Third Year Project in Physics
6CCP3131

An Investigation into Water-Ice Simulation Models for QLL Analysis

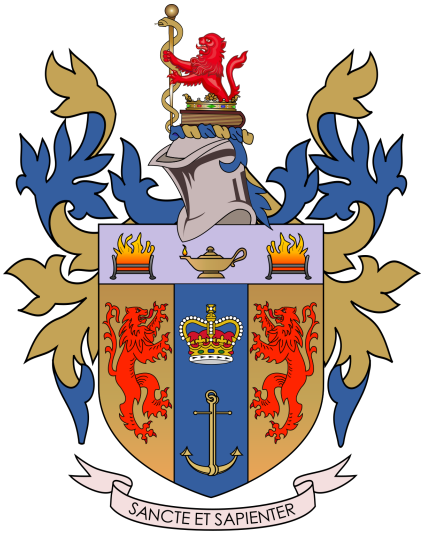
21st April 2022

Student:
Arnav Avad
K19077318

Collaborators:
Bernado Atolin, Sam Male,
Shyam Kumar Rajput

Supervisor:
Professor Carla Molteni

Assistant Supervisor:
Jihong Shi



Department of Physics
King's College London

Abstract

This paper describes the machine learning algorithm, DeepIce, created to distinguish between different phases of ice. The design of the algorithm is also shown. The Steinhardt parameters are mentioned as a precursor to DeepIce and how it sought to solve the same problem using only spherical harmonics compared to the four different neural networks used by DeepIce. To utilise this machine learning algorithm, the mW potential was used to simulate data with LAMMPS in order to train DeepIce's artificial neural network and predict phases of H_2O when presented with a simulated slab of ice containing quasi-liquid layers (QLLs). Examples of various tools including VMD and its plugins are discussed as a means to visualise DeepIce's predictions and observe radial pair distribution functions and diffusion coefficients of the different phases. Furthermore, the results produced using the mW potential to calculate the thickness of the QLLs are analysed both visually and numerically. They are also compared against results produced by the TIP4P/Ice model from earlier reports regarding DeepIce. DeepIce was used to show ice nucleation at a molecular level. The implementation of DeepIce as well as the analysis of QLLs were the aims of the paper and, hence, these were achieved to show results such as the QLL thickness of the Basal < Primary Prismatic < Cubic slabs.

Contents

1 Introduction

Computational physics has allowed for multiple calculations to be performed in a short period of time. Especially, machine learning has played a large role throughout scientific research. This has enabled programs to be written using python specific packages such as TensorFlow and Keras for making predictions based on prepared training data. DeepIce is an example of such a program which aims to differentiate between the various phases of ice and water accurately.

Ice is one of the most essential molecules on this planet that allows various phenomena to occur such as effects on weather patterns. It is a well-researched and ever improving area within science. Currently, 21 different phases with distinguishable geometric morphologies of ice have been identified. 18 crystalline and 3 non-crystalline amorphous structures are considered as the phases of ice, but, as research continues the figures vary from source to source.[45] Throughout this report, only two phases of ice were considered. Cubic ice (also known as Ic) and hexagonal ice (also known as Ih) were used. Being the most abundant phase of ice, hexagonal ice has C_6 symmetry. The structure of the molecule is extremely close to that of a perfect tetrahedral structure with the oxygen-oxygen-oxygen (O-O-O) bond angles being 109.47° compared to the 109.5° of the tetrahedral structure. Furthermore, hexagonal ice seems to be stable down to the temperature of 5K and up to the pressure of 210MPa. The corresponding Bravais lattice belongs to the space group 194.[15] On the other hand, cubic ice was first discovered in 1942 and first synthesised in its pure form in 2020. Its O-O-O angles are also 109.47° though it has $4C_3$ symmetry. This particular phase is only stable between temperatures of 130K and 220K, after which hexagonal ice starts forming. Also, the space group 227 corresponds to the Bravais lattice.[143]

An important component of ice is its quasi-liquid layers on its surfaces. Quasi-liquid layers (QLLs) is a term given to the layer of material that have melting temperatures below the main bulk temperature. They are not definitively liquid or solid, yet they exhibit properties consistent with both liquids and solids.[40] Surface pre-melting is the process involved with QLL formation. QLLs on ice were first hypothesised by Michael Faraday in 1859 in the published book, 'Experimental Research in Chemistry and Physics'.[27] QLLs and their formations have been shown to cause phenomena including thundercloud electrification,[40] ozone depleting species formation[26] as well as simply the idea of reduction of friction on ice surfaces[34].[28] These features have also allowed ice to be a large part of winter sports. Differentiating between these phases of ice was the main purpose of this project in the pursuit of further analysing QLLs.

Numerous methods have been developed in order to differentiate between ice and water. The Steinhardt Order Parameters, GCIceNet,[24] Geiger-Dellago neural network[20] and DeepIce[19] are all examples of these. The Steinhardt parameters were first introduced by Paul Steinhardt in 1983. They allow for crystalline phases to be distinguished from liquids by analysing and comparing the coherence of the orientational order between the molecules and their nearest neighbours through utilising spherical harmonics. The mathematical representation is shown in equations 1 and 2.

$$q_l(i) = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} |q_{lm}(i)|^2} \tag{1}$$

$$q_{lm}(i) = \frac{1}{N_b(i)} \sum_{j=1}^{N_b(i)} Y_{lm}\left(\mathbf{r}_{ij}\right) \tag{2}$$

In the equations above, all the variables are defined as follows: l is an integer parameter, N_b is the number of nearest neighbours, Y_{lm} are the spherical harmonics and $\mathbf{r}_{ij}$ is the vector between molecules i and j.[25]

These set of equations can differentiate between a solid, where the constituent molecules are ordered, and a liquid, where the molecules tend to not be regularly ordered. Steinhardt parameters boasts advantages such as the fact that no reference frame definition is required. Also, the

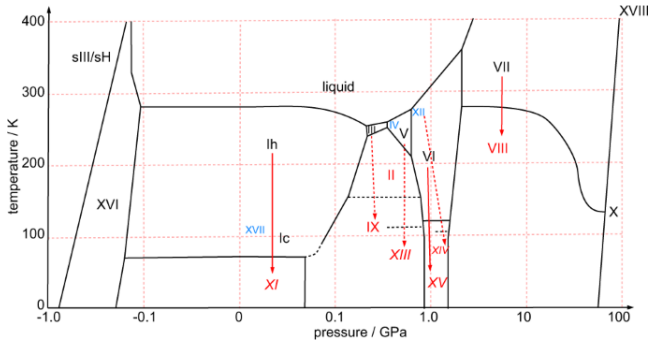


Figure 1: The phase diagram for H_2O showing the different phases and the conditions requires for their existence.[22]

process is time efficient to perform and it was further modified by Lechner and Dellago to use local average parameters to increase the accuracy of the results. Equally, the disadvantages of the Steinhardt parameters are that the definition of a nearest neighbour molecule is ambiguous which can lead to widely varying results. The outputted results also are shown to be unreliable when thermal fluctuations and elastic deformations are modelled. Also, compared to DeepIce, the accuracy is low.[19] In addition, the fact that no specific crystalline structure dependencies are necessary, means that crystalline phases cannot be distinguished from one another.[2529]

