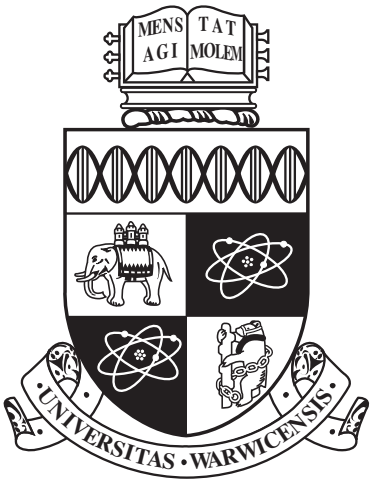


Comparisons of Machine Learning Approaches to the 3D Anderson Model of Localisation

by

Quangminh Bui-Le

Thesis

Submitted to the University of Warwick

for the degree of

Master of Science (by Research) in Physics

Department of Physics

December 2023

THE UNIVERSITY OF
WARWICK

Contents

Acknowledgments **iii**

Declarations **v**

Abstract **vi**

Abbreviations **vii**

Chapter 1 Introduction **1**

1.1 Condensed Matter Physics 1

1.2 A History of Artificial Intelligence 3

1.3 A Brief History of Machine Learning in relation to Artificial Intelligence 4

Chapter 2 Theory and Methodology **8**

2.1 The Anderson model of localisation 8

2.2 The Process of Deep Learning as a subset of Machine Learning . . . 9

2.3 The Application of Convolutional Neural Networks for drawing information from neighbouring data points 11

2.4 The model structures that will be used during training 12

2.5 How the data was generated and prepared for training 13

Chapter 3 Categorisation of Extended and Localised Phases **22**

3.1 Training with the Ohtsuki network for $L = 40$ with Ψ values 22

3.2 Training with the ResNet18 network for $L = 40$ with Ψ values . . . 23

3.3 Comparison of training with Ψ values against $|\Psi|^2$ values 25

3.4 Effects of training using AdaDelta operator compared to Adam operator 25

Chapter 4 Categorisation based on disorder of state **28**

4.1 Training using Images 28

4.1.1 Comparison of effect of L on training of model 28
4.1.2 Comparison of effect of s on training of model 30
4.2 Training using numerical $|\Psi|^2$ values 32
4.2.1 Training using $L = 20$ data 32
4.2.2 Training using $L = 40$ data 34
4.2.3 Training using $L = 100$ data 36
4.2.4 Comparisons of the effects of varying L on $|\Psi|^2$ value training 37
4.3 Training using numerical Ψ values 39
4.3.1 Training using $L = 20$ data 39
4.3.2 Training using $L = 40$ data 39
4.3.3 Training using $L = 100$ data 40
4.3.4 Comparisons of the effects of varying L on Ψ value training . 42

Chapter 5 Conclusions **45**
5.1 Comparisons of training using images against numerical values . . . 45
5.2 Comparisons of training using $|\Psi|^2$ versus Ψ 46
5.3 Overall effectiveness of models produced through this method 48
5.4 Potential points of improvement and future additions to consider . . 49

Appendix A Full confusion matrices **52**

Acknowledgments

I would like the give the majority of my thanks to my supervisor, Dr Rudolf Römer. It was thanks to his teaching that I was even able to complete this work and produce these results. I would like to thank him for always being willing to take questions and offer suggestions as those helped guide me to the creation of this thesis. All the foundations of my machine learning knowledge that I will use in the future are thanks to him.

I would also like to sincerely thank my colleague, Djena Bayo. Her expertise in machine learning through her own work was very helpful in solving the problems I had run into in trying to construct a working model. I would like to thank her for providing the results in Chapter 4.1 of this thesis by training the models for image data of system sizes 20 and 40 as well as the image data for system size 100 at image size 100 and 200.

I thank the Warwick's Scientific Computing Research Technology Platform for providing the computing time and technical support needed in a expedient manner. Without their hard work to maintain the computing systems it would have been impossible to get the work done for this project in this time frame.

I extend my thanks to other members of the Disordered Quantum Systems Research group here at Warwick, Yilun Gao and Syl Shaw. The discussions of our progress in our separate projects was helpful in maintaining a regular pace of work. The help they also provided when I was stuck on issues of PYTHON package instal-

lation and other software issues I am very grateful for. My thanks go especially to Yilun Gao, who I met up with in the office on a daily basis and provided a familiar face I could simply share my day to day with.

I also sincerely thank my family, my father Huy Bui, my mother Kim Ledinh and my sisters Kimvy and Kimmy. They have provided the emotional support upon which stands all my efforts. The love and support they have given me through this whole project and the rest of my life have provided me the ability to dedicate my time into this research with far less worries.

Declarations

I declare that the work provided in this thesis has been produced by me on my work over 2022 to 2023 for the degree of Master of Science. This work has not been submitted wholly or in part to any other institution. The gramatical style used in this thesis follows that which is used for writing in refereed journals. As such it has been written in either the passive voice or the first person plural ("we"). In the majority of cases, the use of "we" can be replaced with "I".

The results shown in Chapter 4.1 for the models trained on images of system size 20, 40 at image size 100 and the models trained on images of system size 100 for images of size 100 and 200 have been provided by Djena Bayo. All figures and analysis of these results has been preformed by the author.

Parts of this work have been presented at the following conference:

Machine Learning the Anderson Transition (poster presentation), German Physical Society Meeting 2023, Technische Universität Dresden, Germany (March 2023)

Parts of this work are also being prepared for publication in peer-reviewed journals. Results in Chapter 4.2 and 4.3 will be provided to a publication on machine learning methods for categorising the Anderson model of localisation.

Abstract

Machine learning and deep learning are methods of statistical analysis using artificial intelligence to categorise complex data and extract deep information. Recently, such methods have gained popularity in areas of advanced physics for the categorisation of experimental data. In addition, data that has been obtained through numerical methods has also seen use, such as data from computing states for models of quantum mechanics. One prominent example of such a model is the Anderson model of localisation. In previous work, it has been shown that one key feature of the Anderson model of localisation is the presence of a metal-insulator transition which means that states of the extended phase show clear differences to the localised phase. In this thesis, we want to utilise machine learning methods to learn the differences between these states to attempt categorising between the two phases. We will also attempt a finer categorisation by categorising on specific disorder values of the Anderson model. We will investigate an image representation of the Anderson model eigenstates generated from the $|\Psi|^2$ values, a numerical representation of those $|\Psi|^2$ values and a numerical representation of the original Ψ values to find which representation is better for machine learning. We will then compare how changing the optimiser, the system size of the data, the size of the images and the format of the data affects the accuracy of the final model. We find that a categorisation of the Anderson model is possible. We find that it is relatively easy to categorise the Anderson model based on whether we are in the phase above or below the metal-insulator transition and that a categorisation of individual disorder points is more challenging. We also find that of the 3 representations we have, using the original Ψ values preforms the best for machine learning.

Abbreviations

CMP - Condensed Matter Physics

AI - Artificial Intelligence

ML - Machine Learning

DL - Deep Learning

Convnets - Convolutional Neural Network

ReLU - Rectified Linear Unit

HPC - High Performance Computing

Chapter 1

Introduction

1.1 Condensed Matter Physics

Condensed matter physics (CMP) was born in the 1960s as an expansion of solid-state physics to create a field of research that focused on the physical properties of both solids and liquids (Kohn [1999]). The field of CMP studies how the macroscopic properties of systems with high enough density that the interaction between the particles of the system effects these properties. Systems with such properties are called condensed systems. In analysing the properties of a given state, we find that some macroscopic properties are qualitatively similar across different systems. This systems are referred to collectively as a phase. Examples of such phases are the phases of atomic spatial structures, or solids and liquids.

Through the localization of electrons it is possible to adjust how a material is able to conduct electricity which is the primary method for altering the electrical properties of systems. One such method of localising electrons is through the use of the Mott phenomenon which uses strong Coloumb repulsion (Imada et al. [1998]). The key feature to this is the strong orbital and spin polarization within the unrestricted Hartree-Fock theory (Brandow [1992]). This produces a large potential energy difference due to the strong attractive self-exchange interaction of occupied orbitals (Brandow [1992]). This potential energy difference has a magnitude that is the same as the direct self interaction energy or the Hubbard U (Brandow [1992]). The band gap of the system is therefore of the order U (Brandow [1992]). Systems with insulating properties due to the Mott localisation are referred to as Mott insulators and differs from the Bloch–Wilson insulator (or band gap insulator) due to the consideration of the electron-electron Coloumb interactions in a Mott insulator

(Roy [2019]).

For this thesis we will be using the Anderson model of localisation. The method of Anderson localisation is through quantum reflections in the lattice causing the wave-function to interfere constructively as a result of the disorder of the state (Guan and Guan [2019]). This leads to the wave decaying and not scattering out of the lattice which produces localisation (Guan and Guan [2019]). Anderson states that if the disorder strength (W) is sufficiently high, the states will always become localised (Anderson [1958]). This can be proved in 1D (Thouless [1972]) and approximated in 2D using renormalisation group analysis (Abrahams et al. [1979]). In 3D, there exists a turning point that represents a transition from insulating behaviour to metallic behaviour (Guan and Guan [2019]). This phase transition occurs as a function of W which develops from random impurities in the system such as vacancies, impurity atoms and extended defects (Vojta [2019]). Disorder can arise naturally in a quantum system as a result of the preparation process and can also be artificially introduced into the system. For high W, there will be enough decay in the wavefunction for localisation to occur but below a critical disorder strength W_c (Schenk et al. [2008b]) the interference of the wavefunctions in the state may not be sufficient to lead to localisation (Guan and Guan [2019]). The insulator that forms from Anderson localisation is called an Anderson insulator (Hibat-Allah et al. [2023]).

The metal-insulator transition (MIT) as a result of Anderson localisation is an example of a quantum phase transition which displays a phase transition at absolute zero temperature (Vojta [2003]). This is of interest in the field of CMP as the behaviour around the quantum critical point of the phase transition can influence properties over a wide range of the phase diagram (Vojta [2003]). The physical properties of the quantum fluctuations also differ greatly from the classical fluctuations for a finite-temperature phase transition which means the dynamics of the phase around the quantum critical point will require new theorems to explain, leaving potential for new developments in the field of CMP (Vojta [2003]). The MIT gives also lots of complexity to the intermediate regime which leads to large change in the physical properties of the system for small variations in other variables (Dobrosavljevic et al. [2012]). Understanding the behaviour of materials around the transition is important as it can inform methods to alter systems to have desired electrical properties (Edwards et al. [1998]).

It is important to be able to identify different phases. As systems display large variation in physical properties either side of a phase transition, identifying these properties will allow analysis of the phase. For instance, to distinguish between the metal and insulator states of the MIT, we could try to identify whether the electrons of the system are extended or localised within the Anderson model. For systems that are very much in the extended or localised domains the difference can be identified visually (Figure 1.1) but when the systems get closer to the critical point the distinction is harder to make (Figure 1.2). To solve this problem we can apply the branch of artificial intelligence (AI) that is machine learning (ML). Using ML algorithms, an AI can find patterns within systems that humans cannot easily define. Once an ML model has been formed using these patterns, it can then be used to make predictions on other unknown data by spotting such patterns. Here we will create categorisation models for the Anderson model using 2 datasets, one for determining the phase of a given state as localised or extended and one for determining the specific W of a given state. These ML models will be trained to take different forms of input, starting with an image recognition model trained on visual representations of the eigenstates then moving onto numerical representations, using the $|\Psi|^2$ values the images were generated from and then moving onto the original Ψ that were generated.

1.2 A History of Artificial Intelligence

AI originates from three sources (Haenlein and Kaplan [2019]). The idea of a machine that could make decisions for itself was first introduced in 1942 by author Isaac Asimov in his short story Runaround (Asimov [1942]). This story revolves around the creation of a robot and the three rules that define its behaviour (Asimov's three laws of robotics) capturing the idea of creating a mechanical system which acts independent of humans. Also during 1942, Alan Turing was working on a machine capable of decrypting the enigma code (Haenlein and Kaplan [2019]). This lead to the creation of The Bombe, what is considered one of the first electro-mechanical computers (Haenlein and Kaplan [2019]). The Bombe's ability to preform calculations in a way that was impossible for humans, leading Turing to further research into the ability of machines to preform calculations, finalising in the production of his 1950 article "Computing Machinery and Intelligence" (Turing [1950]) in which he discusses the methods to create and test "intelligent machines" setting the benchmark for the creation of AI and pioneering the Turing Test. Then in 1956, Marvin Minsky and John McCarthy ran the Dartmouth Summer Research Project on Arti-

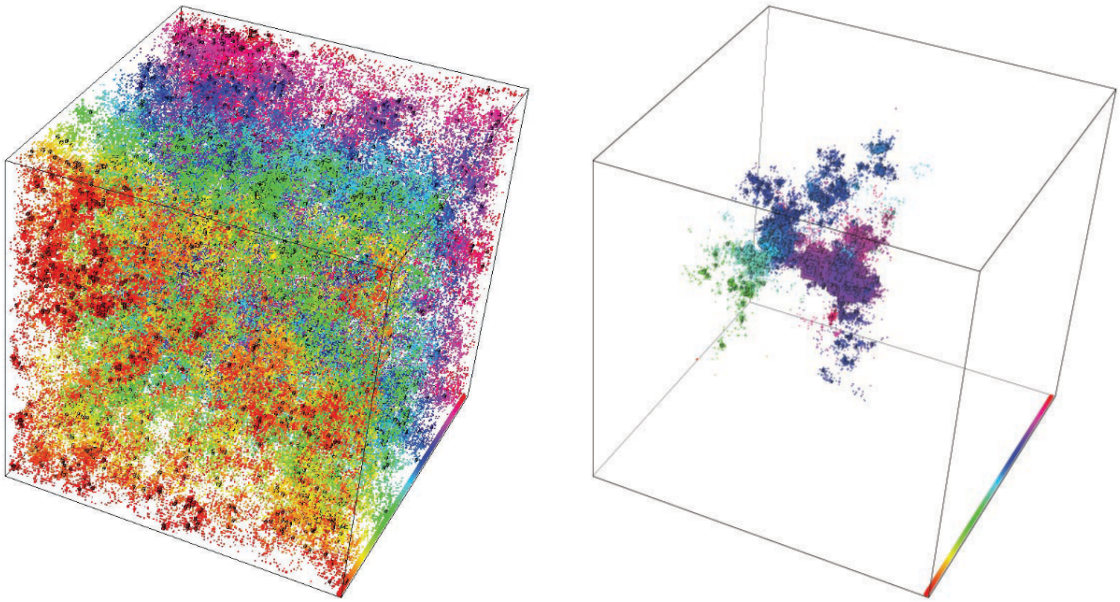
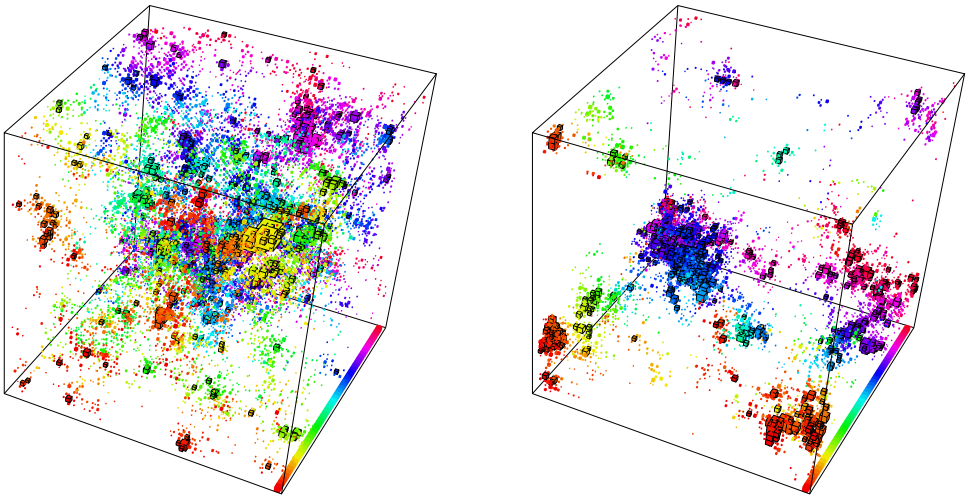


Figure 1.1: Visual representations of Anderson model eigenfunctions. Each point with $|\Psi|^2 > \frac{1}{N^3}$ has been plotted as a cube of size $|\Psi|^2 N$ and those with $|\Psi|^2 N > \frac{1}{\sqrt{1000}}$ have a black border. A colour-map has been applied to the z-axis to help gauge the position of points. The left plot represents states $W \ll W_c$ which are very much in the extended phase. The right plot represents states $W \gg W_c$ and thus are in the localised phase. This image has been sourced from Schenk et al. [2008a].

ficial Intelligence (McCarthy et al. [2006]), reuniting the founding fathers of AI and beginning the advancement of AI to what we see today. It was also when the term "Artificial Intelligence" was first coined (Haenlein and Kaplan [2019]).

1.3 A Brief History of Machine Learning in relation to Artificial Intelligence

ML is a subsection of AI which focuses on trying to automate the creation of the instructions that allow the system to make decisions. While AI is able to take a given input and compare it to a predefined set of human instructions to produce an output, ML can take a set of inputs and expected outputs provided to it and applies a transformation to the inputs such that the machine can differentiate between the expected outputs. This removes the need for humans to define how the machine is supposed to get from an input to the expected output – a task that gets quite complicated when the differences between inputs are less defined. The system improves



(a) $W < W_c$ (b) $W > W_c$

Figure 1.2: Visual representations of Anderson model eigenfunctions for W close to W_c, in the range of W values we want to categorise between. These images have been constructed using the same method as the plots in figure 1.1.

on this process by "training" itself – varying the transformation in order to find one that best predicts the whole set and will thus best predict new inputs - this is called generalization (Chollet [2017]).

The concept of ML as we consider it today originated in 1957 (Fradkov [2020]). The term was introduced by Frank Rosenblatt who created a machine to recognise letters of the alphabet, using ideas based on the behaviour of the human nervous system. This machine was named a "perceptron", named as such due to its model of learning carrying similarities to human/animal learning. Rosenblatt would study the behaviour of the perceptron and discuss how it was possible for a properly designed perceptron to be able to make meaningful classification without the parameters of the classification being defined by its creator (Rosenblatt [1958]) - the basis of ML. Though the mechanics behind the learning of perceptrons was first discussed by Rosenblatt, the more well known theorem was introduced by Novikoff which states that, if a solution exists, a perceptron will train in finite time proportional to $\frac{1}{\gamma^2}$ where γ is the linear separation of the dataset (Novikoff [1962]). This insight is utilised in the modern day as the basis for training ML models.

Different machine learning algorithms exist to construct the predictive model (Pugliese et al. [2021]). They can be divided into 4 categories: *Supervised* learning, *Unsu-*

pervised learning, *Semi-supervised* learning and *Reinforcement* learning (Pugliese et al. [2021]). *Supervised* learning involves the model creator defining a set of inputs and the expected corresponding output (Cunningham et al. [2008]). The ML algorithm will then attempt to modify the model function in order to minimise the error between its own output and the defined output (Cunningham et al. [2008]). One advantage of this method is that by defining the expected output we can determine exactly what categories we want the model to classify data into (Pugliese et al. [2021]). In doing so, we rely on humans to be able to define the expected outputs for the initial data and as such this method would not work for data that we cannot find the categories for already (Pugliese et al. [2021]). By contrast, *Unsupervised* learning methods do not provided any information about the expected outputs to the model (Ghahramani [2004]). This allows the model to sort the data based on any features that it can discover from the training data (Ghahramani [2004]). As such, one of the advantages of an unsupervised learning algorithm is that it may be able to pick up on features that a human could not detect (Pugliese et al. [2021]). In turn, by not defining the desired outputs, the outputs generated by the model may be meaningless (Pugliese et al. [2021]). *Semi-supervised* learning combines training with labelled and unlabelled data to produce a model that predicts better than a purely supervised model (Chapelle et al. [2006]). This only works if the provided unlabelled data contains information that can be used to infer features of the labelled data (Chapelle et al. [2006]). *Reinforcement* learning focuses on the outputs the machine produces. In reinforcement learning, the machine is tasked with making random actions that affects the environment the machine is in (Ghahramani [2004]). The machine is then provided a "reward" based on how well its actions lead to the desired output which the machine will try to maximise (Ghahramani [2004]). One advantage of this is that reinforcement learning can solve problems that cannot be solved by other ML techniques (Pugliese et al. [2021]). Reinforcement learning methods will require more data and computing time than other ML techniques which must be considered when deciding which ML technique to use (Pugliese et al. [2021])

We are using supervised learning methods in order to generate models capable of categorising states based on phase or disorder. ML categorisation is where we train a model by providing a set of inputs and the group we want them to be sorted into (Chollet [2017]). These set of groups are the categories of the data, C with each input having an expected output category, c. The end result will be a model that can take an input and output a set of values corresponding to how well the model thinks the input matches with the categories it has trained on (Chollet [2017]). One

alternative method of supervised learning is a regressional model (Chollet [2017]). A regressional model once trained will take an input and output a value between zero and one corresponding to how well this input aligns with the extremes of the data the model was trained on (Chollet [2017]). This value can then be mapped to the range of outputs to determine what value the input corresponds to (Chollet [2017]). As such, regressional methods are preferred for inputs that take on a continuous range of values whereas categorisation methods are ideal for inputs that take on discrete values (Chollet [2017]).

