. It accepts feature vectors as inputs that convert multi-dimensional inputs such as images to one-dimensional arrays. Inputs are processed through multiple layers of hidden units with activation functions and then produce a single binary output as a prediction. 24

3.2 Adapted from Ref. [35]. CNN structure processing a single image. We can see how the image is used as inputs going through convolution layers of feature extraction, and activation functions in between with skip connections to preserve specific input information and classification stage. 28

3.3 Basic architecture of an autoencoder-decoder. It has an input layer of size 100 and encodes an image into a latent space of size 4 through two dense layers with sizes 50 and 25, respectively. The decoder then reconstructs the image by increasing the dimensionality. 29

3.4 Adopted from Ref. [72]. The Mean Teacher method has one data point as an input. The image is fed through the student and teacher models, adding noise (η, η') in their computation. In this process, the softmax output of the student model is compared to the one-hot label to compute the classification loss and to the teacher model output in order to compute the consistency loss. The last step is a weighted sum of these two loss terms with the application of the resulting weight. 32

3.5 Adopted from Ref. [3]. A showcase of distribution alignment, we can see how the theoretical model "guesses" labels for unlabelled data points, and then we adjust this distribution according to ground-truth labels divided by average model predictions. 34

4.1 Averaged raw ion emission imaging for $|1111\rangle$ state. 37
4.2 A bar chart illustrating the ideal uniform classification of 10,000 half-pi images of four-qubit projective measurements, resulting in 16 basis states from $|0000\rangle$ to $|1111\rangle$. The bar height indicates the frequency with which each state was classified by the ML model. 38
4.3 The result of a classification of mixed state by an early-stage feedforward ML model. The frequency distribution is highly inaccurate and biased towards $|0000\rangle$. 40
4.4 Raw images from the dataset. All of these images are labelled as bright, meaning we should observe four distinct bright spots from the ions' fluorescence. 41
4.5 ROIs in different images we targeted and cropped for further use. Half-pi labels mean that states of ions are random, dark means that we should see no fluorescence from ions, and bright means that we should expect four ions to be fluorescent. 42
4.6 Illustration of the training and validation metrics for the CNN model over 300 epochs. The left panel shows the dynamics of training and validation loss (a measure of the model's prediction error), while the right panel tracks the accuracy of the validation. 47
4.7 Loss to epoch graph for deep autoencoder for reconstructing input images. Trained for 300 epochs and showed a positive result in reducing loss to an optimal level, showing a plateau after 150 epochs. 50
4.8 Architecture of the custom FFNN model for quantum state classification. The model processes each of the four ions separately through parallel channels. Each 5×5 pixel ion image is flattened to a 25-dimensional vector by a reshape (R) function and passed to a linear layer that reduces dimensionality to 16 features. The features from all four ions are then combined in a reshape layer before final classification, finally outputting the complete four-qubit state (represented as four binary digits). This modular architecture allows independent processing of each ion to capture its unique patterns while allowing for a unified quantum state classification. 53
4.9 Early-stage classification distribution with an obvious bias toward $|0000\rangle$ and $|1111\rangle$ states. 54
4.10 Distribution of classified mixed states by custom FFNN model trained on constructed labelled images of all 16 possible states. 55

5.1 Two-stage model. The first stage shows the pipeline with four labelled images processed by the encoder and classifier to learn the main features of images. The second stage uses the frozen weight of the first-stage encoder additionally implementing a Coupler to combine all the weights and biases from images, followed by a pre-trained classifier with a softmax function at the end. Finally, we use our generated distribution to tune the model automatically and enhance the results of classification after each epoch. . . 61
5.2 Comparison of performance of 2 models: one with tuning turned off and one with a Coupler block not working. This demonstrates the importance of the Coupler shown in Figure 5.1 as a tool for sharing combined information from all images. 64
5.3 We show the first improvement in the working version of the model. Also, the sum distance from a uniform distribution of states was introduced as a measure of success. 65
5.4 A small fine-tuning of hyperparameters shows better results in distribution uniformity for both M_1 and M_2 models. Interestingly, when bias towards edge states is reduced, we observe more instability in middle states. 66
5.5 Averaged bright images are shown for each power level: -2 dBm, -3 dBm, -4 dBm, -5 dBm, and -6 dBm. The average brightness distribution is given in each heatmap's grid. Yellow corresponds to a high intensity, while purple corresponds to a low one. Power levels in the first and second rows decrease from left to right. 67
5.6 The untuned results of D_3 show a significantly larger bias towards classifying states as $|1111\rangle$. Our hypothesis is that the improved brightness in the new dataset causes more borderline cases, where ions with ambiguous brightness levels that were previously classified as $|0\rangle$ are now more likely to be classified as $|1\rangle$ by default. 68
5.7 Comparison of the predicted state distributions of the M_2 and M_1 models against the ideal Uniform Distribution for 16 quantum states ranging from $|0000\rangle$ to $|1111\rangle$. The M_2 model has a much lower sum distance of 0.0334 to the uniform target compared to the M_1 model, which had a sum distance of 0.1722. This reflects that there is less bias for all states, which is a good sign for future generalization. 69

List of Tables

4.1 Data shapes and their descriptions. 45
4.2 Performance comparison of autoencoder architectures showing reconstruction accuracy, final loss, and validation accuracy on labelled data, when combined with deep neural network classification. 51

5.1 Definition and interpretation of M_1 and M_2 models 70

Chapter 1

Introduction

1.1 Background

Quantum computing could enable breakthroughs in many fields by solving problems intractable with classical computers. That power comes from leveraging two key quantum mechanical properties: superposition and entanglement. Whereas a classical bit can only be 0 or 1, a qubit can exist in a superposition of both states, and multiple qubits can represent an exponentially large number of superposition combinations simultaneously. These features enable groundbreaking algorithms such as Grover's search, which quadratically speeds up unstructured searches [26], and Shor's factoring [64], which could break widely used cryptographic schemes by factoring large numbers exponentially faster than any known classical algorithm. Recently, trapped-ion platforms have emerged as one of the leading technologies in the field of quantum computation, owing to high-fidelity gate execution and very long coherence times over one hour [1, 22, 68, 75]. However, many challenges must be overcome before fault tolerance in quantum computing is achieved; such a computer would be able to compute deep circuits with the potential to bring real value to the world and revolutionize certain fields of technology [24, 44]. Among existing problems with current trapped ion machines is the very precise preparation and readout of states of ions in such systems.

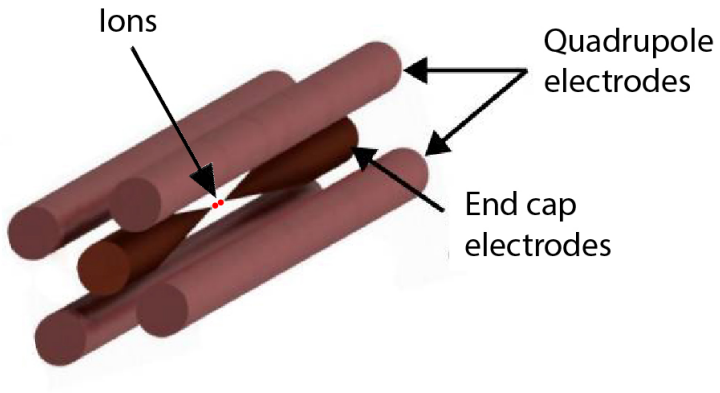


Figure 1.1: Adopted from Ref. [51]. Ions confined in a trap. Trapped ions are located inside the electro-magnetic field and their fluorescence is measured by a camera.

The figure above shows a simplified ion trap, often referred to as a "rod trap". It belongs to the broader category of Paul traps [7]. In this setup, a trap holds individually addressable $^{171}Yb^{+}$ ions, which, after being cooled by a laser beam, line up in a row and are imaged using a camera after being addressed accordingly [51]. It is one of many ways to confine ions in a trap and control them, but we are interested in the concept in general, we will talk about the trap and what happens during the manipulation with ions in the following Chapter 2. After ions are held in place, we can manipulate their energy levels using precise lasers to flip spins, create superposition, induce entanglement, or put them into a fluorescent cycling state. Fluorescence is our primary focus in this process; collecting photons from the fluorescent emission provides us with information on whether an ion (or the entire set of ions) is in the $|0\rangle$ or $|1\rangle$ state.

1.2 Key Principles

Suppose the reader is unfamiliar with trapped ion platform concepts and wants to learn additional details. In that case, this paper is highly recommended [29], in a simple and approachable manner, the author explains a brief history and key developments on the way to build a trapped ion quantum computer (TIQC). If a more extensive or up-to-date overview is required, Ref. [8] provides a broader review. In particular, Chapter 5 outlines the physics part of TIQC. In this book [48], readers can familiarize themselves with the principal physical derivations necessary to fully understand the operation of the Paul trap.

These resources collectively cover the key principles of ion trapping, qubit initialization, quantum gate operations, and the role of entanglement in TIQCs, offering a well-rounded foundation for anyone entering this field.

The intersection of quantum mechanics and machine learning in this work introduces a novel framework for the goal of state detection. This chapter sets up the basic principles from both disciplines, which, when put together, will finally enable our approach of chapters three and four.

1.2.1 Quantum State Fundamentals

The fundamental distinction between classical and quantum computing is how information is encoded and manipulated. Bits in classical computing can only exist in two states: 0 and 1. In contrast, qubits, which are used in quantum computers, have fundamentally different states thanks to the laws of quantum mechanics. The diversity of states given to the qubit by its nature allows, in many ways, many more possible operations, but at the cost of extra complexity with readout, error correction, and more.

The mathematical representation of a qubit's state is done using Dirac notation, where the basis states are denoted as $|0\rangle$ and $|1\rangle$. These states correspond to two distinct energy levels of the ion. $|0\rangle$ - represents the ground electronic state and $|1\rangle$ - could be encoded into a range of stable or metastable excited states. Another way to represent this notation is in matrix form: $|0\rangle$ is just the column vector:

$$|0\rangle \doteq \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{1.1}$$

and similarly $|1\rangle$:

$$|1\rangle \doteq \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1.2}$$

This vector notation allows tractable mathematical representations of quantum systems. For instance, it is useful to calculate how quantum gates, which are also representable in matrix form, can evolve given states.

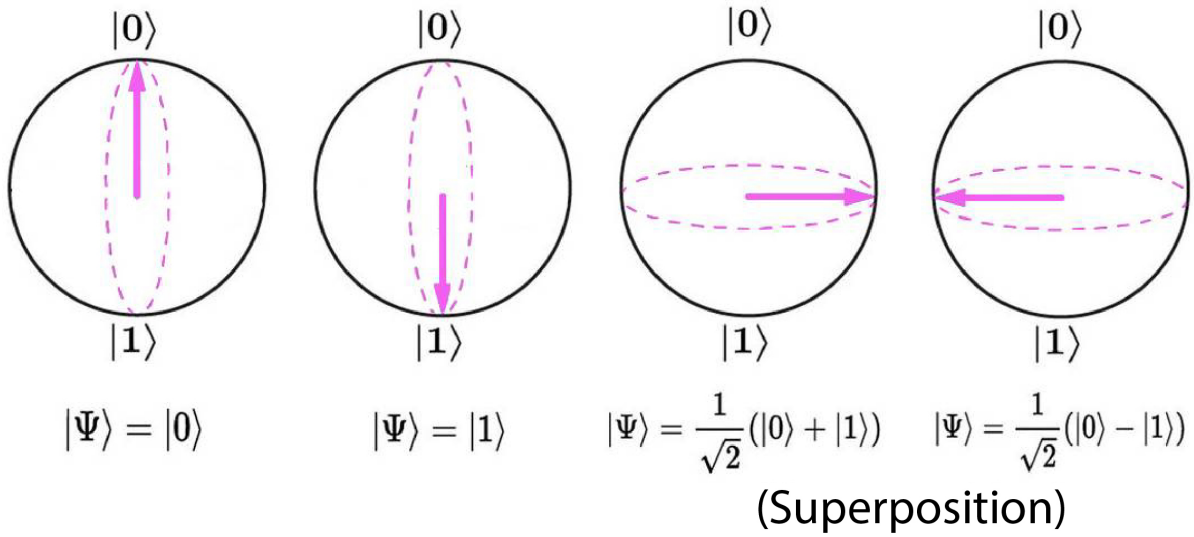


Figure 1.2: Depicting 4 states of a qubit using Bloch sphere representation.

A very intuitive visualization of the state space of a qubit is something called the Bloch sphere, where any point on the surface of the sphere represents a superposition state of the qubit shown in the figure in Fig. 1.2. The state vector pointing to the north or south pole corresponds to the computational basis states $|0\rangle$ and $|1\rangle$, respectively. If the vector is pointing along the equator, the state is an equal superposition; the measurement of the qubit will return 0 or 1, each with 50% probability. Mathematically, a qubit state $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$ maps directly to spherical coordinates, with the computational basis states and geometric representations help to better visualize quantum operations and interpret state evolution [54].

The true beauty of quantum computing comes when considering several qubits. A system comprising n qubits may exist in a superposition of 2^n different states all at once. But quantum mechanics adds a condition: this complex quantum state collapses following measurement resulting in precisely n classical bits of information. This outcome of a measurement is itself probabilistic, for instance, the three-qubit state $\frac{1}{2}|000\rangle + \frac{\sqrt{3}}{2}|111\rangle$ would yield the bit string "000" with 25% probability and "111" with 75% probability, corresponding to the squared magnitudes of their respective amplitudes. Hence, quantum algorithms must carefully coordinate quantum operations to exploit superposition throughout the computation, while ensuring that the desired outcome has accessibly high measurement accuracy.

As an example, for two-qubits, where both are put into superposition states $|\psi\rangle = \frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$, the result of each measurement will be some random binary outcome. Repeated measurements of identically prepared states reveal a uniform distribution over all possible two-bit combinations: $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. Again, this is the probabilistic nature of quantum mechanics, and it influences, how we imagine and construct quantum algorithms.

Finally, the measurement of quantum states in trapped-ion systems differs significantly from classical state measurement. In a typical (classical) computer, "state measurement" involves just reading the value of a bit from its storage location (such as a memory cell or CPU register). In a traditional computer, for example, a memory cell can represent a bit with either a low or high voltage. When the system "reads" this cell, a sensing amplifier or comparator (an electronic device that compares two voltages or currents) determines whether the voltage level is closer to zero or to some predetermined voltage value [21]. That technique is practically instantaneous and highly reliable as compared to reading quantum states because classical bits are not susceptible to quantum effects such as decoherence, spontaneous emission, and quantum tunnelling. Classical bits are also more robust against different sources of noise including thermal fluctuations, ambient electronic noise, radiofrequency interference from nearby equipment, and stray electromagnetic fields from the trap-driving electronics. For trapped ions, we utilize the quantum mechanical principle of fluorescence, where ions in different energy states emit photons when excited by a laser at different rates. The number of photons caught on camera is our analogue of voltage levels in classical computers.

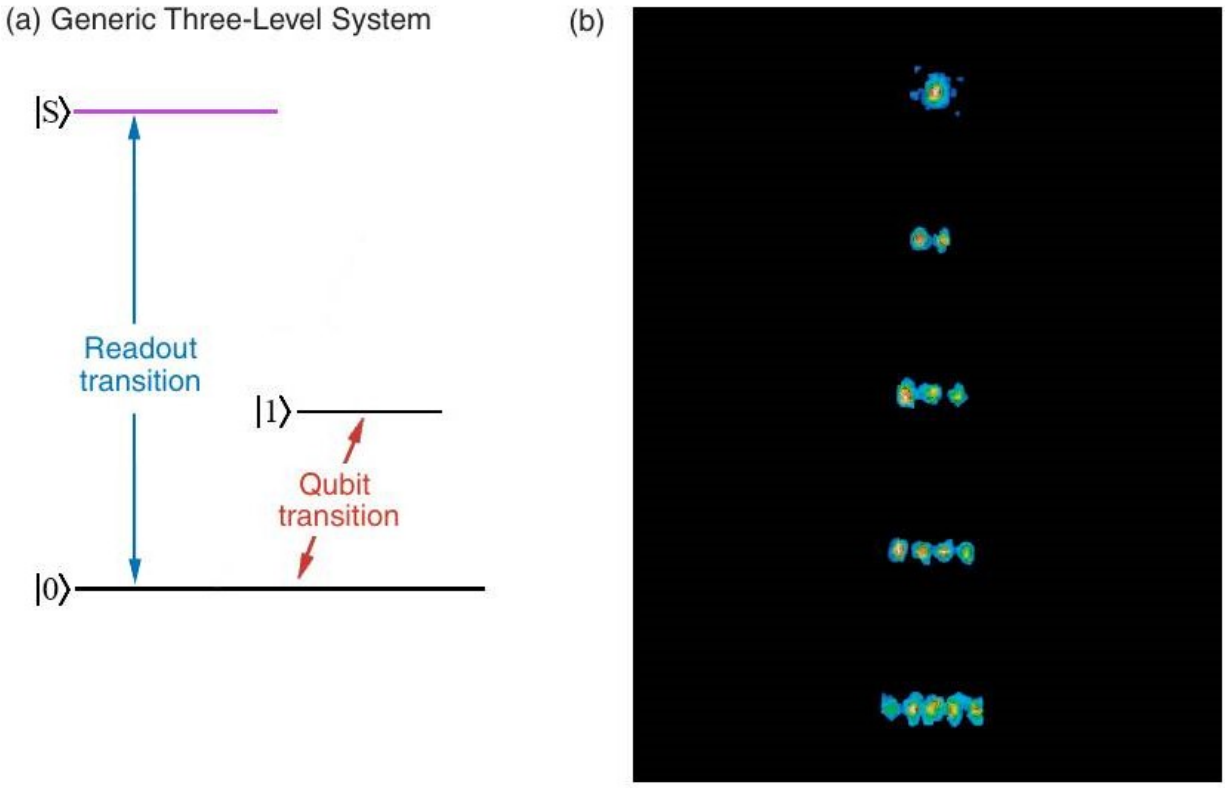


Figure 1.3: Generic three-level system. (a) Ions have three energy levels, each corresponding to a different electronic state of the ion. On the bottom, we have our ground state, which can be transferred to the fluorescent cycling state $|s\rangle$ via laser excitation, which we need for measurement. However, transitions from $|1\rangle$ to $|s\rangle$ are deliberately excluded by tuning the excitation laser frequency to be resonant only with the $|0\rangle$ to $|s\rangle$ transition, exploiting the different energy gaps between states. (b) We can observe the result of the fluorescence, noticing how as we grow a system we encounter more merging between ion signals on camera.

Imagine that, suppose we have performed a certain operation with ions in the system, and now we want to measure the result. This measurement is done using state-dependent fluorescence. In order to produce photon emission, we first excite the ion from its ground state to an auxiliary excited state $|s\rangle$ with a short lifetime, as shown in Figure 1.3. This auxiliary state is critical to the detection process and is coupled together with the ground state.

Applying a detection laser resonant with the $|0\rangle \rightarrow |s\rangle$ transition, an ion in the ground state continuously scatters photons by cycling between these levels. If the ion is in the $|1\rangle$ state, it does not interact with the detection laser and thus appears dark. This large

difference in fluorescence provides a robust method to distinguish the basis states. Other approaches make use of long-lived hyperfine ground state levels for encoding qubit states and use comparable detection principles.

Additionally, Figure 1.3 (b) illustrates this picture of fluorescence. When trapped in a linear Paul trap and laser-cooled to near their motional ground state, the ions form clear images, equally spaced between each other. During detection, each ion in the bright state scatters of the order $10^6 - 10^8$ photons per second. This high flux of photons can be detected efficiently using a camera. However, typical detection systems can only collect up to 10% of these photons on average with numerical aperture (NA) of at least 0.6 [66], because the camera covers a tiny region in space to detect photons, and at the same time, ions emit photons in all directions in three dimensions (3D) making detecting all of them impossible [67]. This kind of detection illustrates another important point: the measurement of quantum states in trapped-ion systems maps a quantum mechanical state into a classical one through the emission of light by atoms. This fluorescence pattern creates a macroscopic, classical signal that can be detected by conventional cameras, providing a reliable method to read out the final results of quantum computation.

1.2.2 Machine Learning Framework

The motivation for combining quantum mechanics with machine learning arises from how they complement each other. Machine learning can find subtle patterns in measurement data, which may support or verify results from traditional methods. The scalability and accuracy of our method of state detection could become a vital ingredient when we evolve to larger quantum systems where the final calculation result could be compromised by measurement errors with way higher probability due to the increasing complexity of read-out signals. As quantum processors scale up, the fluorescence patterns from multiple ions become more difficult to distinguish, with overlapping point-spread functions and increased susceptibility to experimental imperfections [6, 45]. The traditional methods of state detection, which rely on thresholding photon counts or fitting expected distributions to observed fluorescence, can easily suffer from noise, cross-talk between ions, background scattering, limitations in detector sensitivity, and existing hardware flaws [9, 58].

How are machine algorithms able to help this process? We start with transforming raw quantum measurements into machine learning features through this process:

$$\text{Readout transition} \xrightarrow{\text{Measurement}} \text{Data analysis} \xrightarrow{\text{Processing}} \text{Features} \xrightarrow{\text{ML model}} \text{Binary state} \tag{1.3}$$

The complete process of quantum state detection in our trapped-ion system can be described through a series of transformations. We begin with an initial quantum state:

$$|\psi_{\text{initial}}\rangle \xrightarrow{\text{Operations}} |\psi_{\text{final}}\rangle \xrightarrow{\text{Measurement}} |b_1\rangle_1 \otimes |b_2\rangle_2 \otimes |b_3\rangle_3 \otimes \cdots \otimes |b_N\rangle_N \tag{1.4}$$

Here each $|b_i\rangle$ represents the measured basis state ($|0\rangle$ or $|1\rangle$) for the i-th qubit, depending on the measurement outcome.

The quantum system emits photons that our detectors collect, producing a sequence of intensity measurements across the spatial coordinates of our detector. Unlike ideal binary measurements, each pixel in the image produced by a camera captures fluorescence intensity on a continuous scale. Illustrating examples could typically range from 0 to 15 in arbitrary digital units after standard normalization used in our image processing. A count of 0 means no detectable fluorescence, and that the pixel didn't receive any photons. Values closer to 15 thus imply that the fluorescence is at a maximum, meaning the ion is emitting photons at a high rate. This range provides a very convenient way to standardize the intensity measurements and thus interpret and compare the intensity of fluorescence from one experiment to another. Therefore, the mathematical representation of this process could be described as a transformation shown below. Our measurement process can be described as a two-step procedure. First, we collect the fluorescence image of our ion chain:

$$\text{Readout Transition}(|\psi_{\text{final}}\rangle) \rightarrow I_{full}(x, y) \tag{1.5}$$

Here $I_{full}(x, y)$ represents the full fluorescence image of our ion chain, i.e., how many pixels ranging from 0 to 15. We then identify and extract regions of interest (ROIs) corresponding to individual ions:

$$I_{full}(x, y) \xrightarrow{\text{ROIs Extraction}} \{I_1, I_2, I_3, I_4\} \tag{1.6}$$

Here each I_k represents the cropped region containing the k-th ion's fluorescence pattern, i.e., some smaller number of pixels. Our neural network then operates on these individual regions:

$$f_\theta : \{I_1, I_2, I_3, I_4\} \rightarrow \{|0\rangle, |1\rangle\}^{\otimes 4} \tag{1.7}$$

These formulas show how we separate the relevant fluorescence data from each other before performing state classification. Particularly, the neural network learns how to recognize complicated intensity patterns in each ROI without needing to act on individual pixels or

an entire picture. This strategy is more efficient because it allows the network to focus just on the characteristics that discriminate between bright and dark states in each ion's individual fluorescence pattern.