The objective of this report was to outline the resulting predictions made by DeepIce using the mW potential and analysing the suitability of this model when training the neural network to distinguish between water-like and ice-like molecules in multiphase systems. To achieve this, the $1\bar{1}00$ (Primary Prism) and 0001 (Basal) plane of hexagonal ice and the 100 plane of cubic ice were considered. These slabs of ice used are shown above in figure 2 Furthermore, questions such as how the molecules in a multiphase system change in terms of the temperature and the phase were investigated. Quantities such as the number of water molecules and the implied QLL thicknesses were calculated. Previously, the DeepIce neural network was used with the TIP4P/Ice model;[38] applying the network with different potentials should test the versatility of the program as well as highlight the differences when using a different model.

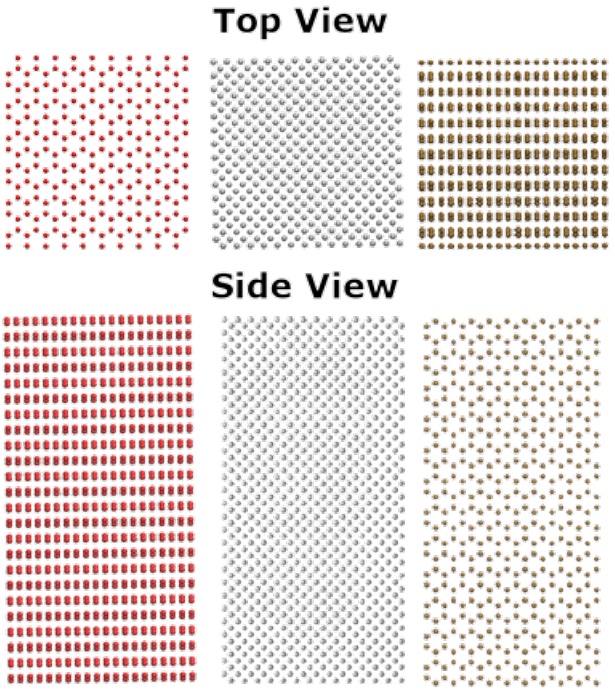


Figure 2: Images showing the top and side view of the slabs of ice used. Basal is shown in red, Cubic is shown in silver and Primary Prism is shown in gold.

2 Simulating & Visualising Molecular Dynamics

Throughout the project, several mathematical and computational tools were used to simulate and visualise molecular structures. Although DeepIce was the main focus for predicting water and ice molecules, data had to be simulated, checked, and portrayed.

2.1 The mW Potential

The mW potential is a model which imitates tetrahedrality between carbon and silicon. Hence, it is a course-grained model for water. It was used to simulate the training data sets necessary to train DeepIce and also to simulate the slab on which DeepIce's prediction was made on. The mW potential models water molecules as monoatomic particles with short-ranged interactions. Electrostatic interactions and hydrogen atoms were not considered within the model. The mathematical definition is shown

Model	Exp.	mW	TIP4P/ Ice
$T_m/K for Ih$	273.15	274.6	272.2
$\Delta H_m(T_m)/kcalmol^{-1}$	1.436	1.26	1.29
$\rho_{liquid}(T_m)/gcm^{-3}$	0.999	1.001	0.985
$\rho_{ice}(T_m)/gcm^{-3}$	0.917	0.978	0.906
$\rho_{liquid}(298K)/gcm^{-3}$	0.997	0.997	0.992
D(298K)$\times 10^{-5}/cm^2s^{-1}$	2.3	6.5	1.19

Table 1: A table showing various physical quantities generated using the mW potential, TIP4P/Ice model and experimental data. The melting temperature, enthalpy of melting, densities of liquid and ice, and diffusion coefficient are shown.[311118]

below.

$$E = \sum_i \sum_{j>i} \phi_2(r_{ij}) + \sum_i \sum_{j\neq i} \sum_{k>j} \phi_3(r_{ij}, r_{ik}, \theta_{ijk}) \tag{3}$$

$$\phi_2(r) = A\varepsilon \left[B\left(\frac{\sigma}{r}\right)^p - \left(\frac{\sigma}{r}\right)^q\right] \exp\left(\frac{\sigma}{r - a\sigma}\right) \tag{4}$$

$$\phi_3(r, s, \theta) = \lambda\varepsilon \left[\cos\theta - \cos\theta_o\right]^2 CD \tag{5}$$

$$C = \exp\left(\frac{\gamma\sigma}{r - a\sigma}\right), D = \exp\left(\frac{\gamma\sigma}{s - a\sigma}\right) \tag{6}$$

Within equations 3, 4, 5 and 6; $A = 7.05$, $B = 0.60$, $p = 4$, $q = 0$ and $\gamma = 1.2$. In addition to this, a is the cutoff point, ϵ is the interaction potential depth, σ is the particle diameter, λ is simply a scale factor which affects the strength of the tetrahedral interaction (tetrahedrality) and θ_o is the O-O-O angle. Each of these parameters can be manipulated to suit particular required specifications.[3135]

Efficiency is greatly increased when the mW potential is used in simulations as the computational cost is reduced by 99% as compared to potentials which model electrostatic interactions. Furthermore, as shown in table 1, when analysing different physical properties, the potential excelled in reproducing the energetics, melting point, density of water and latent heat. Although the potential models are significantly better than most atomistic models, similarly to the density of ice, the diffusion coefficient prediction was far from the experimental value.[31]

2.2 Large-scale Atomic/ Molecular Massively Parallel Simulator: LAMMPS

LAMMPS was the MD software package used for simulating ice and water within the project. The application allows for different ensembles to be simulated using the Nose-Hoover style non-Hamiltonian equations of motion.[2]

In order to simulate the H_2O trajectories, periodic boundary conditions were used. This meant the volume of the structure could be controlled. The unit cell for the simulations were set. Computational cost was also reduced by calculating the interaction between the nearest neighbours rather than all the particles in the simulation.