ML has gained popularity recently, seeing usage in many applications in the field of Physics (Tanaka et al. [2021]). ML categorisation methods have been used to identify phase transitions in states. For example, in the percolation model it has been shown that it is possible to identify physical features of the model such as the order parameter and the permeability (Cheng et al. [2021]) as well as to categorise disorder states (Bayo et al. [2022]). In the 2D ferromagnetic Ising model, ML methods have been used to categorise phases as a function of temperature (Naravane and Mathur [2023]) and for the Rydberg atom array, unsupervised ML algorithms have been used to identify phase boundaries (Patel et al. [2022]). ML methods have also been shown to be able to identify order parameters for conventionally ordered phases without knowing the energy or locality conditions of the Hamiltonian as well as having the potential to represent ground state wavefunctions (Carrasquilla and Melko [2017]). Another recent use of ML methods is solving fluid dynamics problems, such as the viscous Burgers equation (Umeano et al. [2023]) and ML has been successful in detecting topological order in quantum systems (Hibat-Allah et al. [2023]) and can also be used to reduce complexity in quantum many-body problems (Carleo and Troyer [2017]).

Chapter 2

Theory and Methodology

2.1 The Anderson model of localisation

Anderson localisation occurs when electrons are capable of tunnelling between neighbouring sites as quantum waves while also interfering with themselves. When the wavefunction is defined in 3D, the variation in disorder strength, W, corresponds with a transition in the transport properties. In the 3D case, a low disorder means the waves are extended. If the disorder is sufficiently high the wave amplitudes destructively interfere creating a localised system. The transition between extended and localised states also corresponds to the metal-insulator phase transition – the system has metallic behaviour while extended and insulating behaviour while localised.

To understand and interpret the Anderson model we need the Hamiltonian, $\hat{H}$, of the system. We can define the Anderson Hamiltonian as a tight-binding model with nearest-neighbour hopping and random on-site energy (Girvin and Yang [2019])

$$\hat{H} = W \sum_{n=1}^{L^d} \epsilon_n \hat{c}_n^\dagger \hat{c_n} + t \sum_{<n,m>} (\hat{c}_n^\dagger \hat{c_m} + h.c.), \tag{2.1}$$

where n and m are the indices of positions, d is the number of dimensions we are working with, ϵ_n is the on-site energy and $\hat{c}$ is the annihilation operator. In this case, the eigenvalues of the Hamiltonian give the quantum mechanical energy levels and the wavefunctions of the quantum state are defined by the eigenvectors of the real symmetric matrix of the Hamiltonian operator. When we set $t = 1$ for a cubic lattice defined by the set of vectors $\mathbf{r} = n_x\mathbf{e}_x + n_y\mathbf{e}_y + n_z\mathbf{e}_z$ with integers $1 \leq n_x, n_y, n_z \leq N$ where N is the volume of the lattice, we find that a phase transition occurs at a

critical disorder $W_c = 16.5$ (Schenk et al. [2008a])

We can form the Schrödinger equation to be solved (Girvin and Yang [2019])

$$W\epsilon_{\mathbf{r}}\psi(\mathbf{r}) - \sum_{\alpha=x,y,x} \psi(\mathbf{r}+\mathbf{e}_\alpha) + \psi(\mathbf{r}-\mathbf{e}_\alpha) = E\psi(\mathbf{r}), \tag{2.2}$$

and then apply equation (2.1) to $d = 3$ by computing the eigenvalues and eigenstates ψ of the Anderson Hamiltonian according to equation (2.2). To solve for the eigenvalues of the Anderson Hamiltonian many different algorithms have been introduced. For example, the Lanczos algorithm and Cullum-Willoughby implementation was developed in the mid-1980s (Cullum and Willoughby [1985]) for computing the eigenvalues and eigenvectors of symmetric matrices. The algorithm we chose to use was the Jacobi-Davidson method (Geus [2002]). The Jacobi-Davidson method has been shown to have faster calculation time compared to implementations of the Lanczos algorithm while maintaining similar memory costs (Geus [2002]). The Jacobi-Davidson method requires solving the correction equation (Equation (2.3))

$$(I - uu^T)(A - \theta I)(I - uu^T)z = -r \tag{2.3}$$

such that

$$z = (I - uu^T) \tag{2.4}$$

where (u, θ) is the given approximate eigenpair and $r = Au - \theta u$ is the associated residual. The search space, $V_k = [v_1, ..., v_k]$ is then expanded such that z is reorthogonalized with respect to $[v_1, ..., v_k]$ and the new approximate eigenpair is computed from the Ritz approximation, $[V_k, z]^T A[V_k, z]$ (Schenk et al. [2008a]).

2.2 The Process of Deep Learning as a subset of Machine Learning

Deep learning (DL) is an extension of the principles of ML (Chollet [2017]). DL expands on ML by applying successive layers of transformations in order to make a more meaningful representation of the data for prediction. The basic procedure of a deep neural network is to take an input vector, z, and produces an output, a, at each layer, passing the previous output of a layer to the input of the next (Mehta et al. [2019]). The full operation done at each layer is

$$a = \sigma(w \cdot z + b), \tag{2.5}$$

where w is the matrix of weights on the neuron, the trainable parameters that act on the matrix of inputs, b is the bias of the neuron, the linear trainable parameter and σ is the non-linear function applied to the neuron. For our ML algorithm we used the rectified linear unit function (ReLU) for the non-linear function. A loss function, E, is applied to the output of the ML algorithm to measure how much the final predictions vary from the expected results. For the loss of the created ML model to be minimised, we need to calculate the derivative of the loss function with respect to all weights in each layer. In order to compute the gradient at all these points, we can use the linked nature of a deep neural network to create a procedure called the back-propagation algorithm. Back-propagation takes the loss from our final output and uses it to calculate the loss at all previous layers with (Mehta et al. [2019])

$$\Delta_j^l = \frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial a_j^l}\sigma'(z_j^l)\,, \tag{2.6}$$

$$\Delta_j^l = \frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial b_j^l}\frac{\partial b_j^l}{\partial z_j^l} = \frac{\partial E}{\partial b_j^l}\,, \tag{2.7}$$

$$\begin{aligned}\Delta_j^l &= \frac{\partial E}{\partial z_j^l} = \sum_k \frac{\partial E}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= \sum_k \Delta_k^{l+1}\frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= (\sum_k \Delta_k^{l+1} w_{jk}^{l+1})\sigma'(z_j^l)\,, \end{aligned} \tag{2.8}$$

$$\frac{\partial E}{\partial w_{jk}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \Delta_j^l a_k^{l-1}, \tag{2.9}$$

where Δ is the loss, l is the layer number and j and k are neuron numbers. Back-propagation calculates the activation of all neurons in the input layer. The outputs of a previous neuron are read in order to be the input of the next neuron. We compute the value of the next layer layer, calculates the loss in the output using equation 2.6. We then use equation 2.7 to calculate the loss in all layers and finally use equations 2.8 and 2.9 to find the gradients for optimising. An optimiser function is used to determine how much the weights need to vary in order to minimise loss.

2.3 The Application of Convolutional Neural Networks for drawing information from neighbouring data points

We are expanding on the utilisation of DL using a variation of deep neural networks called convolutional neural networks (ConvNets). ConvNets are able to consider data around the point of analysis through the usage of patches - defined selections of data to consider around each point (Figure 2.1) (Chollet [2017]). This differs from a deep neural network which converts the full data into a 1D array and then applies every operation to every point. This allows ConvNets to analyse any potential information that can be drawn about a point based on it neighbours. For example, in the image of an eye, a pixel of the iris will be surrounded by other pixels of the iris. By using ConvNets in this way we can identify parts of the data that correspond to a certain feature. The ConvNets can apply changes to all the patches at once producing a representation of the data that emphasises features that the patches can identify. This representation is called a feature map which the model uses to identify areas that are similar to each other which can then be used to identify patterns that may exist in the data. By using different sizes and shapes for our patches, we can produce different feature maps which pick up on different details. In applying these patches, we will increase the complexity of the input to the DL algorithm. As such we can expect an increased run time. Therefore, it is important to only consider ConvNets when it is apparent that neighbouring data points contain useful information.

One factor that influences the training of a model using a ConvNets is that the patches can only be centered on data with enough surrounding data points in order to fill the patch. As such, applying a patch to some input data map will produce a smaller output map (2.2). Padding is a process ConvNets can apply to account for this. Padding adds additional empty columns and rows to the data so that all data points from the initial input map can have a patch applied to them. This will then produce an output map that has the same size as the unpadded input (2.2).

We also do not have to take every possible patch from the input datamap. One other parameter we can define for our ConvNets layer is the stride. The stride parameter tells the ConvNets layer how many points to skip before selecting the next point for the patch (Chollet [2017]). This will shrink the output feature map in a process called downsampling. Downsampling is important to reduce overfitting. Overfitting is when the model trains too closely to the data provided to the model.

This results in a model that predicts very well on the provided data but will predict poorly with new data. As such the model will be ineffective at categorising data as an overfit model will only accurately predict the data it trained with which is data we already know the category of.

2.4 The model structures that will be used during training

We will be looking at two network structures – the first is detailed in Ohtsuki and Mano [2020], utilising six convolution layers and three pooling layers (Figure 2.4). Pooling layers take each patch of the convolution and calculates a single value representing the whole patch (Nirthika et al. [2022]). This is another method of downsampling which we can use to reduce overfitting. More specifically, for this model we are using MaxPooling which is where we take the maximum value of the patch for the new feature map. The other common form of pooling is AveragePooling where we take the average of the patch for the new feature map. We will then look at the more dense ResNet18 network (He et al. [2015] for images and Tran et al. [2018] for numerical files), utilising four layers which each contain two convolution blocks made of a convolution layer, a batch normalisation layer and a ReLU layer (Figure 2.5). Layers 2 and 3 also include a downsample block which uses a convolution layer with a smaller kernel size and larger stride to reduce the data passing through the model and in turn reduce overfitting. The ResNet18 network also uses both MaxPooling and AveragePooling.

As we are doing multiple category categorisation, we will use the default implementation of categorical cross entropy from the PyTorch library to measure the loss of the model (Foundation [2021]). The categorical cross entropy loss, E_{cce}, is calculated using the function (Foundation [2021])

$$E_{\text{cce}}(\mathbf{x}, c) = -\log\left(\frac{\exp \mathbf{x}_c}{\sum_C \exp \mathbf{x}_C},\right) \tag{2.10}$$

where $\mathbf{x}$ is the final output tensor of the ML categorisation model, containing the probabilities $\mathbf{x}_C$ that the original input to the model matches each category, C. $\mathbf{x}_c$ is the probability that the model predicts that the original input corresponds with the true value defined by the user. Categorical cross entropy is the standard loss function for multiple category classification

For our optimisers, we will be using ADAM (Kingma and Ba [2017]) and ADADELTA (Zeiler [2012]) which are the two optimisers used by T.Ohtsuki (Ohtsuki and Mano [2020]). ADAM and ADADELTA are both predefined optimisers in the PyTorch module and have been implemented with the default learning rate, eps and weight decay parameters as defined in the PyTorch API documentation (Foundation [2021]). In order to find the ideal parameters for a ML algorithm, we preformed a hyperparameter test where we trained using randomly generated values for the optimiser parameters for a small number of epochs in order to find what values produce the best result in that small amount of training. From this we found that the default parameters in the API documentation are ideal.

2.5 How the data was generated and prepared for training

The data being used is Anderson model eigenstate data at energy $E = 0.0$ generated by Dr Römer following the methods explained in Schenk et al. [2008a] using the Jadamilu library in Fortran (Bollhöfer and Notay [2007]). This data is provided as a file of Ψ values where $\Psi = \psi(\mathbf{r})$. We have some scripts to convert this data into the format required for training the models. The first is a Fortran script which utilises the ImageMagick routine (community contributed Github repository [2023]) to convert the numerical values into an image into the .eps format which is then converted into a .jpg or .png format to be used in our training (Figure 1.1). The conversion must be done as by converting from .eps which is a vector format to .jpg or .png which are rasterised formats stores the image as pixels which is necessary for training using CNNs as we want to capture the information stored in data points and their neighbours which relies on all the points of the image being stored individually. When we generate the images for usage in the ML process we remove the black borders around the whole image as this black border contains no physics-based information (Figure 2.6). The second script uses the convert files into a binary format to compress files in a way that they can still be easily read and decoded by Python scripts. To do this we used the Pickle function in Python (Foundation [2023]). This is in order to save space on our work machines and improve data read in speed. We used three machines to develop our models.

We started with our local machines using the CPU to preform initial tests to make sure the method was feasible. We then moved work onto two high performance computing machines in order to use GPU training which will reduce the time it takes to

train. The two HPC machines we used are called Avon and Sulis. Avon has 16 GPU accelerated compute nodes, each with three Nvidia Quadro RTX6000 cards and approximately 22GB GPU RAM. Sulis has Dell PowerEdge R7525 GPU nodes each with three NVIDIA A100 40 GB RAM passively-cooled GPUs, two AMD EPYC 7742 (Rome) 2.25 GHz 64-core processors per node, 30 total nodes, 90 GPUs, 3,840 CPU cores, 512 GB DDR4-3200 RAM per node and 4 GB RAM per core.

We can divide the categories in the dataset in different ways. With two categories, we have split the dataset into data with $W < W_c$ and with $W > W_c$ which we denote as $\tau_{2,L}$ with L denoting the size of the cubic lattice the wavefunction has been calculated over, referred to as the system size. We also split the data into 17 categories to try to preform a more detailed classification of the W of the system. The categories we are using are $W = 15.0, 15.25, 15.5, 15.75, 16.0, 16.2, 16.3, 16.4, 16.5, 16.6, 16.7, 16.8, 17.0, 17.25, 17.5, 17.75$ and 18.0. The dataset for W classification has been denoted as $\tau_{17,L}$ when using the numerical Ψ and $|\Psi^2|$ data and as $\vartheta_{17,L}$ for datasets containing images. Each C in a τ and ϑ will contain 5000 samples except the C in $\tau_{17,100}$ which will only contain 500 samples.

While the datasets are separated by phase or disorder category in our file storage, we still need to attach labels to each file for our model training code to identify each sample. In order to do this, we first create a file listing all the files in each category. We then have a Python script which takes a list file for each category and appends a numerical tag to each item in that list. For convenience, we have labelled items with $W < W_c$ as 0 and $W > W_c$ as 1 and the categories for disorder as labelled ascending from 0 to 16 with $W = 15.0$ having label 0 and $W = 18.0$ having label 16. These lists with associated numerical tags are then collated into a single labels file which can then be passed into the model training code to generate the training, validation and test sets. Before the data is fed into the model it is randomly divided into a training, validation and test set in a ratio of $80 : 15 : 5$. The training set is the set the model uses to compare its predictions to the true values which the model will then use to adjust the weights to get the predictions to match the true values as best as possible. The validation set is used as a part of training to reduce overfitting. The validation set is input into the model after the training step to see how the model is preforming with an unseen dataset. If the model finds that the new set of weights is preforming more poorly with the validation set than the previous set of weights then the model will not save the new weights. This is to optimise the model such that it predicts best on unseen data. The test set is then

passed into the model after the whole training process is complete to determine how effective the final model is.

One other parameter we had to defined for our ML algorithm was the batch size. This parameter determines how the ML algorithm divides up the input τ into random blocks for it to train on. The size of these random blocks will determine how many samples the model can compare in one training step to try to identify features. We chose to use a batch size of 32 for all τ and ϑ except for when using numerical $\tau_{17,100}$. Each individual sample from $\tau_{17,100}$ contains 100^3 values and we found that even with the 40 GB GPU RAM provided Sulis, there was not enough RAM to train a model using a batch size of 32. As such for numerical $\tau_{17,100}$ we used a batch size of 4.

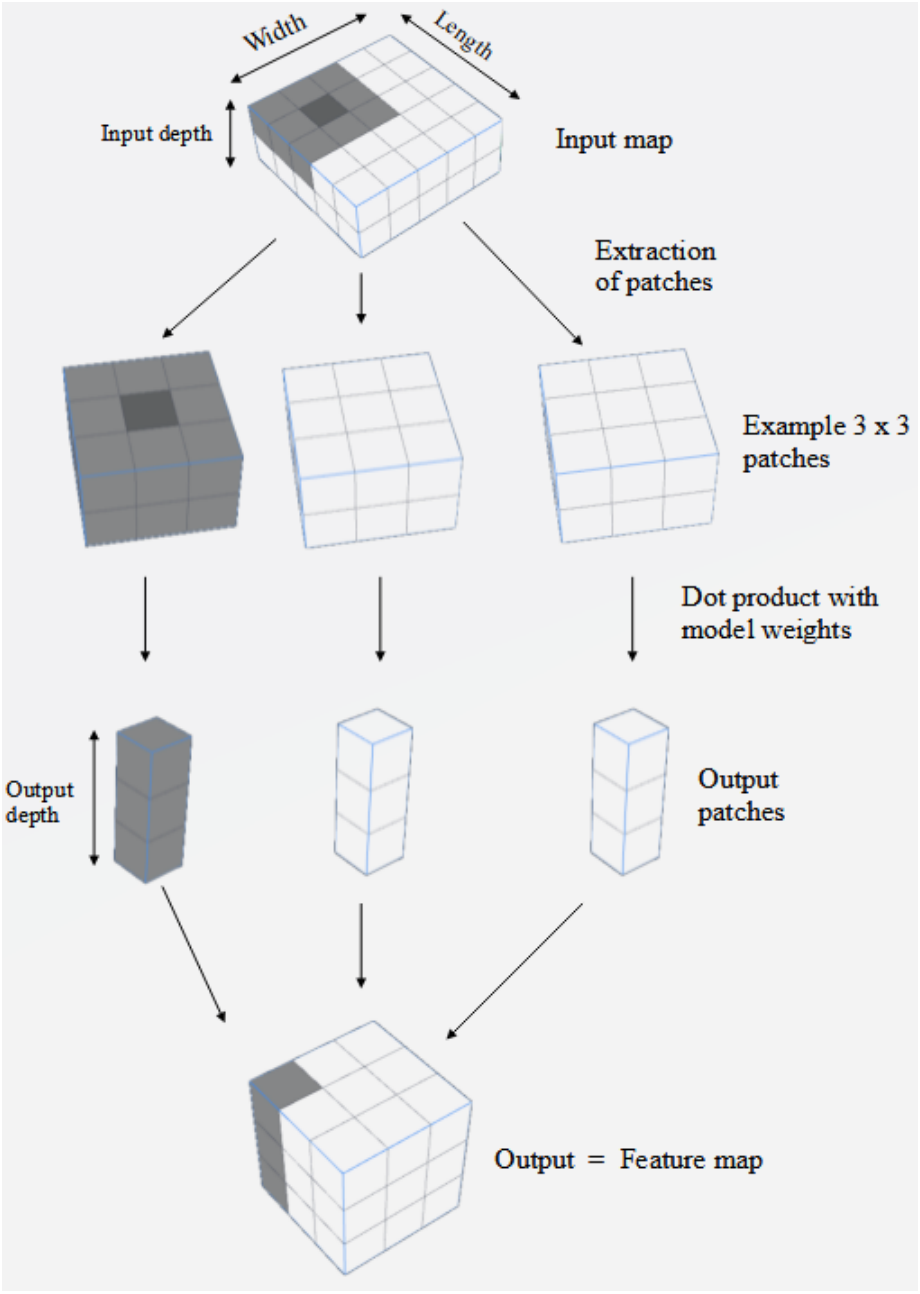


Figure 2.1: Visualisation of how a ConvNets applies patches to data and captures to information stored in neighbouring points. The shaded patch is to illustrate how a single patch is extracted and then transformed into a single output feature. Each patch that is extracted is multiplied with the weights of the network. The output feature map is formed by joining the output patches together such that each output keeps the position of the center of the input patch that created it relative to the other patches (Chollet [2017]).

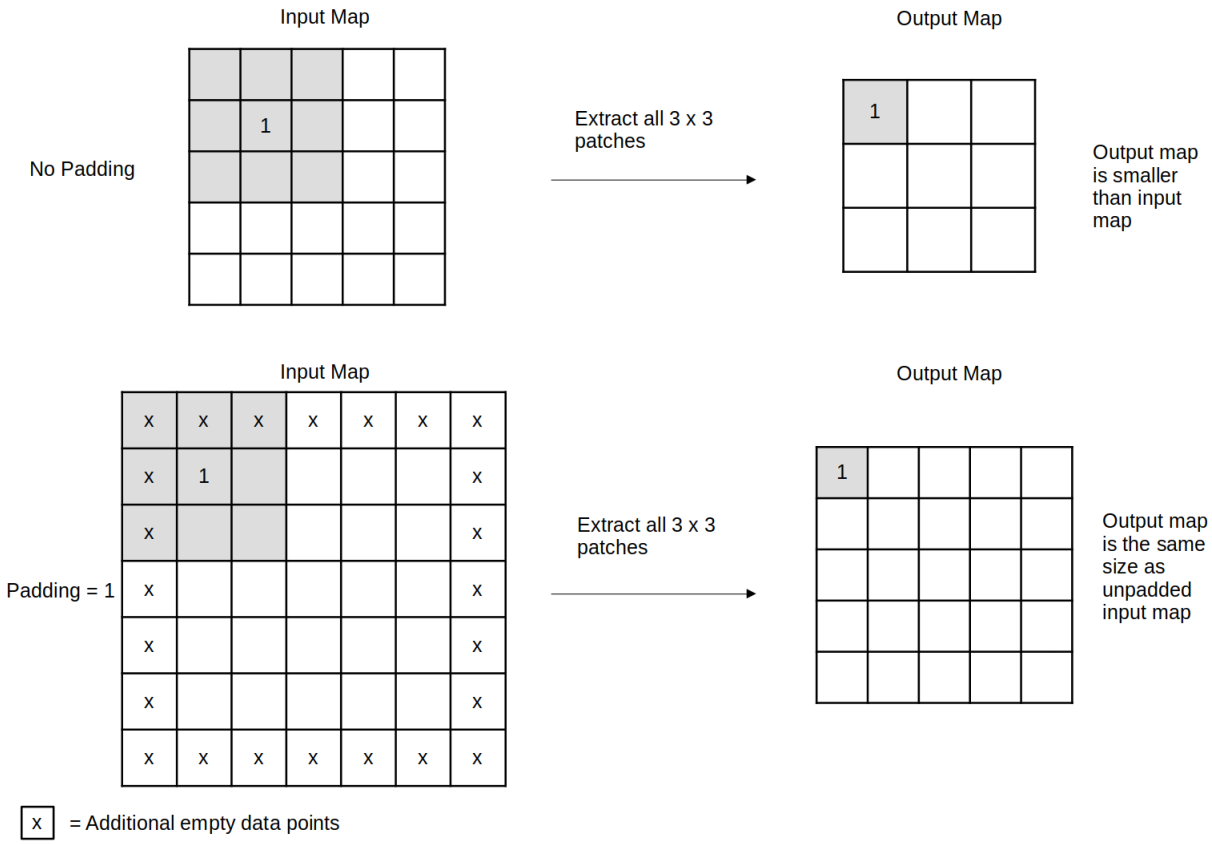


Figure 2.2: Visualisation of how padding is applied to an input map. The shaded region represents the initial 3×3 patch which is condensed into a single output in the output map - one such patch has been marked with the number 1 showing how a single patch is transformed. The squares marked with an x represent empty data points which are added on as a part of the padding procedure to allow patches to be taken from points without enough surrounding data (Chollet [2017]).