This thesis introduces a framework allowing for the analysis of both pure and mixed quantum states. It addresses the state classification problem using machine learning techniques with quite limited labelled data input.

1.3 Previous Work

In previous studies on the machine learning approach to quantum state readout [36, 37], learning on simulated data was explored to test theory and to demonstrate its reliability to classify images using labelled or simulated data [73]. Machine learning models are relatively straightforward to train and test on simulated data because the author manually defines the parameters of the simulated system, computing it using Monte-Carlo integration and it allows to have high confidence in results. Another work on the classification of qubit states with ML showed promising results yet with architectures that could be challenging to scale for larger systems [62]. They employed experimental data from ion trap systems and achieved a 30% improvement over conventional threshold methods employing spatial and temporal features within neural network models [62]. Their approach relied on having a well-understandable noise behaviour within a controlled experimental setup, however, something that is less maintainable when scaling up the size of the quantum system. Unlike our approach, which solves the issue of incomplete data within complex multi-ion structures, theirs is centred on optimizing detection fidelity within well-characterized systems. The next approach shown by researchers is a CNN architecture and specifically a pre-trained ResNet model to classify their dataset. They achieved an impressive $99.53 \pm 0.14\%$ mean individual measurement fidelity (MIMF) of 4 trapped ions [35].

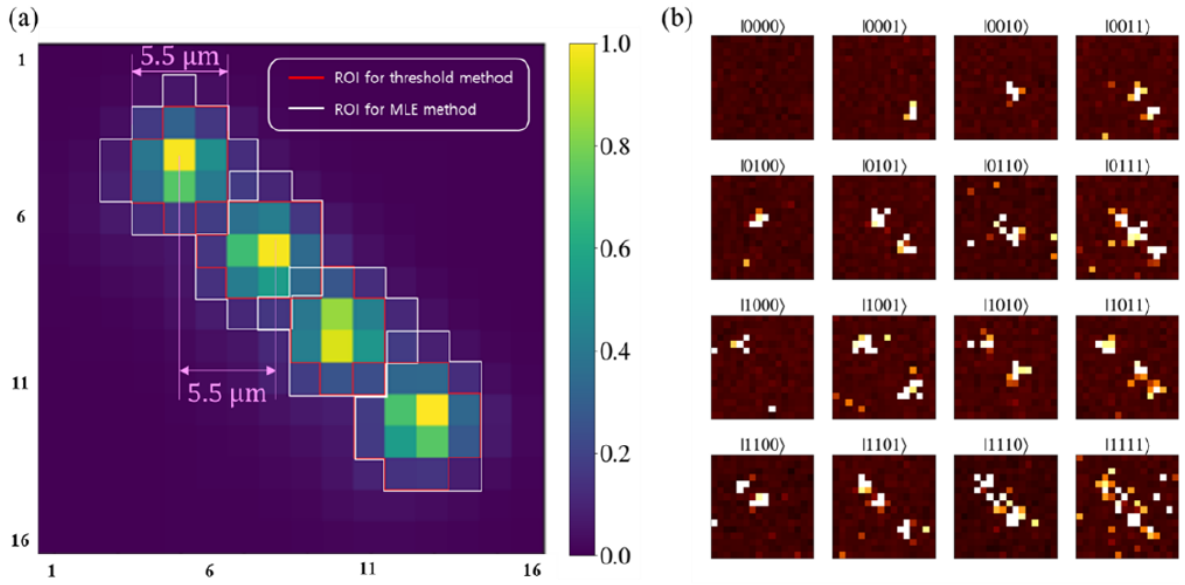


Figure 1.4: Adopted from Ref. [35]. Imaging of Quantum States using Electron Multiplying Charge-Coupled Device (EMCCD) detection. This figure shows two important imaging representations of four-ion configurations prepared in $^{171}Yb^{+}$. Panel (a) shows the time-averaged acquisition, where the original 36×36 pixel frame is binned into a much smaller 16×16 pixel region to enhance clarity. Two regions of interest are segregated, specifically, the thresholding method region enclosed by red borders and the maximum likelihood estimation (MLE) method region represented in white. Panel (b) shows sample single exposures for each of the sixteen possible quantum states in our four-ion system. The single exposures have complex intensity distributions because of the shot-noise-limited conditions. The intensities in a single exposure are inhomogeneous and point to detailed features of the signal that could not be correctly captured by standard thresholding and maximum likelihood methods due to crosstalk (an optical interference between neighbouring ions where fluorescence from one ion affects the measurement of adjacent ions) and other effects of time-varying background signals.

A key experimental distinction between our hardware setup and the one used by the research group in Ref. [35] is that our system operates with lower NA and shorter measurement time. These differences in optical collection parameters affect the quality of fluorescence images we can obtain. The NA is among the most important parameters in optical systems. It determines the light-gathering ability and the resolution of an imaging system. The higher the NA, the better the photon collection efficiency, which can allow increased brightness and details in fluorescence images. With our lower NA configuration, we

face specific challenges in signal-to-noise ratio and photon collection efficiency. Additionally, the lab in Ref. [35] was able to reliably measure four qubits in all 16 states, creating a large dataset of labelled data as shown in Figure 1.4. It is important to note that the ability of the model to work with and classify unlabelled data is believed to be crucial for a scalable solution to the problem of trapped ion state classification using ML, particularly when dealing with varying optical collection parameters. Additionally, the systems classified in Ref. [35] and [62] were limited only to small systems containing 2-4 ions. Those studies did not fully address a challenge in larger ion arrays like the proximity effects. Recent research has also demonstrated high-fidelity state readout [35, 67]. Through spatially resolving the fluorescence from individual ions with a detector, the simultaneous readout of multiple qubits is enabled, resulting in readout fidelity exceeding 99% through the usage of photomultiplier tubes (PMTs) or charge-coupled devices (CCDs) together with adaptive threshold techniques that account for crosstalk between ions [17, 42, 63, 77]. However, this has issues similar to those in the previous papers, as these methods are similarly vulnerable to noise in the process and fail to classify larger batches of ions simultaneously.

To further improve readout performance and scalability, additional work [74] has investigated integrating superconducting nanowire single-photon detectors (SNSPDs) directly into surface electrode ion traps. It was demonstrated that by using such a trap-integrated SNSPD one could achieve state readout of a single $^{9}Ba^{+}$ ion qubit (Beryllium atom trapped in a magnetic field).

Using a 313 nm laser at 2 MHz frequency, the experimentalists were able to knock out one electron and make this atom positively charged and ready to work with, achieving 99% average fidelity, limited primarily by the polarization impurity of the readout laser and off-resonant optical pumping. Using the ion as a calibrated photon source, they characterized the detector's quantum efficiency and its angular dependence, showing agreement with theoretical models. This integrated photonic readout approach frees up optical access and paves the way for scalable parallel qubit readout in large trap arrays without external optics or detectors [52, 74]. The method achieves detection in approximately 145 microseconds, which is comparatively fast for ion-based quantum systems. However, in the context of complex quantum algorithms requiring numerous sequential operations, this detection time could still become a bottleneck when scaling to larger qubit arrays where parallel readout capabilities become essential for maintaining computational efficiency.

In parallel, new machine learning techniques have emerged as a promising approach to optimize readout fidelity and automate the extraction of qubit states from fluorescence images of multi-ion chains. For example, Teoh Yi-Hong at the University of Waterloo developed a FFNN that learns the ions' point spread function (PSF) and classifies their

states from simulated fluorescence data [73]. Interestingly, the author found that the trained network uses learned weights that resemble the PSF and functions similarly to a linear classifier used in adaptive thresholding. To improve scalability, they proposed a two-stage readout process that first determines a ROI for each ion from a calibration image taken during Doppler cooling and then applies the FFNN to the ROI of each ion in the measurement image to predict its state.

1.4 Thesis Goal

This thesis will focus on how to address one problem: reliable classification of projective measurement outcomes from camera images with the potential of integration into trapped-ion quantum computing execution cycles. In the process of this research, we solved the challenge of finding the fluorescence states of the ions of real physical systems using machine learning methods. Working with a system of four ions, which gives a total of 16 different states. The goal is to develop a scalable model that will efficiently classify the states when the number of ions is increased through scalable and reliable machine-learning models. This work considers a dataset of cropped images of these four ions and studies machine learning techniques to efficiently interpret and classify these quantum states.

Traditional methods of labelling all the data or processing images, e.g., using CNNs, may achieve good accuracy for small-sized systems, but they become computationally impractical without specialized tools as the system size grows.. In our case, we decided to work with feed-forward neural networks, considering the fast classification required with minimal computational resources. This is a crucial factor for future integration on FPGA hardware, required to process as many images as possible at speeds compatible with quantum computing operations. Fast and efficient real-time analyses are a necessary ingredient of mid-circuit measurement and validation of final measurements to ensure the accuracy of state determination.

In order to classify such states, we need to develop a process that focuses on an individual ion in our four-ion system. Additionally, we are interested in exploring alternatives to classical FFNN, and in the following chapters, we obtain results by analyzing the same dataset through various machine learning techniques: semi-supervised learning, CNNs, and random forest. We aim to achieve results in designing and testing different architectures of neural networks that can give the best classification accuracy while ensuring the model doesn't systematically favour certain quantum states over others. Fast and accurate state classification is of crucial importance in fault-tolerant quantum computations.

Chapter 2

The Experimental Trapped-ion Platform

Trapped-ion systems have emerged as a promising platform for developing quantum computing and communication. These systems have demonstrated remarkable capability, progressing from early implementations of 2- and 3-qubit algorithms to now handling much larger quantum circuits with dozens of qubits [2, 13, 50, 60]. Experimentalists have conducted complicated operations like the quantum Fourier transform and other quantum simulation operations with scalability and fidelity on a large scale using trapped ions. Fundamentals of quantum communication like teleportation and error correction have been shown on trapped ion platforms and demonstrated that ions can be used for basic studies and for practical applications in quantum technology [48].

In Chapter 3, the trapped-ion platform will play a crucial role in producing high-resolution images of ions. These images are generated during controlled experiments where ions were laser-cooled and confined in a high-vacuum chamber with electromagnetic fields. The emitted fluorescence from the ions is captured using optical systems to achieve clear and accurate imaging of their quantum states. This imaging process provides labelled datasets essential for training machine learning algorithms, enabling the identification and classification of various ion states and transitions. The controlled and consistent nature of the trapped-ion platform ensured the reliability of the data, supporting the development of robust algorithms for feature extraction and classification in quantum systems. For more details about the setup and trapped ion, see Ref. [51].

Ions are atoms that have gained or lost one or more electrons, giving them a net electric charge. This charge allows the ions to be confined and manipulated using electromagnetic fields, effectively isolating them in well-defined regions of space [18, 56]. By employing lasers, the properties and dynamics of these ions can be precisely controlled and measured. The light scattered by the ions provides valuable insights into their behaviour [46]. Notably,

ion traps have facilitated significant technological advancements, including the development of atomic clocks [5, 19] and early prototypes of quantum information processors.

The exploration of trapped ions as a QIP platform began with the pioneering proposal by Cirac and Zoller, who introduced the concept of the first entangling gate for trapped ions [13]. This proposal was soon followed by experimental validation [61]. Mølmer and Sørenson later contributed a more robust entangling gate design that could withstand thermal noise, which has since become the standard approach in the field [53, 71]. The combination of this entangling gate and other state manipulation techniques enables trapped ion systems to meet the DiVincenzo criteria for universal quantum computation [20].

2.1 Properties and Choice of Atoms

The trapped-ion platform serves as the hardware for our dataset generation. Trapped-ion systems are well-known for their long coherence and trapping lifetimes, and usefulness in QIP. It is worth mentioning that scientists frequently use ions with simple electronic structures, such as the singly ionized $^{171}Yb^{+}$, $^{40}Ca^{+}$, and $^{138}Ba^{+}$. These ions contain only one valence electron and behave similarly to hydrogen atoms. Qubit states ($|0\rangle$ and $|1\rangle$) are determined by the electronic states of the ions. There are two basic models for defining qubit states: optical qubits and hyperfine qubits.

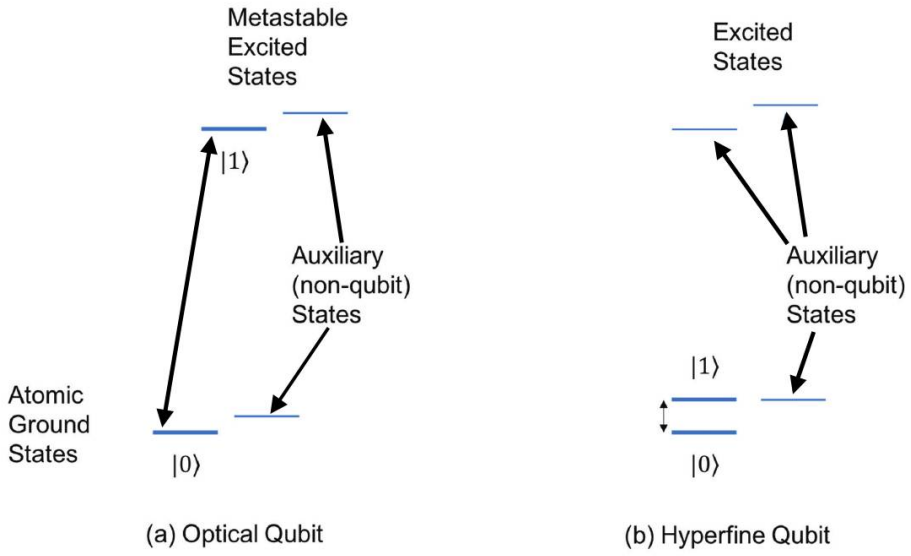


Figure 2.1: Qubits encoded in the internal energy levels of an ion. (a) Optical qubit; (b) Hyperfine qubit.

Atomic ions offer us two distinct ways to realize qubits, depending on which energy levels of the ion are exploited. The first, optical qubits, depend on the large energy difference between a ground state and a metastable excited state. This energy difference lies in the range of optical frequencies 10^{14} to 10^{15} Hz: hence "optical" qubits. In this case, the excited state must be metastable, which is defined as an excited quantum state with an unusually long lifetime due to quantum mechanical selection rules that inhibit rapid decay in order to provide the comparatively long lifespan required for quantum activities.

In hyperfine qubits, much smaller energy separations in the ion's ground state structure are utilized. States that are typically involved in such processes are separated by frequencies in the microwave range from 10^9 to 10^{10} Hz. While this energy separation is much smaller than for optical qubits, the resulting stability is much higher since these states are part of the ground state manifold. In fact, the control of hyperfine qubits frequently requires other excited states that serve primarily as an intermediary level for operations and are not encoded in the qubit itself.

It should be noted that, in both realizations, the energy structure of the ion contains many more states besides the ones chosen for qubit encoding. The auxiliary states (part of the ground state, excited state, and metastable excited state configuration) play essential roles in various tasks of state preparation, manipulation, and readout in the course of

operation, although they do not encode quantum information themselves.

We are also interested in the question: why do we pick one atom over another? For our goals of computing, it is crucial to choose an ion with a hydrogen-like electronic structure — only one valence electron. This simplicity allows for straightforward manipulation and control of the qubits. Ions like Ytterbium and Calcium contain a nuclear spin of one-half and thus offer an energy level diagram like hydrogen, which is also important from a control perspective. This makes them far easier to work with and ensures that the transition frequencies do match up with the many different commonly used optical lasers. One more vital condition is how long-lived electronic states are in the ion. Long-lived states tend to yield better-separated qubit states and so the possibility of better reliability in quantum operations [12, 33]. If an ion is trapped, the quantum computer architecture is set, there is no option to change the ion type. Therefore, choosing the right ion from the start is essential. The major portion of our work was related to the Yb^+ ions from R. Islam's lab [51]. But we also explore alternatives like Ba^+ from Crystal Senko's lab. In the next paragraph, we will expand on what each of these ions is capable of.

$^{133}Ba^+$ is deemed to be one of the most promising ions for quantum computing because it has a variety of desirable properties. What makes $^{133}Ba^+$ such an interesting ion is that it has a nuclear spin of $I = 1/2$, which creates a clean, simple energy-level structure that is easier to control, while the ion also features long-lived metastable states, meaning great flexibility in how we can encode quantum information. In fact, using $^{133}Ba^+$, we could realize at least three different types of qubits: a ground state qubit making use of hyperfine levels, an optical qubit, and even a metastable qubit. That is quite useful flexibility for quantum computing research [16, 28, 47, 69].

One of the more practical advantages is that most of its relevant energy transitions correspond in wavelength to visible light. Though minor, this point is extremely important, namely, it is possible to use commercially available lasers and optics without needing ultraviolet or infrared light during the experiment. There is, however, one major challenge: $^{133}Ba^+$ is synthetic and radioactive, and its half-life is 10.5 years [12, 33]. This does create some problems in the procurement and some issues of safety. However, these problems were mitigated by wise design decisions, such as minimizing the quantity of Barium involved in micrograms and a remote control system. In addition to that, Barium has multiple suitable isotopes for quantum computing, such as $^{137}Ba^+$ and $^{138}Ba^+$. This opens possibilities toward advanced quantum computing architectures that may be implemented with multiple species within the same system.

$^{171}Yb^+$ shares many basic properties with the already discussed Barium ion; it is similarly convenient from the point of view of the hyperfine structure, owing to the nuclear spin

I=1/2. Optical qubits combine a stable ground state and a metastable excited state via an optical-frequency transition, while hyperfine qubits make use of two hyperfine ground states. In $^{171}Yb^{+}$, these hyperfine qubit states are specifically defined as in Fig. 1.3. This ion forms a hyperfine qubit through the states:

$$^{2}S_{1/2},\ F=0,\ m_F=0,\quad ^{2}S_{1/2},\ F=1,\ m_F=0$$

This is illustrated in Fig. 2.2.

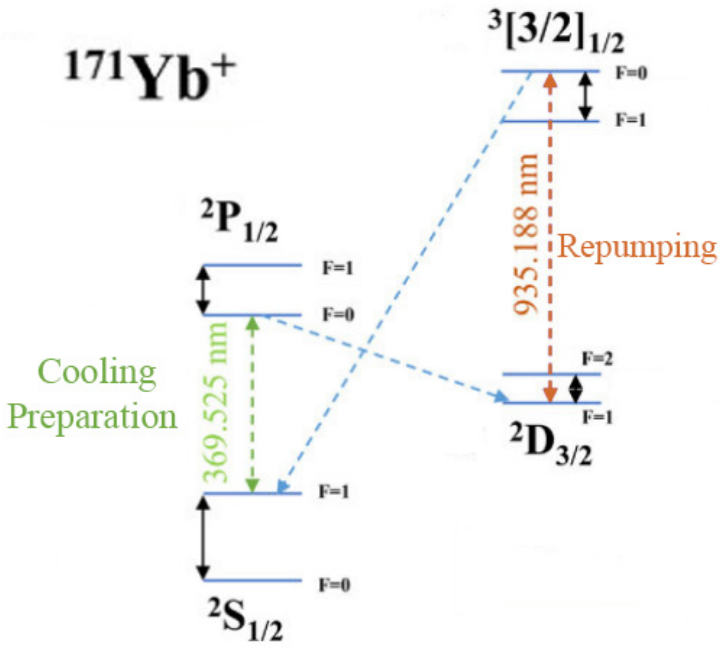


Figure 2.2: Energy levels of $^{171}Yb^{+}$.

The main idea of $^{171}Yb^{+}$'s applicability is based on a cycling transition involving the $P_{\frac{1}{2}}$ state. Its lifetime is as short as 8 ns and stands at the heart of several quantum operations. The main transition from $S_{\frac{1}{2}}$ to $P_{\frac{1}{2}}$ takes place at 369 nm. This transition has high fidelity: the probability of the excited ion ending up in the state $S_{\frac{1}{2}}$ is 99.5%. However, any practical implementation of quantum computing has to consider a small yet relevant complication: the probability that an ion in the $P_{\frac{1}{2}}$ state will decay to the $D_{\frac{3}{2}}$ state is below 0.5%. This leads to a refinement of the $^{171}Yb^{+}$ system: that excited ion must be returned to the computational basis states, and for that purpose, a "repump" laser at 935 nm couples ions from the $D_{\frac{3}{2}}$ state to the $3D\,[3/2]_{\frac{1}{2}}$ level [43].

A second degree of complexity arises from the probability that ions can collide with the background gas to produce the $F_{\frac{7}{2}}$ state. This state is metastable, with a long half-life of 5.4 years, and can remain out of the computation chain for some time. This demands the inclusion of a second repump laser at 760 nm in order to address ions falling into undesired states, specifically, those resulting from the $D_{\frac{5}{2}}$ state in the $^{171}Yb^{+}$ ion system. This 760-nm laser couples ions from the $D_{\frac{5}{2}}$ state into an intermediate state that decays into the ground-state manifold and thereby effectively resets the ion back into the computational subspace [51].

State detection in $^{171}Yb^{+}$ is a particularly beautiful example: the measurement of the qubit state involves illuminating the ion with light tuned to the $S_{\frac{1}{2}}\ |F=1\rangle \rightarrow P_{\frac{1}{2}}$ transition. Detection then relies on a clever mechanism; an ion in the $|1\rangle$ state will fluoresce by cycling on this transition, whereas an ion in the $|0\rangle$ state is dark and cannot be excited by this light.

An essential practical point is that the detection process involves many such cycles because of the small collection efficiency of typical imaging systems; only a few percent of emitted photons are typically collected. This can sometimes involve hundreds of scattering cycles for a particular ion state to be detected with some reliability.

2.2 Readout Details

Understanding the factors that influence the quality and nature of data received from trapped ion systems is essential. Two key parameters in this context are the numerical aperture (NA) of the optical setup, the exposure duration for fluorescence measurements, and the quality of the detector.