The run style verlet was used. Hence, the Velocity Verlet algorithm was used as the time integrator when simulating using LAMMPS. It is a numerical solution to Newton's equations of motion for molecular dynamics.[56] This equation is shown below as equations 7 and 8.

$$\mathbf{r}(t + \Delta t) \approx \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{\Delta t^2}{2m}\mathbf{f}(t) + \mathrm{O}\left(\Delta \mathrm{t}^3\right) \tag{7}$$

$$\mathbf{v}(t+\Delta t) \approx \mathbf{v}(t) + \frac{\Delta t}{2m}[\mathbf{f}(t+\Delta t) + \mathbf{f}(t)] + \mathrm{O}\left(\Delta \mathrm{t}^3\right) \tag{8}$$

2.3 Visual Molecular Dynamics: VMD

VMD is the program used to visualise simulations created by LAMMPS or other simulators and the predictions made by DeepIce. It was particularly useful to visualise the predictions as the different types of molecules were coloured and, hence, differentiated from one another. In addition to visualising molecular structures, various plugins could be installed and used. For example, the radial pair distribution function plugin and the diffusion coefficient plugin were convenient plugins to use in order to cross check the results of the mW potential LAMMPS simulations.

Ice Phase	Diffusion Coefficient / Åns^{-1}
Water 260K	335.2 ± 0.5365
Ic 260K	0.0002488 ± 0.0001668
Ih 260K	0.0001511 ± 0.0001693
Ic 250K	0.0004243 ± 0.0001563
Ih 250K	0.000474 ± 0.0001614

Table 2: A list of diffusion coefficient values calculated using the VMD diffusion coefficient extension.

The radial pair distribution function, $g(r)$, shows the probability of a particle existing at distance r as the distance varies from a reference particle. It can also be referred to the average number density of particle at some distance, r, from the reference particle.

$$g(r) = \frac{1}{4\pi r^2} \frac{1}{N\rho} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \langle \delta \left(r - |\mathbf{r}_i - \mathbf{r}_j| \right) \rangle \quad (9)$$

Equation 9 shows the relation where $g(r)$ is the radial distribution function, ρ is the density, N is the number of molecules and r is the distance from the reference particle.[4]

A non-native plugin diffusion coefficient was added by cloning the relevant repository from Toni Giogino's GitHub page. The plugin also gave the option to calculate the mean squared displacement (MSD) between chosen interval lag times. The equations implemented to calculate these quantities are shown below:

$$M(\tau) = \left\langle |\mathbf{r}(\tau) - \mathbf{r}(0)|^2 \right\rangle \quad (10)$$

$$D(\tau) = M(\tau)/2E\tau \quad (11)$$

where $M(\tau)$ is the MSD at lag time τ, $\mathbf{r}(\tau)$ is the displacement, $D(\tau)$ is the diffusion coefficient and E is the system's dimensionality.[21]

As seen above in table 2, the supercooled 260K water has a considerably higher diffusion coefficient at 335.2 ± 0.5365Å/ns compared to the other phases of ice. This gave an intuitive result to prove the 260K supercooled water was indeed supercooled water.

In addition to the diffusion coefficient, figure 3 shows the radial pair distribution function.

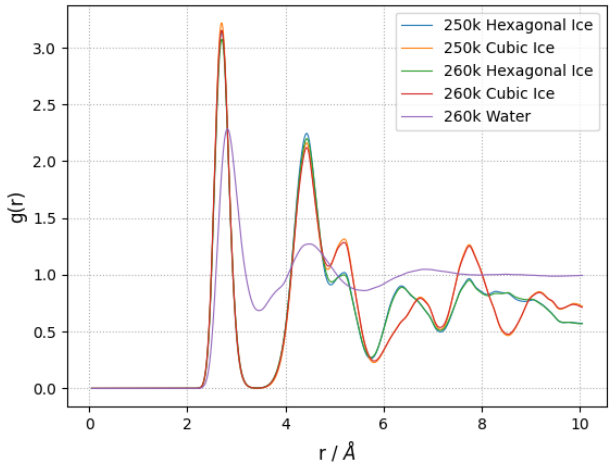


Figure 3: Graphs for simulated phases of H_2O showing the radial pair distribution function of 260K supercooled water, 260K Ih, 260K Ic, 250K Ih, 250K Ic.

The radial pair distribution functions were compared to the paper named 'Neural networks for local structure detection in polymorphic systems' where the graphs followed a similar trend. The radial pair distribution function for water had a lower maximum than the functions for the hexagonal and cubic crystalline structures. The curve for water also contained fewer local maxima and minima compared to ice. However, g(r) stabilised at a higher value for water when compared to the ice phases while the paper showed the opposite.[20] This stability and lack of oscillations observed can be attributed to the lack of crystalline structure in 260K supercooled water compared to ice. Since ice has a fixed structure, there will be gaps between molecules allowing the $g(r)$ to reduce drastically. Supercooled water has no fixed structure, hence, less gaps between molecules meaning the line oscillated to a lesser degree. The shape of the graphs are a reflection of the molecular structure within the unit cells.[8]

3 The Machine Learning Approach

Machine learning (ML), first hypothesised by Alan Turing in 1950,[41] is a part of the larger study of artificial intelligence (AI) which uses data known as training data to learn in order

to predict more accurate results. ML particularly becomes useful when conventional decision making code is inefficient and inaccurate to use. It has therefore become prominent throughout people's lives. For example, google maps, spam filters in emails, social media advertisement and PayPal all use ML to improve daily tasks.[42] The plagiarism detection service, Turnitin, is also researching the potential of using ML.[43] Most notably, an example of the extent of how advanced it could be was set by Alphabet owned DeepMind. AlphaGo was developed to learn and play the Japanese board game Go. AlphaGo proceeded to defeat Lee Sedol, a top ranking Go player, by winning 4-1.[39]

3.1 The Principle of Machine Learning

The formal definition of ML is as follows:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." - *Tom Mitchell*[30]

The three main components of ML are the data, model and the loss function.[23] First, the training data is generated using conventional methods similar to how LAMMPS was used to simulate the trajectories of H_2O molecules in various phases of ice as discussed above. These training data sets also had to be transformed in order for it to fit within the neural network.

The second component of ML was the model itself. Various types of models include supervised learning, unsupervised learning and reinforcement learning.[23] In the case for DeepIce, supervised learning was used as the algorithm attempted to use training data to vary the weights in order for the model to fit a set outcome called the validation data set. Hence, cross validation is used.