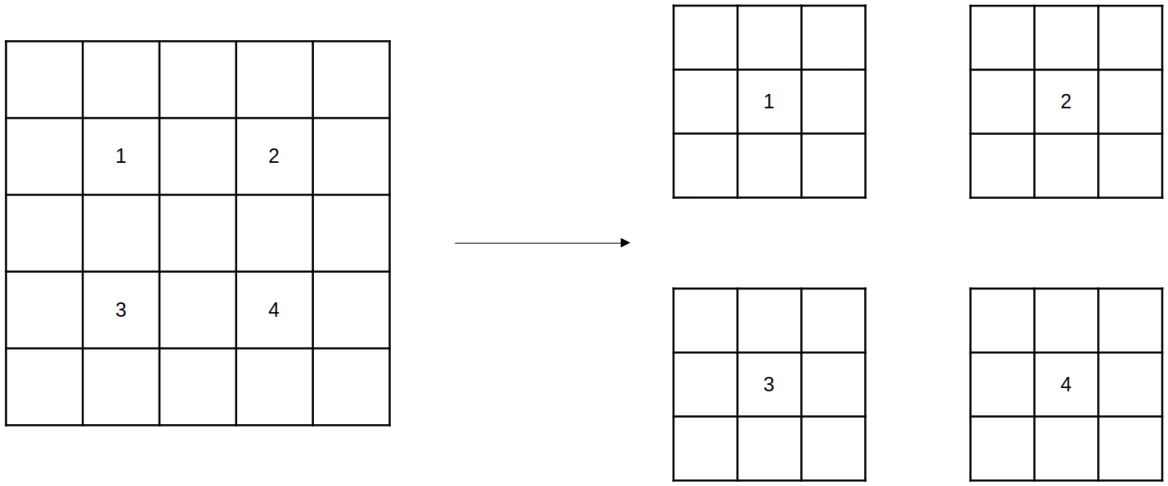


Figure 2.3: The usage of stride in the creation of patches. For an input map of 5×5, if we apply a stride of 2 we take patches from points that are 2 data points apart. As such, if we extract all 3×3 patches, we will end up with the four patches denoted 1,2,3 and 4 (Chollet [2017]).

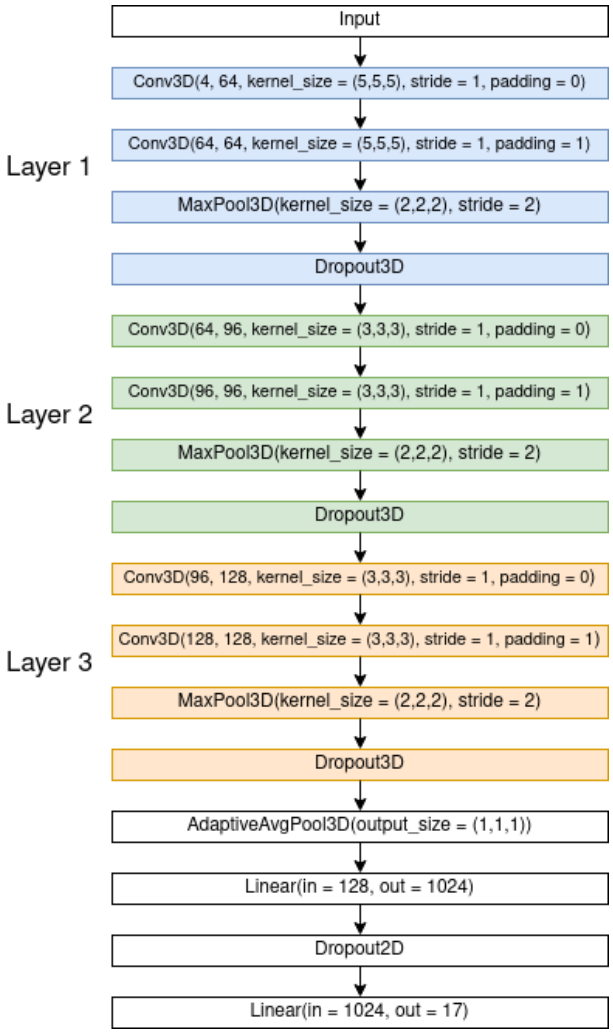


Figure 2.4: Structure of training network as defined in Ohtsuki and Mano [2020] for categorisation of 3D data.

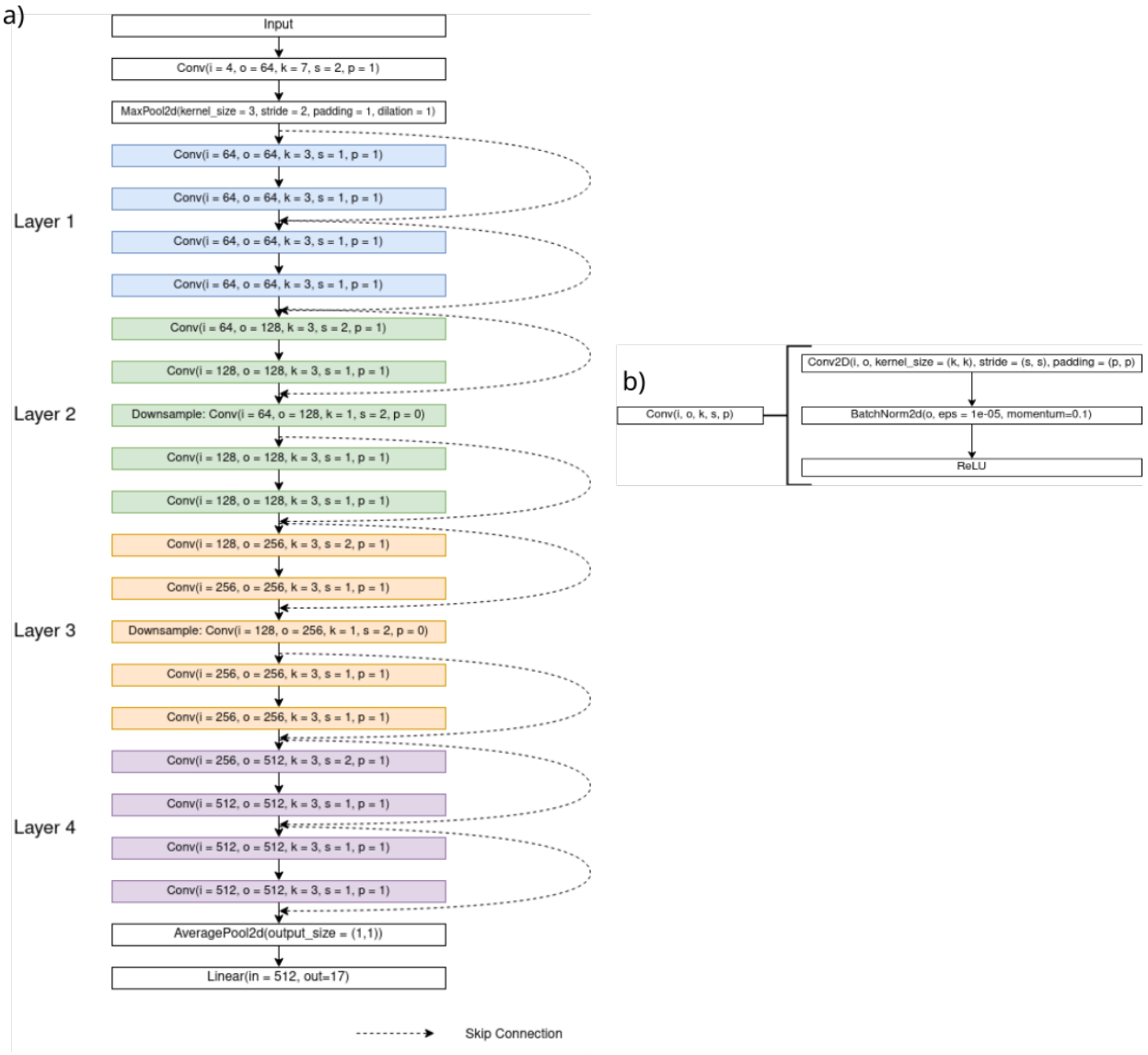
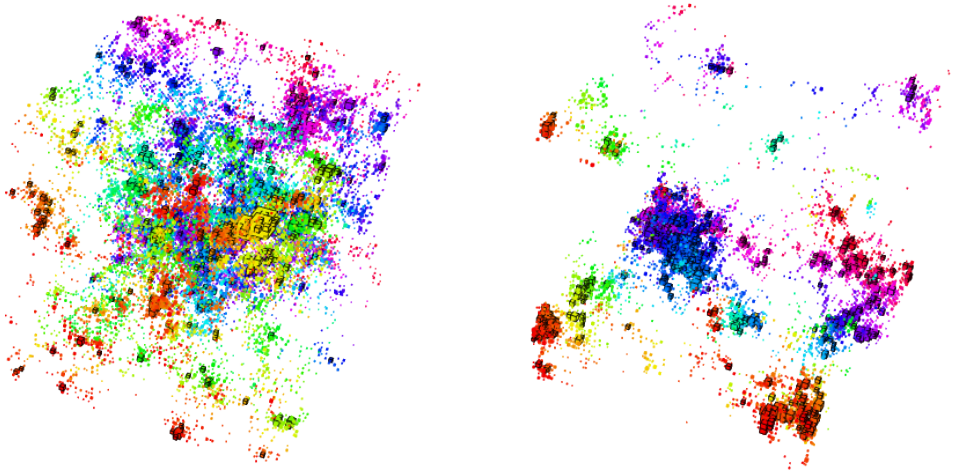


Figure 2.5: Architecture of the ResNet18 network for 2D images. Each conv block in figure 2.5*a* consists of a convolution layer, a batch normalisation layer and a ReLU layer as shown in figure 2.5*b*. The 3D network uses the same structure but with 3D kernels.



(a) Disorder 15.0 - $W < W_c$

(b) Disorder 18.0 - $W > W_c$

Figure 2.6: The images that will be input into the ML algorithm for training. Same as the images from Figure 1.2 but with the black borders removed.

Chapter 3

Categorisation of Extended and Localised Phases

3.1 Training with the Ohtsuki network for $L = 40$ with Ψ values

We begin training using the model described in Ohtsuki and Mano [2020]. Attempting to train the ML model on $\tau_{2,40}$ Ψ values as defined in section 2.5 to identify whether the input state is in an extended or localised phase. The accuracy of the model has been calculated as the number of correct predictions divided by the total number of samples. From this, we find that the training and validation accuracy starts low and increases as the number of epochs, denoted as ε, increases which is the expected behaviour for the training process. If we compare the training accuracy at $\varepsilon = 1$ (Figure 3.1) and $\varepsilon = 70$ (Figure 3.1), we can see that the accuracy of the predictions has increased from 49.5% to 98.4%. We can then plot a confusion matrix for these results which shows how many of the models predictions matches with the desired categories. Looking at how the model behaves for $\varepsilon = 1$ to 70 (Figure 3.1) we can see the diagonal fit of the confusion matrices improve as ε increases showing that the model is predicting more data correctly. We must also consider the behaviour of the validation matrices. Comparing the validation accuracy at $\varepsilon = 1$ (Figure 3.1) and the best validation accuracy at $\varepsilon = 70$ (Figure 3.2) we get an improvement from 50.0% to 98.1%. When we look at the behaviour of the model for all $\varepsilon = 1$ to 70 (Figure 3.2) we can see that the model is stagnate for the $\varepsilon = 1$ to 8 before the model begins to improve in accuracy. When we plot the accuracy (Figure 3.3) measured at each ε, we can see how the model improves as the number of training steps increases. We also plot a line representing what the accuracy

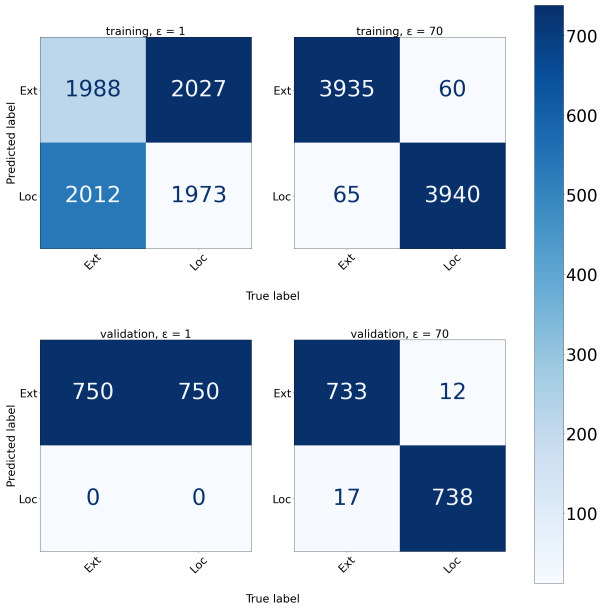


Figure 3.1: Confusion matrix of initial and final ε for Ψ trained model. Model trained using the Ohtsuki network, the ADAM optimiser and $L = 40$ data.

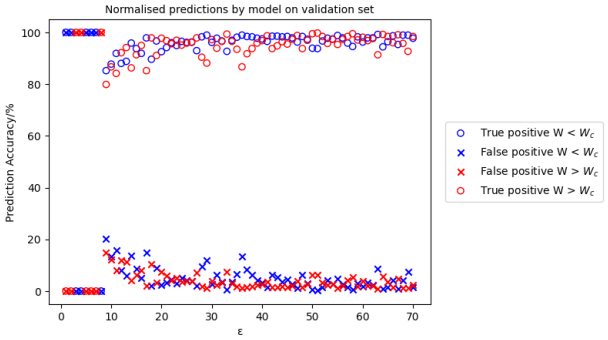


Figure 3.2: True and False positive rates at each ε for Ψ data model. Model trained using the Ohtsuki network, the ADAM optimiser and for $L = 40$ data.

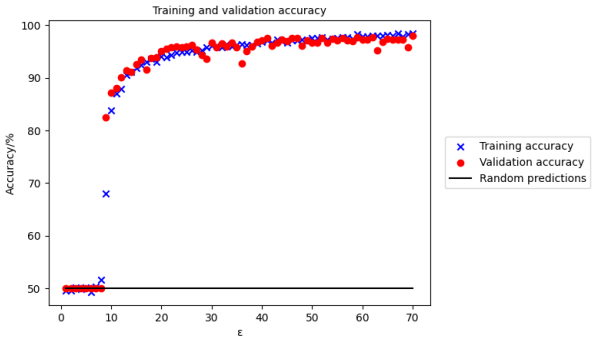


Figure 3.3: Accuracy of model for training and validation sets. Model trained using the Ohtsuki network, the ADAM optimiser for $L = 40$ data with Ψ values.

would be if the model predicted completely randomly in order to see if the model is working better than random chance. In this case, the accuracy starts around the random expectation chance of 50% for $\varepsilon = 1$ to 8 before it begins improving from random predictions. Overall, this shows that the model trained using the Ohtsuki network is able to categorise states of the Anderson model into extended and localised phases that it has never seen before with an accuracy of 98.1% and thus we can conclude that the model works effectively.

3.2 Training with the ResNet18 network for $L = 40$ with Ψ values

Changing to the RESNET18 network, we can see that the model preforms differently to the Ohtsuki model. The model starts with a training accuracy of 95.1% (Figure 3.4) and peaks with 100% at $\varepsilon = 20$ (Figure 3.4), fluctuating slightly due to random variations in how the batches passed into the training step are divided.

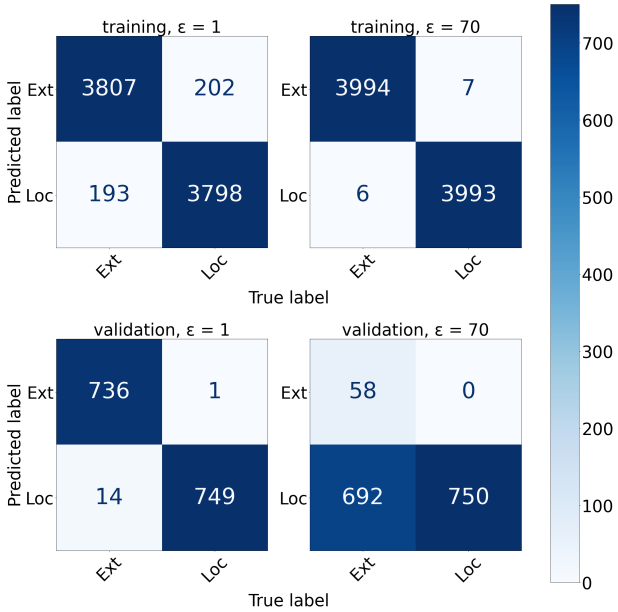


Figure 3.4: Confusion matrix of initial and final ε for Ψ trained model. Model trained using the ResNet18 network, the Adam optimiser and with $L = 40$ data.

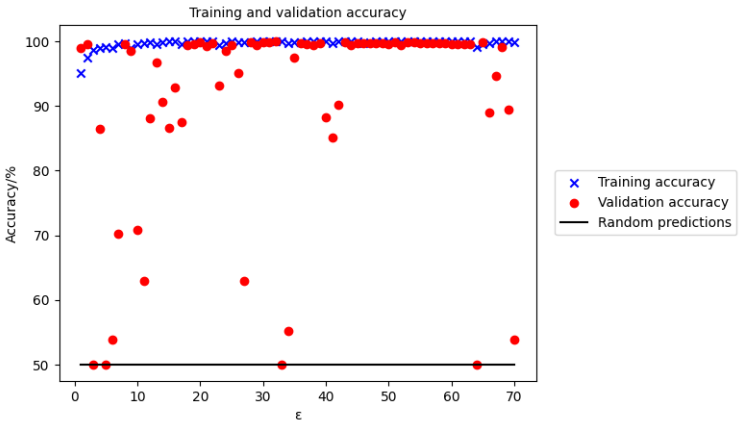


Figure 3.5: Accuracy of model for training and validation sets. Model trained using the ResNet18 network, the Adam optimiser, with $L = 40$ data and Ψ values.

The validation accuracy also starts really high at 99.0% (Figure 3.4) and also peaks at 99.8% at $\varepsilon = 20$ (Figure 3.4). Unlike with the Ohtsuki model, the validation accuracy does not remain close to the peak once it has reached it but fluctuates between 99% and 50% (Figure 3.5). The training structure of the model is complex enough that the model can reach an effective accuracy for low ε and so any addition features the model can identify are likely not generalised leading to large variations in the validation accuracy as the model is overfit for some ε before the validation step discards the changes. Compared to the Ohtsuki network, we can say that the ResNet18 network trained model preforms even better at categorising extended and localised phases of the Anderson model as the ResNet18 trained model has a better validation accuracy by 1.7%. This is due to the increased depth of the ResNet18 network leading to better feature detection in the model.

3.3 Comparison of training with Ψ values against $|\Psi|^2$ values

The above results correspond to training using generated Ψ values. If we instead switch to using Ψ^2 values the model trains differently. For the Ohtsuki network, looking at the accuracy (Figure 3.6) curve, we find that the peak training accuracy is 82.7% and the peak validation accuracy is 83.0% compared to the 98.4% and 98.1% training and validation accuracy for the Ψ trained models. On figure 3.6 we can see that for $\varepsilon = 1$ to 20 the model remains at 50% accuracy which also occurs in the Ψ trained models for $\varepsilon = 1$ to 8 (Figure 3.3). When we then plot at the accuracy (Figure 3.7) for the ResNet18 network we find a maximum training accuracy of 100% first at $\varepsilon = 25$ and a maximum validation accuracy of 99.7% at $\varepsilon = 38$. Compared to the Ψ models (Figures 3.5) with maximum training and validation accuracy of 100% and 99.8% respectively, the accuracy for the Ψ and $|\Psi|^2$ trained models are effectively the same. Looking at the validation accuracy, we can see that the main difference is that there is more variation in the validation accuracy as ε varies. Together these results show that the $|\Psi|^2$ value model produces a slightly less effective result when training with the Ohtsuki network and a result of similar effectiveness when training with the ResNet18 network for $L = 40$.