The NA plays a crucial role as it affects the resolution and the distance between observed ions on the camera. A higher NA allows for a greater separation between ions, making them more distinguishable. This characteristic is beneficial for developing methods to determine the fluorescence level of each ion and, consequently, identify its state (e.g., $|0\rangle$ or $|1\rangle$) at a local level. How NA influences the pattern of an ion is illustrated by the graph below:

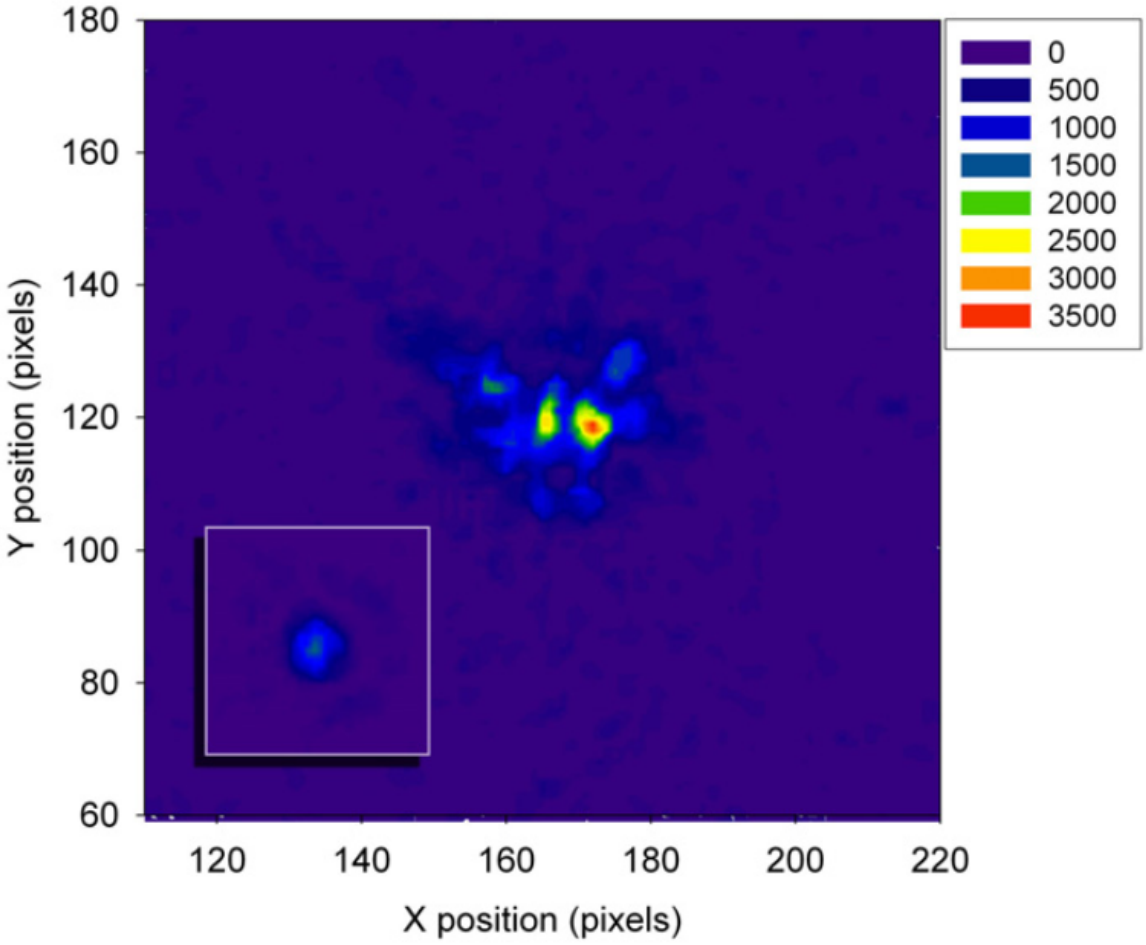


Figure 2.3: Adopted from Ref. [65]. The EMCCD camera used a 0.9 N.A. spherical mirror to capture a single Ba^+ ion picture, which was then compared to a 0.25 N.A. multi-element lens with a total magnification of 70 (see inset). The exposure duration was 0.8 seconds for both images. The pixel size is 10 μm. The picture intensity is expressed in the same (arbitrary) units. Mirrors have a much greater integrated intensity of the ion picture compared to multi-element lenses.

Based on these observations, we can construct a hypothesis that pre-trained NN will be able to generalize, but not enough accurately to be useful for any arbitrary trapped ion setup. Meaning, we can reuse or expand the architecture of the NN, but we need to train it specifically to facilitate each trapped ion device.

Equally important is the exposure time to detect fluorescence. The length of time an ion spends cycling between energy levels and emitting photons directly impacts the clarity of the captured image. Longer exposure times lead to brighter images and im-

prove the visibility of the ion's fluorescence, enhancing the accuracy of state classification. However, during this detection period, off-resonant scattering to dark states represents an important bottleneck in trapped-ion quantum computing. This process occurs when ions transition out of the cycling transition used for fluorescence detection into non-fluorescing "dark" states, decreasing the visibility of ion fluorescence and reducing detection fidelity. Additionally, longer cycling also heats up the ions, requiring more cooling. Extended exposure also increases the camera's susceptibility to noise, creating a trade-off between image brightness and signal-to-noise ratio. Managing this balance between detection time, off-resonant scattering effects, and noise susceptibility is vital for obtaining reliable data that can be processed effectively by machine learning algorithms in the next sections.

We should also take into account different trapped ion systems. Trap designs come with their own unique noise characteristics and environmental factors [76]. The noise levels may vary depending on the system and can be unpredictable, impacting the overall stability and reliability of the measurements. Consequently, the variability in these parameters should be accounted for when estimating the nature and level of noise.

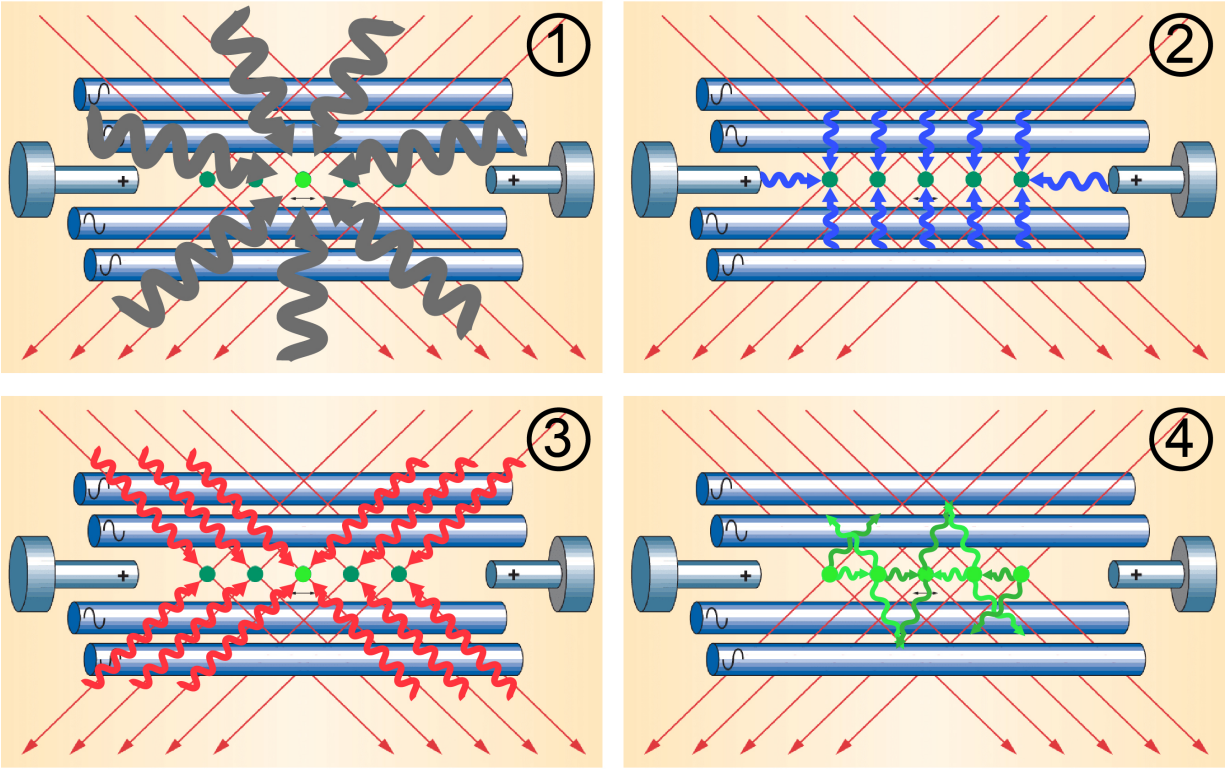


Figure 2.4: Sources of noise impacting ion measurements visualized: (1) Environmental noise, i.e., external vibrations and electromagnetic interference, which disrupt ion positioning and cause fluctuations in fluorescence collection; (2) Electrical noise from RF electrodes used for trapping ions, which can modulate the trapping potential and cause ion motion that blurs fluorescence images; (3) Intensity and frequency fluctuations from laser sources, which directly alter fluorescence rates and detection efficiency; and (4) Cross-talk noise from neighbouring ions, which can lead to spatial overlap of fluorescence signals. All these noise sources directly influence the fluorescence signal captured by our camera, affecting both the spatial and intensity characteristics of the detected photons, thereby impacting the accuracy of quantum state detection.

We should also take into account different trapped ion systems. Trap designs come with their own unique noise characteristics and environmental factors. The noise levels may vary depending on the system and can be unpredictable, impacting the overall stability and reliability of the measurements. Therefore, the variability in these parameters should be accounted for if estimating the nature and level of noise.

These various noise sources contribute to what are broadly classified as state preparation and measurement (SPAM) errors [23], which present significant challenges to scaling trapped-ion quantum computers. During bright state ($|0\rangle$) detection, an ion should ideally

remain in the cycling transition, continuously scattering photons throughout the detection window. However, off-resonant pumping can transfer ions to states outside this cycling transition, temporarily extinguishing fluorescence. Similarly, collisions with background gas particles may cause state changes during measurement. Even in dark state ($|1\rangle$) measurements, where few photons should be scattered, unwanted excitations can occur due to experimental imperfections, background scattering of control lasers, and technical noise in detection equipment.

These detection challenges create an inherent time optimization problem. Longer measurement times improve photon statistics and bright-state identification reliability, but extended measurement intervals simultaneously increase the likelihood of off-resonant pumping and other phenomena that degrade measurement accuracy. As quantum systems grow larger, reliable simultaneous detection of many ions becomes increasingly critical, particularly for quantum error correction protocols where accurate syndrome measurements determine the success of error mitigation strategies. Understanding and addressing these sources of error motivates the development of more robust measurement approaches, including our machine learning-based projective measurement classification method for quantum states that will be explored in the following chapters.

Chapter 3

Machine Learning Theory

To address these hardware challenges, ensuring stable measurement parameters and developing strategies to mitigate noise are essential steps in refining the data extraction process. These considerations are critical for preparing high-quality data that will be analyzed by machine learning algorithms to classify ion states, whether locally (as $|0\rangle$ or $|1\rangle$) or in general ($|1010\rangle$, $|1100\rangle$, $|1001\rangle$, etc.), for four-ion state systems.

In a broad sense, an artificial neural network (NN) is a collection of interconnected nodes to detect patterns and interpret information. Each of these artificial neurons, which is connected with many other neurons in a layered fashion, accepts an input signal or input data, interprets the information, and then sends its output signal to the next layer in a cascading mode. In some cases, these neurons can pass information in a looped or recurrent manner as well.

NNs are constantly involved in optimizing interconnectivity between individual artificial neurons across different layers. During the training process, the network is presented with input data and compared with what is to be expected. The difference between what is true output and what is predicted causes systematic modifications to the network parameters—most importantly, the "weights" controlling nodes' interconnectivity. This careful optimization allows the network to improve increasingly to identify relevant features in information whether it is for image classification, language processing, or prediction models. Learned weights capture patterns and relations discovered in the training set. In the following section, we will introduce different NN architectures and ways to learn patterns.

3.1 Feedforward Neural Network

Feedforward neural networks are a general class of artificial neural networks where information moves only in one direction from input through some intermediary hidden layers to output, without cycles or back coupling. An example of this network is shown below in Figure 3.1.

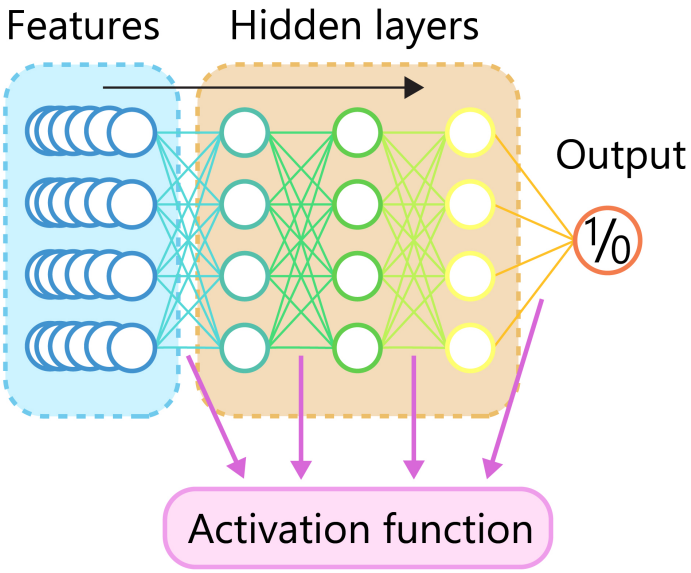


Figure 3.1: The FFNN architecture can be represented by a diagram. It accepts feature vectors as inputs that convert multi-dimensional inputs such as images to one-dimensional arrays. Inputs are processed through multiple layers of hidden units with activation functions and then produce a single binary output as a prediction.

In developing our model we utilize a specific architecture, which was developed for dealing with the particular challenges given by our dataset's noisy images and interaction complexity among the neighbourhood ions. Our approach prioritizes effective noise management by ion cropping and accurate classification of individual ion states despite interference effects. This targeted separation of ions within our model framework is critical for ensuring both scalability to larger ion arrays and precision in real experimental quantum computing applications.

Each layer of FFNNs consists of thousands of neurons, where each one calculates the

weighted sum of the entering inputs, transmitting the result via a nonlinear activation function. We can express this process mathematically as follows:

$$y_i = \sigma\left(\sum_{j=1}^{n} w_{ij}x_j + b_i\right) \tag{3.1}$$

Here:

- y_i - is the output of the i-th neuron,
- x_j - represents the j-th input,
- w_{ij} - denoted as the weight connecting the j-th input to the i-th neuron,
- b_i - is the bias term,
- $\sigma(z)$ - is the sigmoid activation function.

The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

The sigmoid function introduces nonlinearity into the model, allowing it to learn complex patterns in the data. By compressing the output to the range $[0, 1]$, the sigmoid function is particularly suitable for binary classification tasks or for outputs requiring a probabilistic interpretation.

This ability to approximate complex relations between input features and output comes from the non-linearity introduced by this activation function. Complex nonlinear patterns in data cannot be modelled by a neural network without non-linearity since it would just be a sequence of linear transformations. Adding an activation function allows the network to approximate large classes of functions, turning it into a potent tool for modelling real-world phenomena.

This is formalized in the Universal Approximation Theorem [30], which states a feed-forward neural network with a single hidden layer containing a finite number of neurons will be able to approximate any continuous function on a compact subset of R_n, a closed and bounded region of n-dimensional space, such as a finite interval, disc or cube [14, 31]. Given appropriate activation functions and weights, it is one of the main reasons why neural networks could be applied to all sorts of tasks including regression, classification, and pattern recognition. However, in practice, deeper networks with multiple layers are frequently

preferred because of their enhanced generalization and efficiency owing to hierarchical data representations learned by the model.

The explicit choice of the activation function is critical for achieving this property. The usual nonlinear activation functions like sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) all have certain properties that yield different abilities of the network to model different relations. While they introduce the required nonlinearities, they do so in a manner that keeps them computationally efficient in both the forward and backward propagations [15].

FFNNs are usually trained in a supervised way. The aim is to learn the mapping of inputs to outputs by minimizing some loss functions, such as Mean Squared Error (MSE) for regression and cross-entropy for classification on labelled training data. FFNNs do this by a process called backpropagation [59], where the gradient of the loss concerning each weight in the network is computed and used to update the weights so that the network constantly makes improved predictions. Weight and bias are updated in the network, including two of the most common ways, namely, stochastic gradient descent (SGD), and its variants such as Adam [38], which are two of the most significant ways through which the networks usually get trained.

The flexibility of a feedforward architecture has some particular downsides; a major one is a regularization against overfitting when a model learns too much from the training data and fails on new and unseen data. This is a general risk that can be reduced, for example, by regularization methods such as dropout, weight decay, and reduction in model complexity. In this thesis, we design the FFNN in such a way that it manages the challenges imposed by both noisy data and interactions among nearby ions. Afterwards, an advanced architecture in which every piece of data had been processed alone before feeding into a shared layer or final classification is designed. Thus, the network can abstract and make certain streams of information distinguishable before gathering common features from all input sources. This shared layer represents the crucial part of the network where the relationships between the diverse streams of data are compiled together before the network proceeds for further learning. During each learning phase, model complexity is restricted via the limitation in depth and number of dense layers. This gradual regularization approach will hopefully keep overfitting at a minimum, a problem that typically emerges when training NNs on datasets with limited samples relative to the model's parameter count (generally fewer than 10x examples than parameters). In an FFNN, all the layers are fully connected dense layers, meaning every neuron in one layer receives input from all neurons in the previous layer and sends its output to all neurons in the subsequent layer — creating the comprehensive connectivity pattern characteristic of these networks. Such a structure allows the network

to remember intricate dependencies of the features in the input data. However, adding too many dense layers, especially for relatively small data sets, quickly leads to overfitting. Because of this, we reduce the number of layers to have enough capacity in the network to avoid overfitting and model important patterns.

Eventually, the sigmoid function used in the output layer provides a probabilistic prediction by its nature, the output lying between 0 and 1. It is, therefore, quite useful for applications involving binary classification problems since the network can decode the result as the probability score assigned to each class. Such probabilistic outputs have specific value when dealing with ambiguous or noisy data, as, besides providing the class prediction, they provide an associated confidence level, thus enabling richer classification decisions.

3.2 Convolutional Neural Network

A convolutional neural network (CNN) is a type of deep learning model that is specially designed for tasks requiring images as input, and suitable for tasks like image classification, object identification, and segmentation. [55]. These types of networks employ convolution, a type of linear mathematical procedure. A CNN's hidden layers contain convolutional layers [25]. Their core structure is similar to the way the human visual cortex processes images, eventually extracting increasingly complex details from input data [10, 32]. Convolutional layers apply a sequence of filters to an input image to recognize features like edges, textures, and forms. These features are then integrated into subsequent layers to form hierarchical representations, where deeper layers detect increasingly complex patterns by combining simpler features from earlier layers—progressing from basic edges to textures to object parts to complete structures. Figure 3.2 illustrates this hierarchical feature extraction process.

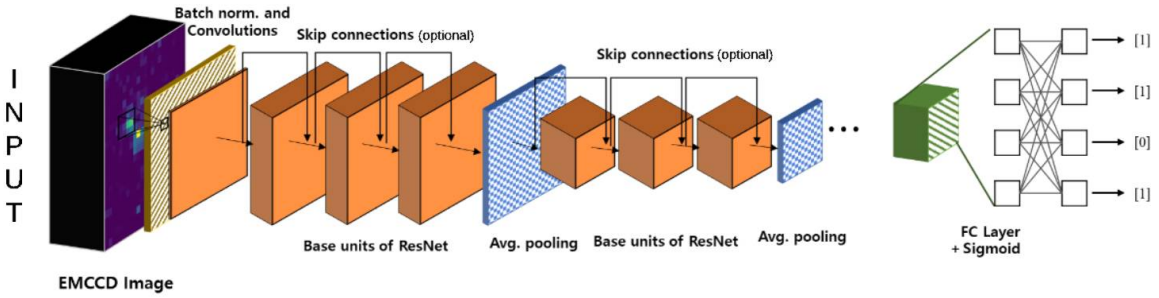


Figure 3.2: Adapted from Ref. [35]. CNN structure processing a single image. We can see how the image is used as inputs going through convolution layers of feature extraction, and activation functions in between with skip connections to preserve specific input information and classification stage.

The most highlighted advantage of CNNs is their ability to leverage spatial content. CNNs can theoretically identify minor details such as edges or corners and patch them up to form more complicated shapes such as textures or objects. This hierarchical feature extraction method is highly useful for large images containing numerous disparate regions containing crucial information. Furthermore, the use of shared weights in convolutional layers (in which the same filter is applied to different sections of the image) allows CNNs to learn spatially invariant patterns, making them both computationally efficient and capable of generalizing across different image positions. This is especially important when we work with ROIs, which could be shifted in the dataset or scaled.

CNNs implement several key architectural elements beyond convolutions. They use activation functions such as ReLU to introduce non-linearity, allowing the network to model complex relationships between features. Additionally, these networks incorporate pooling layers, such as max-pooling, which systematically reduce the spatial dimensions of feature maps while preserving the most important features. This dimensionality reduction is illustrated in Figure 3.2, where you can see how pooling operations maintain critical information while significantly decreasing computational requirements. Together, these components create an efficient architecture for processing our high-dimensional fluorescence data.

Because of the hierarchical nature, the shared weight structure has made these CNNs indispensable for modern image processing. However, they heavily rely on the complexity of the input data and require a vast amount of data for better performance. In the case that the dimensions of the input data are lower or have fewer samples, simpler architectures or traditional machine learning methods perform better since these CNNs either overfit or

fail to extract relevant features.

3.3 Variational Autoencoders

From the original publication Variational Autoencoder (VAE) [39] had a two-stage structure consisting of encoding and a decoding network, combining elements of NNs adopting concepts from Bayesian inference and variational methods. Unlike legacy autoencoders [49], where inputs are connected directly into a fixed output space, VAEs encode inputs as probability distributions in the latent space.