The loss function is the last main component of machine learning. Otherwise known as the cost function, it attempts to quantify the cost of the model. Essentially, the cost is how accurate an iteration of the model is compared to the actual value. To optimise the loss function, a lower number is preferred for enhanced predictions.[36]

3.2 Artificial Neural Networks

Artificial neural networks (ANN) are computing systems which are comprised of multiple hidden layers between an input layer and an output layer. Each layer consists of artificial neurons called units. The idea is to attempt to mimic a biological brain and how neurons use synaptic transmission to transport information. Neural networks are part of the larger ML subject.[447] They can be described as a black box due to the fact the process of converting the input into the output is obfuscated.[17]

Parameters which influence the learning process of a neural network are known as hyperparameters. Batch size and the number of epochs are two separate hyperparameters which could be controlled when using DeepIce. The batch size is the number of samples a neural network has processed before back-propagation occurs while the number of epochs are the number of times the neural network will process the full data set. In addition to this, gradient descent is an optimisation algorithm used to find the lowest error value.[1] There are various different types of gradient descents depending on how the batch size is defined: batch gradient descent is when the batch size is the same size of the training set, stochastic gradient descent is when the batch size is 1 and mini-batch gradient descent is when the batch size is between 1 and the size of the training data set.[32] The idea is to vary these hyperparameters such that the error between the theoretical and real data is minimal.

An activation function is a function used to decide whether a neuron is fired or not. The linear function, sigmoid function and tanh function are all examples of activation functions. Vital to the performance of a neural network, a

loss function or cost function is a function created to evaluate the set of weights to quantify its weight in order to improve the model as it runs.[12] Back-propagation is dependant on this loss function as it is the process of differentiating it to either minimise or maximise it depending on what is desired.[3717]

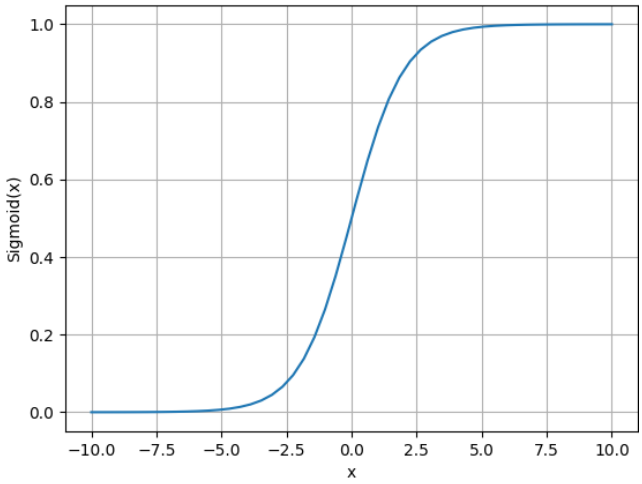


Figure 4: The sigmoid activation function.

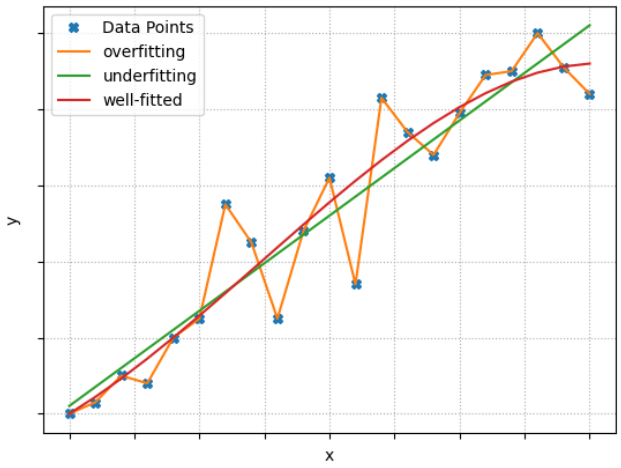


Figure 5: A graph visualising underfitting and overfitting in machine learning.

Depending on how the hyperparameters are varied, underfitting and overfitting can transpire. Underfitting is the idea that the model is not able to grasp pattern recognition or a relationship between the validation and training sets. This can also be referred to as the phase space of solutions not being explored. In this case, the model's error is low on the training and validation data sets.[10] On the other hand, overfitting occurs when the model fits the training data exactly such that when the ANN is exposed to new data, relationships cannot be extracted. Therefore, the neural network is not able to serve its purpose. Hence, the phase space of solutions is said to be too well defined. In other words, the model's error is high on the training data but low on the validation data.[9]

3.3 DeepIce: Implementation of Neural Networks

DeepIce implemented neural networks into its ML model. This was achieved using a combination of the TensorFlow and Keras python libraries. DeepIce is comprised of four different neural networks which are the Cartesian coordinate network, spherical harmonic network, spherical coordinate network and Fourier transform network as shown in figure 6.

All the different neural networks used the same non-linear Rectified Linear Unit (ReLU) activation function for each node/neuron.[19] In terms of mathematics, it is defined as such:

$$f(x) = x^+ = \max(0, x) \tag{12}$$

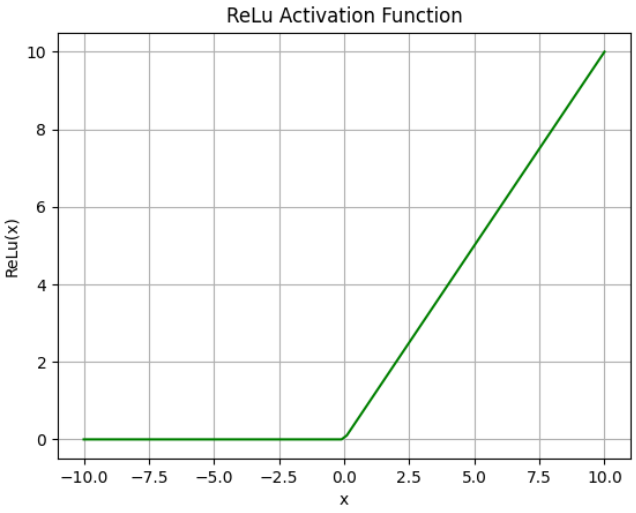


Figure 7: A graphical representation of the ReLU activation function.

This activation function is more computationally efficient when compared to the more complex tanh and sigmoid activation functions. This function also makes the network lighter as less neurons are firing within the network, reducing