3.4 Effects of training using AdaDelta operator compared to Adam operator

One other factor we have considered in the training is the optimiser used. For the previous example models we have been using the optimiser Adam (Kingma and Ba [2017]). Another potential optimiser that we could use is the AdaDelta optimiser (Zeiler [2012]) which has also been mentioned in the Ohtsuki paper (Ohtsuki and Mano [2020]). Our results (Figure 3.9) show that training with the AdaDelta optimiser is not as effective as using the Adam optimiser. The training accuracy shows no growth, instead showing a large spread around the 50% mark and the validation accuracy remains constantly at 50%. When considering that a model that predicts completely randomly would produce an accuracy of 50% we can conclude that the model trained using the AdaDelta optimiser has not produced a usable model. Plotting the true and false positive rates for the validation data (Figure 3.8), we can see that the model is predicting all the data into 1 category. If we compare this to the same model trained using the Adam optimiser (Figure 3.6), we see that for the $\varepsilon = 1$ to 20 (Figure 3.10), the Adam model remains at 50%

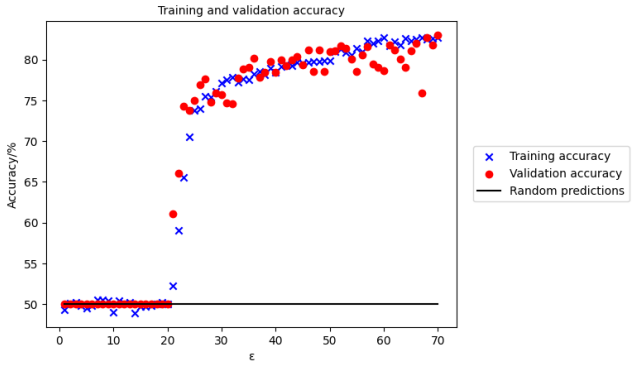


Figure 3.6: Accuracy of model for training and validation sets. Model trained using the Ohtsuki network, the ADAM optimiser, with L = 40 data and Ψ^2 values.

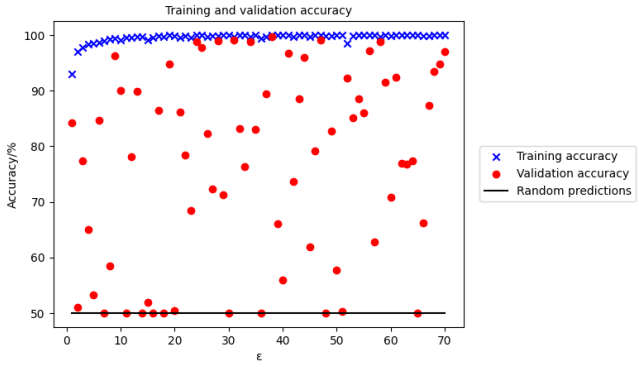


Figure 3.7: Accuracy of model for training and validation sets. Model trained using RESNET18 network, ADAM optimiser, with L = 40 data and Ψ^2 values.

accuracy before beginning to improve as it has found an appropriate set of weights for effective predictions. This means it could be that the ADADELTA model has not found the appropriate set of weights to begin training from and thus is stuck in a local minima and by continuing the training for $\varepsilon > 70$ we could see the model find an appropriate starting point and then train properly. Overall, it would be more effective to train using the optimiser ADAM as a model trained with that optimiser will reach an effective result in $\varepsilon < 70$ and therefore will take less time to train due to needing a smaller range of ε.

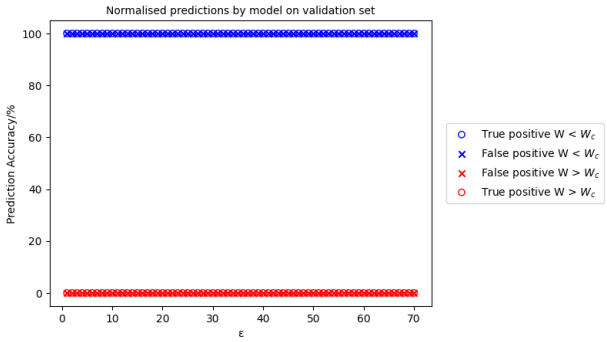


Figure 3.8: True and False positive rates at each ε for Ψ^2 data model. Model trained using the Ohtsuki network, the AdaDelta optimiser for L = 40 data.

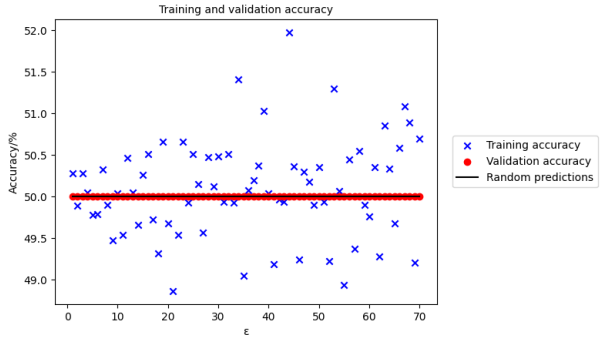


Figure 3.9: Accuracy of model for training and validation sets. Model trained using the Ohtsuki network, the AdaDelta optimiser with L = 40 data and Ψ^2 values.

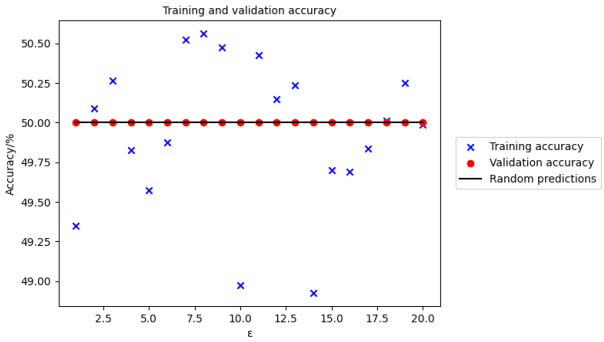


Figure 3.10: Accuracy of model for training and validation sets. Model trained using the Ohtsuki network, the Adam optimiser with L = 40 data and Ψ^2 values. Chart cut at $\varepsilon = 20$.

Chapter 4

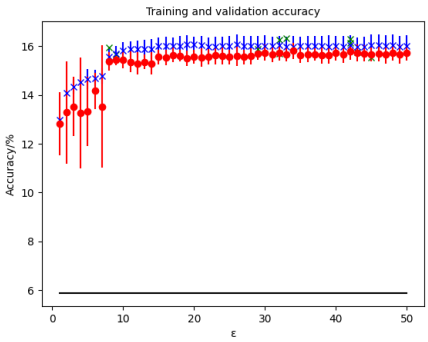
Categorisation based on disorder of state

4.1 Training using Images

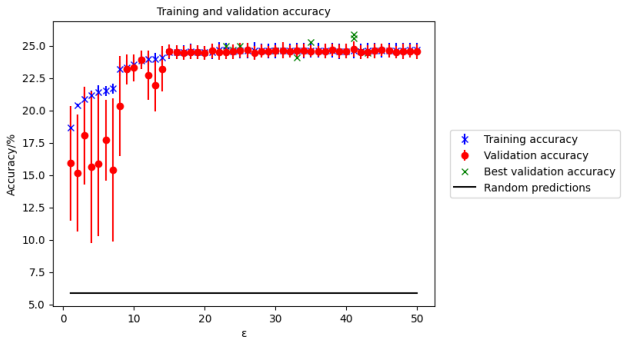
4.1.1 Comparison of effect of L on training of model

We now train the model using dataset $\vartheta_{17,20}$. We start with data with an image size, s, of 100×100 pixels from this dataset. Training the model with this we find the model reaches a maximum training accuracy of 16.0 ± 0.4% at epoch (ε) 50 (Figure 4.1a). Looking at the validation accuracy when $\varepsilon = 50$ we find it is 15.7 ± 0.4%. The maximum validation accuracy we obtain is 15.8 ± 0.3% at $\varepsilon = 34$. The model accuracy has stopped increasing after $\varepsilon = 10$, showing that the model has completed training. This shows that with $L = 20$ and $s = 100$ the model cannot train to an effective accuracy. One reason for this is that the $L = 20$ data only provides information of the phase of the wavefunction for 8000 sites leading to the model not having enough data to train with. Compared to the black line in figure (Figure 4.1a) which represents the expected accuracy if the model predicted entirely randomly, we can see that the model is 10% better than random chance.

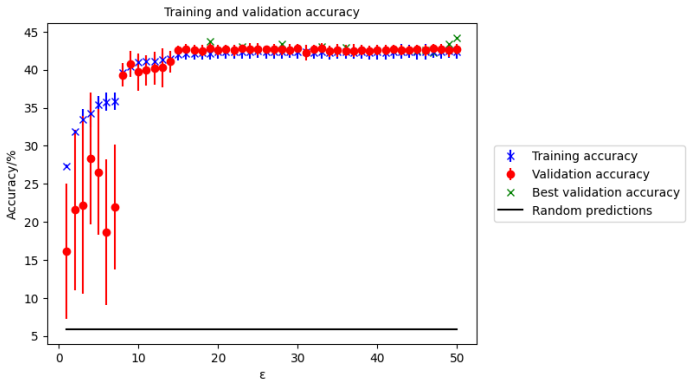
Increasing to $L = 40$ with dataset $\vartheta_{17,40}$, we find that for the $s = 100$ images, the model is able to reach a final training accuracy of 24.7 ± 0.6% and validation accuracy of 24.5 ± 0.6% (Figure 4.1b). The maximum validation accuracy is 24.7 ± 0.6% for $\varepsilon = 41$. Compared to the models for $L = 20$ data, we can see that there has been an improvement in the training accuracy from 16.0% to 24.7% and in the validation accuracy from 15.8% to 24.7%. This shows that increasing L has improved the models effectiveness - by giving the model more data to work with the



(a) Accuracy for model trained at $L = 20$.



(b) Accuracy for model trained at $L = 40$.



(c) Accuracy for model trained at $L = 100$.

Figure 4.1: Accuracy graphs of training and validation sets for models trained using the ResNet18 network, the Adam optimiser for $s = 100$ and varying sizes of L.

model can more easily identify the patterns necessary for categorisation.

Lastly, we can train the model using $\vartheta_{17,100}$ $s = 100$ images (Figure 4.1c). With this we find that the model is able to reach a final training accuracy of $42.3 \pm 0.9\%$, a final validation accuracy of $42.7 \pm 0.7\%$ and a max validation accuracy of $42.8 \pm 0.6\%$ for $\varepsilon = 30$. Once again, increasing L has increased the training and validation accuracy as more data has been provided.

Plotting the accuracy of the models trained for $L = 10, 40$ and 100 with $s = 100$ (Figure 4.1c) overall, we can see that training in this way leads to very little overfitting. The training and validation increase at the same rate and when the training accuracy has reached its maximum, the validation accuracy does not begin to decrease. As figure 4.2 shows, increasing s has led to an increase in the training and

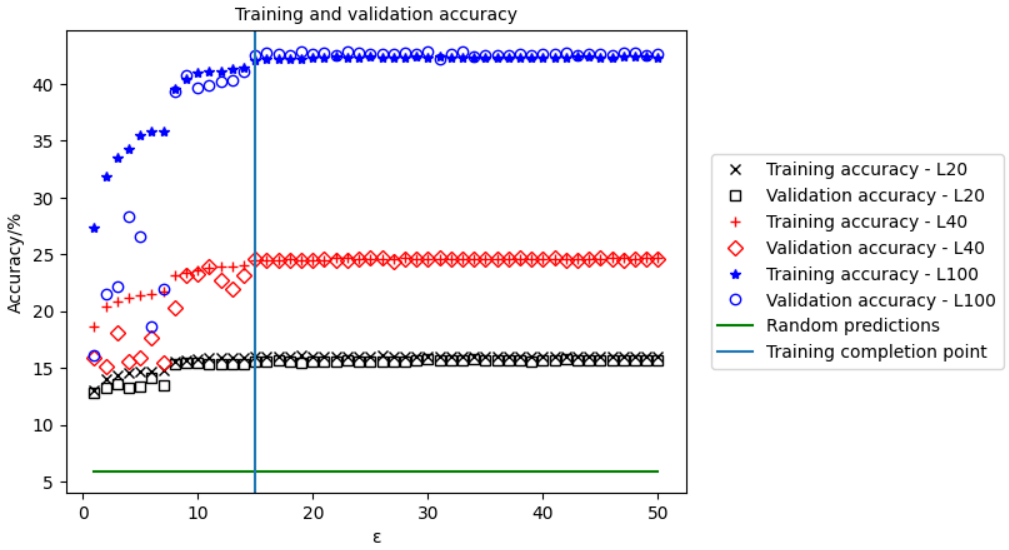


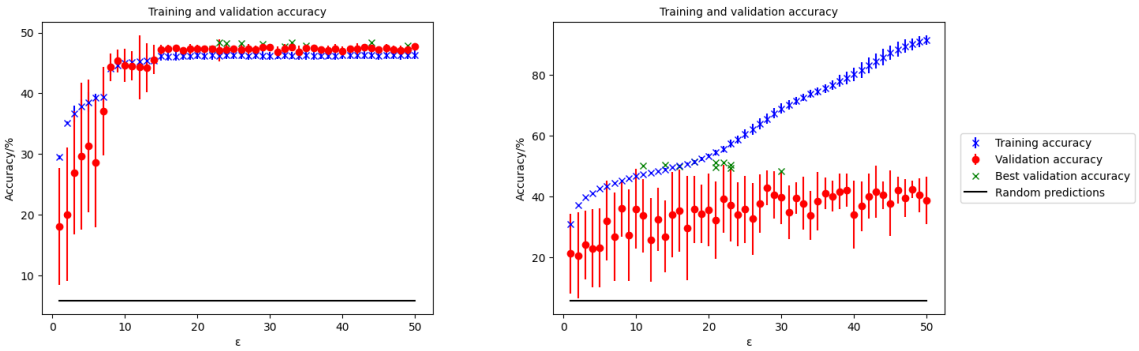
Figure 4.2: Consolidated accuracy for training and validation sets for $L = 20, 40$ and 100 models trained using $s = 100$ images. Models trained using the ResNet18 network and the Adam optimiser.

validation accuracy. This is due to the increased L equating to a larger cube of the Anderson model being plotted in the image making it more likely for the differentiating patterns to form and be detected for the model to train towards. We can also see that all 3 models stop increasing at approximately $\varepsilon = 15$ which shows that the training has been completed at $\varepsilon = 15$ and any training beyond this point is not necessary.

4.1.2 Comparison of effect of s on training of model

We can also test how different image sizes affects the model. For this, we will training all models for $L = 100$. Starting with the $s = 100$ image training (Figure 4.1c), the model has a final training accuracy of $42.3 \pm 0.9\%$ and max validation accuracy of $42.8 \pm 0.6\%$. Looking at $s = 200$ (Figure 4.3a), we see that the model reaches a final accuracy of $46.3 \pm 0.7\%$. The peak validation accuracy is $47.7 \pm 0.4\%$ when $\varepsilon = 50$. With the $s = 500$ images the model reaches a final training accuracy of 91.3 $\pm$ 1.6% and a peak validation accuracy of $43.0 \pm 5.7\%$ at $\varepsilon = 28$ (Figure 4.3b).

Comparing the three image sizes, we can see that increasing s from 100 to 200, increases the training and validation accuracy. When s increases from 200 to 500 we find that the training accuracy increases by 45% compared to the 4% increase when increasing s from 100 to 200. The validation accuracy falls by 4.7% when



(a) Accuracy for model trained for $s = 200$.

(b) Accuracy for model trained for $s = 500$.

Figure 4.3: Accuracy graphs of training and validation sets for models trained using the ResNet18 network, the Adam optimiser for $L = 100$ and varying sizes of s.

changing from $s = 200$ to 500. This behaviour of very high training accuracy and a decrease in validation accuracy fits the behaviour of overfitting. This could show that the increase in s has increased the information volume by too much leading to over fitting. One key factor to also consider is that the $s = 500$ model was trained using a different machine. At the time, Sulis usage was being reserved for other tasks and as such the $s = 500$ training was run on Avon while Djena ran the $s = 100$ and 200 tasks at a later point on Sulis. This could lead to slight differences in memory partitions and batch formation even with the same random number seed leading to different behaviour. The behaviour at $s = 500$ has shown that it is important to consider that providing the maximum amount of data is not necessarily optimal due to overfitting and that maintaining as much consistency between machines is important in produce reproducible models.

Changing s does not improve the training and validation accuracy as much as changing L does. Increasing s from 100 to 200 increases the training accuracy by 4% and the validation accuracy by 4.9% whereas increasing L from 20 to 100 improves the training accuracy by 26.7% and the validation accuracy by 27.0%. As such, we can say that while training using the highest possible s is useful, training with the largest L is more important. This matches up with the effects on the data of L and s. Increasing s takes the same data and scales it up. This can produce more detail within the finer points of the image which will improve model performance. Increasing s does not change the information about the physical state whereas increasing L will increase the size of cube of the Anderson model state being provided to the model. Therefore, by increasing L we will provide more information about the physical system which in turn will help the model predict more accurately.

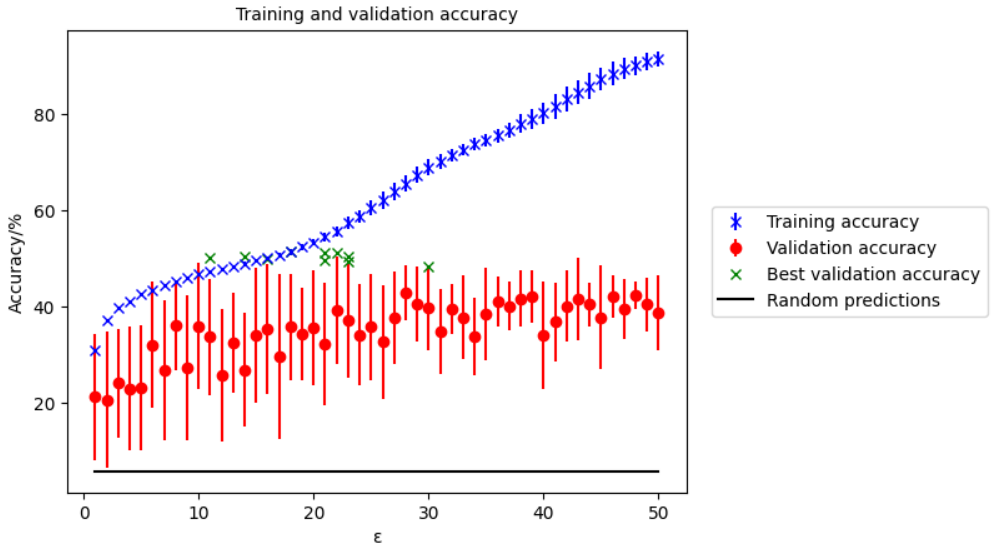


Figure 4.4: Consolidated accuracy for training and validation sets for $s = 100, 200$ and 500 at $L = 100$. Models trained using the ResNet18 network and the Adam optimiser.

4.2 Training using numerical $|\Psi|^2$ values

4.2.1 Training using $L = 20$ data

The next step of training was to use at the $|\Psi|^2$ values we generated the images from. Starting with dataset $\tau_{17,20}$, we find that the training accuracy is 98.4 ± 0.2% (Figure 4.5). Looking at the validation accuracy, we see that the max average validation accuracy is 10.9 ± 7.9%. Compared to a model trained on the equivalent images (4.1a) we find that the training accuracy has improved from 16.0 ± 0.4% whereas the validation accuracy has decreased from 15.8 ± 0.3%. While an improvement in training accuracy may seem beneficial, it represents a model that is only able to predict well on the data that has already been provided to it. As such, a model with very high training accuracy but low validation accuracy represents a model which cannot predict well on unseen data and thus is useless for categorisation.

There is an issue with measuring the validation accuracy of the ten trials in this way. When we plot the best and worst validation confusion matrix at $\varepsilon = 8$ for the ten trials it shows that while the best confusion matrix has a diagonal shape, the worst matrix shows that for this trial the model has failed at this point and is predicting all the data into a single category. Since model performance at each individual ε will not be the same across different trials,

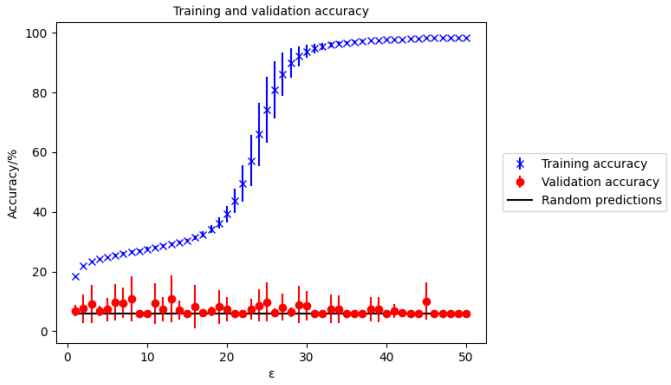


Figure 4.5: Accuracy of model for training and validation sets. Model trained using the ResNet18 network, the Adam optimiser with numerical $|\Psi|^2$ values for $L = 20$.

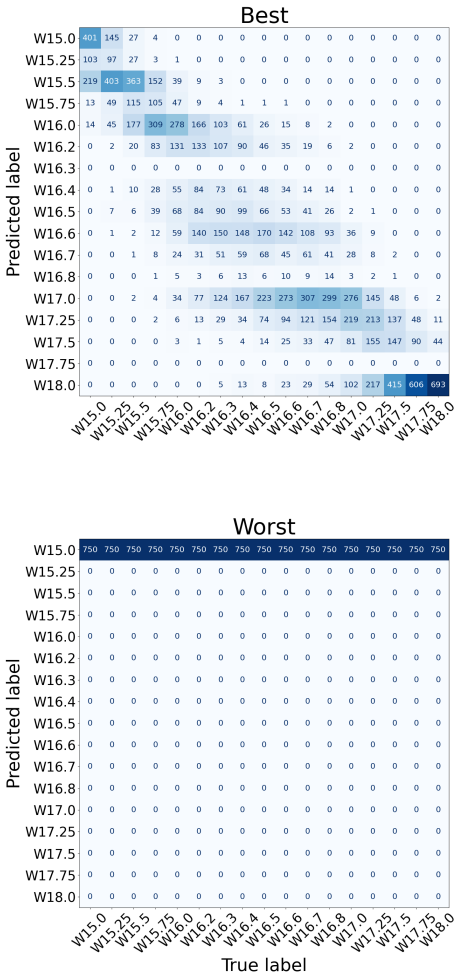


Figure 4.6: Best and worst validation confusion matrices of the 10 training trials for $L = 20$ with $|\Psi|^2$ values at $\varepsilon = 8$. See Appendix A for the full set of 10 confusion matrices.

averaging across the validation accuracy at a single ε does not provide the full picture. As such, when we measure the best validation accuracy of the trained models we get a maximum validation accuracy of $30.2 \pm 0.3\%$ (4.7) which is better when compared to the $15.8 \pm 0.3\%$ for the same data in the form of images. By plotting the confusion matrices for the ε in each training run with the best validation accuracy, we can see that most of the models predict closer to the diagonal of the confusion matrix. Overall, this shows that compared to models trained on images, models trained with $|\Psi|^2$ values can reach higher validation accuracy and thus create models that are better for categorisation. Plots of the confusion matrices for all ten trials can be found in Appendix A.