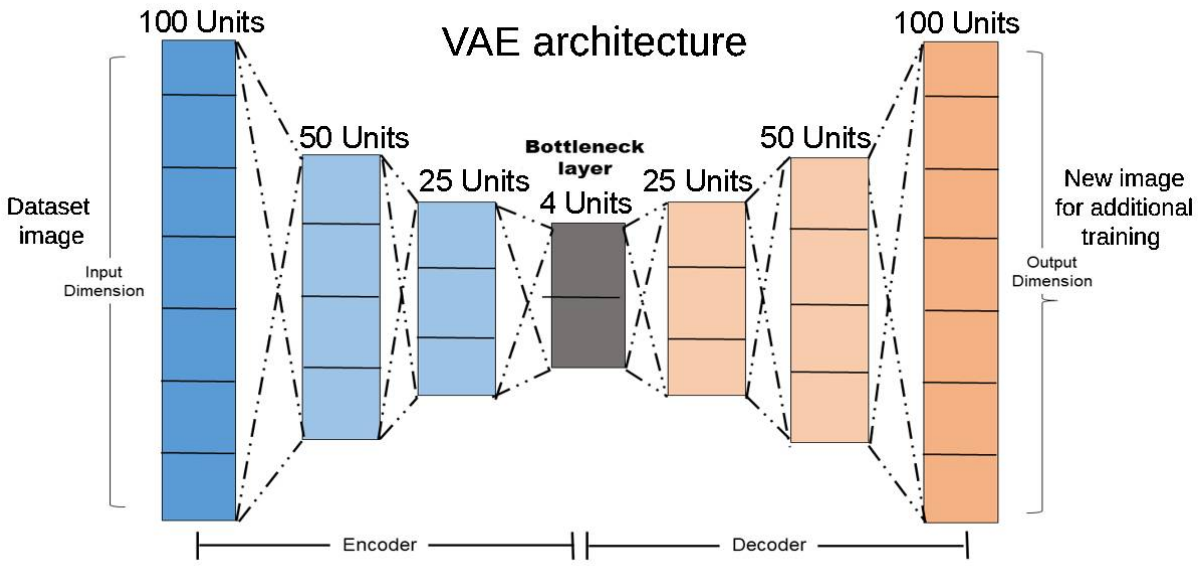


Figure 3.3: Basic architecture of an autoencoder-decoder. It has an input layer of size 100 and encodes an image into a latent space of size 4 through two dense layers with sizes 50 and 25, respectively. The decoder then reconstructs the image by increasing the dimensionality.

3.3.1 Theoretical Framework

The encoder maps an input $\mathbf{x}$ to parameters of a probability distribution (typically Gaussian) in latent space, rather than to a fixed point. Formally, for a latent variable $\mathbf{z}$, the encoder produces $q_\phi(\mathbf{z}|\mathbf{x})$, parameterized by ϕ, which approximates the true posterior distribution $p(\mathbf{z}|\mathbf{x})$ [39].

For a Gaussian distribution, the encoder outputs two vectors: a mean vector $\boldsymbol{\mu}$ and a variance vector $\boldsymbol{\sigma}^2$. The latent representation is then sampled:

$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I}) \tag{3.3}$$

Then, the decoder, which is parameterized with θ, samples from this latent representation. We return to the original data space to obtain $p_\theta(\mathbf{x}|\mathbf{z})$.

3.3.2 Optimization Objective

VAEs are optimized through increasing evidence lower bound (ELBO) as much as possible, the minimum value for the log-likelihood for data [39, 57]:

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{3.4}$$

This objective function balances two components:

- It's a reconstruction stage that makes the decoder function accurately to reconstruct the input.
- A regularization term (the Kullback-Leibler divergence), which will push the distribution $q_\phi(\mathbf{z}|\mathbf{x})$ toward an original distribution $p(\mathbf{z})$, typically a standard normal distribution [70].

3.4 Semi-Supervised Learning

Following the previous experience, we move on to designing our semi-supervised learning (SSL) strategy [11]. Semi-supervised learning is a special class of classification that makes use of both labelled and unlabelled data for training. Traditional classifiers rely mainly on labelled data, which is often quite hard and time-consuming to get, as the creation of such data requires significant human work. In contrast, unlabelled data is relatively inexpensive and easier to collect. Intending to deal with this problem, semi-supervised learning incorporates a small amount of labelled data together with a large pool of unlabelled data to create a classifier. It minimizes the heavy dependence on human efforts while yielding better results with great value in theoretical research and real-world scenarios [80]. There are numerous ways to implement SSL on a given dataset, most of the methods include a combination of existing tools in ML, while others develop unique approaches to a specific

problem. For example, we can distinguish at least four categories in the deep SSL domain: deep generative methods, consistency regularization methods, graph-based methods, and pseudo-labelling methods [78].

In our trapped-ion quantum state classification problem, the availability of only two labelled states among 16 possible states creates a unique challenge that SSL is particularly well-suited to address. While established SSL frameworks may seem overly complex for our specific application, examining these approaches is crucial for understanding the theoretical foundations that informed our methodology. The following sections present two sophisticated SSL techniques which served as direct inspiration for our distribution-based approach. We adapted key concepts from these frameworks, particularly the idea of distribution alignment, while simplifying their implementation to suit the constraints of quantum measurement data. By presenting these frameworks and then our adapted method, we demonstrate a progression from established machine learning techniques to quantum-specific applications, revealing how theoretical insights can be translated into practical solutions for quantum computing challenges.

We will introduce only two SSL techniques, which are quite similar to what we developed starting with the consistency regularization approach and its specific realization, Mean Teacher.

3.4.1 The Mean Teacher

The Mean Teacher [72] is an SSL method proposed, which elevates the improvement in performance from both the labelled and unlabelled data. It extends some of the ideas from Temporal Ensembling [41], but instead of depending on the predictions across time, it guides the training using the exponential moving average (EMA), giving more importance to recent changes. EMA is a smoothing technique where the current value is a weighted sum of past values. This indicates that, over training time, the teacher model that is being provided is a more stable and smoother version of the student model. EMA is calculated using the following formula:

$$\mathrm{EMA}_t = \alpha \cdot x_t + (1 - \alpha) \cdot \mathrm{EMA}_{t-1}$$

Here:

- EMA_t: The exponential moving average at time t.
- x_t: The current data point or value at time t.

- EMA_{t-1}: The exponential moving average at the previous time step.
- α: The smoothing factor, where $0 < \alpha \leq 1$. It determines the weight given to the current data point versus past averages.

This framework has two models working together: a standard NN with dropout applied during training. This model learns directly from the labelled input data, and the weights of the teacher are computed as an EMA of the student model's weights over the training process. Another important concept this model uses is image augmentation; this framework creates two randomly augmented versions of all labelled and unlabelled images, while the "student" predicts the first image, and the "teacher" predicts the second one. A resulting difference between predictions is measured by square difference and treated as a consistency loss. The workflow is shown in Figure 3.4.

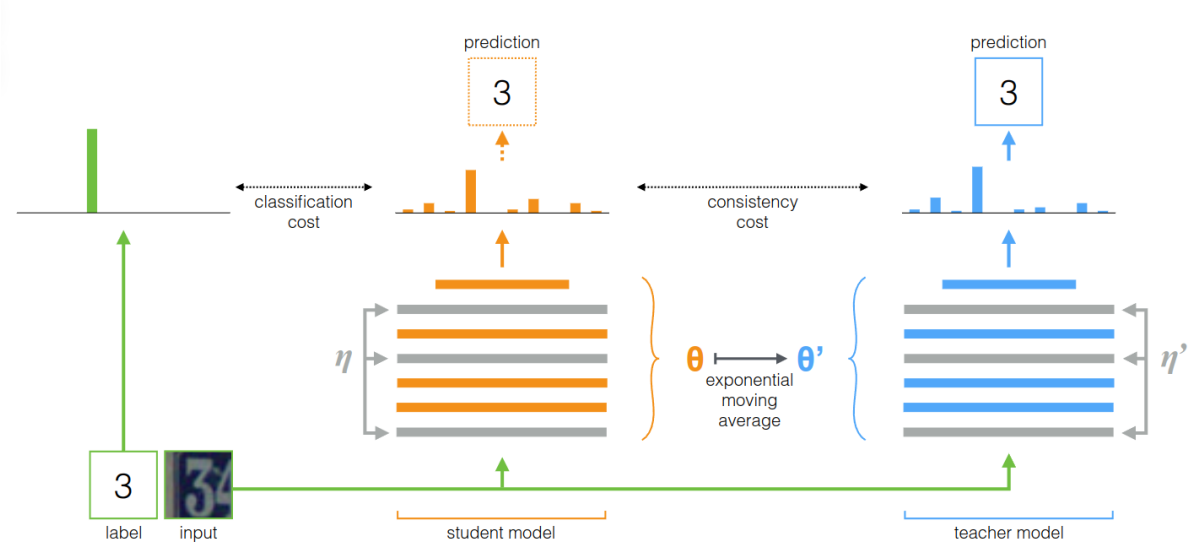


Figure 3.4: Adopted from Ref. [72]. The Mean Teacher method has one data point as an input. The image is fed through the student and teacher models, adding noise (η, η') in their computation. In this process, the softmax output of the student model is compared to the one-hot label to compute the classification loss and to the teacher model output in order to compute the consistency loss. The last step is a weighted sum of these two loss terms with the application of the resulting weight.

3.4.2 ReMixMatch

Another interesting and similar model to our SSL framework is ReMixMatch [3], which is an improved distributed matching approach to SSL called MixMatch [4], see Fig. 3.5. The

MixMatch algorithm first generates "guessed labels" for each unlabelled sample, and then, utilizing both the original labelled data and the guessed labels, uses a fully supervised approach.

From the math point of view, we can represent a combination of labelled data $X = \{(x_b, p_b) : b \in (1, \ldots, B)\}$ and unlabelled data, $U = \{u_b : b \in (1, \ldots, B)\}$ and use it to improve training. For labelled examples, x_b are inputs with one-hot encoded labelsp_b, and for unlabelled examples u_b, the model predicts class distribution $p_{\text{model}}(y|x;\theta)$.

First, K weakly augmented versions $\hat{u}_{b,k}$ of each unlabelled example are generated. A "guessed label" $\bar{q}_b$ is computed as the average prediction across these augmentations:

$$\bar{q}_b = \frac{1}{K} \sum_{k=1}^{K} p_{\text{model}}(y|\hat{u}_{b,k};\theta)$$

and sharpened by adjusting the temperature T (raising probabilities to $1/T$ and renormalizing). A temperature technique where each probability value in the distribution is raised to the power of 1/T and then renormalized, effectively making high probabilities higher and low probabilities lower when $T < 1$. After that labelled and unlabelled data are combined, and pairs $(x_1, p_1), (x_2, p_2)$ are mixed using MixUp:

$$x' = \lambda x_1 + (1 - \lambda)x_2, \quad p' = \lambda p_1 + (1 - \lambda)p_2$$

Here $\lambda \sim \text{Beta}(\alpha, \alpha)$. The final training uses standard cross-entropy loss (a measure of the difference between predicted probability distributions and the true distribution) for labelled data and mean square error for unlabelled data (weighted by λ_U).

Key hyperparameters include K (number of augmentations), T (sharpening temperature), α (MixUp Beta parameter), and λ_U (unlabelled loss weight).

Now, we will discuss the relevant details of the ReMixMatch model.

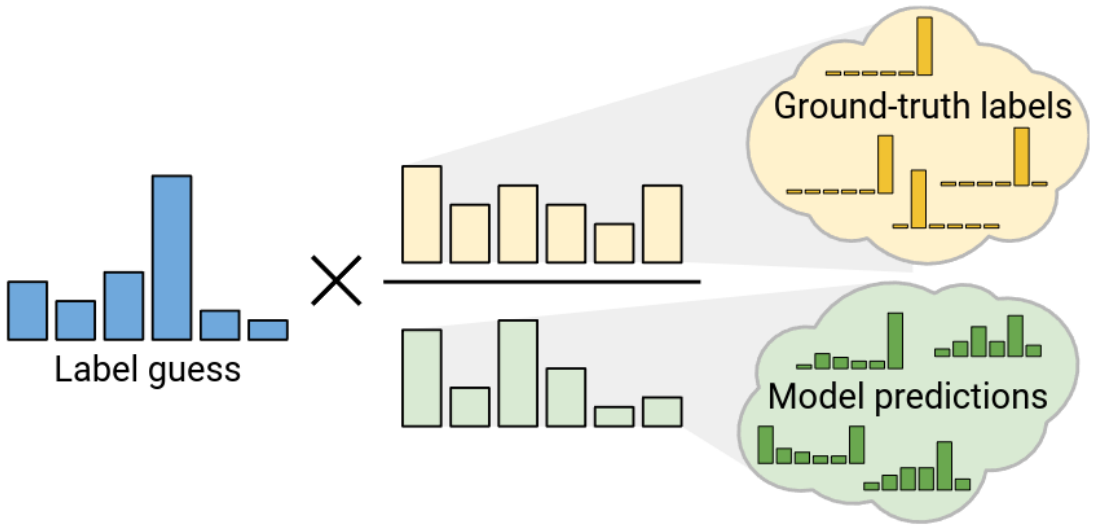


Figure 3.5: Adopted from Ref. [3]. A showcase of distribution alignment, we can see how the theoretical model "guesses" labels for unlabelled data points, and then we adjust this distribution according to ground-truth labels divided by average model predictions.

This new paper [3] introduces two major innovations: distribution alignment and augmentation anchoring. The first contribution is Distribution Alignment, which imposes consistency between the model's predictions on unlabelled data and the class distribution of the labelled dataset. The algorithm will basically ensure that the predictions on unlabelled data match the statistical distribution of the labelled data. While a similar technique called sharpening (a sort of entropy minimization) is also used in the paper, the ReMixMatch improves it and by directly aligning the predicted distribution to the distribution of the labelled data. This has a very simple intuition behind it: in an ideal case, every class should be predicted with the same frequency, or at least with a frequency reflecting the one in the labelled dataset.

To do this, the algorithm maintains a running average of its predictions on unlabelled data. This running average provides an estimate of the class distribution across all of the predictions on unlabelled data and is represented by the variable $\hat{p}$. The model then adjusts its predictions so that they more closely align with the true class distribution q of the labelled dataset. Formally, the alignment is:

$$\tilde{p}(x) = \frac{p(x)}{\hat{p}} \cdot q$$

Here:

- $p(x)$ - represents the initial predictions for the unlabelled data,
- $\hat{p}$ - is the running average of predictions over the unlabelled data,
- q - is the class distribution from the labelled data.

This technique reduces biases in predictions, improves generalization, and avoids introducing additional loss terms or sensitive hyperparameters (the configuration settings such as learning rate, batch size, and model architecture choices that must be set before training and are not learned from the data).

The second innovation, Augmentation Anchoring, complements Distribution Alignment. The flow is as follows: for each unlabelled image, the model generates two augmented versions. The student model predicts the class distribution for one augmentation, while the teacher model — using an EMA of the student's weights—predicts the distribution for the second augmentation (similar to what we have seen in the MixMatch paper [4]).

Overall, the model is trained to minimize the squared difference between these two predictions, enforcing consistency between augmented versions. This consistency loss is combined with the standard cross-entropy loss for labelled data, weighted by a schedule $w(t)$, which increases the contribution of consistency loss over training time.

As a next step of the research, an autoencoder was used in the classification to support our semi-supervised methods. Through the application of an autoencoder with an internal state size of four, we were able to train the model without labels initially in order to produce a series of model states matching the size of our target data before feeding these states into both simple and complex classifiers trained on the labelled data to unshuffle the states. This was done as even if the autoencoder were to properly separate the data into 16 differing states in its internal nodes, it would not necessarily match our needs with it potentially labelling the $|1011\rangle$ class with all high values in the internal nodes or any other combination. Additionally, when structuring our autoencoder, three methods were used, consisting of a simple one-layer autoencoder, a nine-layer deep NN autoencoder using RELU activations, and a CNN autoencoder, using 2×2 pooling and upsampling without padding. Each of these methods additionally was scored three times, firstly through direct interpretation of the states, secondly through being fed into an SVM classifier, and finally being used in a deep neural net. This resulted in nine comparative results cases to use in order to identify the best approach for semi-supervised learning before moving forward into testing a feed-forward neural network.

In this chapter, we explored foundational concepts of machine learning theory. We covered the mechanics of feed-forward neural networks, understanding their structure and how they operate. Additionally, we discussed the basic ideas and tools required to construct and train neural networks effectively. Furthermore, we introduced several key models, such as CNNs, and explored SSL approaches. These include various frameworks relevant to our project, like the Mean Teacher framework and RemixMatch. Each of these models and strategies highlights practical ways to handle data in scenarios similar to ours, providing useful insights for handling training with limited labelled data.

Chapter 4

Data and Methodology

This chapter aims to walk through the process of data acquisition: how our data was gathered, how it was analyzed, and how it was prepared to feed it to machine learning models. Figure 4.1 offers a visual example of raw ion emission imaging for the $|1111\rangle$ state, averaged over 10,000 images, which illustrates how the ions appear prior to any preprocessing.

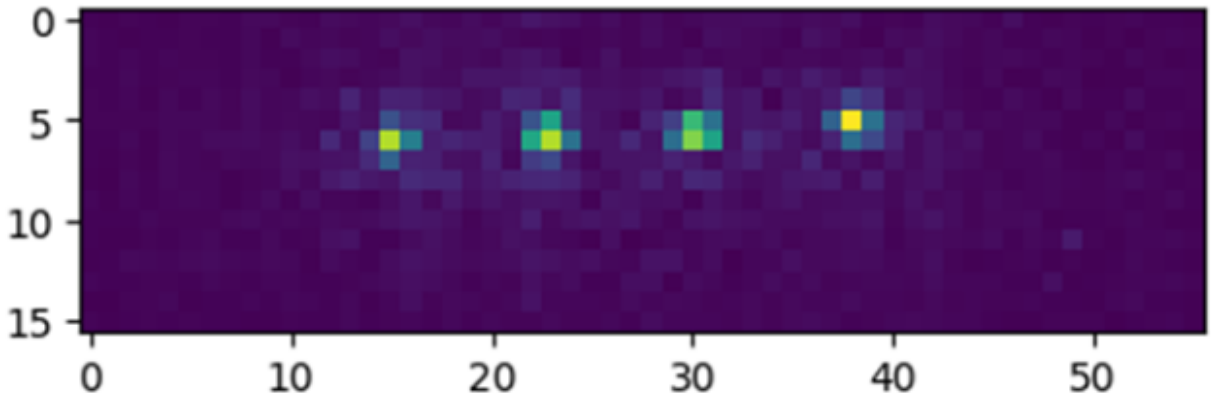


Figure 4.1: Averaged raw ion emission imaging for $|1111\rangle$ state.

We also take a look at how the basic ML architectures introduced in the previous chapter are modified in such a way as to take this data in and utilize it effectively. This chapter will go into a deeper discussion about how different architectures performed on a given dataset, provide results, and provide arguments as to why some methods failed in comparison with others.

Furthermore, we discuss the results of training and outline the progress, from the very

first steps of our FFNN development to more advanced ones that have been used when composing the results for Chapter 5.

4.1 Data Collection

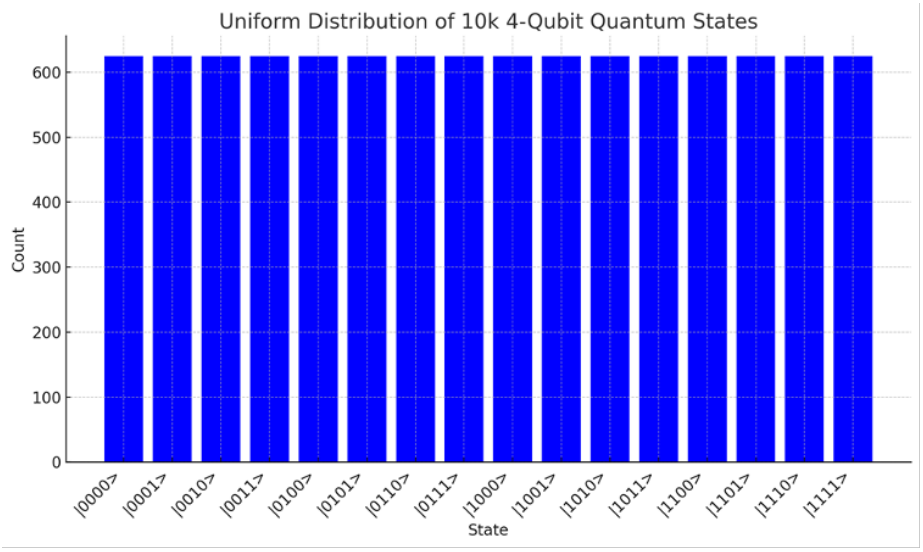


Figure 4.2: A bar chart illustrating the ideal uniform classification of 10,000 half-pi images of four-qubit projective measurements, resulting in 16 basis states from $|0000\rangle$ to $|1111\rangle$. The bar height indicates the frequency with which each state was classified by the ML model.

The experimental data acquisition was performed by researchers in R. Islam's laboratory at the University of Waterloo. Throughout our analysis, we assume the experimental apparatus maintained stable trapping conditions and consistent optical alignment during data collection. The raw image data was provided to us in HDF5 format without any pre-processing applied. An initial dataset for training contained 30,000 images overall; the resolution of the images was 56×16 pixels (see Fig. 4.1). The distribution was as follows: 10,000 images of bright state ($|1111\rangle$), 10,000 images of dark state ($|0000\rangle$) and 10,000 images of half-pi or equal-superposition states. Mixed states are ones wherein each state of the system has an equal probability of occurring among 2^n instances. These are uniform counts over all these measurement outcome variations. In our case, taking $n = 4$, there are $2^4 = 16$ possible measurement outcomes. This distribution may be obtained by starting with an arbitrary ion state in superposition and applying a half-pi pulse (a $\pi/2$ rotation) on the Bloch sphere. This would create an equal superposition of the basis states

$|0\rangle$ and $|1\rangle$. As a result, a distribution was acquired, repeating the measurement of this system 10,000 times. We should expect a uniform distribution of states to learn from. Figure 4.2 shows what we expect our dataset to be if it is perfectly random and doesn't have errors. The high fidelity of the $\pi/2$ pulse (with coherence times of seconds) ensures that state preparation errors are minimal, making deviations from uniformity primarily attributable to measurement noise rather than preparation imperfections. We use this uniform distribution as a target because it's the only stable anchor for our ML model to fit to, allowing us to separate state preparation from measurement effects. In practice, even with perfect state preparation, shot-to-shot variations would cause a 10,000-sample measurement to show slight deviations from perfect uniformity.

It is important to note that the half-pi ($\pi/2$) pulse used to create the superposition states in our dataset has inherent limitations that affect measurement statistics. In the experimental setup used by Islam's team, the microwave-driven transitions have coherence times (T_2) on the order of seconds, significantly longer than the π-time of approximately 20 μs used for state manipulation. This allows for high-fidelity state preparation with minimal phase errors. The optical pumping process used for state initialization achieves infidelities of 10^{-5} to 10^{-6} with typical pumping times of 20-30 μs, substantially longer than the 1-2 μs required for 1/e population transfer.

Our machine learning approach implicitly assumes that state preparation errors are significantly smaller than measurement errors. This is a reasonable assumption given that the detection process introduces approximately 5% standard deviation in photon counts due to shot noise, while state preparation fidelities exceed 99.999%. This asymmetry establishes an upper bound on our method's performance—we cannot exceed the fidelity of the state preparation itself. However, since preparation errors are substantially smaller than measurement errors in this experimental platform, our approach remains valid for practical quantum computing applications where the goal is to minimize the impact of measurement errors rather than preparation errors.