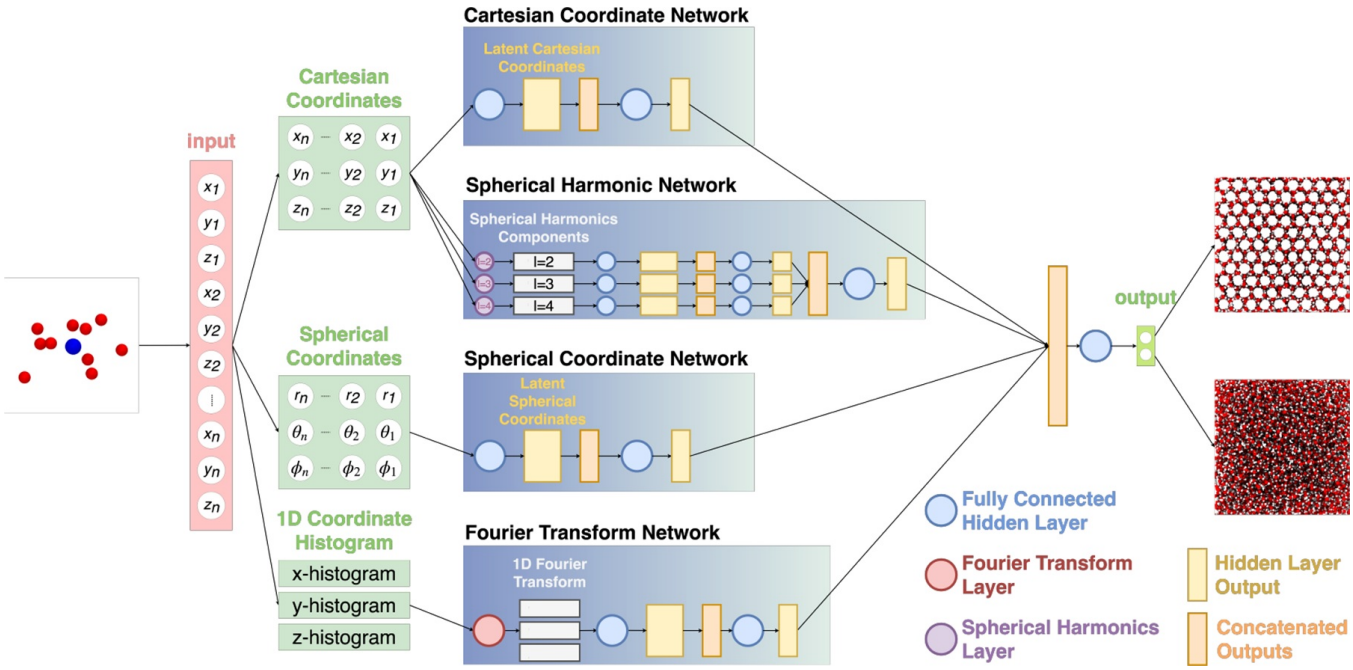


Figure 6: Diagram showing how the DeepIce neural network is designed in order to train efficiently and accurately.[19]

the computational processing power necessary to run the neural network successfully. Furthermore, there is no vanishing gradient problem which is a description of the gradient of the activation function reducing such that when x changes, $f(x)$ doesn't change significantly. An example of this is seen in the Sigmoid function in figure 4. On the other hand, disadvantages such as the dying ReLU problem can botch the ML model. This is because if too few activations of neurons occurs, the neural network is said to die out.[37]

The input of DeepIce is made of a one dimensional vector of the Cartesian coordinates of the specified number of nearest neighbours. Therefore, the size of this vector is $3n$ where n in the number of nearest neighbours.[19]

A Cartesian coordinates network was one of four neural networks used. The network worked by sharing the Cartesian coordinates with hidden layers of the network. These coordinates are then transformed into latent Cartesian coordinates. These are then concatenated and sent through another set of hidden layer. Weights and biases calculated for the neurons within the hidden layers are optimised during DeepIce's training.[19]

The spherical Harmonics network is the second neural network which DeepIce is comprised of. Similar to the Steinhardt approach, various spherical harmonics are computed depending on the Cartesian coordinates of the nearest neighbours. Hence, when the nearest neighbour Cartesian coordinates are inferred from the input, they also transmit to this network as well as the Cartesian coordinates network. Only the $l = 2$, $l = 3$ and $l = 4$ order spherical harmonics are computed. The pathway of the coordinates used are shown in figure 6. These three orders passed through three separate subnetworks. While similar to the Steinhardt parameters, the main benefit of this network is the fact it learns to optimise the accuracy of prediction rather than having a distinct definition of liquid and solid structures of ice.[19]

The Spherical coordinates network is extremely similar to the Cartesian coordinates network in terms of the network structure. Instead of using Cartesian coordinates, spherical coordinates are used in this network. Hence, the initial input needs to be converted into spherical coordinates.[19]

Fourier Transform network works by analysing the Fourier fingerprint of each phase. Three different Fourier transforms are computed using the histograms made by considering the x, y and z coordinates of the nearest neighbours separately. Again, these Fourier transform outputs are processed through multiple hidden layers in the neural network.[19]

By concatenating the results of the four different neural networks, encoding the local molecular surroundings, a vector is produced. The output network consists of the softmax function, otherwise known as multi-class logistic regression, which is a type of activation function. This function converts a vector of real values into another vector of real values which sum to one in order to treat the values as probabilities. It is simply used to weigh inputs at the concatenation layer of the network. The softmax function is given in equation 13.[1913]

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{13}$$

The loss function used within DeepIce is the cross entropy loss function shown below.

$$E = -\frac{1}{n_t} \sum_{i}^{n_t} \left[\tilde{y}_t^{(i)} \log(A) + \left(1 - \tilde{y}_t^{(i)}\right) \log\left(1 - A\right) \right] \tag{14}$$

where:

$$A = y_p\left(X^{(i)}\right) \tag{15}$$

The variable are defined as the following: n_t is the total number of values in the training set, $\tilde{y}_t^{(i)}$ is the correct output and A is the predicted output. The value of E is minimized in an attempt to reduce the error as the model learns.[19]

Since the cross entropy function is used to account for the neural network's performance, the loss varies as the probability changes as shown in figure 8. As the probability decreases, the loss increases rapidly. This is because as the probability of the output is close to the true value, the error is minimal and hence, the loss is close to 0.

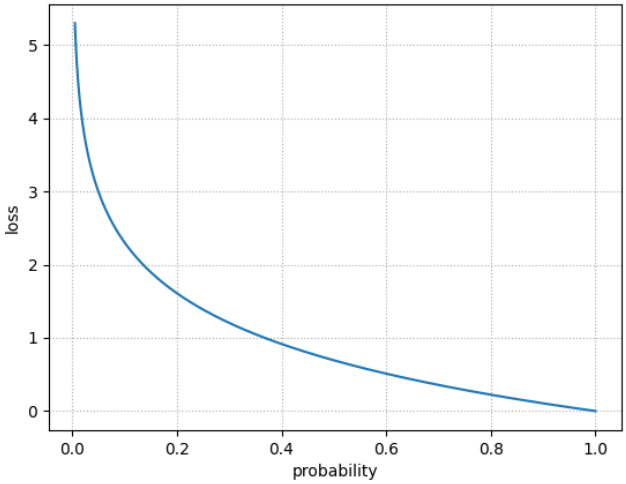


Figure 8: A graph showing how the loss changes with probability when using the cross entropy loss function

4 DeepIce Data Preparation, Training and Visualisation

A series of steps were taken to lead to the final results of the simulations. The following steps were taken:

0. Create a virtual environment and install necessary libraries.
1. Simulate data for various ice phases using LAMMPS.
2. Use VMD to visualise simulated data and check using different extensions.
3. Convert simulated data to csv and then to npz files.
4. Convert slab data to csv and then to npz files.
5. Use training data to train DeepIce and store weights files.
6. Predict using DeepIce.
7. Evaluate using DeepIce to find the accuracy and error rate.
8. Analyse and depict prediction using python scripts and VMD.