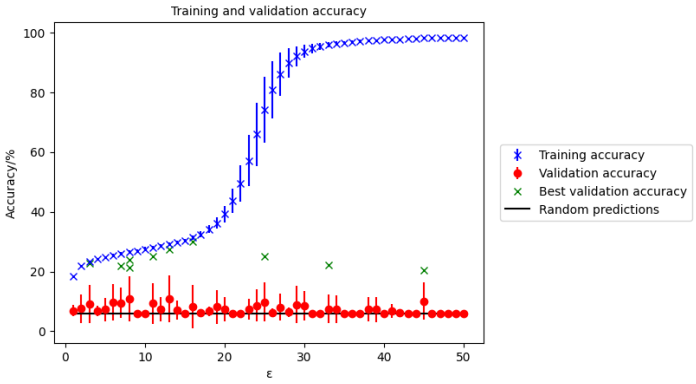


Figure 4.7: Accuracy of model for training and validation sets with best validation accuracy from each model trained plotted in green. Model trained using the ResNet18 network, the Adam optimiser with numerical $|\Psi|^2$ values for $L = 20$.

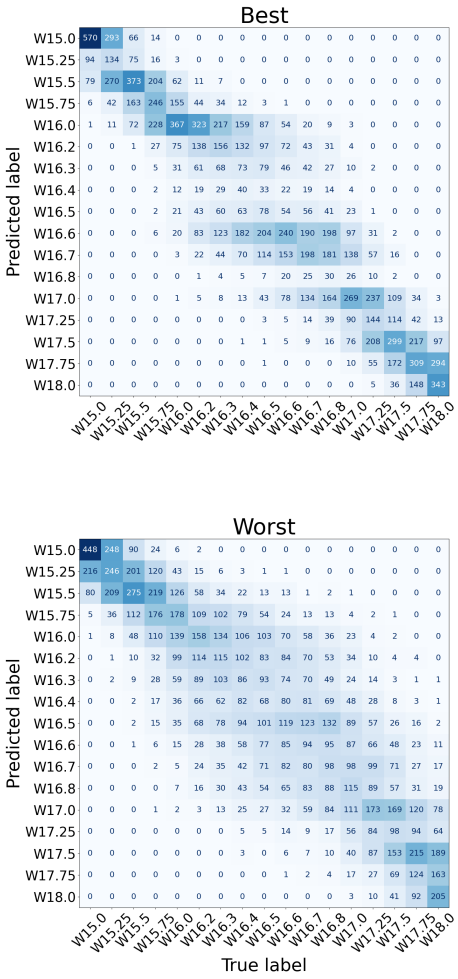


Figure 4.8: Best and worst validation confusion matrices at the ε with the best validation accuracy of the 10 trials when trained for $L = 20$ with $|\Psi|^2$ values.

4.2.2 Training using $L = 40$ data

Moving onto the $L = 40$ data, we find a training accuracy of 95.3 $\pm$ 1.4% in the final ε and a maximum average validation accuracy of 15.1 $\pm$ 13.5% when $\varepsilon = 37$ (Figure 4.9). The validation accuracy for the best model across each trial is 49.0 $\pm$ 13.7%. When we plot the accuracy graphs for both we can see that the $L = 20$ data has begun to level off around $\varepsilon = 30$, whereas the $L = 40$ model is only just starting to stop increasing, showing that the $L = 40$ model is not as over-fit. This maximum validation accuracy occurs for $\varepsilon = 38$ compared to $\varepsilon = 15$ for the $L = 20$ data. This is explained by the less overfitting in the $L = 40$ training leading to the validation accuracy being able to improve further into the training of the model.

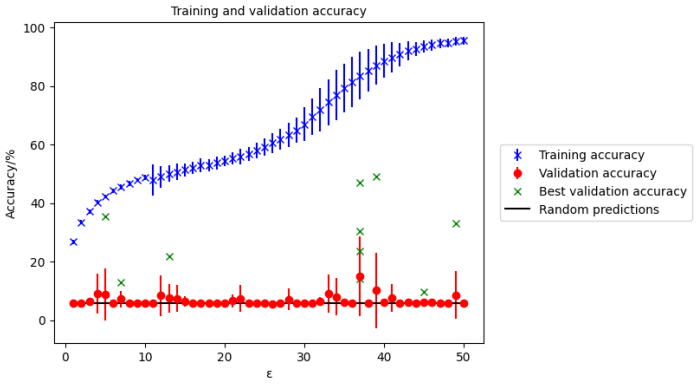


Figure 4.9: Accuracy of model for training and validation sets. Model trained using the ResNet18 network, the Adam optimiser with numerical $|\Psi|^2$ values for $L = 40$.

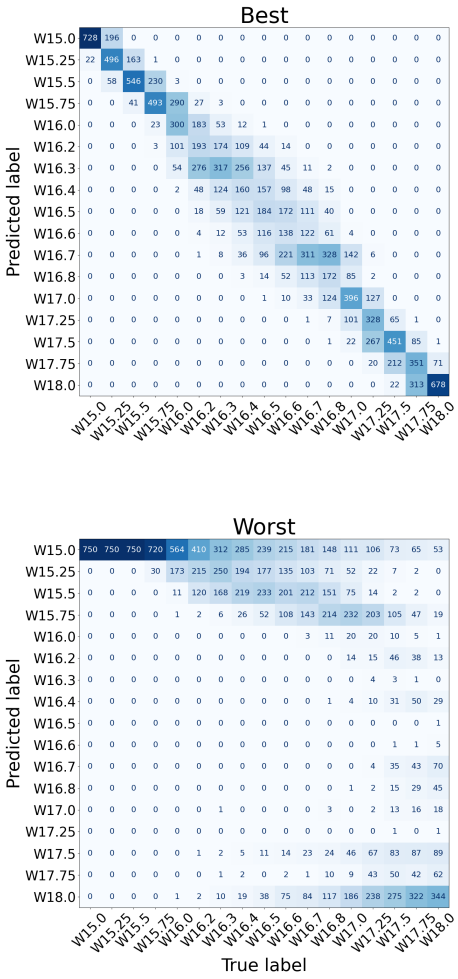


Figure 4.10: Best and worst validation confusion matrices at the ε with the best validation accuracy of the 10 trials when trained for $L = 40$ with $|\Psi|^2$ values.

With the best and worst confusion matrices for the ε with the maximum validation accuracy (Figure 4.10), there is an improvement in the best matrix as the predictions are tending closer to the diagonal which reflects the improved best validation accuracy. Comparing the worst confusion matrix with the one for $L = 20$ (Figure 4.8) shows that the worst result for $L = 40$ is worse that for $L = 20$. This is a result of the increased complexity of the data resulting in predictions becoming more difficult. This can also be seen in the plot of the best validation accuracy for each trial in figure 4.9. There is much more spread in the y-axis of the green plots compared to the points in figure 4.5 showing that while the increased complexity

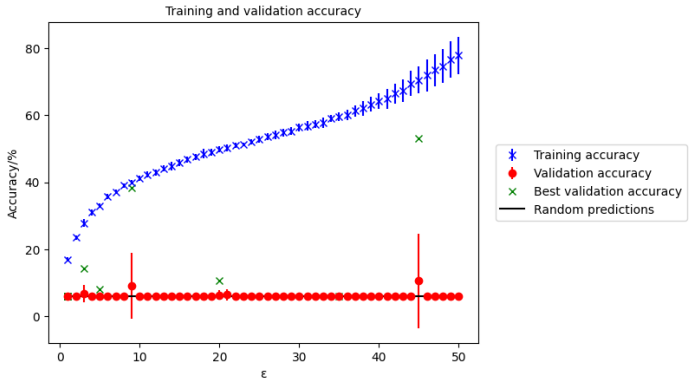


Figure 4.11: Accuracy of model for training and validation sets. Model trained using the ResNet18 network, the Adam optimiser with numerical $|\Psi|^2$ values for $L = 100$.

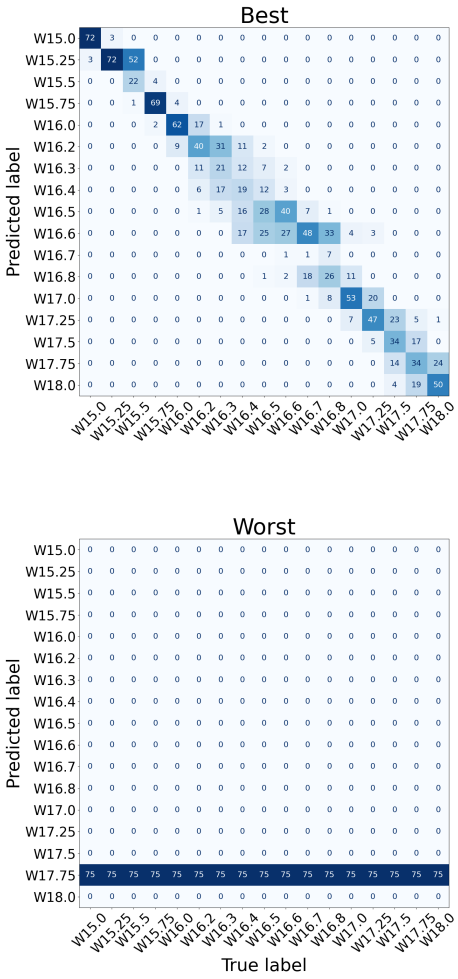


Figure 4.12: Best and worst validation confusion matrices at the ε with the best validation accuracy of the 10 trials when trained for $L = 100$ with $|\Psi|^2$ values.

does allow the model to reach a higher peak by providing more information for the model to draw patterns from, it also makes it more difficult for a model to find those patterns, leading to more variation in the best validation accuracy.

4.2.3 Training using $L = 100$ data

Finally, with the $L = 100$ data, we get a final training accuracy of $77.9 \pm 5.5\%$, that the maximum of the average validation accuracy is $10.6 \pm 14.1\%$ (Figure 4.11) and that the best validation accuracy is $53.1 \pm 16.6\%$. Plotting the best and worst confusion matrices for the maximum measured validation accuracy (Figure 4.12) we find again that the best confusion matrix is aligned well with the diagonal and the worst confusion matrix is has failed completely and is only predicting into one

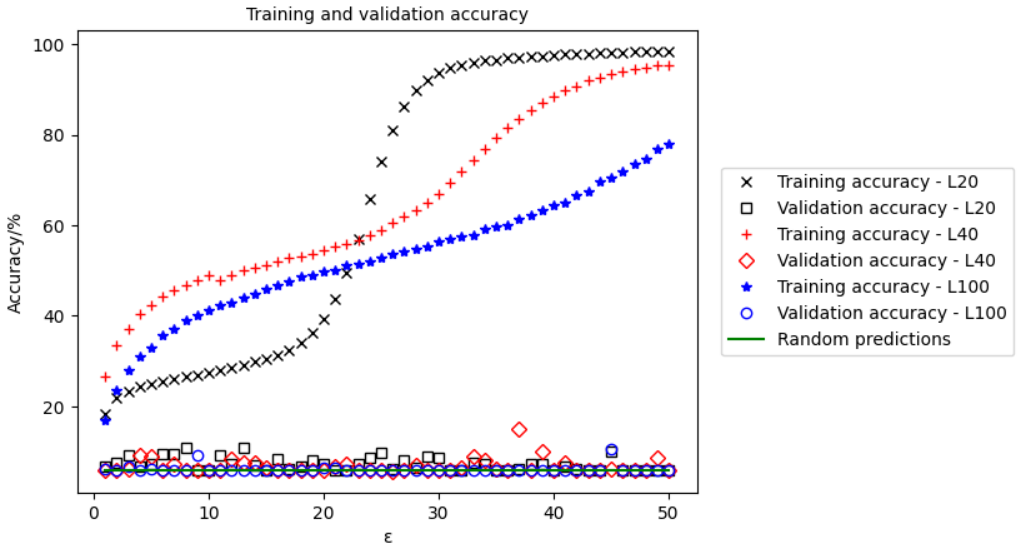


Figure 4.13: Consolidated accuracy for training and validation sets for $L = 20$, 40 and 100 model trained using $|\Psi|^2$ values.

category. Plotting all of the best validation accuracy (Figure 4.11) again shows a larger spread in the best validation accuracy values compared to $L = 40$ shown by the increased error from $\pm$ 13.7% for $L = 40$ to $\pm$ 16.6% for $L = 100$. Again, this shows that an increase in complexity can lead to a more accurate model by providing more information for determining patterns and in turn this increased complexity will also lead to the model having more trouble with identifying those patterns in the first place, leading to a higher best validation accuracy and more variation in that value between trials.

4.2.4 Comparisons of the effects of varying L on $|\Psi|^2$ value training

Combining the training and validation accuracy for $L = 20, 40$ and 100 (Figure 4.13), we see that the trends match up with behaviour in the Ψ models. As L increases, the final training accuracy decreases as the increased complexity of the input makes the model take longer to train. Comparing the validation accuracy, the increase in L leads to increasing best validation accuracy, even between $L = 40$ and 100. Considering that the average validation accuracy still decreases as L goes from 40 to 100, the increase in complexity is causing lower average validation accuracy but leads to higher potential best validation accuracy which will be reached as more epochs are trained. This can be seen by comparing the 10 best confusion matrices for the 40 and 100 models.

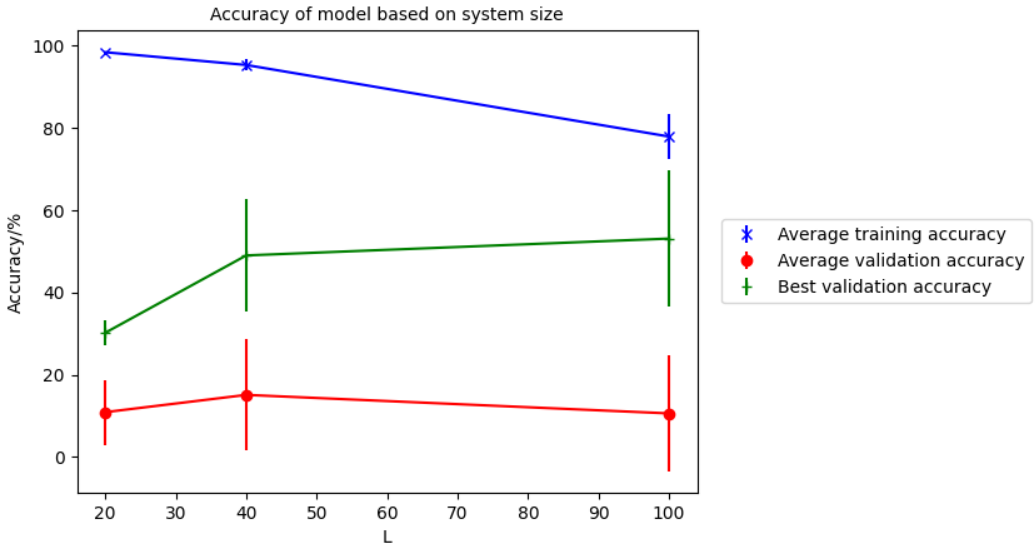


Figure 4.14: Training, average validation and best validation accuracy for $L = 20$, 40 and 100 models trained using $|\Psi|^2$ values.

Plotting the training accuracy, average validation accuracy and best validation accuracy against L (Figure 4.14), we can see that increasing L leads to a decrease in training accuracy from 98.4% for $L = 20$ to 95.3% for $L = 40$ and 77.9% for $L = 100$. The decrease in training accuracy is due to a decrease in overfitting. The increase in the volume of data supplied to the model will increase the number of epochs it takes to train the model which in turn leads to taking more epochs to begin overfitting, resulting in lower training accuracy. Despite the training accuracy being lower for higher L, less overfitting is ideal for training as it leads to a higher validation accuracy which is the key factor in measuring the effectiveness of a model. This can be seen by comparing the best validation accuracy for each L. For $L = 20$ the best attained validation accuracy is 30.2% which increases to 49.0% for $L = 40$ and then 53.1% for $L = 100$. From this, we can conclude that increasing L has produced a more effective model as the higher validation accuracy represents a model which is better at predicting on unseen data which is the factor we want to maximise. Comparing the average validation accuracy we find that at first the accuracy increases between $L = 20$ and 40 from 10.9% to 15.1%. This fits in with the expected behaviour where increasing L causes less overfitting and thus leads to a higher validation accuracy. When L increases to 100 the average validation accuracy falls from 15.1% to 10.6%. This is due to the increased L causing the model to get caught in a local maxima more often which skews the average accuracy down which can be seen when comparing the confusion matrices of the best validation for

$L = 40$ (Figure 4.10) and $L = 100$ (Figure 4.12). This increase in input data also allows the model to reach higher best accuracy when it escapes the local maxima which is why the best accuracy is still higher. The main issue this causes is that the model can take more attempts to reach high best validation accuracy leading to needing to do more runs.

4.3 Training using numerical Ψ values

4.3.1 Training using $L = 20$ data

The next step of training is to train using the original Ψ values in order to keep the phase information. Starting with the $L = 20$ data the training accuracy reaches a maximum of 92.5 ± 2.7% in the final ε (Figure 4.15)). The average validation accuracy peaks at 25.1 ± 17.5% in $\varepsilon = 32$ and the best validation accuracy is 60.0 ± 5.5%. Comparatively, the model trained with $|\Psi|^2$ values at the same L has a training accuracy of 98.4 ± 0.2%, an average validation accuracy of 98.4 ± 0.2% and a best validation accuracy of 30.2 ± 3.0%. This means that compared to a model trained using $|\Psi|^2$ values, the Ψ train models have decreased training accuracy and increased average and best validation accuracy. As the main factor we wish to increase is the validation accuracy, we can conclude from this that the Ψ model has preformed better than the equivalent $|\Psi|^2$ model. Plotting the best and worst confusion matrix of the best result across all trials (Figure 4.16), there is much more alignment with the diagonal in both the best and the worst case showing how the model is predicting much better compared to using $|\Psi|^2$ values (Figure 4.8).

4.3.2 Training using $L = 40$ data

Increasing to $L = 40$, we get a final training accuracy of 81.3 ± 1.0% and a peak average validation accuracy of 28.1 ± 26.9% (Figure 4.17) and the best validation accuracy we have obtained is 85.9 ± 10.4%. Plotting the best and worst confusion matrices out the ε representing the maximum validation accuracy for each trial (Figure 4.18), we can see that the model is predicting even closer to the diagonal when compared to the best and worst for $L = 20$ showing again how the increase in L has lead to an improved model. One other factor to consider is that for the $|\Psi|^2$ value trained models (Figure 4.10), while the best confusion matrix was becoming more aligned with the diagonal, the worst confusion matrix was beginning to predict into one category as the increased complexity was making it harder to spot patterns. This behaviour is not seen in the Ψ model trained at the same L showing that the

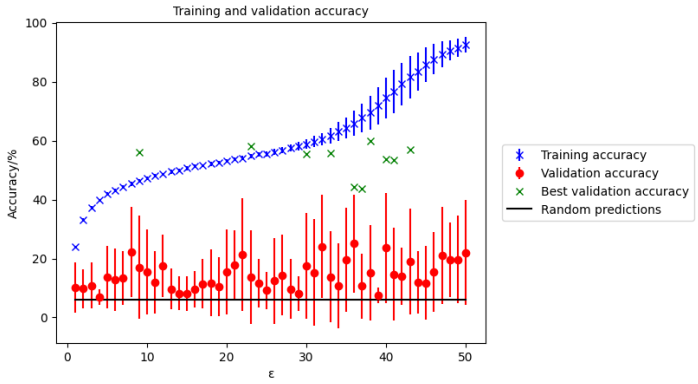


Figure 4.15: Accuracy of model for training and validation sets, best validation accuracy from each model trained plotted in green. Model trained using the ResNet18 network, the Adam optimiser with numerical Ψ values at $L = 20$.

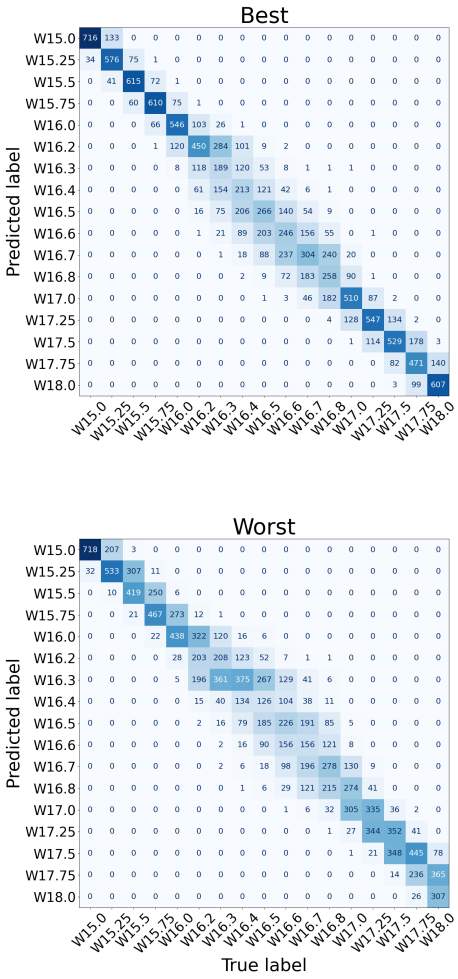


Figure 4.16: Best and worst validation confusion matrices at the ε with the best validation accuracy of the 10 trials when trained for $L = 20$ with Ψ values.

patterns required for classification are more identifiable in Ψ data.