A main challenge with this setup is that a large portion of our state distribution is unlabelled. Only 2 out of 16 qubit states we want to predict correctly are labelled and this creates a complication for learning. In other words, with the limited amount of training data we have, complications arise in reaching the necessary accuracy to classify all possible 16 classes in the dataset.

4.1.1 Initial Approach and Challenges

Our initial technique involved taking the trapped-ion system's original images and uploading them into a feed-forward neural network model (see Section 3.1). We attempted to classify all 16 states at the same time in order to evaluate the method's overall accuracy and utility. However, despite multiple attempts with different hyperparameters and deeper designs, the results remained unsatisfactory. The classification accuracy stayed around 50%, with little change regardless of the number of layers or fine-tuning approaches used. The model struggles with the complexity of categorizing all states from the complete image, as evidenced by its low accuracy, shown in Figure 4.3.

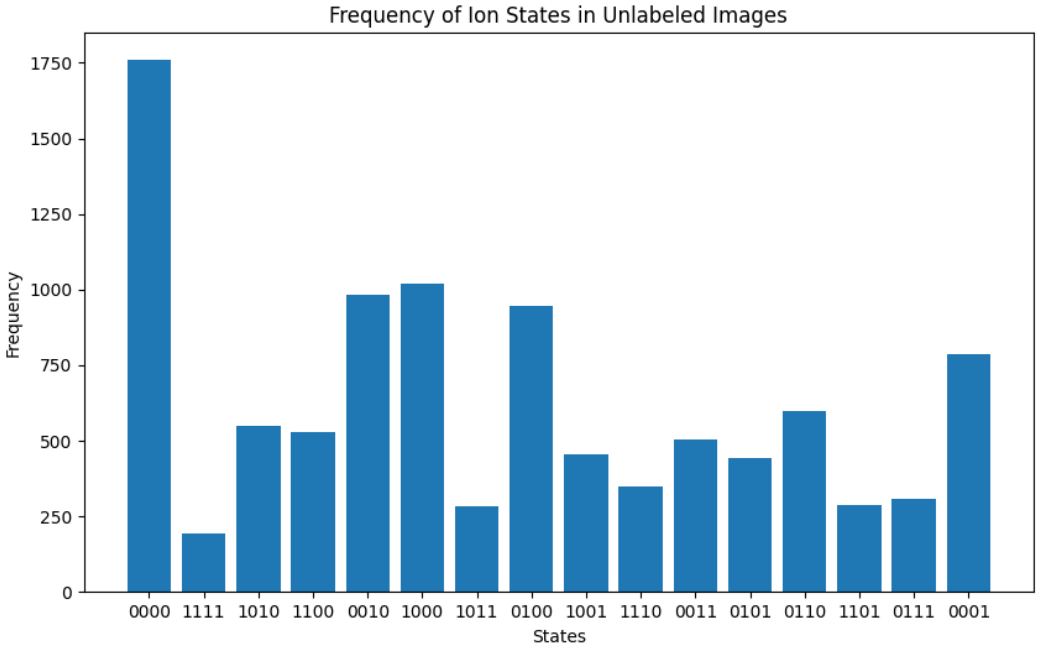


Figure 4.3: The result of a classification of mixed state by an early-stage feedforward ML model. The frequency distribution is highly inaccurate and biased towards $|0000\rangle$.

As a result, we research a more advanced approach: cropping the region of interest around each ion to create a new, separate dataset. This allows us to focus on classifying ions based on their individual image data rather than trying to interpret the full system at once.

4.1.2 Improved Approach: Classifying Individual Ions

In this modified strategy, each ion is treated as a distinct entity, and the FFNN model is trained to classify ion states based on their position in the sequence. After determining the classification result for each individual ion, we pool the results to determine one of the 16 possible ion states. This precise method proves to be substantially more effective, resulting in much higher accuracy than the initial approach.

4.1.3 Cropping of Ion Images

The choice to crop the ion images in the dataset was based on two factors: the existence of considerable noise and irrelevant dark areas around the ions. The raw photos showed a high level of noise as well as large areas of darkness around the ions, which provided no usable information for the training process (see Figure 4.3).

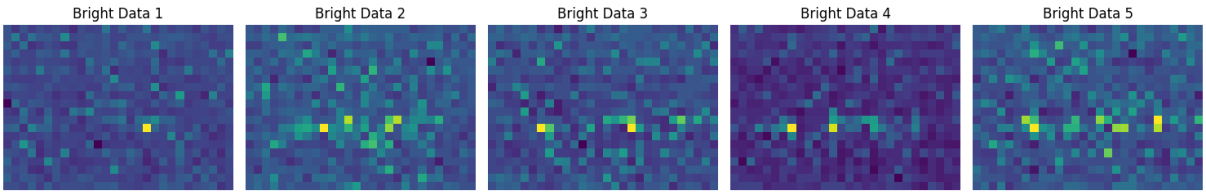


Figure 4.4: Raw images from the dataset. All of these images are labelled as bright, meaning we should observe four distinct bright spots from the ions' fluorescence.

These regions could potentially mislead the machine learning model since they include noise that could be misinterpreted as ion signals. Our ROIs are the bright, fluorescing ions that appear locally in the shots. However, the fluorescence is just slightly brighter than the surrounding noise, and the whole image contains 25 pixels of brightness for each ion, the representation of how it looks is shown in Figure 4.5:

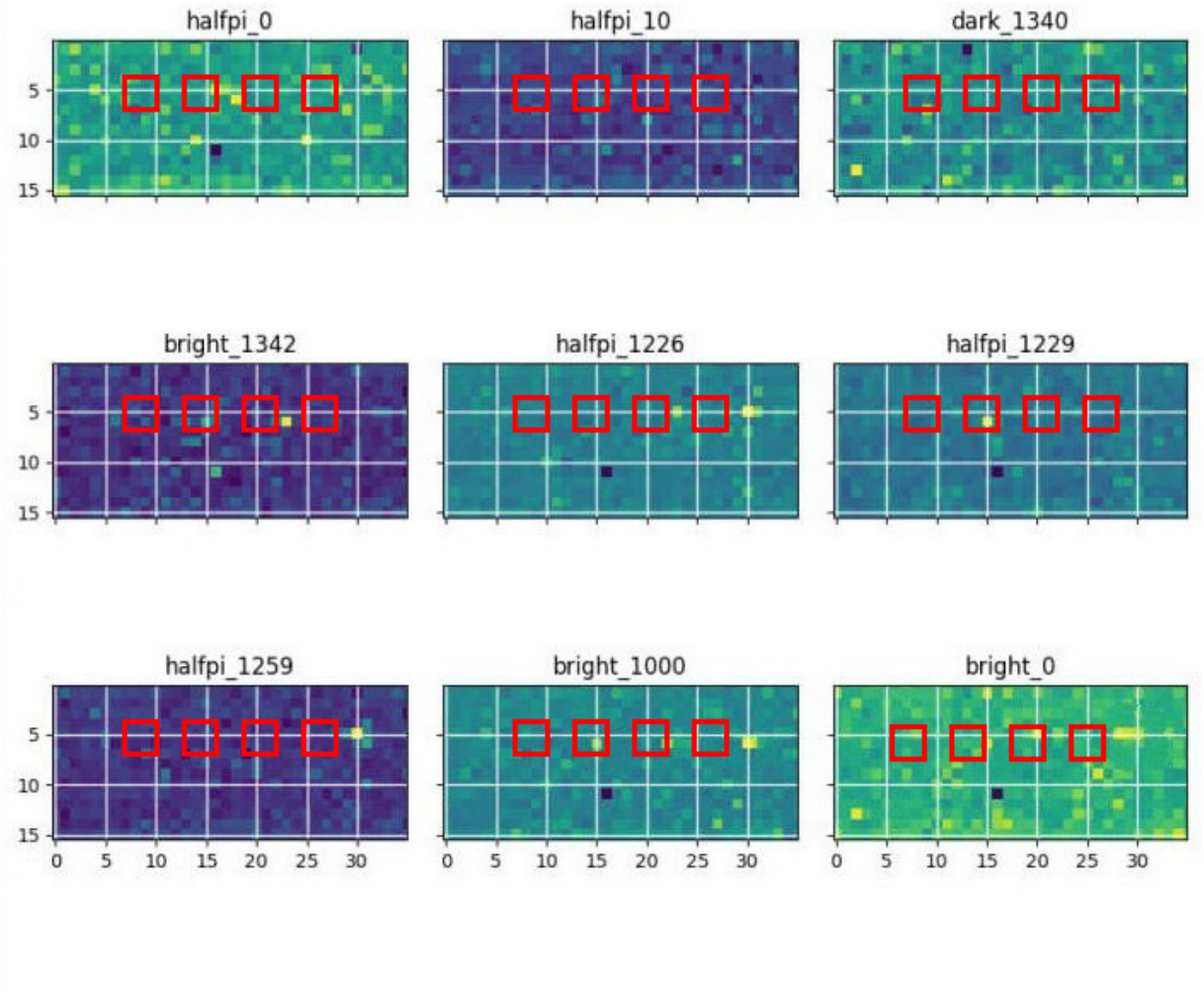


Figure 4.5: ROIs in different images we targeted and cropped for further use. Half-pi labels mean that states of ions are random, dark means that we should see no fluorescence from ions, and bright means that we should expect four ions to be fluorescent.

This noise could easily be mistaken as part of the ion's fluorescence by the model, resulting in incorrect classifications. To address this, we chose to crop the areas of interest surrounding the ions. Cropping also reduces the effect of ion crosstalk. By separating each ion in the image, the model can focus on the relevant information for each ion separately, rather than being misled by interference from neighbouring ions. This strategy also improves scalability because the algorithm can focus on learning the specific properties of each ion in a parallel manner. If the number of ions increases, we simply add more feedforwards for classification, ensuring that the process remains efficient. To implement

the cropping, we took bright images from the dataset (10,000 images) and printed their mean image; such an image is shown in Figure 4.1. We then found the centres of the ions in this averaged image and cropped a 5×5 pixel square around each ion's centre. This method resulted in a new dataset composed of cropped ion images that are more focused on the relevant information. However, this change required us to work with the dataset much more closely, manually arranging and preparing the cropped ion images to ensure optimal performance during model training.

The dataset for this study was provided in HDF5 (H5) format, which is a common file type for storing large amounts of structured data, particularly in scientific computing. While H5 is efficient and frequently used, it may not always be optimal depending on the user's experience with the format and requirements of metadata. As a result, we decided to first convert the H5 data into a more traditional format, such as NumPy arrays or Tensor data types, which are more often used in machine learning tasks.

After the transformation, it was important to ensure that the dataset was well-organized. This required iterating over the full dataset and providing unique labels to each image. The labels and matching keys were gathered into a systematic list, and each key was aligned with its image for fast and easy access. This involved iterating over the full dataset and providing unique labels to each image. The labels and keys were organized into a hierarchical list, with each key aligned with its relevant image. This strategy significantly improved the data extraction process during model training, resulting in faster and more efficient access to the required data. For data splitting we used a commonly known practice in machine learning: we divided the dataset into three separate parts: the training set - 70% of the data; the validation set - 20% of the data, and the test set - 10% of the data.

The training set is used for model learning, the validation set for tuning hyperparameters and performance assessment throughout training, and the test set for final evaluation. The test set stays "unseen" by the model during the training and is reserved for final evaluation. The model does not learn from the test set throughout the training and validation process, ensuring that performance metrics on this set represent the model's capacity to generalize to new, previously unknown data. This methodology provides an excellent way to evaluate the trained model's real-world performance, but it does not guarantee high accuracy on future data because it is dependent on the training set's quality and representativeness. Also regarding model and dataset structure alignment, before beginning training, it is essential to understand both the structure of the dataset and the architecture of the machine learning model. In this scenario, the dataset was required to be formatted in a way that was consistent with the architecture of the model being trained. Our approach requires training four distinct feedforward algorithms, each representing a different

ion. This method reduces bias, which is responsible for systematic errors in classification, giving more weight to some specific states. Additionally, this approach allows for more precise adjustment of the model, especially when labelled data is limited.

In our particular setup, each image in the dataset is associated with a key that is indexed. The labelling system specifies whether the image depicts a bright, dark or unknown mixed ion state, as well as the image number and ion index. This ordered structure facilitates our model and helps to detect biases.

Overall, we have 20,000 blocks of 4 individual index-dependent images of labelled ions - $|0000\rangle$ and $|1111\rangle$ states and 10,000 created batches uniformly distributed across the 16 possible states. As we know from a state detection procedure [51] we can expect a uniform distribution of classes in the unlabelled data. While we do not know if the model is perfectly accurate, we have an advantage of learning from those images more effectively by applying uniform distribution labels to the dataset. This then means that each of our mixed states has an equal representation of 625 images per class, which presents a unique opportunity to employ the MSE loss function for model adjustment, aiming at a predicted distribution that aligns with this uniformity. For example, after training a basic FFNN model on the labelled data, we are able to run predictions on the unlabelled data and compare the model's effectiveness through a comparison of the predictions to the uniform distribution. This allows for the validation of labelled machine learning models on our entire dataset and will allow for the comparison of supervised and semi-supervised methods. Additionally, this provides a unique bias-control tool, as without this addition, a model trained to only predict states of $|1111\rangle$ and $|0000\rangle$ would inevitably be overfitted and non-generalizable.

4.1.4 Data Preparation

After the creation of our raw data files, we had to prepare the data for processing by our various ML methods. To begin, we define a PyTorch Dataset class, named *IonImagesDataset*, to streamline the process of loading and preprocessing the data, ensuring that each image has the correct format and making sure we support the automatic shuffling of the data. Following standard image processing techniques, we applied normalization to the image data, scaling pixel values to an acceptable range for a ReLU, allowing for simplified data processing. Due to the limited amount of data, leaving in the colour channels was considered in this phase, but, based on the findings in the existing research on the issue, we chose to use grayscale conversion of the images based on established research findings [27]. We then transformed the images to align with the input requirements of our neural networks, including the addition of a channel dimension with the unsqueeze(0) function,

which adds a new dimension of size 1 at the start of a tensor, used to simulate a batch dimension, alongside creating copies of the dataset of shapes:

Table 4.1: Data shapes and their descriptions.

Shape	**Description**	**Purpose**
100	Flat array, representing a single-dimensional dataset.	Used for simpler analysis and initial preprocessing.
4×25	Grid with 4 rows and 25 columns, representing a 2-dimensional structure.	Prepares the data for fully connected layers in neural networks.
$4 \times 5 \times 5$	4 layers of 5×5 grids, representing a 3-dimensional structure.	Designed for input into convolutional layers for spatial pattern recognition.

We then partitioned the data, allocating 80% of the dataset for training and reserving 20% for validation. There is no need to allocate 10% for testing data here, as our testing will take place on the unlabelled data through comparison to either shuffled states or the uniform distribution. To handle the batches in our learning, we employed PyTorch's *DataLoader* to optimize the batch processing for better memory management and introduced shuffling to reduce the risk of overfitting to our training data. Finally, with all of this in place, we were then able to move forward with the creation of our ML models for testing.

4.2 Other Architectures

In this section, we will show and discuss alternatives to our FFNN architectures like CNN, autoencoder-based SSL with support vector machine (SVM), and simple SSL method. We will show their results, advantages, and drawbacks.

4.2.1 Convolutional Neural Network results

Another experimental architecture we employed was a CNN classifier on our images to identify the effectiveness of simple ML approaches on our problem (architecture example shown in Fig. 3.2). For the architecture of the CNN the first convolution layer takes The input tensor has a shape of $(B, 4, 5, 5)$, where B is the batch size. A total of 32 convolutional filters of size 3×3 are applied without padding to extract low-level features from the data.

After that, a ReLU activation function was applied to the output of the convolutional layer. This introduces non-linearity to the data, allowing more complex patterns to be identified. Then, a pooling layer is applied, with a 2×2 window and a stride of 2. This makes the CNN more robust to small translations of the input images by reducing the dimensionality of the feature maps. Following this, we implement a second convolutional layer that takes an input of 32 channels from the previous layers and then applies 64 3×3 convolutional filters. This layer should assist in extracting higher-level features from the data, allowing for the model to hopefully identify the centre of the ions. Another set of ReLU and pooling layers are then applied with the same parameters as before, and after that, the output of the second pooling layer is fed into a flattening layer to get a 1D vector before being fed into the fully connected layers and applying linear transformations to get the final output. The first fully connected layer has 128 units, and the second has 4. 128 was chosen as the initial size based on the available literature [40] and common approaches, whereas we chose a size of four for our output to match our four-ion setup in the experimentation. Finally, this output is fed to a sigmoid function, mapping the output to a [0,1] range. For prediction of a 4-bit binary class, if the sigmoid output is greater than 0.5, then the ion is assigned a label 1 and a 0 otherwise, additionally allowing us to treat the data as a probability prediction and use dynamic thresholding where needed if the model is to be used in a context that requires additional information.

In Figure 4.6 we find that the training and validation loss for the CNN model both rapidly reduced and were nearly equal for the first 25 epochs, indicating rapid learning of the patterns in the data and a tendency to learn without overfitting early on. As the number of epochs increased further, the validation loss began to increase, surpassing the training loss, and slowly diverged further from the training loss. This divergence of the two losses indicated the beginning of overfitting to the training data. The accuracy of the CNN reached a peak of roughly 66% between 150-200 epochs when validating the labelled data. This was quite poor overall and does not show the level of model fidelity expected for our application at all. Moreover, when the model processes test data this lack of fidelity is likely due to a mismatch between the simplicity of the images and the complexity of the CNN-based approach. While CNNs excel at processing larger images by detecting and recognizing features across spatial dimensions, their strengths are less relevant to this task. As with each image only being of dimension 5×5, multiple processing filters cannot capture much pattern variety. This suggests that the problem may not require the complexity of a CNN and could be addressed more effectively with simpler models.

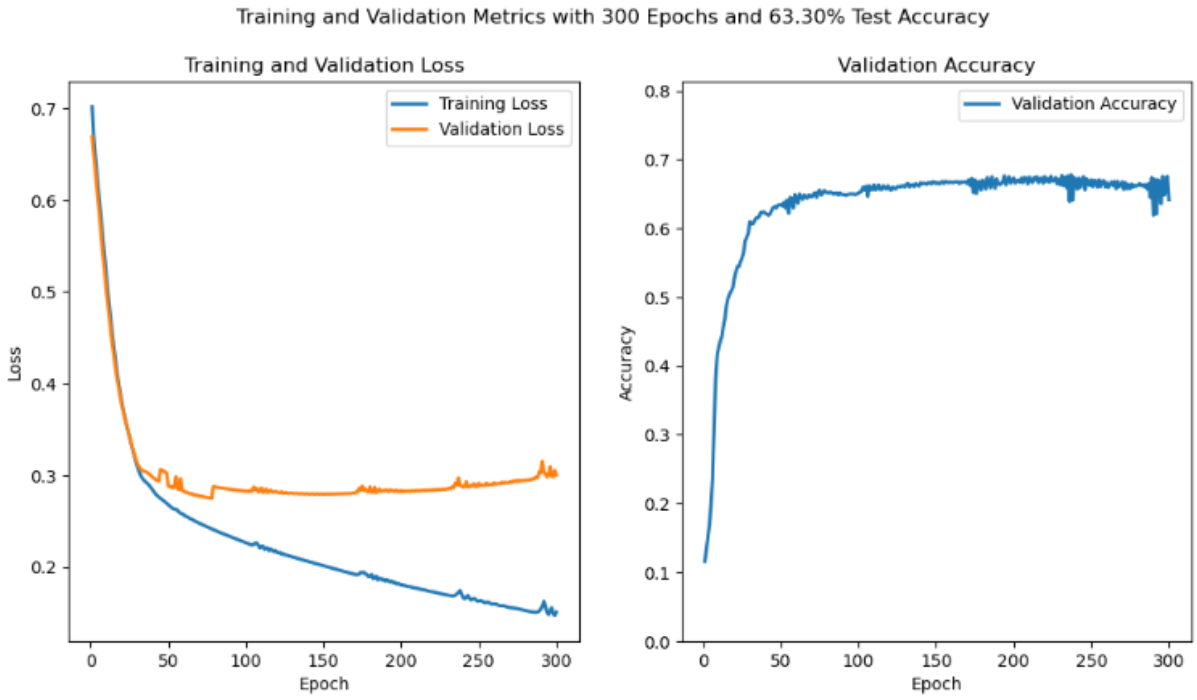


Figure 4.6: Illustration of the training and validation metrics for the CNN model over 300 epochs. The left panel shows the dynamics of training and validation loss (a measure of the model's prediction error), while the right panel tracks the accuracy of the validation.

Thus, although CNNs are powerful in complex tasks on big and high-dimensional data, their results on such tasks with simple low-dimensional input are not as good. In such cases, another approach might yield better results at least without the superfluous complexity introduced by deep convolutional layers, using other methods such as fully connected neural networks or even classic machine learning methods like random forests. While the performance difference is narrowing, CNNs are still among the best ways to do any form of image processing, provided the data is rich enough to exploit their hierarchical feature extraction capability.

4.2.2 Autoencoder-Based Semi-Supervised Learning Approach

Building upon previous conclusions, we designed an autoencoder [79] and wrapped it into a classification framework that could support our SSL method discussed in Section 3.3. The purpose of autoencoding is to learn a mapping from high-dimensional observations to a lower-dimensional representation space so that the original observations may be approximated (reconstructed) from the lower-dimensional representation. The assumption is that

if the model is introduced with reconstructed images of ions in addition to regular labelled data, we can produce better results in classification accuracy.

The autoencoder that was used had a dimensionality of four matching the number of ions within our system, and it was initially trained unsupervised. It transformed the input data into smaller representations, which could be transformed back to obtain the original data. The training objective is set to minimize reconstruction error between input and output. The received reconstructed states were fed into the classifier, trained on labelled data, to decode or "unshuffle" the states. The term "unshuffled the states" refers to a process of decoding or interpreting the latent space representations given by the autoencoder in the right way to identify and assign the correct fluorescent states. This was necessary because, although the VAE can generate a representation of 16 different states for the 4 ions, these generated states may not accurately reflect the real images. Since the VAE does not inherently understand the underlying physics, it could produce states that are physically implausible or inconsistent with the actual ion configurations.

We tested three different autoencoder architectures:

- **Basic Autoencoder**: This single-layer design uses a single dense layer to encode the input directly into a latent space of dimension 4. One dense layer was used in the decoding stage to rebuild the input.
- **Deep Autoencoder**: This architecture has multiple layers, including three dense layers that are used for encoding and decoding tasks, respectively. This design gradually decreased the dimensionality of the input using layers of different sizes: $\frac{n}{2}$ and $\frac{n}{4}$, where n is an image size. While the latent space size is equal to 4. The decoding mirrored this process in reverse, progressively reconstructing the data back to the original size.
- **Convolutional Autoencoder**: This convolutional architecture is designed to exploit spatial patterns in data. To minimize dimensionality, the encoder uses 2D convolutional layers with 1×4 filters and max-pooling. The decoder uses up-sampling layers to restore the latent space to the original image dimensions.