4.1 Setting up the Environment

The precursor to the method, hence the 0^{th} step, used was to create a virtual environment

to download all the libraries with the dependencies. Python 3.7 with the keras, tensorflow and relevant packages were installed in order to use DeepIce. Furthermore, to use the data preparation scripts, another virtual environment with MDAnalysis, numpy, pandas and scikit-learn libraries was created.

4.2 Simulating Data using LAMMPS

The first step required utilising LAMMPS to simulate 5ns of various ice phases at different temperatures. These phases included hexagonal ice, cubic ice and water. All of these phases simulated data at 250K and 260K, except for super-cooled water which was only simulated for 260K. Several .in files were used to script the simulations. The units and the boundary condition were set as femtoseconds and periodic, respectively. The boundaries were created so a bulk structure would be simulated and QLLs would not form. The required bulk was imported to match the phase structure and the potential used was defined as the mW potential. Fix conditions were used to define the temperature, time-steps were defined and the run commands forced the simulation to run at the fixed variables. The simulation boxes were made of 2744 molecules for cubic ice and 2880 molecules for hexagonal ice. The isothermal-isobaric ensemble, NPT, was used to run the first simulation in order to initialise the temperature and pressure. Then, the canonical ensemble, NVT, was used to first equilibrate the simulation and secondly to simulate the necessary data. The simulation for 260K water was similar. The NPT ensemble was used to initialise hexagonal ice at 260K, the NVT ensemble was then consequently used to bring the temperature up to 400K, down to 300K and then finally to 260K. The tables 3, 4 and 5 shows the pathways taken to simulate the bulk structures.

Ensemble	Temp/ K	Timestep/ fm	No. of Timesteps
NPT	50	0.01	100
		0.10	100
		5.00	100
	100	5.00	100
	150	5.00	100
	200	5.00	100
	250	5.00	1000000
NVT	250	5.00	1000000
NVT	250	5.00	1000000

Table 3: A table showing the pathway taken to simulate 250K cubic and hexagonal ice using LAMMPS.

Ensemble	Temp/ K	Timestep/ fm	No. of Timesteps
NPT	50	0.01	100
		0.10	100
		5.00	100
	100	5.00	100
	150	5.00	100
	200	5.00	100
	260	5.00	1000000
NVT	260	5.00	1000000
NVT	260	5.00	1000000

Table 4: A table showing the pathway taken to simulate 260K cubic and hexagonal ice using LAMMPS.

Ensemble	Temp/ K	Timestep/ fm	No. of Timesteps
NPT	50	0.05	200
		0.50	200
		5.00	200
	100	5.00	200
	150	5.00	200
	200	5.00	200
	260	5.00	1000000
NVT	400	5.00	1000000
NVT	300	5.00	1000000
NVT	260	5.00	1000000

Table 5: A table showing the pathway taken to simulate 260K supercooled water using LAMMPS.

4.3 Training Data Transformation

Simulated data had to be converted from their trajectory dcd files to npz files which specified ice and water using a binary base. The idea was to fit the data into the DeepIce neural network. This was done by using a python script. This was achieved by writing a csv file of the input and targets using the trajectories and structural pdb files. The number of nearest neighbours also had to be specified for this function. The python script was utilised to convert the csv to an npz file with arrays named X_train and y_train by making use of the numpy library to import methods named *asarray* and *savez_compressed*. The slabs also had to be converted to fit into the DeepIce neural network. Therefore, a similar method was used to also convert these files into the npz file format.

The training data set, Γ, encodes the coordinates of the nearest neighbours and their relevant phase identities, $\tilde{y}$. The below formalism depicts the structure of the training data set.

$$\Gamma = [\Lambda, \tilde{y}] \tag{16}$$

where:

$$\tilde{y}_{ice} = [1, 0] \tag{17}$$

$$\tilde{y}_{water} = [0, 1] \tag{18}$$

4.4 DeepIce Training and Predicting

DeepIce was subsequently trained using the train command. The number of nearest neighbours was set to 10, the batch size was specified to either 10 or 30 and the number of epochs was also inputted. In addition, the previously outputted weights file, the ice phase npz file and the output weights were all specified with the allocated paths. An example is shown below:

```
python main_deepIce.py --Train --nearest_neighbours 10 --batch_size 30 --n_epochs 1 --weights_file="C:\Users\arnav\Documents\3rd_Year_Group_Project\Output_weights\not_concat\Ic_wlh\batch30\deepice_Ic_water_trained_epoch1_batch30.h5" --data="C:\Users\arnav\Documents\3rd_Year_Group_Project\Datasets_Preparation\Ic_wlh\bulk_Ic_wlh_260K_training.npz" --output_weights="C:\Users\arnav\Documents\3rd_Year_Group_Project\Output_weights\not_concat\Ic_wlh\batch30\deepice_Ic_water_trained_epoch2_batch30.h5"
```

After DeepIce had been trained to differentiate between water and ice, each slab was inputted for prediction results to be formed. The slabs used were the basal hexagonal ice slab, the primary prism hexagonal ice slab and cubic ice slab. The names of the slabs represent the planes of ice slabs used. To run the prediction method, the according weights file and the slab npz file were used. Furthermore, the number of molecules and number of nearest neighbours had to be defined because it was different for cubic and hexagonal ice. An output dat file was produced. The following is an example of the command used.

```
python main\_deepIce.py --Predict --data_file="C:\Users\arnav\Documents\3rd_Year_Group_Project\Models\Slabs\basal_npz\Trajectory1.NVT.Zbasal.240K.npz" --weights_file="C:\Users\arnav\Documents\3rd_Year_Group_Project\Output_weights\not_concat\Ih_wlh\batch30\deepice_Ih_water_trained_epoch1_batch30.h5" --nearest_neighbours=10 --num_mols=5760
```

The evaluate command was then ran to find the accuracy and the error rate between an evaluation data set and a weights file outputted from training the DeepIce neural network.

```
python main_deepIce.py --Evaluate --data_file="C:\Users\arnav\Documents\3rd_Year_Group_Project\Datasets_Preparation\Ih_wlh\bulk_Ih_wlh_260K_training.npz'' --weights_file="C:\Users\arnav\Documents\3rd_Year_Group_Project\Output_weights\not_concat\Ih_wlh\batch30\deepice_Ih_water_trained_epoch1_batch30.h5'' --nearest_neighbours=10
```

4.5 Prediction Analysis

The data produced from this procedure still required to be visualised by the human eye to understand it. Therefore, many python scripts had to be created to achieve this. A colouring python scripts was created to replace the binary 1s and 0s, which identified which molecule was ice and water, with the letters O and A to differentiate them with colours in VMD. This script required using the prediction dat file and the particular structural xyz file depending on whether cubic or hexagonal ice was being used.