4.3.3 Training using $L = 100$ data

With the $L = 100$ data, we find that the final training accuracy is $67.6 \pm 1.0\%$. The best average validation accuracy is $40.1 \pm 27.0\%$ and the best validation accuracy overall is $79.6 \pm 8.2\%$ (Figure 4.19). This is interesting as the best validation accuracy has decrease going from $L = 40$ to $L = 100$, from $85.9 \pm 10.4\%$ (Figure 4.17) to $79.6 \pm 8.2\%$. When we plot the best and worst confusion matrices for the best validation accuracy of each trial (Figure 4.20), we can see that the model is predicting well near $W = 15.0$ and 18.0 and is struggling more around the critical point where differentiating the categories is more complex and thus will take more training steps.

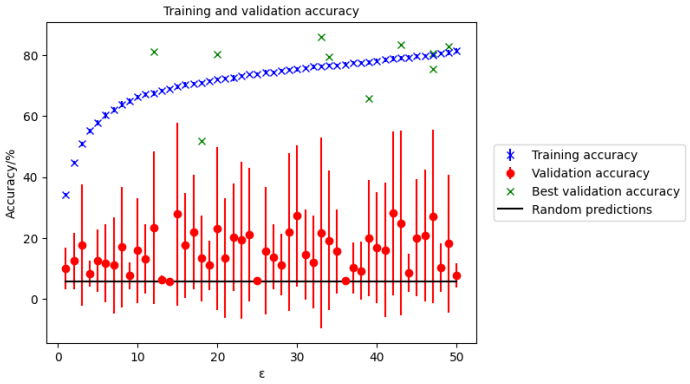


Figure 4.17: Accuracy of model for training and validation sets with best validation accuracy from each model trained plotted. Model trained using the ResNet18 network, the Adam optimiser for numerical Ψ values at $L = 40$.

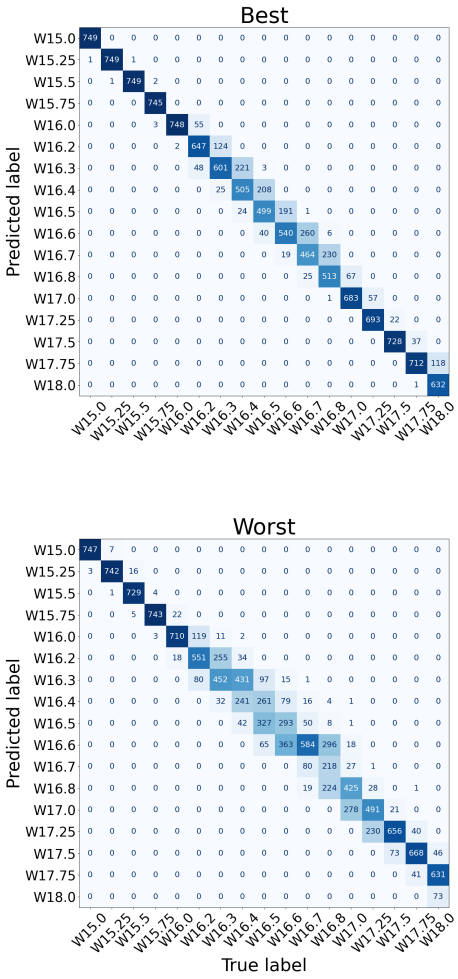


Figure 4.18: Best and worst validation confusion matrices at the ε with the best validation accuracy of the 10 trials when trained for $L = 40$ with Ψ values.

It is also important to consider that in training the $L = 100$ data, we could only use 500 samples instead of 5000 due to GPU memory constraints and run-times. While this is still more data ($100^3 \times 500 = 5 \times 10^8$ vs $40^3 \times 5000 = 3.2 \times 10^8$) the increased complexity of $L = 100$ data may mean that 500 samples is insufficient for the full training.

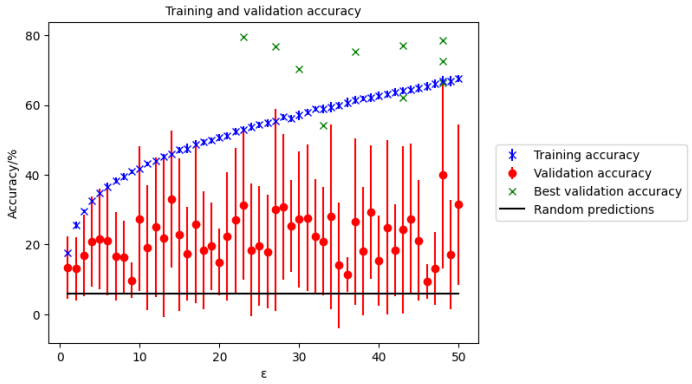


Figure 4.19: Accuracy of model for training and validation sets with best validation accuracy from each model trained plotted. Model trained using the ResNet18 network, the Adam optimiser with numerical Ψ values at $L = 100$.

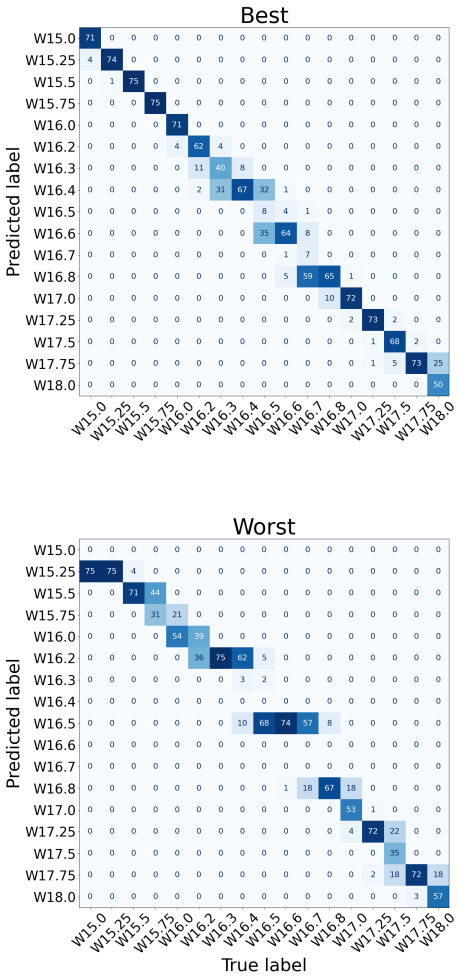


Figure 4.20: Best and worst validation confusion matrices at the ε with the best validation accuracy of the 10 trials when trained for $L = 100$ with Ψ values.

4.3.4 Comparisons of the effects of varying L on Ψ value training

Overall, training across $L = 20, 40$ and 100 (Figure 4.21), we can see that the increase in L increases the number of epochs it takes for the model to fully identify the training set. This is seen in how the training accuracy on $\varepsilon = 50$ is lower as L increases. Increasing L increases the validation accuracy when going from 20 to 40 but not from 40 to 100. While increasing L does help with increasing the validation accuracy, it is still important to factor in the increased training time. As such it is important to balance the best validation accuracy with the number of epochs it takes to train - while using the largest possible L will give a better validation accuracy, it will take less time with smaller Ls and thus can be more efficient.

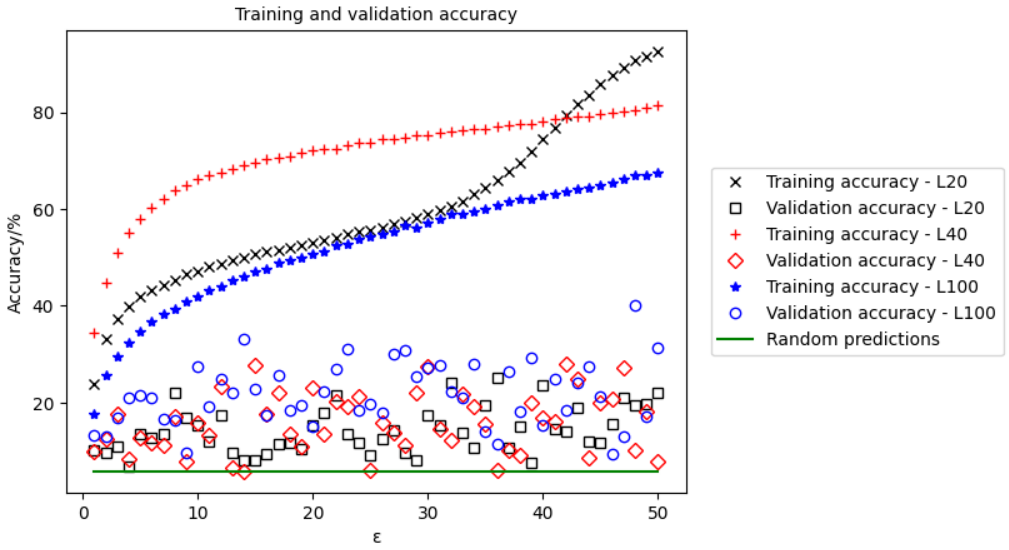


Figure 4.21: Consolidated accuracy for training and validation sets for $L = 20$, 40 and 100 model trained using Ψ values.

Plotting the training accuracy, average validation accuracy and best validation accuracy against L (Figure 4.22), we can see that increasing L leads to a decrease in training accuracy from 92.5% for $L = 20$ to 81.3% for $L = 40$ and 67.6% for $L = 100$. Again, this shows a decrease in overfitting as L increases which is ideal behaviour. Comparing the average validation accuracy for each L, we find for $L = 20$ the peak average validation accuracy is 25.1% which increases to 28.1% for $L = 40$ and then 40.1% for $L = 100$. This improvement in the average validation accuracy shows that increasing L is improving the model's ability to predict on unseen data. If we then compare the best validation accuracy we find that at first the accuracy increases between $L = 20$ and 40 from 60.0% to 85.9% which is due to less overfitting again. When L increases to 100 the best validation accuracy falls from 85.9% to 79.6%. The potential explanations for this are that the random split in the dataset when forming the training, validation and test sets and the random split in the batches was more beneficial to the training process for the $L = 40$ data than the $L = 100$ data producing a better model for the $L = 40$ dataset. Alternatively, the decrease in individual samples in the $L = 100$ dataset compared to the $L = 40$ dataset could have made it more difficult for the model to identify the categories correctly and thus produce a worse final model.

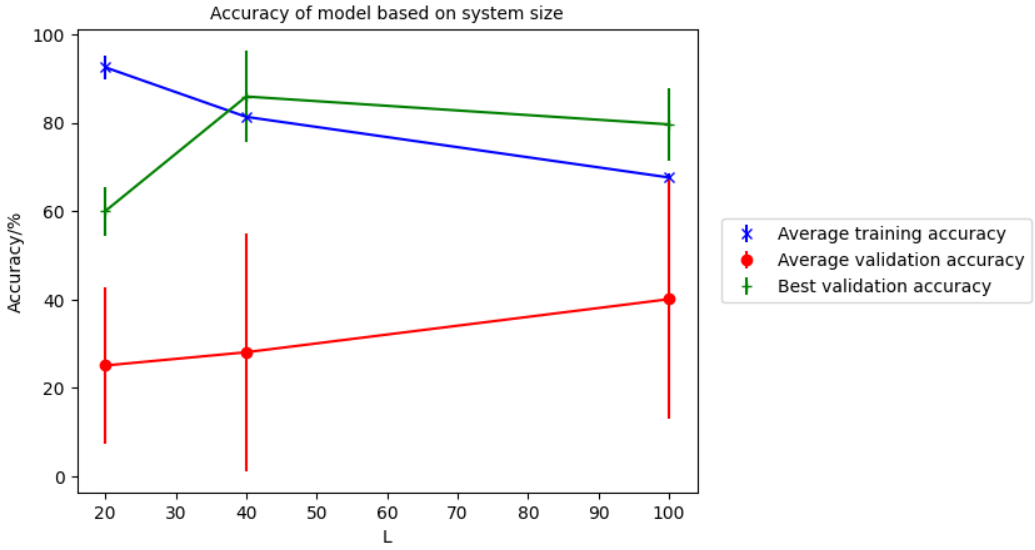


Figure 4.22: Training, average validation and best validation accuracy for $L = 20$, 40 and 100 models trained using Ψ values.

Chapter 5

Conclusions

5.1 Comparisons of training using images against numerical values

When comparing models trained using images to models trained using Ψ or $|\Psi|^2$ values, we can see that image-trained models preform similarly to the $|\Psi|^2$ models with maximum validation accuracy of the image model reaches being 46.7% (Figure 4.3a) for the model of $s = 200, L = 100$ compared to the maximum validation accuracy for the $|\Psi|^2$ model at $L = 100$ of 53.1% 4.11). When compared to the models also trained with Ψ values at $L = 100$, we find that the image models preform more poorly as the Ψ models can reach a maximum validation accuracy of 79.6% for $L = 100$ (Figure 4.19). Another factor to consider here is how the model behaves towards the end of the training. For the image models, after approximately epoch 15, the training and validation accuracy stops increasing. While the training and validation accuracy are very similar showing that there is no overfitting, this also means that the model will not improve beyond this point. Comparatively, the numerical models do not have the training accuracy level off towards the end. This means that the models still have potential to improve. When we plot the $L = 20$ numerical models, we can see the training accuracy has begun overfitting meaning that training the numerical models for more epochs will not necessarily improve the validation accuracy. By employing more methods that minimise overfitting, it may be possible to train the numerical models in order to further increase the validation accuracy. From this we can conclude that it is better to use the numerical values over images for training classification models. This makes sense as by converting the raw values into images, we are sacrificing some of the underlying information of the system in order to convert it into a visual format. In addition, the process of

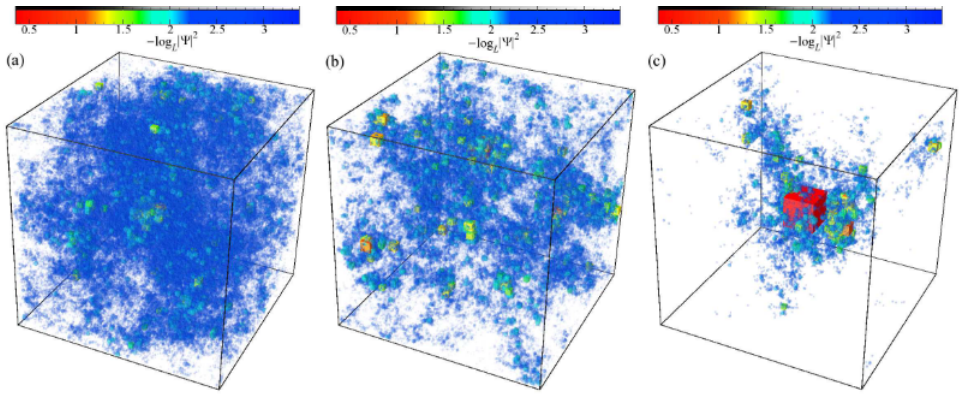


Figure 5.1: Alternative visual representation of the Anderson model. Here sites that contribute to 98% of the wavefunction norm are plotted as cube of volume $|\Psi|^2$. The colour map for this figure corresponds to the value of $-\log_L|\Psi|^2$ with $L = 120$. These plots correspond to disorders spanning the MIT with a) W = 15.0, b) W = 16.5 and c) W = 18.0 (Rodriguez et al. [2011]).

converting a numerical file into an image requires more information to be stored by the computer and as such leads to larger individual file sizes and thus more storage will be required on the machines being used for ML. The visual representation for the Anderson model we are using has been chosen for convenience. As such, we can consider other potential ways to represent the data in a 3D format (Figure 5.1) or even by converting the 3D representation into a series of 2D images by taking cross-section slices out of the 3D image and placing them in a 2D grid and seeing how that compares (Figure 5.2). Some other things to consider are the training time and the error. While the image models have a lower accuracy, they also have a lower error compared to the numerical models. In addition, the accuracy of the models trained on images stops increasing after the 15th epoch, showing that the training has completed within the first 15 epochs. As such, if we can develop a image model that can predict to similar accuracy to the best raw models, then it would be more effective to use images over raw values as the models would train in less steps, taking less time and training with a lower error.

5.2 Comparisons of training using $|\Psi|^2$ versus Ψ

Separating the raw values into Ψ and $|\Psi|^2$, we can see the impact they have on the models. Overall, we can confirm that switching from Ψ to $|\Psi|^2$, increases the training accuracy but decreases the validation accuracy - for both the average and the best validation accuracy (Figure 5.3). From this we can say that it is ideal to use Ψ values for training classification models. While the $|\Psi|^2$ models display a better training accuracy the more important factor is the validation accuracy as

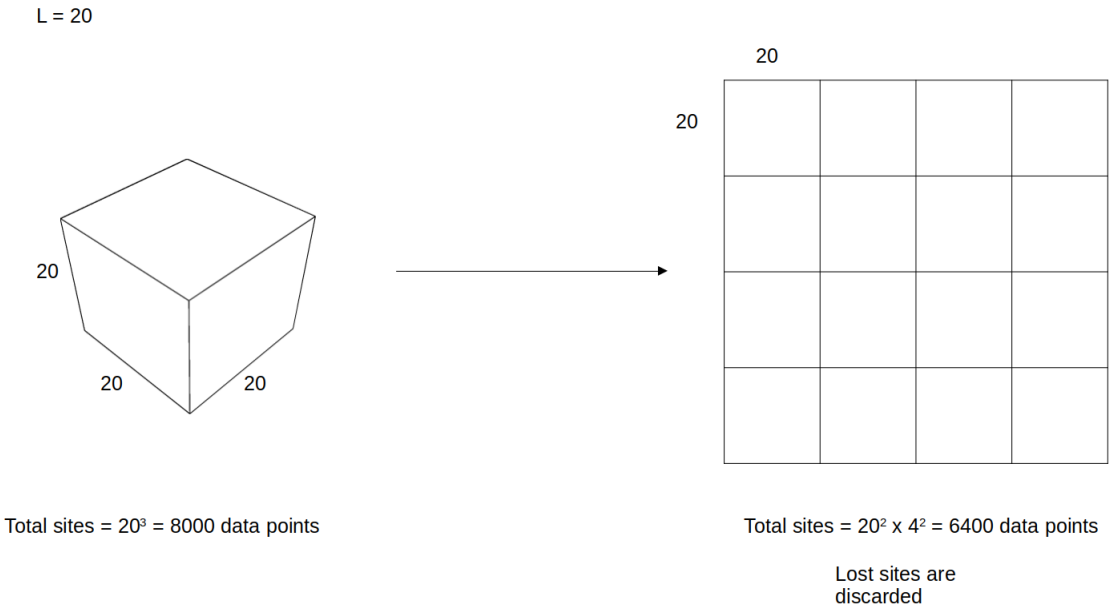
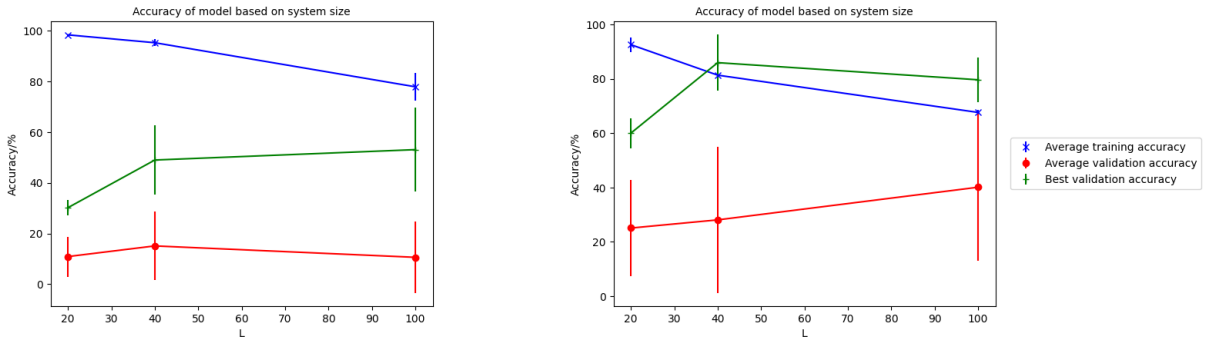


Figure 5.2: Potential method of converting 3D representation into 2D. This method discards some of the cross-section slices of the original image in order to maintain a square shape. This is so that the kernel size, padding, stride and other parameters can remain the same in the equivalent 2D model in order to try to keep similar training behaviour.

that represents how the model is training on unseen data. We want the validation accuracy to be as high as possible as that would represent a model that is effective at predicting unseen data and therefore a model that can be used to identify a new input. As such, the significantly lower average and best validation accuracy for $|\Psi|^2$ trained models means that it is more effective to use Ψ values where possible. This is representative of the phase problem. By training with the $|\Psi^2|$ values we only have access to the amplitude of the wavefunction at each point which only takes positive values whereas by training with Ψ values we also have the information of the phase of the wavefunction which can be positive or negative. The extra information of whether the phase is positive or negative is beneficial to ML algorithms as additional information provides more points of comparison for an ML model to preform categorisations.



(a) Accuracy for models trained using numerical $|\Psi|^2$ values.

(b) Accuracy for models trained using numerical $|\Psi|^2$ values.

Figure 5.3: Average training accuracy, average validation accuracy and best validation accuracy for $|\Psi|^2$ and Ψ trained at $L = 20, 40$ and 100. Models trained using the ResNet18 network and the Adam optimiser.

5.3 Overall effectiveness of models produced through this method

The model produced through this method has shown promise for predicting states of the Anderson model. When preforming classifications of $L = 40$ data between the phases of $W < W_c$ and $W > W_c$, we have attained an accuracy of 98.1% when attempting to recreate the model shown in Ohtsuki and Mano [2020]. This results is preforming slightly worse than the result attained in Ohtsuki and Mano [2020], where the authors state that their model can predict between localised and delocalised states with an accuracy $> 99\%$. The authors also state in this paper that they are expecting validation accuracy values of over 97% so our model is still preforming within expected values. Any difference in performance is most likely due to the different datasets used between the two studies. In addition, we have shown that it is possible to attain a validation accuracy $> 99\%$ by switching to the ResNet18 to increase the depth of the training without significant increases to the training time. With the ResNet18 network we were able to attain an accuracy of 99.8% showing that the ML implementation we have used here has preformed to the same level as previous studies.