We evaluated each autoencoder architecture consistently with a three-step approach for assessing both Accuracy during reconstruction and classification performance:

1. **Latent Space Analysis**: To assess reconstruction fidelity and information preservation, we print the latent space representations generated by VAE and compare them with the original images.

2. **Performance classification**: The Support Vector Machine (SVM) classifier takes encoded features as input features, thus enabling a test for the effectiveness with which latent representations capture patterns from original images. SVMs are supervised algorithms that classify data by determining the optimum hyperplanes to divide various classes with the highest margin. The SVM decision function can be written as $y = wx' + \gamma$, where y is the predicted class, w is the weight vector, x' is the input feature representation, and γ is the bias term. For an ion with latent representation Z_i, the SVM performs the following classification:

$$h(Z_i) = \text{sign}(\langle w, Z_i \rangle + b)$$

Here:

- $\langle w, Z_i \rangle$: is the inner product (dot product) between w, Z_i
- w: The weight vector that defines the hyperplane
- sign$(\cdot)$: Returns $+1$ or -1 based on which side of the hyperplane Z_i lies
- b: The bias term

Rather than attempting to classify all four ions simultaneously (a complex 16-class problem), we used an RBF (Radial Basis Function) kernel SVM to analyze each ion's encoded representation separately. This modular approach allowed us to combine individual ion classifications to determine the complete quantum state of the system.

3. **Training Enhancement**: We incorporated the reconstructed data generated by the autoencoders into our training dataset to enhance the model's generalizability and reduce overfitting.

The deep encoder model we paid more attention to fairly quickly learned the patterns within the data, which can be observed from the sharp drops of both training and validation losses at the beginning stages of training (shown in Figure 4.7), indicating effective learning and generalization beyond mere memorization. As this training progresses, the learning curve flattens out to indicate successful generalization of the problem space.

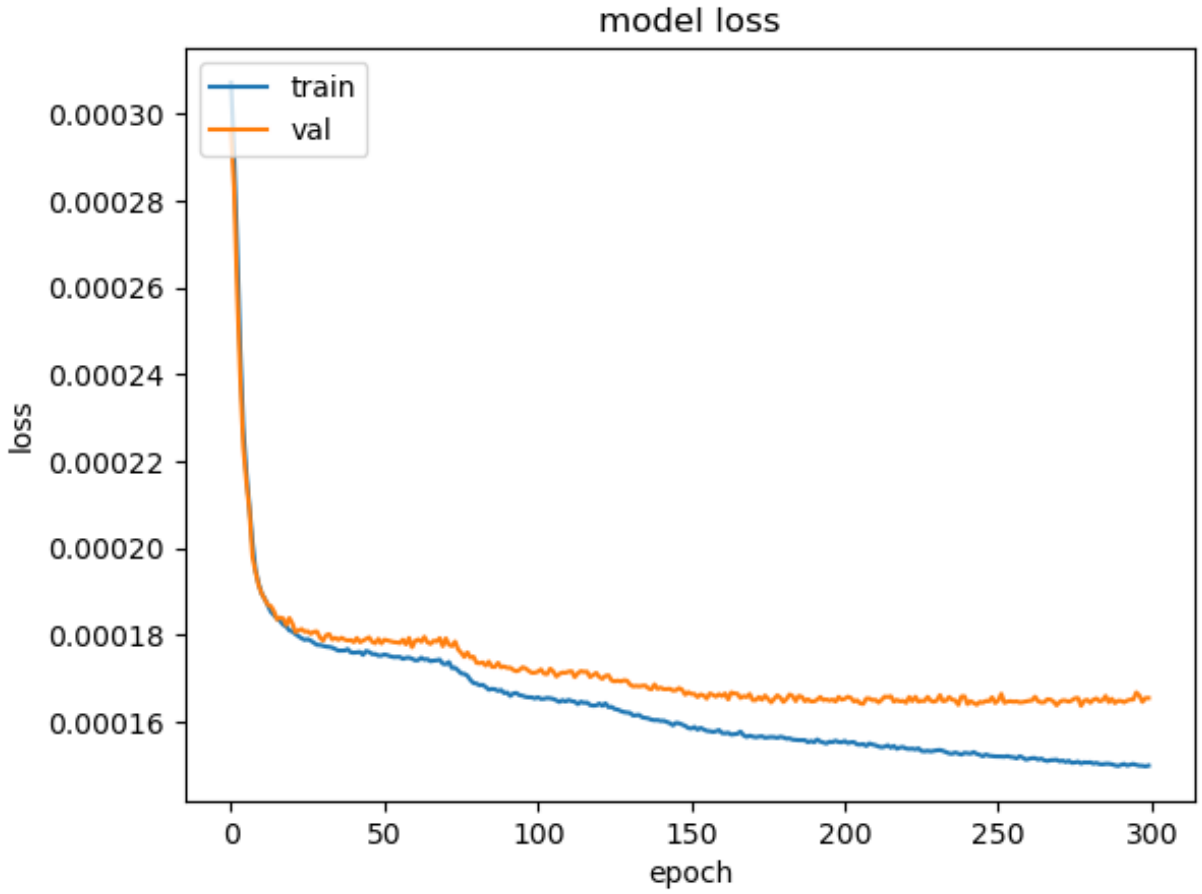


Figure 4.7: Loss to epoch graph for deep autoencoder for reconstructing input images. Trained for 300 epochs and showed a positive result in reducing loss to an optimal level, showing a plateau after 150 epochs.

Among the three architectures that were tested, the deep autoencoder outperformed the others, achieving a classification accuracy of 99% on reconstructed images it created when combined with an SVM. The convolutional autoencoder reached 98%, demonstrating its strength in spatial feature extraction, and finally, the basic autoencoder reached 96% reconstruction accuracy. The low final loss values reflect the autoencoder's strength in compressing and reconstructing data, validating our semi-supervised strategy. It's important to mention that these accuracies aren't representative before we test these architectures on real data.

Following the analysis of the reconstruction data we then incorporated a SVM within our multi-ion classification framework, allowing the SVM to classify four different class boundaries as separate binaries rather than taking a 16-class approach. Despite the accuracy in encoding, the resulting bit accuracy hovered around the 50 percent mark. Specif-

ically, the error rate stood at 49.7 %, meaning nearly half of the classifier's predictions deviated from the actual labels. This performance was just marginally above what would be expected from random guessing in a binary classification scenario, implying the classifier's inability to learn any patterns from the encoded features to make accurate predictions. This held true for each of the three autoencoding methods, with each scoring within 5% of the 50%, and when tested against the interpretation of the raw states, both performed similarly, indicating that the internal state of the autoencoder is not simply interpretable as the four bits from our problem.

As such, we moved forward with testing the deep neural net approach in order to create a proper prediction framework. Of the three autoencoder methods used, the best was still the deep autoencoder where the final epoch concluded with a training loss of 0.35, a validation loss of 0.35, and a validation accuracy of approximately 84.9%. These metrics indicate strong model performance, with the validation accuracy reflecting a high degree of generalizability. The other two methods performed slightly less well on the dataset with accuracies of 75% for the one-layer encoding and 82% for the convolutional encoder, showing that the choice between deep and convolutional encoding is only slightly impactful on the performance of the model with the deep encoder performing barely better in testing. To sum up the results, consult the table below:

Architecture	**Reconstruction Accuracy (%)**	**Final Loss**	**DNN Classification Accuracy (%)**
Deep Autoencoder	99	0.35	84.9
Convolutional Autoencoder	98	≈ 0.25	82
Basic Autoencoder	96	≈ 0.38	75

Table 4.2: Performance comparison of autoencoder architectures showing reconstruction accuracy, final loss, and validation accuracy on labelled data, when combined with deep neural network classification.

4.3 FFNN development

After observing poor performance with our initial approach on uncropped data in Section 4.1.1, we experimented with multiple ML models and 2 different data preprocessing techniques. While CNN models are commonly the first choice for image processing due to their

proven effectiveness, our specific case required additional considerations. It is important to note that the approach that the previous team employed on this problem was random forest algorithms. As a result, they achieved high accuracy in classifying 2-qubit images. Nevertheless, we decided to move on to different architectures. The reasoning behind it was the fact that we want this algorithm to be fairly scalable and potentially more effective in reducing bias during the classification of unknown mixed states, since all of our data for mixed states is unlabelled. Originally, our approach was simple: the FFNN was designed to learn from labelled images and classify whole 4-qubit states to zeros and ones. This idea worked, but the final result was far from accurate, which we discuss in detail in Chapter 5.

During this phase of the project, we chose to manually locate the ROIs. This option was practical given the nature of our data, which consisted of ions positioned in specific positions with limited movement. Identifying ROIs by hand was helpful for this dataset since it allowed us to focus on the most relevant regions for classification, making the process easier for the network. We then created a new dataset that highlighted these regions of interest.

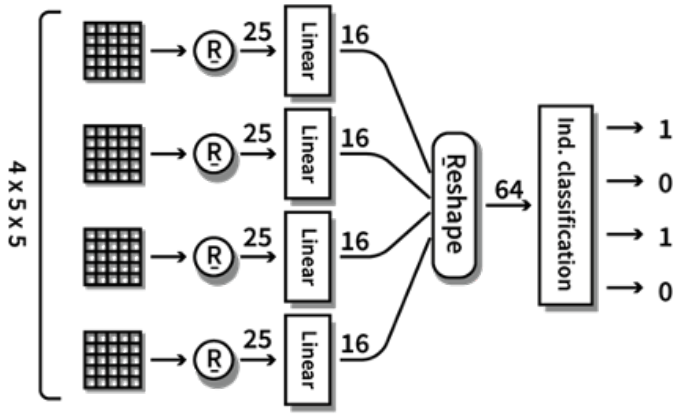


Figure 4.8: Architecture of the custom FFNN model for quantum state classification. The model processes each of the four ions separately through parallel channels. Each 5×5 pixel ion image is flattened to a 25-dimensional vector by a reshape (R) function and passed to a linear layer that reduces dimensionality to 16 features. The features from all four ions are then combined in a reshape layer before final classification, finally outputting the complete four-qubit state (represented as four binary digits). This modular architecture allows independent processing of each ion to capture its unique patterns while allowing for a unified quantum state classification.

The first stage of training focused on achieving an accuracy of at least 90% on labelled data. This goal was necessary to minimize bias towards particular states represented by only zeros or ones. The challenge was to classify 16 possible states, corresponding to different ion configurations. Despite the inherent difficulty, achieving high accuracy in this stage is critical for the eventual implementation of semi-supervised learning techniques in future iterations of the project in Chapter 5.

Previously, we used a simple FFNN to process and classify images altogether. We improved that algorithm by also parallelizing the classification to a specific ion, followed

by a sigmoid activation function to assign a probability to label each ion image. We tied the classifications of the given images together to create a state classification representation, which consists of four binary digits. These combined digits were compared to true labelled data. We'll discuss this architecture in more detail in Chapter 5.1, but for now, it satisfied our need for 92% accuracy on labelled data, and we were ready to move on to the next stage. However, before that, we wanted to run one last experiment in order to boost the accuracy further, if possible.

We tried to use the same FFNN with concatenated and cropped images. This step was part of an architectural upgrade that involved processing ions together in order to improve the network's capacity to recognize ROIs. Although we predicted that this method would result in only slight increases in accuracy, the outcomes exceeded our expectations. We received more than 10% improvements and decided to test the newly trained network on unlabelled data. The result of the classification was too biased, unfortunately, as shown in Figure 4.9.

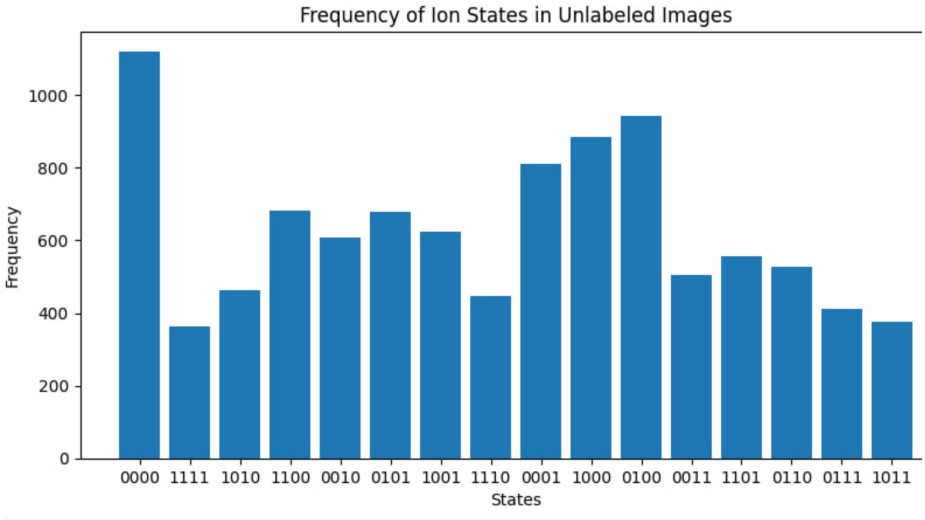


Figure 4.9: Early-stage classification distribution with an obvious bias toward $|0000\rangle$ and $|1111\rangle$ states.

At this point, we came up with the notion of constructing a synthetic dataset containing 16 possible states of four ions from our labelled data. To do so, we proportionately blend labelled ion pictures, just manually combining randomly selected bright and dark ion images. We now had an equal number of organized mixed states for each class, made up of previously marked $|0000\rangle$ and $|1111\rangle$. This is an important step in understanding the nature of the data and how our crosstalk operates.

Despite the observed improvements, the distribution of classified unlabelled data showed additional uniformity but remained far from ideal for enabling effective semi-supervised learning. This limitation is illustrated in Figure 4.10, which highlights the resulting distribution.

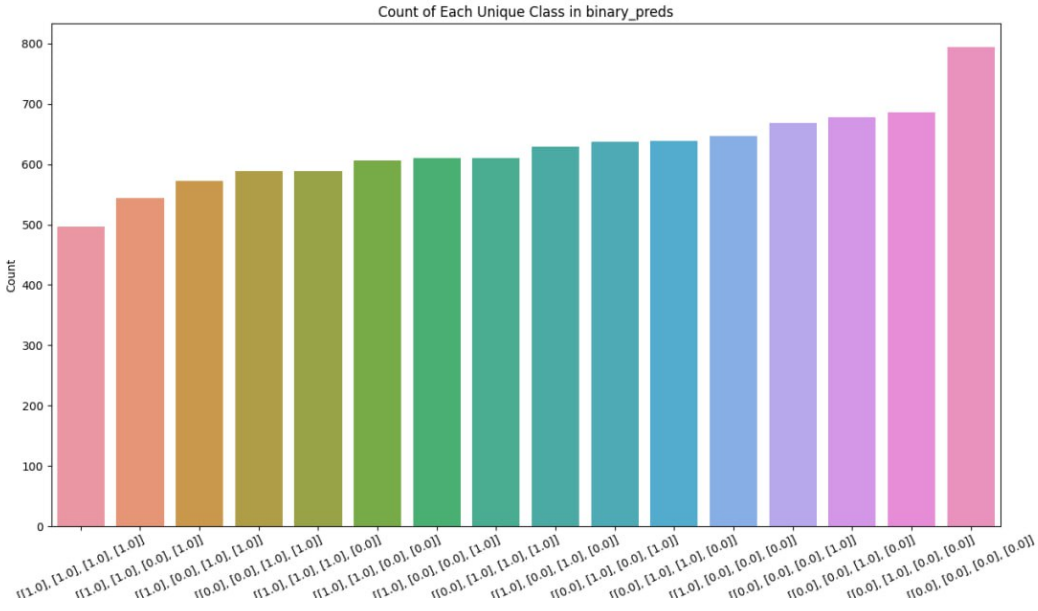


Figure 4.10: Distribution of classified mixed states by custom FFNN model trained on constructed labelled images of all 16 possible states.

After additional research, we discovered various factors that contributed to this poor performance. First of all, we realized that mixing labelled ion images would not help because we have to account for crosstalk sources that affect neighbouring ions or other noise sources. Border ions have different fluorescence patterns because they lack neighbouring ions on one side, and ions separated by gaps experience less neighbouring fluorescence interference. These position-dependent effects prevented the model from learning reliable combinations of ion states. In many cases, neighbouring ions released photons that interfered with ion classification, confusing the algorithm. Furthermore, the synthetic composition of ion images failed to replicate the complex optical interactions present in genuine mixed quantum states.

The main objective of the new architecture was to make it flexible enough for future improvements. Our goal was to maximize the extraction of useful features by optimizing the architecture of this particular dataset. The idea was to include unlabelled data and address any biases that surfaced during the second training stage once the model had been

sufficiently trained on labelled data. We built an advanced FFNN with an encoder-decoder architecture in order to achieve this goal; we discuss details of the implementation in the next chapter. To make sure the network learned the most important features, we only used labelled data during the first training phase. After that, we added a different coupling mechanism that merged the biases and weights of four distinct feed-forward networks. This coupling allowed the network to jointly classify all ion states, integrating the outputs from the separate pathways. This structured approach provided a solid foundation for moving toward more effective classification of unlabelled data in future stages.

Chapter 5

Results

In Chapter 4, we focused on the methods used to approach the problem, including the features of our datasets and the processes for organizing and preparing the data. Additionally, we evaluated various alternative architectures and their performance on labelled and unlabelled datasets. However, this chapter did not cover the successful implementation of a solution. The present chapter is devoted to determining and outlining the most optimal architecture and model tailored to the problem at hand.

Previously, we set up an architecture for labelled data, and we observed consistent results in learning, resulting in achieving a high accuracy of 92% in Section **??** for all bright and dark states. To move forward, we need to understand how to make unlabelled data work in order to classify multi-ion systems and make them scalable enough without losing too much accuracy in classification. On the first try, we implement the simple idea of using a pre-trained NN just to classify unlabelled images. The classification worked, but we lacked a reliable method to evaluate its performance. To test that, we used an important feature of the mixed dataset that was explained in Chapter 4, specifically that it should have a uniform distribution across all states. This crucial feature helps us to construct a custom loss function, which enables classification of $\frac{\pi}{2}$-rotated (mixed) state data. The main objective for the final stage of development is to change the data loading and training processes for unlabelled data. Effectively, it means that we do not need to initialize labels in the data loader, and we need to implement a custom loss. The final architecture for multi-ion classification leads to a successful attempt at solving all of the critical issues: modelling correlated data, cancelling biases and managing noise arising from ion interactions. It is a scheme under which several conceptual modules are glued together to increase the probability of high-performance scalability in the model.

What are the building blocks of the model? An important portion of the model is

ion-specific feature extraction; it is realized by a special approach where each ion has its own weight matrix and bias vector. In such a way, the model learns the unique properties of each ion: a characteristic noise profile, its brightness, and other possible ways a signal may behave. Unlike the standard dense layer that shares weights across all input data, this treats each ion separately to take into account its certain behaviours and responses to environmental changes or interactions with other ions. That special architecture keeps unique patterns of each ion separate yet still computationally efficient, increasing generalization and preventing interference due to shared parameters.

The first step is an encoder module, which starts with the step-by-step conversion of raw input data into higher-dimensional representations through ion-specific feature extraction. This forms a hierarchical representation of the data through a series of layers with non-linear activation, capturing both low-level signal characteristics and high-level abstractions. Early layers are responsible for basic attributes, like brightness variations and noise, while subsequent layers reveal much more complex relationships, such as dynamics of ion interaction and influence from the environment. This hierarchical approach ensures robust feature extraction and forms a critical foundation for accurate classification and interaction modelling. This classification stage transforms encoded features into class probabilities. This shows an idea of confidence in the state of every ion. A log-softmax activation function does this, it normalizes output probabilities and keeps numerical stability while training. The probabilistic outputs come out with such a function, which is very interpretable; thus, predictions turn out to be very reliable even in noisy or uncertain conditions. This stage becomes specifically critical for the analysis of individual ion behaviour and the overall reliability assessment of the system. Mathematically, we can describe the process as follows:

The encoder maps the raw input data $\mathbf{x}_i$ (specific to each ion) into a higher-dimensional latent representation $\mathbf{z}_K$ through a series of layers:

$$\mathbf{z}_k = \sigma(\mathbf{W}_k \mathbf{z}_{k-1} + \mathbf{b}_k), \quad k = 1, 2, \ldots, K$$

Here, term $\mathbf{W}_k$ and $\mathbf{b}_k$ are the weight matrices and bias vectors, while $\sigma(\cdot)$ represents the activation function.

The classification layer transforms the final encoded features $\mathbf{z}_K$ into logits $\mathbf{o}_i$ for each ion:

$$\mathbf{o}_i = \mathbf{W}_{\text{cls}} \mathbf{z}_K + \mathbf{b}_{\text{cls}}, \quad i = 1, 2, 3, 4$$

Here i indexes the 4 ions being processed in parallel.

Since there are only two local classes (0 and 1), the log-softmax activation function is applied to normalize the logits into probabilities:

$$\log \mathbf{p}_i[c] = \mathbf{o}_i[c] - \log \sum_{k=0}^{1} \exp(\mathbf{o}_i[k]), \quad c = 0, 1$$

Here $\mathbf{p}_i[c]$ is the probability of ion i belonging to class c.

The binary classifications for each ion ($\hat{y}_i$) are then combined to form a 4-qbit state class $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4], \quad \text{where } \hat{y}_i = \arg\max_c \mathbf{p}_i[c]$$

The final state predicted class $\hat{\mathbf{y}}$ is compared to the true 4-bit class $\mathbf{y}_{\text{true}}$ using the negative log-likelihood (NLL) loss:

$$\mathcal{L}_{\text{NLL}} = -\sum_{i=1}^{4} \log \mathbf{p}_i[y_{\text{true},i}]$$

This ensures that the model learns to classify each ion correctly and accurately reconstructs the complete 4-qubit global class.