Individual quantities were also analysed by creating python scripts. The two quantities were the number of water molecules at each time throughout the simulation of the slabs and the QLL thickness corresponding to each slab at

various temperatures and ice phases. The number of water molecules was simply measured by counting the binary output of the prediction files. On the other hand, the QLL thickness was measured by two different methods. These methods were called the Molecular Counting method (MolC) and the Monte Carlo method (MC) as shown in equations 19 and 20.

The first method consisted of using the following equation:

$$d_{Molc} = \frac{\overline{N_w} M}{2\rho N_A L_y L_z 10^{-24}} \tag{19}$$

In this formula, $\overline{N_w}$ represents the relevant average number of water molecules, M is the molar mass of water, ρ is the density of water, N_A is Avogadro's number and L_y, L_z are the dimensions of the QLL.[16]

The other method comprised of taking into account the fact that the ratio between the number of water molecules and the total number of molecules would be the same as the ratio between the thickness of the QLL and the total slab thickness.[35]

$$d_{Mc} = \frac{\overline{N_w} d_t}{2N_t} \tag{20}$$

In the above equation, $\overline{N_w}$ is the average number of water molecules in the whole slab, N_t represents the total number of molecules in the slab and d_t is the overall thickness of the slab. The thicknesses were calculated by using the xyz structure files to find the highest and lowest coordinate values. To do this, the xyz structure file could be opened and the necessary coordinate values extracted. Both equations are divided by 2 to account for two QLLs.

5 Results and Analysis

5.1 Visual Representations

All the results, except for the results which specify the batch size used, used a batch size of 30.

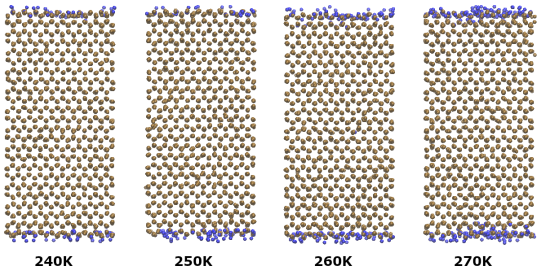


Figure 9: Coloured DeepIce predictions of the YPrism $(1\bar{1}00)$ hexagonal ice slab at various temperatures 10K apart visualised and rendered using VMD. Ice molecules are shown as gold and water molecules are shown to be blue.

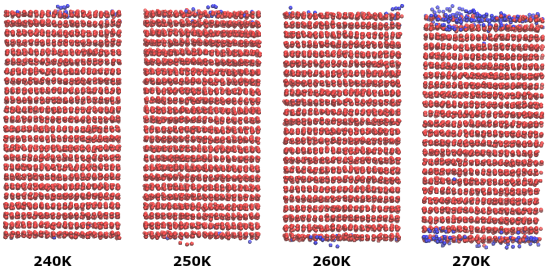


Figure 10: The Basal plane (0001) of the hexagonal ice slab DeepIce predictions from 240K to 270K. The red molecules are what DeepIce detected as ice and water molecules are blue.

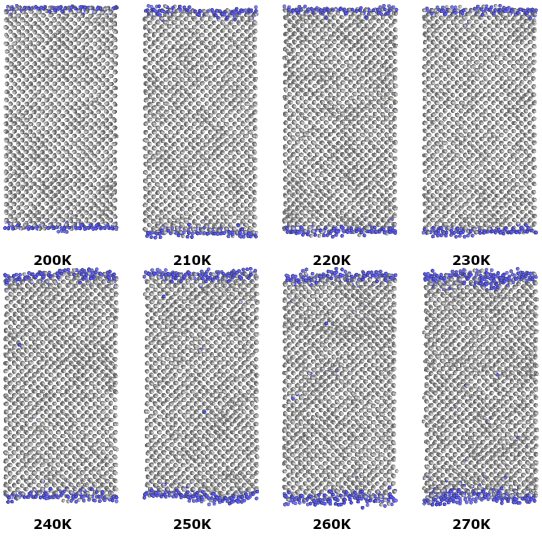


Figure 11: Ice slab XIc (100) DeepIce predictions from 200K to 270K. Ice molecules are shaded silver and water molecules are blue.

The visualised results produced using DeepIce are shown in figure 9, 10 and 11. Evidently, the size of the QLL is visually shown to increase with higher temperatures in all the slabs. Furthermore, for a fixed temperature, the QLL thickness ranged from largest to smallest in the order cubic, prism and basal, respectively.

Figure 12: Graphs showing the time series of the number of water molecules of both QLLs on each slab as it is simulated for 100ns.