Increasing the complexity of the dataset by introducing discrete categories of W, we have found that for models trained on Ψ values, we attain a peak validation accuracy $> 60\%$ for all system sizes. The model can preform at around 80% accuracy for $L = 40$or100, leading to approximately 20% of samples being categorised incorrectly. For comparison, ML classification for datasets with more then two cate-

gories shown in Bayo et al. [2022] and Çivitcioğlu et al. [2022] has produced models that categorise 5% and 10% of samples incorrectly. While both cases use less than 17 categories, with Bayo et al. [2022] using twelve categories and Çivitcioğlu et al. [2022] using seven, as the model developed in Bayo et al. [2022] has shown that despite the large number of categories, it is possible to attain $> 90\%$ accuracy, there is still improvements that can be made to the model we have developed.

There is also the idea of generalisation. While the model developed preforms well on unseen data of the same system size, as shown by the high validation accuracy attained for each system size, we find that the model preforms more poorly when the system size of the unseen data and the model are mismatched. This is due to the patterns that exist in the data that the model has identified vary with the system size of the data leading to each model identifying different key features for different values of L. This becomes apparent when testing the model for lower system sizes than the model was trained on as models trained on $L = 100$ data will predict much more poorly for $L = 40$ or 20 data. This is linked to the improved accuracy for larger systems sizes, as when the model is trained for larger system sizes, it is able to find more extensive patterns within the data to preform a more accurate classification. When the system size is then reduced, these patterns may not develop in the data, leading to the patterns that the model attempts to identify not existing and therefore leading to failed classification. When models of a lower system size are trained on data of a larger system size, the issue is that the method the data reads in will cause data points to be lost leading to key identifying features of the category being lost. As such, while the models can be used to predict categories for systems sizes that are different to the ones they were trained on, they will preform much more poorly.

5.4 Potential points of improvement and future additions to consider

From this point there are many things to consider with improving the ML methods for classifying physical systems. Firstly, training models in such a way has a large time sink - this is both due to the training process taking a large amount of time but also due to the resource requirements for training effectively. This time factor increases as L gets larger as the data volume increases by a factor of L^3. As such, this has required us to utilise GPU clusters in order to train our models - these also come with a queue time, adding to the time requirement. There is also the time

taken to generate the data, compress it/convert it into images and transfer this data onto the work clusters. All in all, there are many time factors to consider which must be account for when attempting ML work and determining what's the best level of data to be used for training.
The next factor to consider is memory requirements. Training models requires a substantial amount of RAM in order to process the data and the training being done on it. For example, when training using $\tau_{17,20}$ and $\tau_{17,40}$, we were able to utilise all 5000 samples in the training process. When we attempted to train on $\tau_{17,100}$, the data volume increase by a factor of L^3 meant that even with the full 40 GB GPU RAM available on Sulis, we were unable to use the full 5000 samples and had to reduce the dataset to 500 samples. In addition, the data and copies of the models being stored must also be considered in the disk space they take up. The total $\tau_{17,20}$ dataset takes up 5.5 GB of memory to store the compressed numerical files. The $\tau_{17,40}$ dataset increases this to 43.6 GB of memory for its numerical files. In total the memory taken up by all datasets used is 190.8 GB. If we wanted to use the full 5000 samples for $\tau_{17,100}$ this memory requirement goes to 804.8 GB as the memory requirements also increase by a factor of L^3 as well as s^2 for images. These factors must also be considered when trying to determine how much data and of what size should be used for training.
It can also be important to consider the systems being used for training. While storing the random number seed used while the model is training is helpful in creating reproducible models, external factors such as memory partitions on hard drives and how the machine chooses to separate data during training will impact how the model achieves its training. This can be noticed when trying to transfer and re-run a training process across our HPC machines Avon and Sulis. As such, it is important to consider what the properties of the different machines being used to train ML models as trying to keep such parameters constant across all machines in a project team will be difficult. The usage of a work machine with consistent parameters has helped avoid this issue but being aware of the repeatability issues is important to remember. We chose to mainly focus on using Sulis as the available resources of Sulis allowed training to be completed slightly quicker than on Avon. While the training is not guaranteed to be reproducible, the models, once trained, will continue to behave the same and can therefore be transferred to many other system.

In the future, we can consider the creation of a regression based classifier. By stretching the prediction region over the disorder range of 15.0 to 18.0 and scaling that to the 0 to 1 scale of the regression and training on that, we should be able to

create a model that can predict to any disorder value on the scale of 15.0 to 18.0 rather than just the defined categories set by our training process.

Appendix A

Full confusion matrices

Here we plot the full confusion matrices from which we took the best and worst confusion matrices for use in Section 4.2 and 4.3.

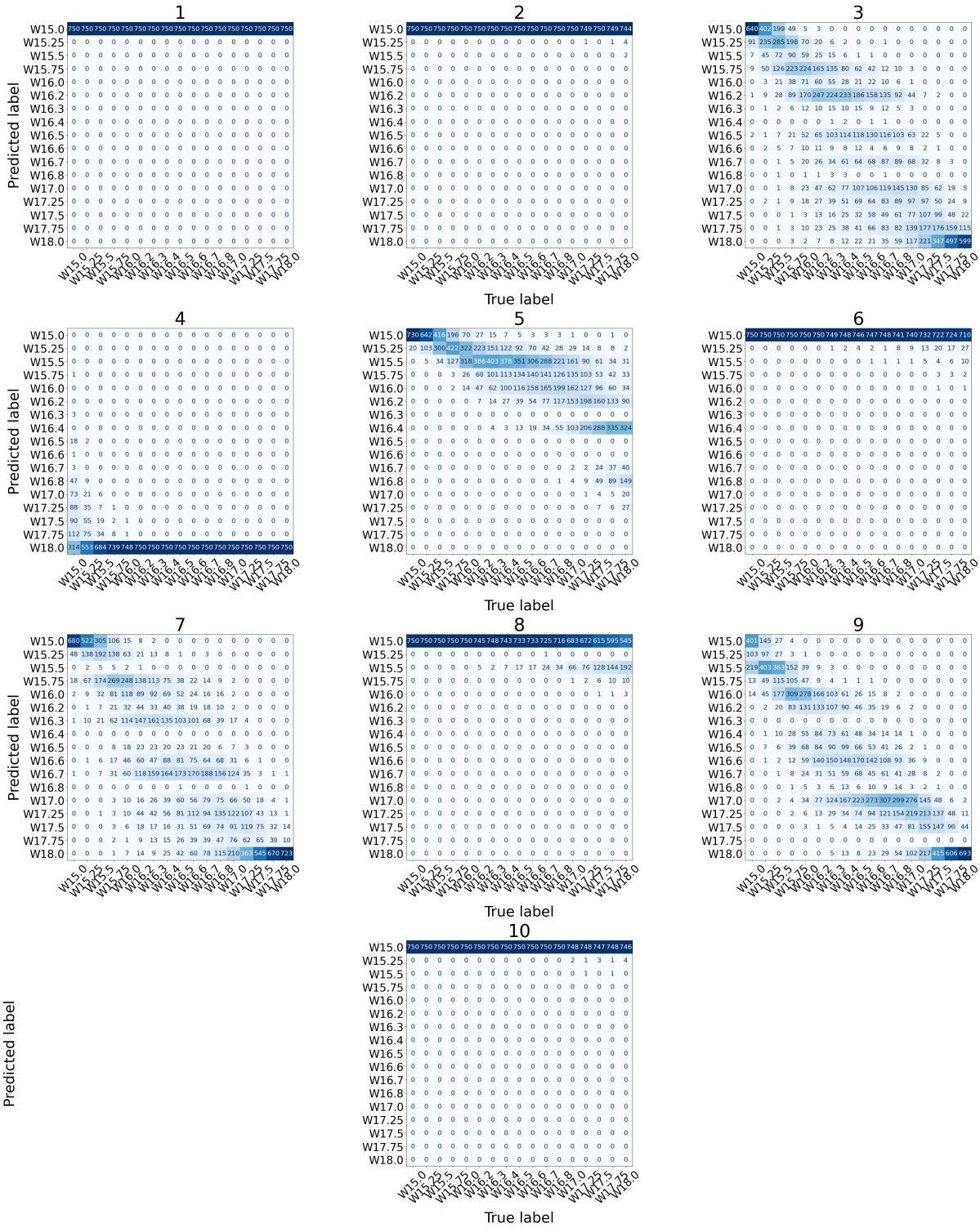


Figure A.1: Confusion matrices for all ten trials at $\varepsilon = 8$ for models trained using the ResNet18 network, the Adam optimiser with numerical $|\Psi|^2$ values for $L = 20$.

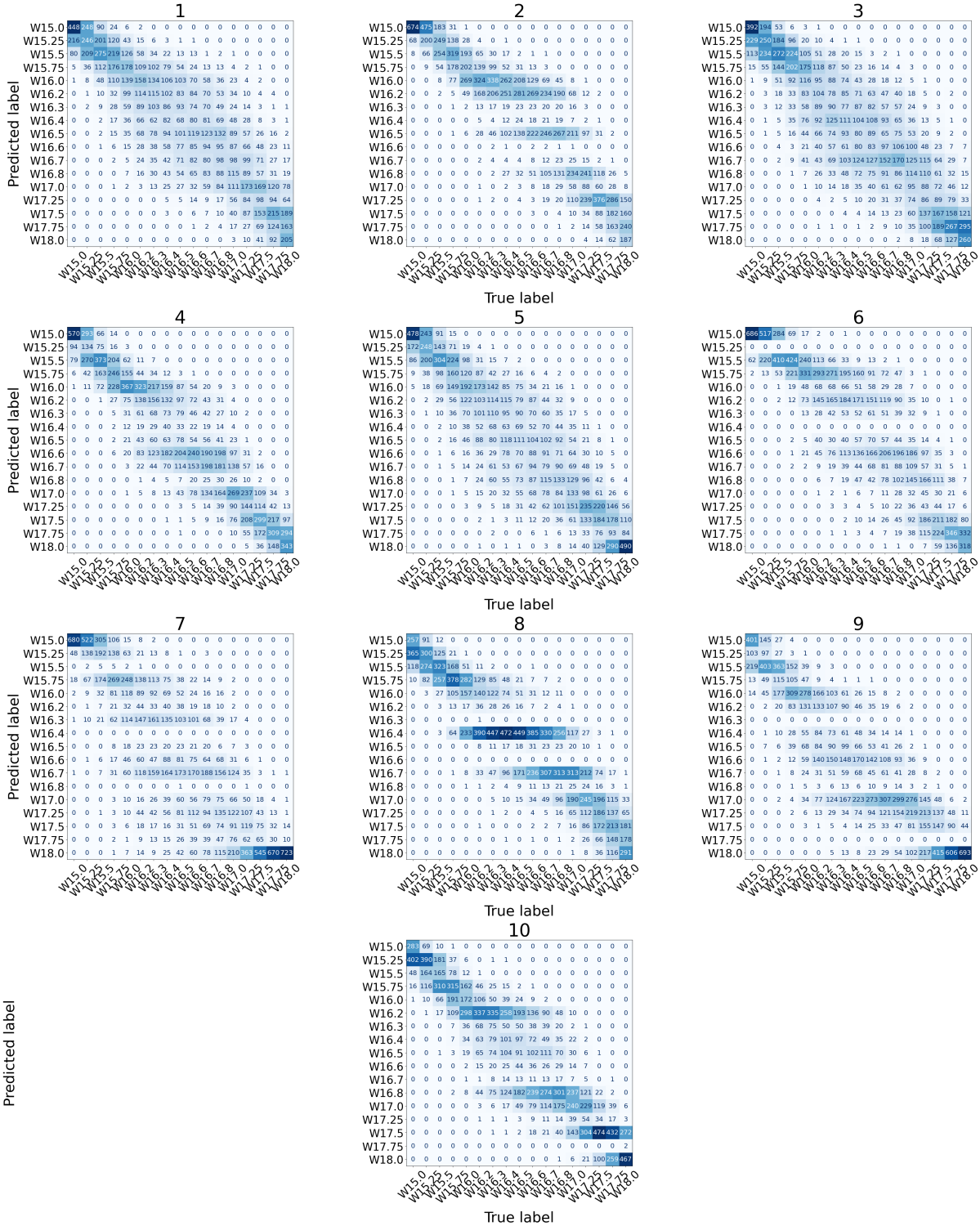


Figure A.2: Confusion matrices for all ten trials at the ε with the best validation accuracy for models trained using the ResNet18 network, the Adam optimiser with numerical $|\Psi|^2$ values for $L = 20$.

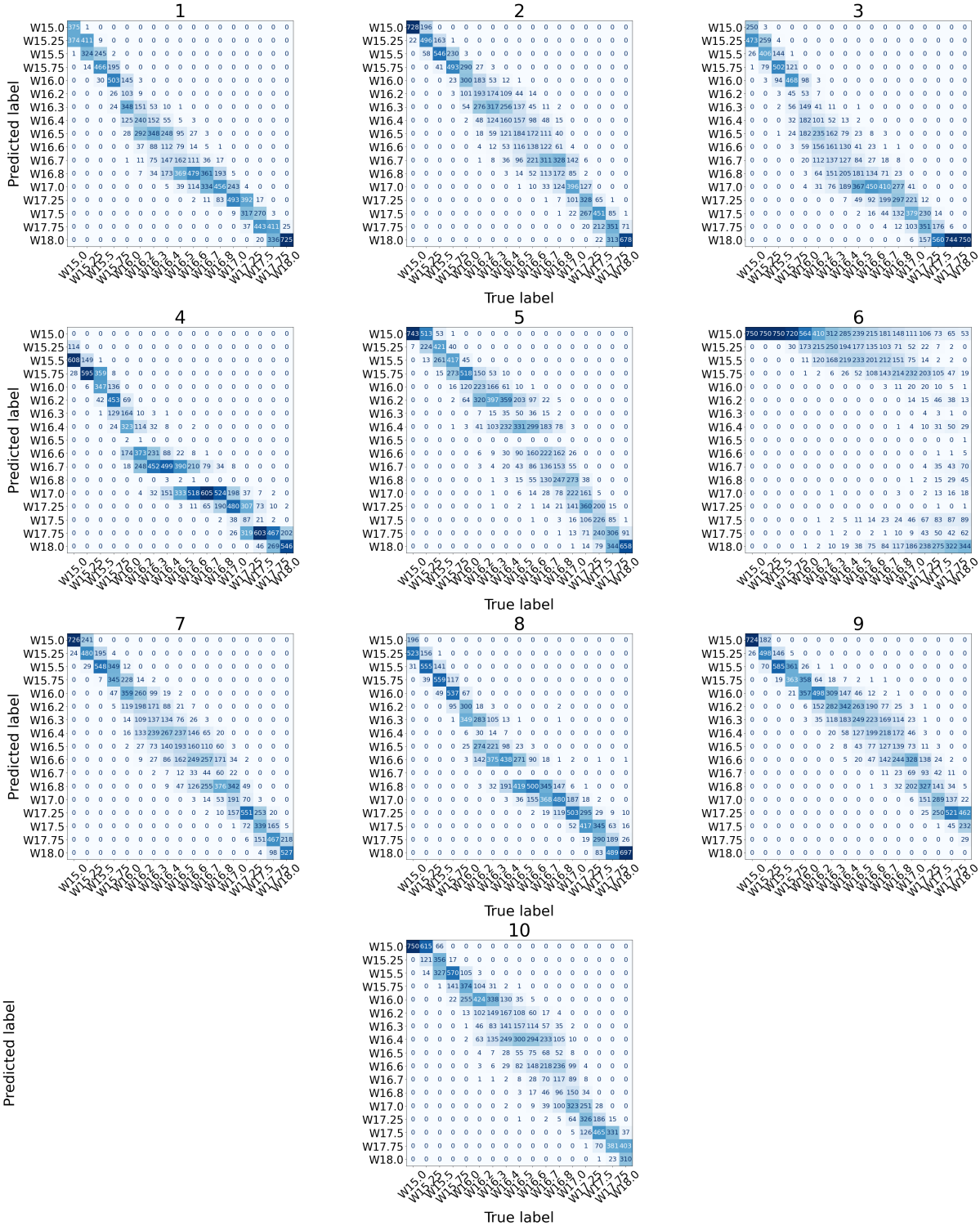


Figure A.3: Confusion matrices for all ten trials at the ε with the best validation accuracy for models trained using the ResNet18 network, the Adam optimiser with numerical $|\Psi|^2$ values for $L = 40$.

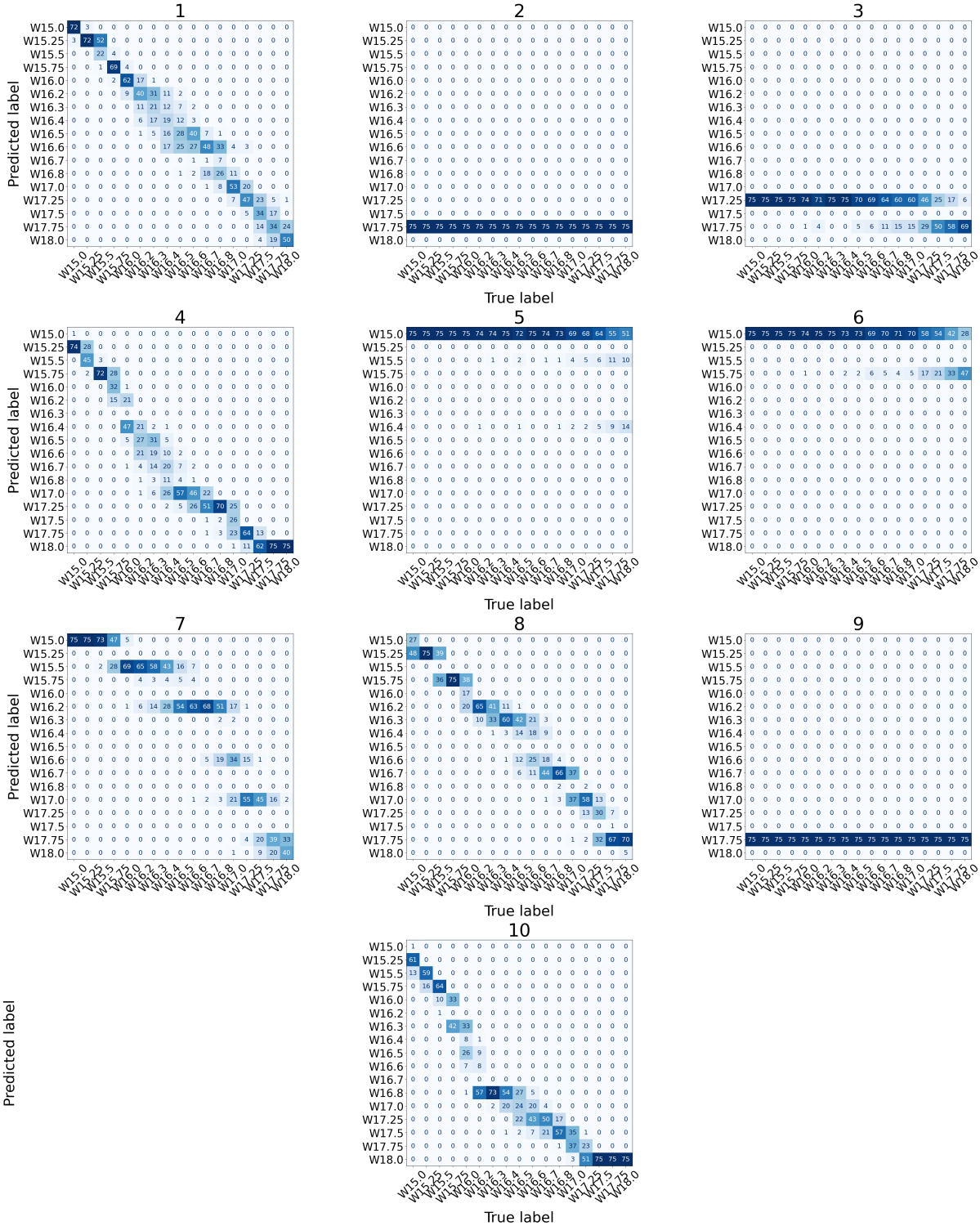


Figure A.4: Confusion matrices for all ten trials at the ε with the best validation accuracy for models trained using the ResNet18 network, the Adam optimiser with numerical $|\Psi|^2$ values for $L = 100$.