One of the major weaknesses of this original architecture lies in assuming ion independence without considering any form of correlation or interaction between ions. This enhanced model (M_2) architecture incorporates one more module: a Coupler to model the interaction between such ions. The Coupler transforms the encoded features into a global representation, modelling the relationships among ions concerning how one ion's behaviour would affect its neighbours. A mathematical representation of the process is provided below:

The encoded output from the Encoder y is adjusted using the Coupler. The Coupler applies a transformation f_{Coupler} to y:

$$y_1 = f_{\text{Coupler}}(y)$$

Here f_{Coupler} includes a series of dense layers with ReLU activation:

$$f_{\text{Coupler}}(y) = \text{ReLU}(W_3(\text{ReLU}(W_2(\text{ReLU}(W_1 y + b_1)) + b_2)) + b_3)$$

Then, the Coupler output y_1 is added back to the original encoder output y, which creates an enhanced representation:

$$y' = y + y_1,$$

which then goes to the previously transferred classifier module. Cross-talk and shared noise are considered to enrich the feature representation with interaction-specific information in order to close the gap between independent and collective ion behaviour. More importantly, this enrichment is crucial in those systems that defy understanding ion behaviours in an isolated way. This updated version contains the Coupler module but keeps all the strengths of the basic model. It puts the base model encoder and classifier, whose parameters are frozen during training to not interfere with the knowledge acquired from the labelled data. The Coupler introduces interaction-specific learning focused on unlabelled data, which might better capture the underlying structure of ion correlations. The output from the Coupler is concatenated with the encoded features to enhance the generalization capability for various kinds of data distributions. The modular architecture of this sort will let the M_2 model exploit the stability and accuracy of the base model by remediating its limitations.

The architecture itself works as follows: the base model (M_1) first trains on labelled data to learn about the behaviour of ions individually. This will make for a solid foundation for further enhancements. In the next step, the Coupler will be trained separately on unlabelled data, where methods of unsupervised learning are used to model inter-ion interactions while preserving the individual features learned. Fine-tuning is done with both labelled and unlabelled data to bring out better adaptability in complex scenarios and to ensure good generalization.

Below, in Figure 5.1 we will show how the model looks in its final form and then break down the steps to develop it with respect to the results of the training.

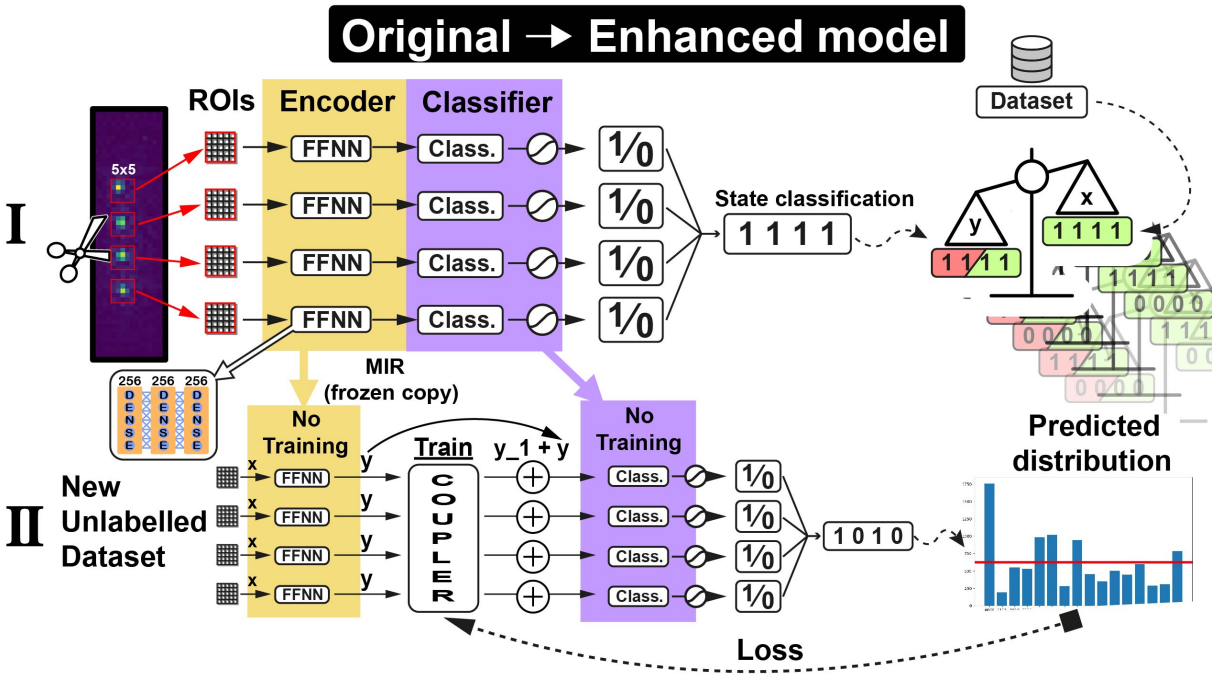


Figure 5.1: Two-stage model. The first stage shows the pipeline with four labelled images processed by the encoder and classifier to learn the main features of images. The second stage uses the frozen weight of the first-stage encoder additionally implementing a Coupler to combine all the weights and biases from images, followed by a pre-trained classifier with a softmax function at the end. Finally, we use our generated distribution to tune the model automatically and enhance the results of classification after each epoch.

5.1 Final Model Development and Results

At first, we used the most common approach and implemented mean square error (MSE) loss for this classified distribution. MSE calculates the average of the squared differences between predicted and target values, commonly used to measure the accuracy of regression models. However, we observed that our loss did not change through epochs, we concluded that it could be an optimization problem. After the investigation, we realized that the gradient signals weren't effectively propagating through our network because of our discrete distribution counts and the architecture of our model. This indicated we needed a different approach to the loss function that would better handle our specific distribution matching task.

An alternative approach that we tried was implementing the Kullback-Leibler divergence for the distribution we received from classification versus the uniform distribution

we were aiming for. This strategy worked, and we observed a decreasing loss function; however, our accuracy did not improve. Finally, we concluded that additional changes to the architecture were required. We implemented one-hot encoding and Gumbel softmax [34] in the process along with new combined loss.

One of the common representations of categorical data as a binary vector is one-hot encoding, where each class gets assigned to a particular position in that vector. The element representing a class is 1, and the rest of the elements are 0. Mathematically, for a class $c \in \{1, 2, \ldots, C\}$, the one-hot encoded vector $\mathbf{e}_c$ is:

$$\mathbf{e}_c[i] = \begin{cases} 1 & \text{if } i = c, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, 2, \ldots, C$$

Here C is the total number of classes.

Gumbel softmax is a technique used to approximate discrete sampling in a differentiable manner. It introduces noise from the Gumbel distribution into the logits $\mathbf{o}_i$ and applies a softmax function to produce a continuous approximation of a one-hot vector. For a class c, the Gumbel softmax output $\mathbf{p}_i[c]$ is given by:

$$\mathbf{p}_i[c] = \frac{\exp((\mathbf{o}_i[c] + g_c)/\tau)}{\sum_{j=1}^{C} \exp((\mathbf{o}_i[j] + g_j)/\tau)}$$

Here $g_c \sim \text{Gumbel}(0, 1)$ is noise sampled from the Gumbel distribution, and $\tau > 0$ is the temperature parameter controlling the sharpness of the distribution. As $\tau \to 0$, the output approaches a one-hot vector.

These techniques were employed to make the classification process more robust and differentiable. Unfortunately, the gradient flow issue persisted.

After further consideration, we realized that only a custom loss function would work. We designed a correlation loss function, which solved our gradient problem, the process is shown below: The total loss combines:

1. NLL loss:

$$\mathcal{L}_{\text{NLL}} = -\sum_{i=1}^{N} \sum_{k=1}^{C} y_{i,k} \log(\hat{y}_{i,k})$$

   Here C is the number of classes and $\hat{y}_{i,k}$ is the predicted probability for the k-th class.

2. Regularization Loss (Optional, e.g., Correlation Loss): The correlation loss $\mathcal{L}_{\text{corr}}$ penalizes dependencies between the predicted outputs:

$$\mathcal{L}_{\text{corr}} = \text{Penalty Function(Correlations of Outputs)}$$

or more precisely:

$$\mathcal{L}_{\text{corr}} = \frac{1}{|\text{UpperCorr}|} \sum_{i,j \in \text{UpperCorr}} |\text{Corr}_{i,j}| + \frac{1}{C} \sum_{i=1}^{C} |\mu_i|$$

Here:

$$\text{Corr}_{i,j} = \frac{\sum_k y_{k,i} y_{k,j} - \mu_i \mu_j}{\sqrt{\sigma_i^2 \sigma_j^2}}$$

$$\mu_i = \frac{1}{N} \sum_k y_{k,i}, \quad \sigma_i^2 = \frac{1}{N} \sum_k (y_{k,i} - \mu_i)^2$$

- $y_{k,i}$: The i-th component of the output vector for the k-th sample, transformed to range $[-1, 1]$ using $y = 2 \cdot \text{classify}(x) - 1$.
- μ_i: The mean of the i-th component across all samples.
- σ_i^2: The variance of the i-th component across all samples.
- $\text{Corr}_{i,j}$: The Pearson correlation coefficient between components i and j.
- $|\text{UpperCorr}|$: The number of unique correlations in the upper triangle (excluding the diagonal) of the correlation matrix.
- C: The total number of components (or classes).
- $\mathcal{L}_{\text{corr}}$: The total correlation loss, consisting of the absolute mean correlation and the mean absolute component bias.

This approach gave us immediate results and provided our first working output:

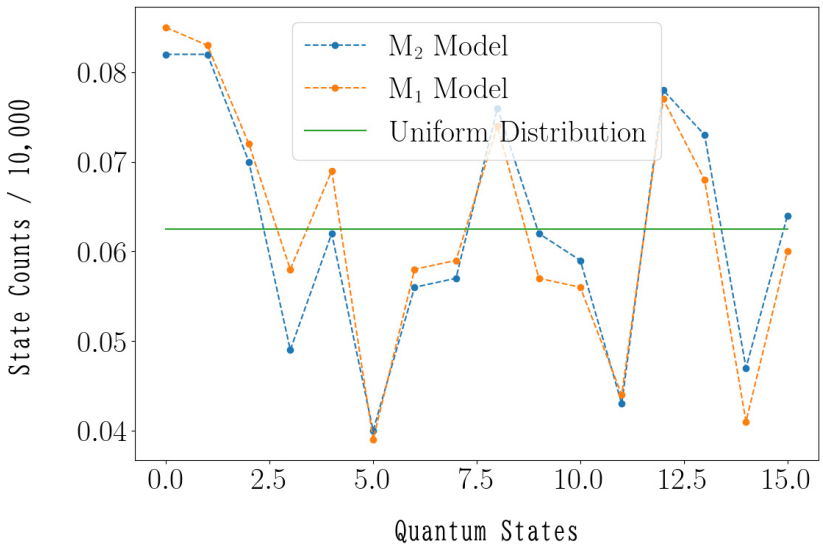


Figure 5.2: Comparison of performance of 2 models: one with tuning turned off and one with a Coupler block not working. This demonstrates the importance of the Coupler shown in Figure 5.1 as a tool for sharing combined information from all images.

In detail, Figure 5.2 shows a count of the states the model classified on the left as 0.01 equivalent to 100. On the bottom, we can see a representation of 16 unique states created out of 4 qubits. And a green line represents the uniformity we want to converge towards. A positive result for a classification of unlabelled states is a maximum alignment of the blue and orange lines to a horizontal uniformity.

After implementing the working Coupler, we can see that the lines of our distribution became more stable and closer to the uniform we are aiming for; consult Figure 5.3. Also, it is visible how better the M_1 model performs after tuning.

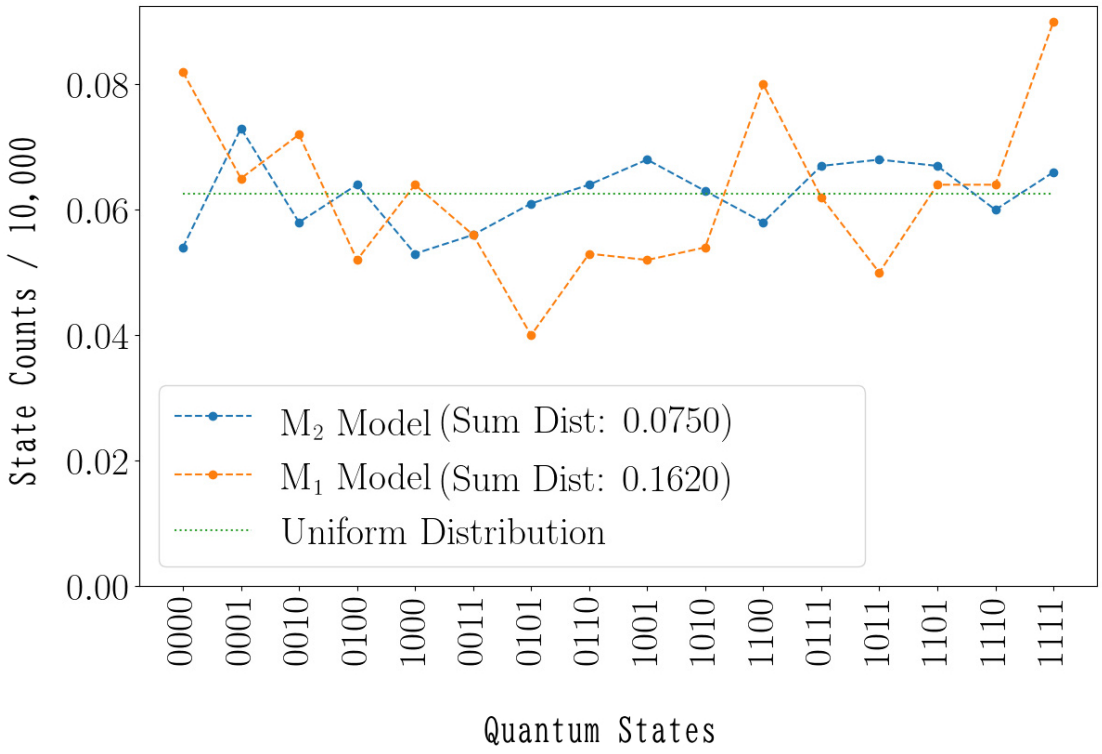


Figure 5.3: We show the first improvement in the working version of the model. Also, the sum distance from a uniform distribution of states was introduced as a measure of success.

Additionally, we tuned our model using the *torch.optim* optimization tool. This is a PyTorch module that provides optimization algorithms for training machine learning models using gradients computed during backpropagation. Effectively, we run training hundreds of times, testing M_1 model with different hyperparameters, arriving at the optimal: N epochs: 25 and learning rate: 0.00035. Then, the same procedure was executed on M_2 model with the result of optimal N epochs: 15, learning rate: 0.00016, and weight decay: 5.65e-5, which gave a better final output illustrated in Figure 5.4. We can clearly see the sum distance to the uniform distribution line. Here, we define the sum distance to uniform as the sum of absolute differences between each predicted state probability and the ideal uniform probability (1/16 for our 16-state system), providing a quantitative measure of classification bias. Lower sum distance values indicate more uniform predictions across all quantum states, with zero representing perfect uniformity. We want to highlight the importance of automatic optimization to receive the best performance from the existing architecture.

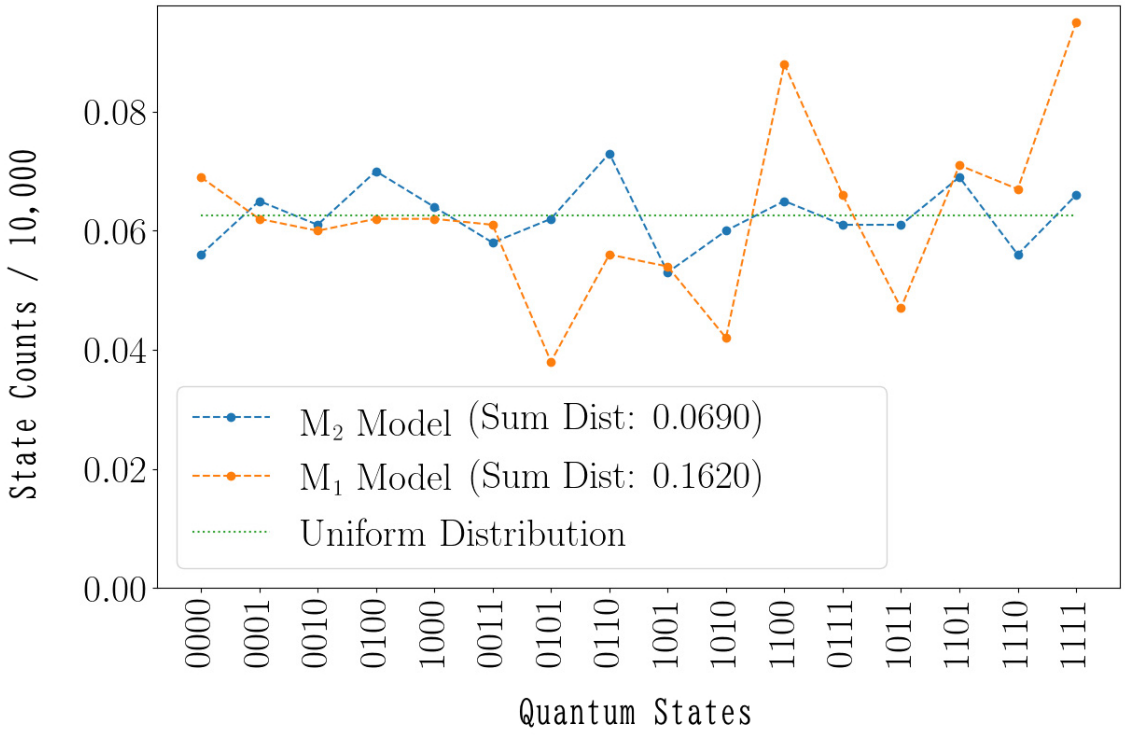


Figure 5.4: A small fine-tuning of hyperparameters shows better results in distribution uniformity for both M_1 and M_2 models. Interestingly, when bias towards edge states is reduced, we observe more instability in middle states.

After adding the new dataset (D_2), Figure 5.5, from the lab, we nearly doubled our overall data points and it provided a great opportunity to see how added data points would impact both the reduction of bias and classification accuracy. We tested the encoder-decoder pre-trained model on images coming from the D_2 dataset first but got unexpected results: instead of decent classification accuracy, we had a notable drop in accuracy. This prompted us to revise our methodology and split our D_2 and D_1 into a new separate test, train and validation datasets.

The important difference between the D_1 and D_2 was fluorescence intensity. In order to maximize the contrast from zero to one state, we decided to focus only on images with the best contrast (-2 dBm), assuming such adjustment could enhance the training process because the lab provided us with the same images but with different brightness intensities illustrated in Figure 5.5. Initially, though, there were no measurable gains in the classification performance with this additional data.

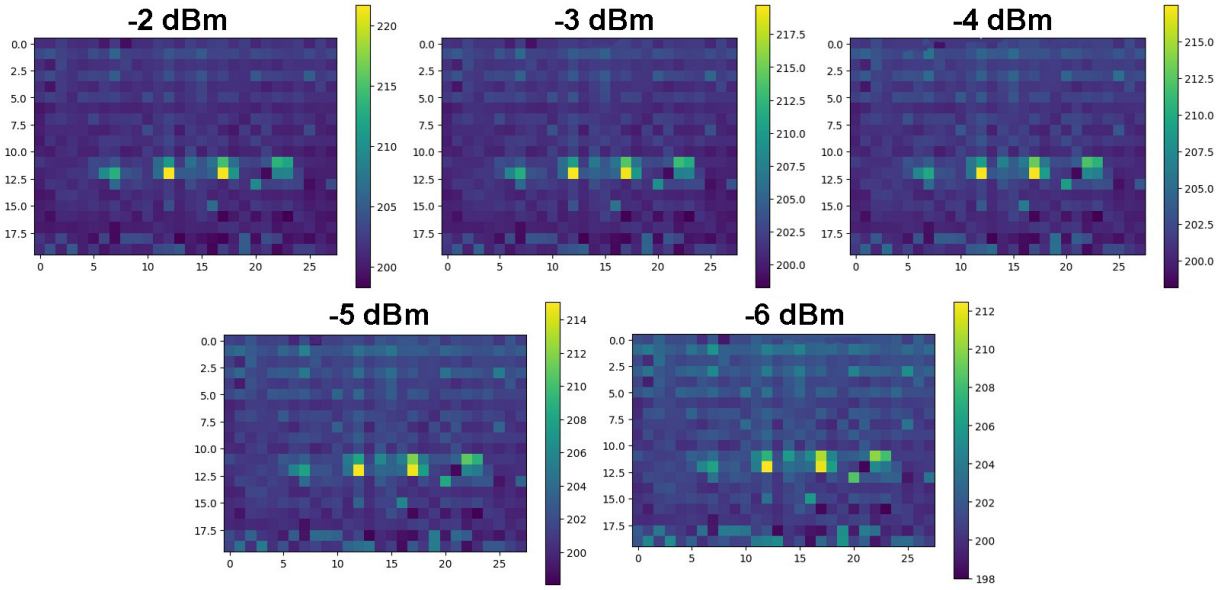


Figure 5.5: Averaged bright images are shown for each power level: -2 dBm, -3 dBm, -4 dBm, -5 dBm, and -6 dBm. The average brightness distribution is given in each heatmap's grid. Yellow corresponds to a high intensity, while purple corresponds to a low one. Power levels in the first and second rows decrease from left to right.

While further tuning, the power of the combined dataset D_3 started becoming more apparent. The M_1 model failed on the new data with growing bias and reduced accuracy. The M_2 model, however, did very well with the new training data; it adapted well to the new variations in ion behaviour. Training results of the M_2 model on the extended dataset had the best classification results shown in Figure 5.6.

Another important observation was that the addition of this new D_2 dataset changed the data distribution. The D_3 dataset presented some variations that the original D_1 did not have, which would make the work of the classifier slightly different. Specifically, it caused a significant shift in the bias towards the $|1111\rangle$ state, likely due to the increased frequency of this pattern.

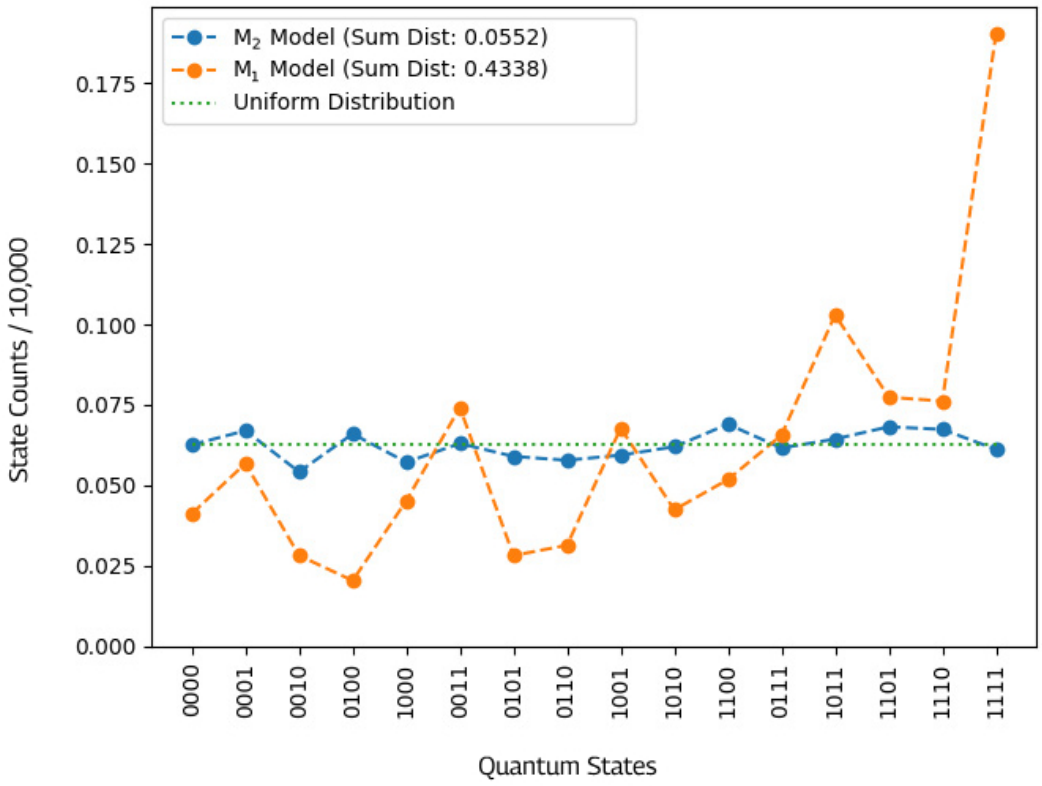


Figure 5.6: The untuned results of D_3 show a significantly larger bias towards classifying states as $|1111\rangle$. Our hypothesis is that the improved brightness in the new dataset causes more borderline cases, where ions with ambiguous brightness levels that were previously classified as $|0\rangle$ are now more likely to be classified as $|1\rangle$ by default.

After a second optimization of the parameters using *torch.optim* we got our all-time best result with close to perfect bias - sum distance to the uniform was 0.033 with a great accuracy of 87% on test labelled data, in addition to that.

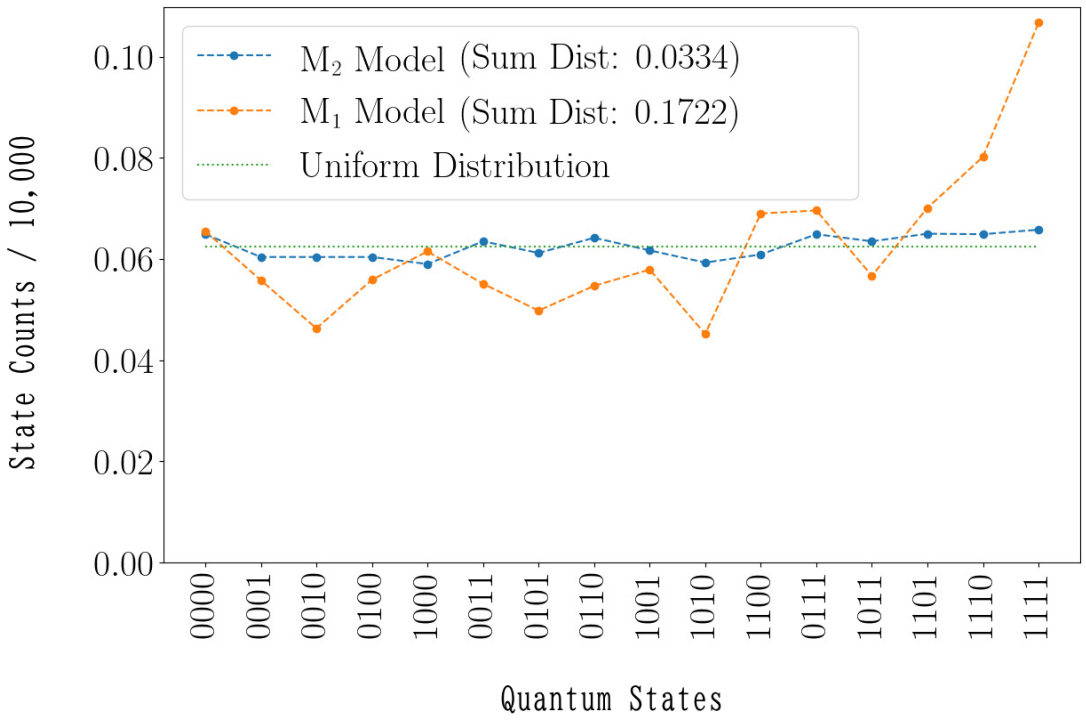


Figure 5.7: Comparison of the predicted state distributions of the M_2 and M_1 models against the ideal Uniform Distribution for 16 quantum states ranging from $|0000\rangle$ to $|1111\rangle$. The M_2 model has a much lower sum distance of 0.0334 to the uniform target compared to the M_1 model, which had a sum distance of 0.1722. This reflects that there is less bias for all states, which is a good sign for future generalization.

5.2 Final Comparison

Table 5.1: Definition and interpretation of M_1 and M_2 models

Aspect	M_1 **Model**	M_2 **Model**
Architecture	Independent encoder and classifier, processing each ion separately.	Incorporates a Coupler module to model interactions between ions, in addition to the encoder and classifier.
Interaction Modeling	None; assumes ions are independent and ignores inter-ion correlations.	Explicitly models inter-ion correlations using the Coupler module, capturing shared noise and cross-talk.
Training Strategy	Fully trained on labelled data, focusing on ion-specific features.	Trains the Coupler separately using unlabelled data while freezing the base model's parameters. Fine-tuning is performed using a combination of labelled and unlabelled data.
Strengths	Simpler and faster to train, effective for independent ion behaviours.	Handles correlated noise and cross-talk, improving classification accuracy and generalization in complex systems.
Limitations	Unable to capture inter-ion correlations, limiting performance in systems with significant interaction.	Increased complexity and computational demands require careful training of the Coupler to avoid overfitting.

Despite its strengths, the architecture faced several challenges. At times, performance would be saturated due to the limited amount of labelled data available or intrinsic noise in the dataset itself. In order to mitigate some of these problems, the Coupler was taken a step further along with unlabelled data in order to capture the hidden pattern. Other limitations have been class distribution biases, wherein class imbalance further results in imbalanced predictions. Simulated datasets were taken instead, where class distributions are balanced. Also, noise and cross-talk from neighbouring ions complicated the classification, and the ability of the Coupler to model interactions helped here.

It's also worth noticing that our evaluation framework purposefully relies on accuracy and sum distance to uniform rather than traditional metrics such as precision, recall, F1-scores, or ROC curves. This choice reflects our research priority of achieving correct overall state distribution rather than optimizing for specific error types in individual state classifications. Categorizing the entire multi-qubit state distribution is more important than accurately categorizing each ion state individually in our applications of trapped ion measurement. Our approach aligns with this priority by directly measuring how uniformly the model classifies unlabeled mixed states - a physical condition that our quantum system is guaranteed to meet. Building a model that classifies the entire 16-state space with the least degree of distributional bias is our primary goal, while precision/recall metrics would only give insight into binary-class performance on an individual ion level. The accuracy on labelled data serves as an essential foundation to ensure the model correctly learns the boundary states ($|0000\rangle$ and $|1111\rangle$), while the sum distance to uniform provides the critical evaluation of whether the model generalizes correctly across the complete quantum state space.

Chapter 6

Conclusions

The following thesis discusses the intersection of machine learning with trapped-ion quantum computing and ways in which one technology could be leveraged in order to overcome some of the difficult challenges facing quantum information processing. With the help of high control and stability of trapped-ion platforms, we created excellent-quality datasets that served as a basis for developing sophisticated machine-learning models. Such models enabled the accurate classification of ion states by relieving a set of key problems like noise, crosstalk, and scalability within quantum systems.

While providing decent coherence times together with precise manipulations of qubit states, trapped-ion devices finally offer a reliable experimental environment for creating the datasets. The hyperfine qubits, like the ion $^{171}Yb^{+}$, were used for reliable quantum information encoding, while state-of-the-art readout techniques were used to capture high-quality partially labelled datasets. This way, experimental quantum computing combined with machine learning techniques underlined the multidisciplinary character of quantum research.

A model that was developed in this thesis, adapted to the specificities of the trapped ion platform, was an FFNN with a unique architecture. This would improve the gain in generalization and accuracy of models toward classification tasks with the inclusion of ion-specific dense layers and interaction modelling through means of a coupler module. Other than these developments, several issues that should be addressed in further research were also noted. The small size of the labelled dataset constrained the performance of the model concerning bias reduction and a plateau in accuracy, among others.

While the dataset was well organized and the ML model was tuned perfectly for such a problem, inherent noise and probably crosstalk between proximate ions made the process of classification a lot more difficult than it really needed to be. These difficulties leave

open several paths of research for exploring data broadening, increased labelled dataset sizes, and other noise reduction methods that could help further improve machine learning results on quantum systems. The increases in future capabilities within quantum hardware, especially integrated photonic systems and scalable ion traps with good control, will help significantly to improve models for classification.

6.1 Outlook

In terms of future research, one obvious approach is to collect and incorporate additional data, maybe from various quantum computing devices, including improved trapped-ion quantum setups. This could boost our model's generalizability and accuracy. A larger dataset would allow the model to handle a broader range of circumstances, providing more resilient performance even as the complexity of quantum operations increases.

Another important consideration is investigating the integration of our ML model with specific hardware, such as an FPGA. We can accomplish real-time image classification by putting it straight onto an FPGA system—this could happen while the trapped-ion quantum device is performing its computations. The speed and model weight are crucial here: the classification process must keep up with quantum processing, delivering quick feedback and readouts.

At the same time, we must keep aware of the model's size. Keeping the model compact and well-tuned is critical since a bigger model would require more memory and longer processing times. If the classification takes too long, it will not be compatible with the rapid cycle times required by quantum computing. So there's a balance to strike between enhancing performance and accuracy through extra training data and retaining a lightweight footprint that enables quick and direct integration into the quantum computing operation.

Finally, if we can achieve that balance, our machine-learning technique might serve as a strong foundation for more advanced, real-time, closed-loop quantum computing research. The combination of precise machine learning inference, fast FPGA execution, and tight coupling with trapped-ion quantum devices has the potential to propel the industry ahead, allowing for more efficient and responsive quantum data processing as these technologies advance.

References

[1] C.J. Ballance, T.P. Harty, N.M. Linke, M.A. Sepiol, and D.M. Lucas. High-fidelity quantum logic gates using trapped-ion hyperfine qubits. *Physical Review Letters*, 117(6), August 2016.

[2] Stephane Beauregard. Circuit for shor's algorithm using 2n+ 3 qubits. *arXiv preprint quant-ph/0205095*, 2002.

[3] David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv (Cornell University)*, January 2019.

[4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv (Cornell University)*, January 2019.

[5] S. M. Brewer, J.-s. Chen, A. M. Hankin, E. R. Clements, C. W. Chou, D. J. Wineland, D. B. Hume, and D. R. Leibrandt. Al+27 quantum-logic clock with a systematic uncertainty below 10. *Physical Review Letters*, 123(3), July 2019.

[6] Kenneth R Brown, John Chiaverini, Jeremy M Sage, and Hartmut Häffner. Materials challenges for trapped-ion quantum computers. *Nature Reviews Materials*, 6(10):892–905, 2021.

[7] Lowell S. Brown. Quantum motion in a paul trap. *Physical Review Letters*, 66(5):527–529, February 1991.

[8] Colin D. Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M. Sage. Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, 6(2), May 2019.

[9] Alice Heather Burrell. *High Fidelity Readout of Trapped Ion Qubits*. Oxford, 2010.

[10] Charles F. Cadieu, Ha Hong, Daniel L. K. Yamins, Nicolas Pinto, Diego Ardila, Ethan A. Solomon, Najib J. Majaj, and James J. DiCarlo. Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS Computational Biology*, 10(12):e1003963, December 2014.

[11] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[12] Justin E. Christensen, David Hucul, Wesley C. Campbell, and Eric R. Hudson. High-fidelity manipulation of a qubit enabled by a manufactured nucleus. *Npj Quantum Information*, 6(1), April 2020.

[13] Juan I Cirac and Peter Zoller. Quantum computations with cold trapped ions. *Physical review letters*, 74(20):4091, 1995.

[14] Balázs Csáji. *Approximation with Artificial Neural Networks*. PhD thesis, Eötvös Loránd University, 06 2001.

[15] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals and Systems*, 2(4):303–314, December 1989.

[16] Andrew J. Daley, Martin M. Boyd, Jun Ye, and Peter Zoller. Quantum computing with alkaline-earth-metal atoms. *Physical Review Letters*, 101(17), October 2008.

[17] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe. Demonstration of a small programmable quantum computer with atomic qubits. *Nature*, 536(7614):63–66, August 2016.

[18] Hans Dehmelt. Experiments with an isolated subatomic particle at rest. *Reviews of Modern Physics*, 62(3):525–530, July 1990.

[19] Hans G. Dehmelt. Monoion oscillator as potential ultimate laser frequency standard. *IEEE Transactions on Instrumentation and Measurement*, IM–31(2):83–87, June 1982.

[20] David P. DiVincenzo and Ibm. The physical implementation of quantum computation. *arXiv.org*, February 2000.

[21] Ulrich Drepper. What every programmer should know about memory. *Red Hat, Inc*, 11(2007):2007, 2007.

[22] J. P. Gaebler, T. R. Tan, Y. Lin, Y. Wan, R. Bowler, A. C. Keith, S. Glancy, K. Coakley, E. Knill, D. Leibfried, and D. J. Wineland. High-fidelity universal gate set for $^{9}Be^{+}$ ion qubits. *Phys. Rev. Lett.*, 117:060505, Aug 2016.

[23] Michael R Geller and Mingyu Sun. Toward efficient correction of multiqubit measurement errors: pair correlation method. *Quantum Science and Technology*, 6(2):025009, December 2020.

[24] Joshua J. Goings, Alec White, Joonho Lee, Christofer S. Tautermann, Matthias Degroote, Craig Gidney, Toru Shiozaki, Ryan Babbush, and Nicholas C. Rubin. Reliably assessing the electronic structure of cytochrome p450 on today's classical computers and tomorrow's quantum computers. *Proceedings of the National Academy of Sciences*, 119(38), September 2022.

[25] Ian Goodfellow. Deep learning, 2016.

[26] Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79(2):325–328, July 1997.

[27] Ali Güneş, Habil Kalkan, and Efkan Durmuş. Optimizing the color-to-grayscale conversion for image classification. *Signal Image and Video Processing*, 10(5):853–860, October 2015.

[28] H Haffner, C Roos, and R Blatt. Quantum computing with trapped ions. *Physics Reports*, 469(4):155–203, October 2008.

[29] Michael H Holzscheiter. Ion-trap quantum computation. *Los Alamos Science*, 27:264–283, 2002.

[30] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[31] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[32] David H. Hubel and Torsten N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154, 1962.

[33] David Hucul, Justin E. Christensen, Eric R. Hudson, and Wesley C. Campbell. Spectroscopy of a synthetic trapped ion qubit. *Physical Review Letters*, 119(10), September 2017.

[34] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv (Cornell University)*, January 2016.

[35] Junho Jeong, Changhyun Jung, Taehyun Kim, and Dongil "Dan" Cho. Using machine learning to improve multi-qubit state discrimination of trapped ions from uncertain emccd measurements. *Opt. Express*, 31(21):35113–35130, Oct 2023.

[36] Junho Jung. Emccd-based multi-qubit state discrimination of trapped ions enhanced by machine learning, 2023.

[37] Ilia Khomchenko, Andrei Fionov, Artem Alekseev, Daniil Volkov, Ilya A. Semerikov, Nikolay N. Kolachevsky, and Aleksey K. Fedorov. Detection states of ions in a paul trap via conventional and quantum machine learning algorithms. *arXiv (Cornell University)*, December 2024.

[38] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *arXiv (Cornell University)*, January 2014.

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv (Cornell University)*, January 2013.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[41] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *arXiv*, 2018.

[42] K. A. Landsman, Y. Wu, P. H. Leung, D. Zhu, N. M. Linke, K. R. Brown, L. Duan, and C. Monroe. Two-qubit entangling gates within arbitrarily long chains of trapped ions. *Physical Review. A/Physical Review, A*, 100(2), August 2019.

[43] Thomas LauprÃªtre, Lucas Groult, Bachir Achi, Michael Petersen, Yann KersalÃ©, Marion Delehaye, and ClÃ©ment LacroÃte. Absolute frequency measurements of transitions in ytterbium. *OSA Continuum*, 3(1):50, January 2020.

[44] Joonho Lee, Dominic W. Berry, Craig Gidney, William J. Huggins, Jarrod R. McClean, Nathan Wiebe, and Ryan Babbush. Even more efficient quantum computations of chemistry through tensor hypercontraction. *PRX Quantum*, 2(3), July 2021.

[45] P J Lee, K-a Brickman, L Deslauriers, P C Haljan, L-m Duan, and C Monroe. Phase control of trapped ion quantum gates. *Journal of Optics B Quantum and Semiclassical Optics*, 7(10):S371–S383, September 2005.

[46] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland. Quantum dynamics of single trapped ions. *Reviews of Modern Physics*, 75(1):281–324, March 2003.

[47] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland. Quantum dynamics of single trapped ions. *Reviews of Modern Physics*, 75(1):281–324, March 2003.

[48] Shayan Majidy, Christopher Wilson, and Raymond Laflamme. *Building Quantum Computers: A Practical Introduction*. Cambridge University Press, 2024.

[49] Umberto Michelucci. An introduction to autoencoders. *arXiv (Cornell University)*, January 2022.

[50] T. Monz, P. Schindler, J. T. Barreiro, M. Chwalla, D. Nigg, W. A. Coish, and R. ... Blatt. 14-qubit entanglement: Creation and coherence. *Physical Review Letters*, 106(13):130506, 2011.

[51] Sainath Motlakunta. Developing a large-scale, programmable trapped ion quantum simulator with in situ mid-circuit measurement and reset, September 2023.

[52] A. H. Myerson, D. J. Szwer, S. C. Webster, D. T. C. Allcock, M. J. Curtis, G. Imreh, J. A. Sherman, D. N. Stacey, A. M. Steane, and D. M. Lucas. High-fidelity readout of trapped-ion qubits. *Physical Review Letters*, 100(20), May 2008.

[53] Klaus Mølmer and Anders Sørensen. Multiparticle entanglement of hot trapped ions. *Physical Review Letters*, 82(9):1835–1838, March 1999.

[54] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[55] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv (Cornell University)*, January 2015.

[56] John Preskill. Fault-tolerant quantum computation. *arXiv (Cornell University)*, January 1997.

[57] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv (Cornell University)*, January 2014.

[58] M.S. Robbins and B.J. Hadwen. The noise performance of electron multiplying charge-coupled devices. *IEEE Transactions on Electron Devices*, 50(5):1227–1232, 2003.

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

[60] F. Schmidt-Kaler, H. Häffner, M. Riebe, S. Gulde, G. P. T. Lancaster, T. Deuschle, and R. ... Blatt. Realization of the cirac-zoller controlled-not quantum gate. *Nature*, 422:408–411, 2003.

[61] Ferdinand Schmidt-Kaler, Hartmut Häffner, Mark Riebe, Stephan Gulde, Gavin P. T. Lancaster, Thomas Deuschle, Christoph Becher, Christian F. Roos, Jürgen Eschner, and Rainer Blatt. Realization of the cirac–zoller controlled-not quantum gate. *Nature*, 422(6930):408–411, March 2003.

[62] Alireza Seif, Kevin A Landsman, Norbert M Linke, Caroline Figgatt, C Monroe, and Mohammad Hafezi. Machine learning assisted readout of trapped-ion qubits. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 51(17):174006, aug 2018.

[63] Alireza Seif, Kevin A Landsman, Norbert M Linke, Caroline Figgatt, C Monroe, and Mohammad Hafezi. Machine learning assisted readout of trapped-ion qubits. *Journal of Physics B Atomic Molecular and Optical Physics*, 51(17):174006, July 2018.

[64] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[65] G Shu, M R Dietrich, N Kurz, and B B Blinov. Trapped ion imaging with a high numerical aperture spherical mirror. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 42(15):154005, jul 2009.

[66] G. Shu, N. Kurz, M. R. Dietrich, and B. B. Blinov. Efficient fluorescence collection from trapped ions with an integrated spherical mirror. *Physical Review A*, 81(4), April 2010.

[67] Gang Shu, Chen-Kuan Chou, Nathan Kurz, Matthew R Dietrich, and Boris B Blinov. Efficient fluorescence collection and ion imaging with the âtackâ ion trap. *JOSA B*, 28(12):2865–2870, 2011.

[68] R. Srinivas, S. C. Burd, H. M. Knaack, R. T. Sutherland, A. Kwiatkowski, S. Glancy, E. Knill, D. J. Wineland, D. Leibfried, A. C. Wilson, D. T. C. Allcock, and D. H.

Slichter. High-fidelity laser-free universal control of trapped ion qubits. *Nature*, 597(7875):213, September 2021.

[69] A. Steane. The ion trap quantum information processor. *Applied Physics B*, 64(6):623–643, June 1997.

[70] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *arXiv (Cornell University)*, January 2016.

[71] Anders Sørensen and Klaus Mølmer. Quantum computation with ions in thermal motion. *Physical Review Letters*, 82(9):1971–1974, March 1999.

[72] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv (Cornell University)*, January 2017.

[73] Yi Hong Teoh. Machine learning and optimization techniques for trapped-ion quantum simulators, August 2021.

[74] S. L. Todaro, V. B. Verma, K. C. McCormick, et al. State readout of a trapped ion qubit using a trap-integrated superconducting photon detector. *Phys. Rev. Lett.*, 126(010501), January 6 2021.

[75] Pengfei Wang, Chun-Yang Luan, Mu Qiao, Mark Um, Junhua Zhang, Ye Wang, Xiao Yuan, Mile Gu, Jingning Zhang, and Kihwan Kim. Single ion qubit with estimated coherence time exceeding one hour. *Nature Communications*, 12(1), January 2021.

[76] Ying-Xiang Wang. Systematic investigations on ion dynamics with noises in paul trap. *Journal of Physics a Mathematical and Theoretical*, 56(46):465302, October 2023.

[77] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J.-s. Chen, N. C. Pisenti, M. Chmielewski, C. Collins, K. M. Hudek, J. Mizrahi, J. D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. M. Ducore, A. Blinov, S. M. Kreikemeier, V. Chaplin, M. Keesan, C. Monroe, and J. Kim. Benchmarking an 11-qubit quantum computer. *Nature Communications*, 10(1), November 2019.

[78] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):8934–8954, November 2022.

[79] Yifei Zhang. A better autoencoder for image: Convolutional autoencoder. In *ICONIP17-DCEC. Available online: http://users. cecs. anu. edu. au/Tom. Gedeon/-conf/ABCs2018/paper/ABCs2018_paper_58. pdf (accessed on 23 March 2017)*, 2018.

[80] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.