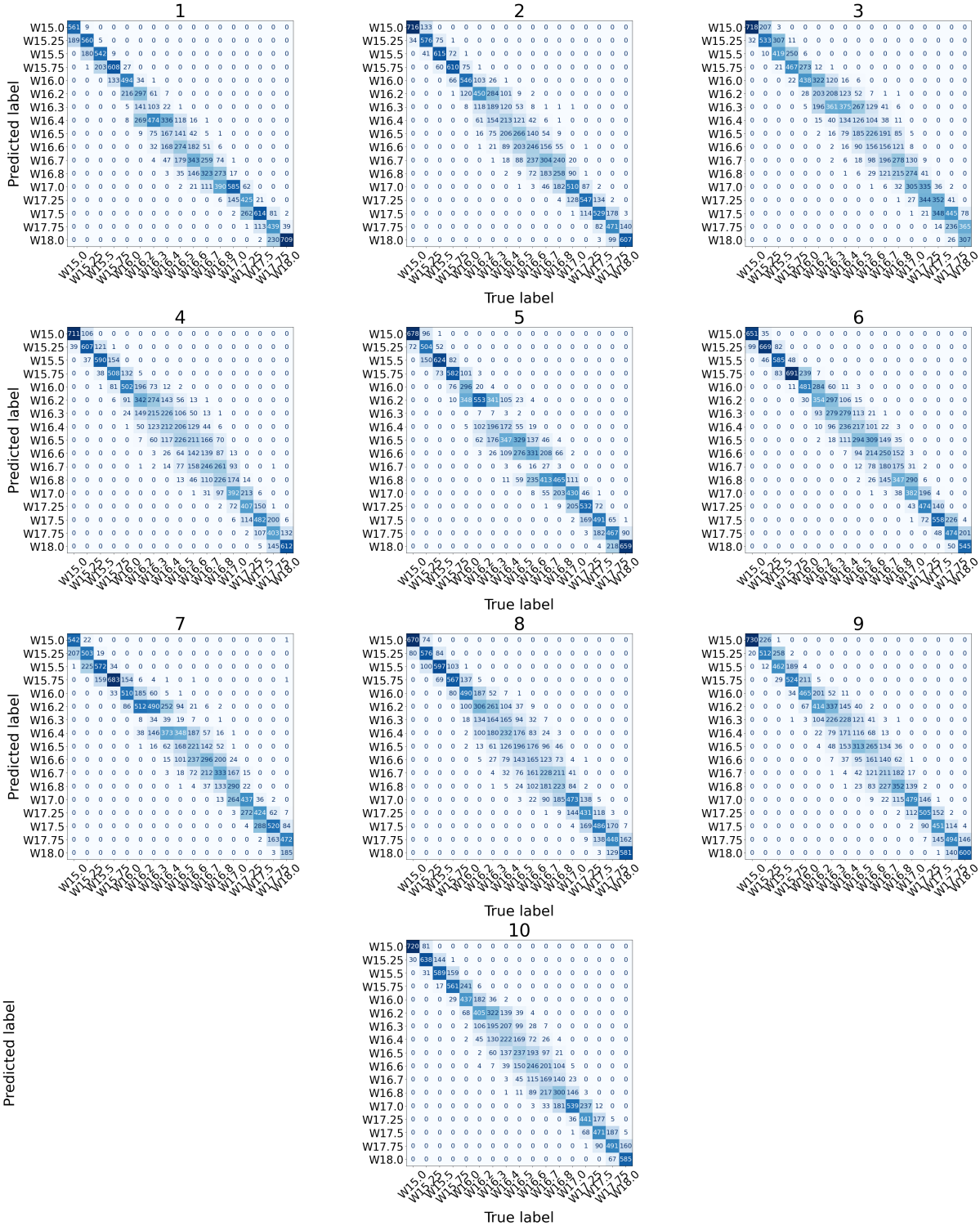


Figure A.5: Confusion matrices for all ten trials at the ε with the best validation accuracy for models trained using the ResNet18 network, the Adam optimiser with numerical Ψ values for $L = 20$.

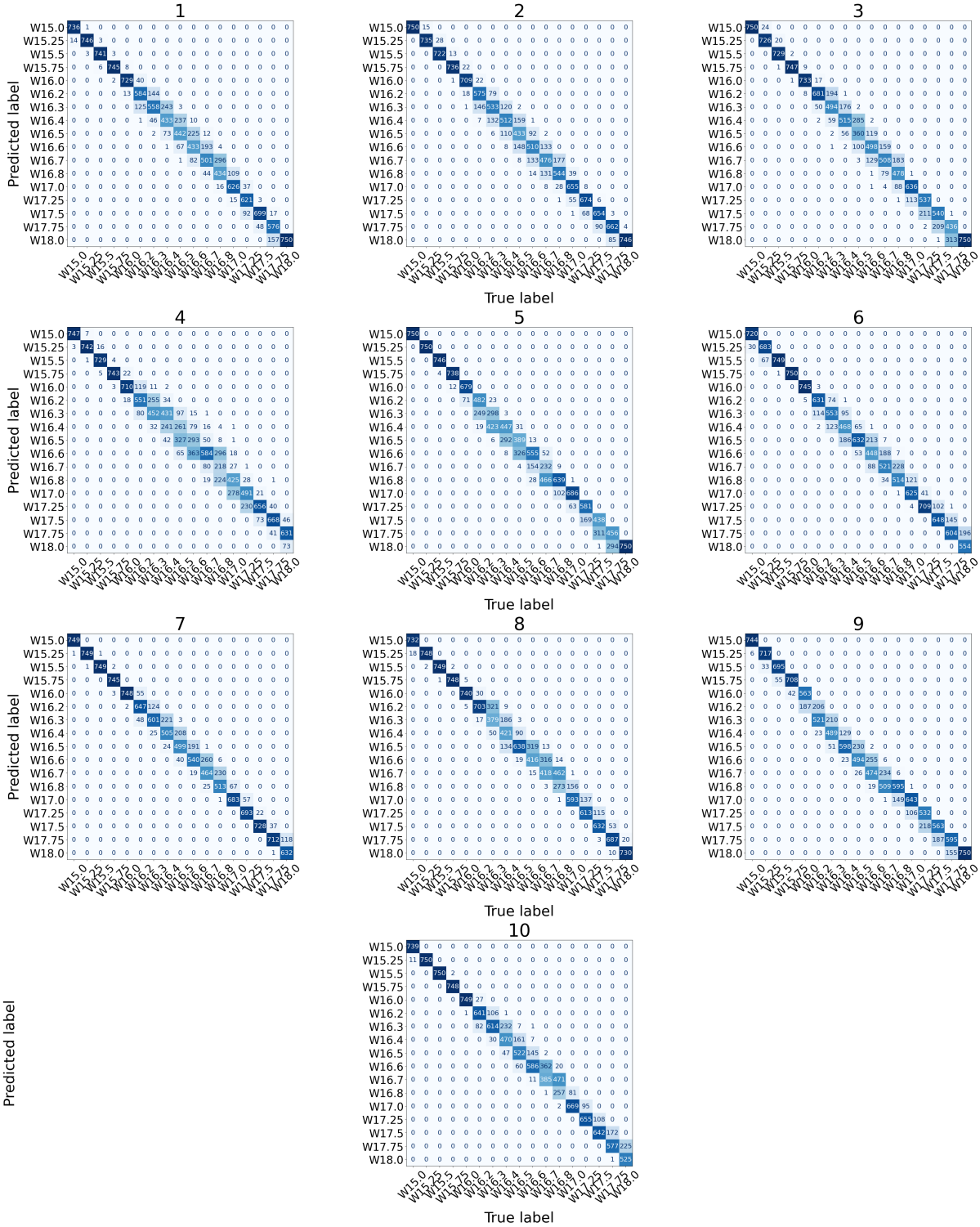


Figure A.6: Confusion matrices for all ten trials at the ε with the best validation accuracy for models trained using the ResNet18 network, the Adam optimiser with numerical Ψ values for $L = 40$.

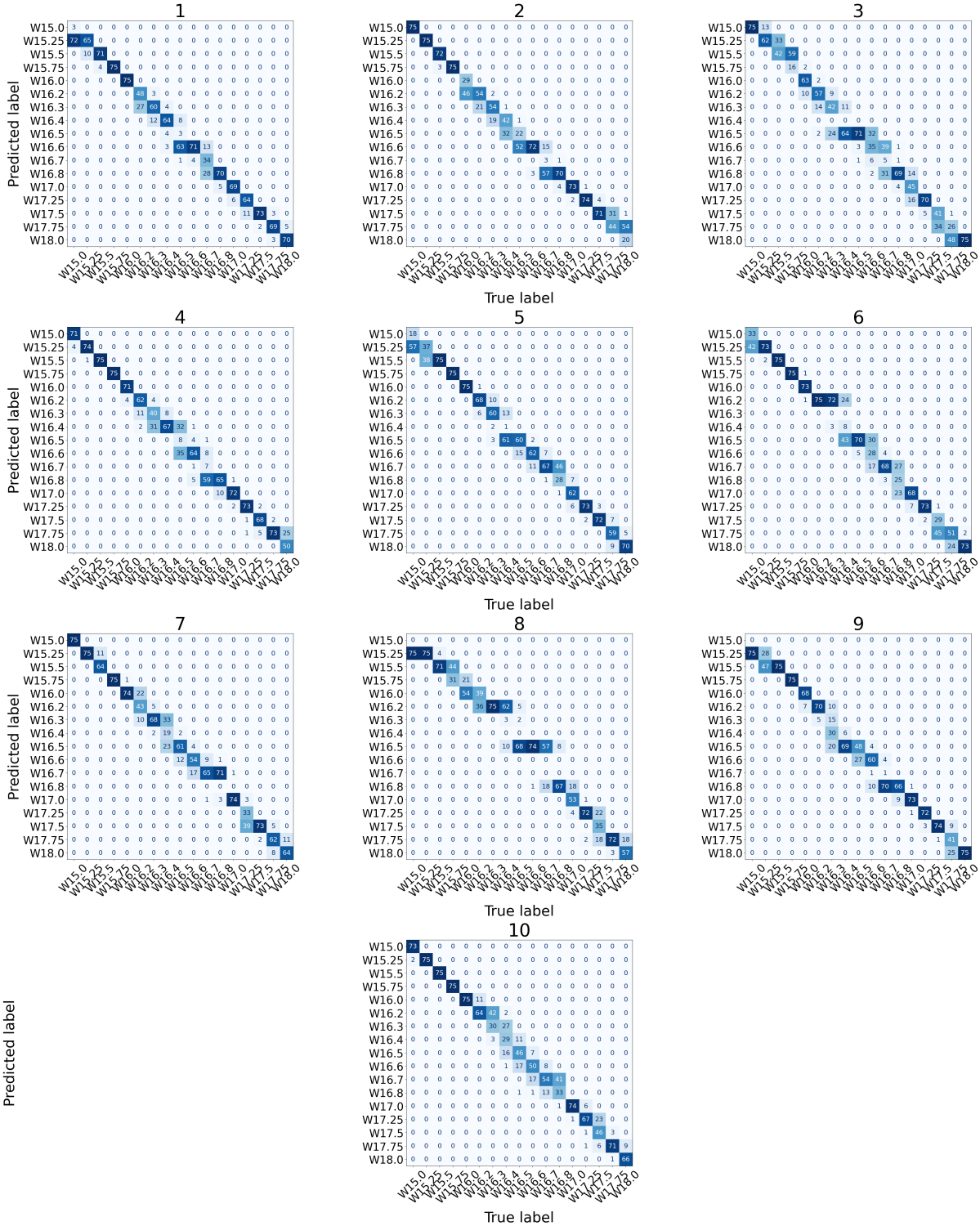


Figure A.7: Confusion matrices for all ten trials at the ε with the best validation accuracy for models trained using the ResNet18 network, the Adam optimiser with numerical Ψ values for $L = 100$.

Bibliography

E. Abrahams, P. W. Anderson, D. C. Licciardello, and T. V. Ramakrishnan. Scaling theory of localization: Absence of quantum diffusion in two dimensions. *Phys. Rev. Lett.*, 42:673–676, Mar 1979. doi: 10.1103/PhysRevLett.42.673. URL https://link.aps.org/doi/10.1103/PhysRevLett.42.673.

P. W. Anderson. Absence of diffusion in certain random lattices. *Phys. Rev.*, 109:1492–1505, Mar 1958. doi: 10.1103/PhysRev.109.1492. URL https://link.aps.org/doi/10.1103/PhysRev.109.1492.

Isaac Asimov. Runaround. *Astounding science fiction*, 29(1):94–103, 1942.

Djénabou Bayo, Andreas Honecker, and Rudolf A. Römer. Machine learning the 2d percolation model. *Journal of Physics: Conference Series*, 2207(1):012057, mar 2022. doi: 10.1088/1742-6596/2207/1/012057. URL https://dx.doi.org/10.1088/1742-6596/2207/1/012057.

Matthias Bollhöfer and Yvan Notay. Jadamilu: a software code for computing selected eigenvalues of large sparse symmetric matrices. *Computer Physics Communications*, 177(12):951–964, 2007. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2007.08.004. URL https://www.sciencedirect.com/science/article/pii/S0010465507003700.

Baird Brandow. The physics of mott electron localization. *Journal of Alloys and Compounds*, 181(1):377–396, 1992. ISSN 0925-8388. doi: https://doi.org/10.1016/0925-8388(92)90334-6. URL https://www.sciencedirect.com/science/article/pii/0925838892903346. Proceedings of the 19th Rare Earth Research Conference.

Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355 (6325):602–606, 2017. doi: 10.1126/science.aag2302. URL https://www.science.org/doi/abs/10.1126/science.aag2302.

Juan Carrasquilla and Roger G. Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, May 2017. ISSN 1745-2481. doi: 10.1038/nphys4035. URL https://doi.org/10.1038/nphys4035.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006. ISBN 978-0-262-03358-9.

Shu Cheng, Fei He, Huai Zhang, Ka-Di Zhu, and Yaolin Shi. Machine learning percolation model, 2021. URL https://doi.org/10.48550/arXiv.2101.08928.

F. Chollet. *Deep Learning with Python*. Manning Publications Company, 2017. ISBN 9781617294433. URL https://books.google.co.uk/books?id=Yo3CAQAACAAJ.

community contributed Github repository. Imagemagick open source software, Oct 2023. URL https://github.com/ImageMagick/ImageMagick.

Jane Cullum and Ralph A. Willoughby. A survey of lanczos procedures for very large real 'symmetric' eigenvalue problems. *Journal of Computational and Applied Mathematics*, 12-13:37–60, 1985. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(85)90006-8. URL https://www.sciencedirect.com/science/article/pii/0377042785900068.

Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. *Supervised Learning*, pages 21–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-75171-7. doi: 10.1007/978-3-540-75171-7_2. URL https://doi.org/10.1007/978-3-540-75171-7_2.

Vladimir Dobrosavljevic, Nandini Trivedi, and James M. Valles, Jr. *Conductor-Insulator Quantum Phase Transitions*. Oxford University Press, 06 2012. ISBN 9780199592593. doi: 10.1093/acprof:oso/9780199592593.001.0001. URL https://doi.org/10.1093/acprof:oso/9780199592593.001.0001.

P. P. Edwards, R. L. Johnston, C. N. R. Rao, D. P. Tunstall, P. P. Edwards, Johnston R. L., C. N. R. Rao, D. P. Tunstall, and F. Hensel. The metal–insulator transition: a perspective. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356(1735):5–22, 1998. doi: 10.1098/rsta.1998.0146. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsta.1998.0146.

Python Software Foundation. Python 3.12.0 documentation, Oct 2023. URL https://docs.python.org/3/index.html.

PyTorch Foundation. Pytorch 1.9.0 api documentation, Jun 2021. URL https://pytorch.org/docs/1.9.0/generated/torch.nn.CrossEntropyLoss.html.

Alexander L. Fradkov. Early history of machine learning. *IFAC-PapersOnLine*, 53(2):1385–1390, 2020. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2020.12.1888. URL https://www.sciencedirect.com/science/article/pii/S2405896320325027. 21st IFAC World Congress.

Roman Geus. *The Jacobi-Davidson algorithm for solving large sparse symmetric eigenvalue problems with application to the design of accelerator cavities*. PhD thesis, ETH Zurich, 2002.

Zoubin Ghahramani. *Unsupervised Learning*, pages 72–112. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-28650-9. doi: 10.1007/978-3-540-28650-9_5. URL https://doi.org/10.1007/978-3-540-28650-9_5.

Steven M. Girvin and Kun Yang. *Modern Condensed Matter Physics*. Cambridge University Press, 2019. doi: 10.1017/9781316480649.

Chenguang Guan and Xingyue Guan. A brief introduction to anderson localization. *Department of Physics, Massachusetts Institute of Technology*, 2019.

Michael Haenlein and Andreas Kaplan. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review*, 61(4):5–14, 2019. doi: 10.1177/0008125619864925. URL https://doi.org/10.1177/0008125619864925.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://doi.org/10.48550/arXiv.1512.03385.

Mohamed Hibat-Allah, Roger G. Melko, and Juan Carrasquilla. Investigating topological order using recurrent neural networks. *Phys. Rev. B*, 108:075152, Aug 2023. doi: 10.1103/PhysRevB.108.075152. URL https://link.aps.org/doi/10.1103/PhysRevB.108.075152.

Masatoshi Imada, Atsushi Fujimori, and Yoshinori Tokura. Metal-insulator transitions. *Rev. Mod. Phys.*, 70:1039–1263, Oct 1998. doi: 10.1103/RevModPhys.70.1039. URL https://link.aps.org/doi/10.1103/RevModPhys.70.1039.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://doi.org/10.48550/arXiv.1412.6980.

W. Kohn. An essay on condensed matter physics in the twentieth century. *Rev. Mod. Phys.*, 71:S59–S77, Mar 1999. doi: 10.1103/RevModPhys.71.S59. URL https://link.aps.org/doi/10.1103/RevModPhys.71.S59.

John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Magazine*, 27(4):12, Dec. 2006. doi: 10.1609/aimag.v27i4.1904. URL https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1904.

Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, may 2019. doi: 10.1016/j.physrep.2019.03.001. URL https://doi.org/10.1016%2Fj.physrep.2019.03.001.

Adwait Naravane and Nilmani Mathur. Semi-supervised learning of order parameter in 2d ising and xy models using conditional variational autoencoders, 2023. URL https://doi.org/10.48550/arXiv.2306.16822.

Rajendran Nirthika, Siyamalan Manivannan, Amirthalingam Ramanan, and Ruixuan Wang. Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study. *Neural Computing and Applications*, 34(7): 5321–5347, Apr 2022. ISSN 1433-3058. doi: 10.1007/s00521-022-06953-8. URL https://doi.org/10.1007/s00521-022-06953-8.

Albert BJ Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622. New York, NY, 1962.

Tomi Ohtsuki and Tomohiro Mano. Drawing phase diagrams of random quantum systems by deep learning the wave functions. *Journal of the Physical Society of Japan*, 89(2):022001, feb 2020. doi: 10.7566/jpsj.89.022001. URL https://doi.org/10.7566%2Fjpsj.89.022001.

Zakaria Patel, Ejaaz Merali, and Sebastian J Wetzel. Unsupervised learning of rydberg atom array phase diagram with siamese neural networks. *New Journal of Physics*, 24(11):113021, nov 2022. doi: 10.1088/1367-2630/ac9c7a. URL https://dx.doi.org/10.1088/1367-2630/ac9c7a.

Raffaele Pugliese, Stefano Regondi, and Riccardo Marini. Machine learning-based approach: global trends, research directions, and regulatory standpoints. *Data Science and Management*, 4:19–29, 2021. ISSN 2666-7649. doi: https://doi.org/10.1016/j.dsm.2021.12.002. URL https://www.sciencedirect.com/science/article/pii/S2666764921000485.

Alberto Rodriguez, Louella J. Vasquez, Keith Slevin, and Rudolf A. Römer. Multifractal finite-size scaling and universality at the anderson transition. *Physical Review B*, 84(13), October 2011. ISSN 1550-235X. doi: 10.1103/physrevb.84.134209. URL http://dx.doi.org/10.1103/PhysRevB.84.134209.

F. Rosenblatt. *Two Theorems of Statistical Separability in the Perceptron: (Project Para) Contract Nonr-2381(00).* Cornell Aeronautical Laboratory, Inc. Cornell Aeronautical Laboratory, 1958. URL https://books.google.co.uk/books?id=5Q0-AQAAIAAJ.

Sindhunil Barman Roy. *Mott Insulators*. 2053-2563. IOP Publishing, 2019. ISBN 978-0-7503-1596-8. doi: 10.1088/2053-2563/ab16c9. URL https://dx.doi.org/10.1088/2053-2563/ab16c9.

Olaf Schenk, Matthias Bollhöfer, and Rudolf A. Römer. On large-scale diagonalization techniques for the anderson model of localization. *SIAM Review*, 50(1):91–112, 2008a. doi: 10.1137/070707002. URL https://doi.org/10.1137/070707002.

Olaf Schenk, Matthias Bollhöfer, and Rudolf A. Römer. On large-scale diagonalization techniques for the anderson model of localization. *SIAM Review*, 50(1):91–112, 2008b. doi: 10.1137/070707002. URL https://doi.org/10.1137/070707002.

Akinori Tanaka, Akio Tomiya, and Koji Hashimoto. *Deep Learning and Physics*. Springer, 2021. ISBN 978-981-33-6108-9.

D J Thouless. A relation between the density of states and range of localization for one dimensional random systems. *Journal of Physics C: Solid State Physics*, 5(1):77, jan 1972. doi: 10.1088/0022-3719/5/1/010. URL https://dx.doi.org/10.1088/0022-3719/5/1/010.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition, 2018. URL https://doi.org/10.48550/arXiv.1711.11248.

A. M. Turing. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950. ISSN 0026-4423. doi: 10.1093/mind/LIX.236.433. URL https://doi.org/10.1093/mind/LIX.236.433.

Chukwudubem Umeano, Annie E. Paine, Vincent E. Elfving, and Oleksandr Kyrienko. What can we learn from quantum convolutional neural networks?, 2023. URL https://doi.org/10.48550/arXiv.2308.16664.

Matthias Vojta. Quantum phase transitions. *Reports on Progress in Physics*, 66(12):2069, nov 2003. doi: 10.1088/0034-4885/66/12/R01. URL https://dx.doi.org/10.1088/0034-4885/66/12/R01.

Thomas Vojta. Disorder in quantum many-body systems. *Annual Review of Condensed Matter Physics*, 10(1):233–252, mar 2019. doi: 10.1146/annurev-conmatphys-031218-013433. URL https://doi.org/10.1146%2Fannurev-conmatphys-031218-013433.

Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012. URL https://doi.org/10.48550/arXiv.1212.5701.

Burak Çivitcioğlu, Rudolf A. Römer, and Andreas Honecker. Machine learning the square-lattice ising model. *Journal of Physics: Conference Series*, 2207(1): 012058, March 2022. ISSN 1742-6596. doi: 10.1088/1742-6596/2207/1/012058. URL http://dx.doi.org/10.1088/1742-6596/2207/1/012058.