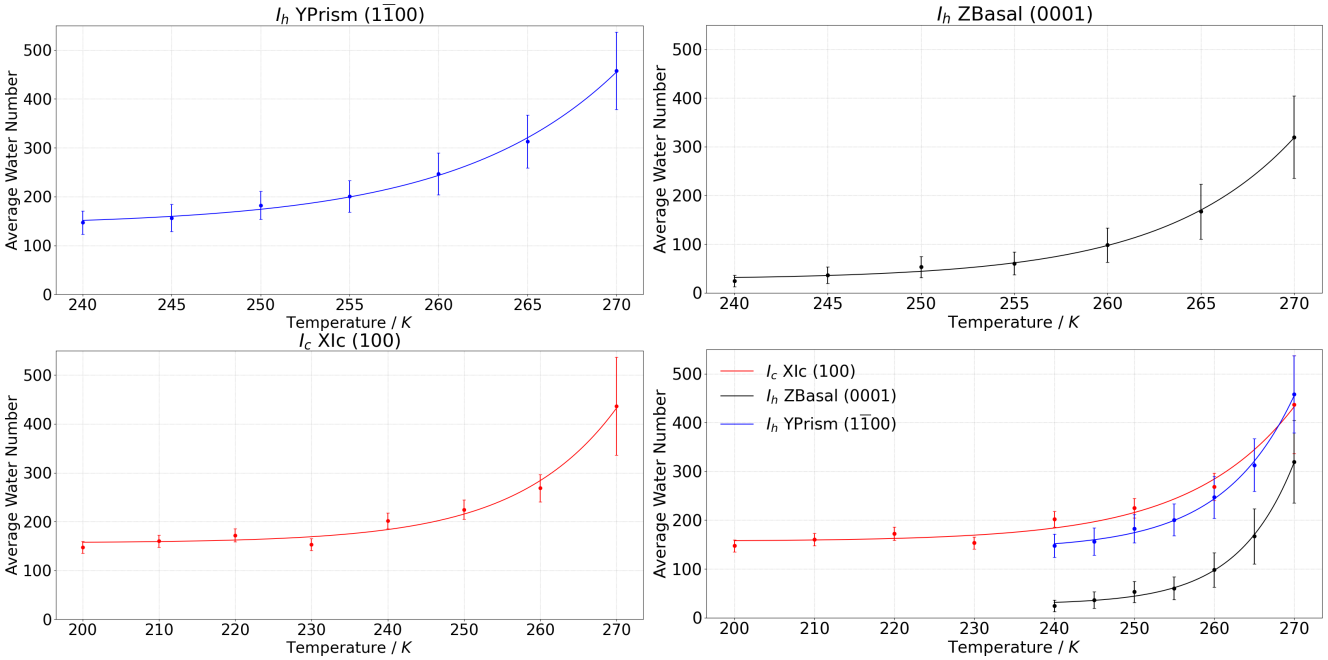


Figure 13: Graphs showing the average number of water molecules of both QLLs in each slab at various temperatures and H_2O phases detected by DeepIce when using the mW potential.

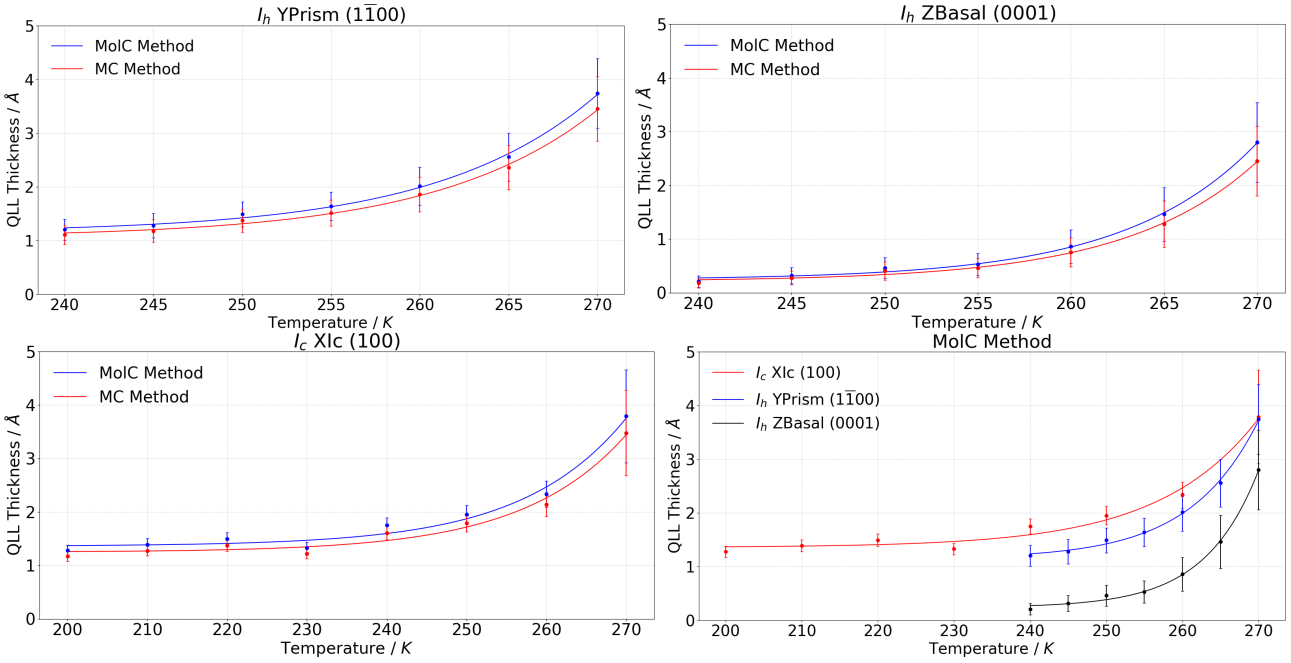


Figure 14: The above graphs depict the individual QLL thicknesses of the three different slabs used as a function of temperature. The two coloured lines represent two different methods used to calculate the thickness: MolC and MC.

5.2 Water Number

Figure 12 and 13 show the evolution of the number of water molecules through time and the average water number, respectively. Features from both graphs have led to conclusive observations. The first main observation is the increase in water number as the temperature increases. This agrees with intuition and literature.[3816] This could be explained by the Maxwell-Boltzmann distribution as an analogy; as the temperature increases, the molecules gain in kinetic energy and the number of molecules past the kinetic threshold increase relative to the temperature.

Furthermore, another visible trend is the increase in standard deviation as the temperature increases as shown in figure 13. The data hints at the fact that the number of water molecules fluctuate more aggressively with higher temperatures. The Maxwell-Boltzmann distribution can, again, help analogously show why the temperature and fluctuations are positively correlated. As the temperature increases, the threshold kinetic energy boundary increases vertically eluding to the fact that more water molecules have the threshold kinetic energy. Hence, more water molecules will fluctuate relative to the boundary.

Another observation made was the fact that less water molecules were distinguished by DeepIce in the Basal hexagonal ice slab. This, with the original paper considering DeepIce, contradicts the paper named 'The thickness of a liquid layer on the free surface of ice as obtained from computer simulation'. In the referred paper, the basal plane showed a larger QLL thickness as well.[16]

5.3 QLL Thickness

The data showing the thickness of the QLLs showed similar results to the water number. As shown in figure 14, the Basal Hexagonal ice slab showed the thinnest QLLs whilst the primary Prism hexagonal ice slab and the cubic ice slab showed the second thickest and thickest QLLs, respectively. Furthermore, the thickness also increased with temperature agreeing with intuition. In addition to this, the MolC method measured the thickness to be higher than the MC method. This could be explained by the idea that the MolC method did not account for overlapping water molecules, hence it overestimated the thickness of the QLLs. But, since the standard deviation error bars do overlap with one another, there is no statistically significant difference that can be inferred between both sets of data.

5.4 Network Size

Training Description	Avg Epoch Time/s
Ic and Water (Batch 10)	1109.7
Ic and Water (Batch 30)	400.4
Ih and Water (Batch 10)	805.6
Ih and Water (Batch 30)	407.05

Table 6: A table portraying how the average time taken for an epoch to run varies with different number of batches for different phases of H_2O. *Note: Investigation was conducted on a Desktop with a AMD Ryzen 5 3600 6-Core Processor and 16.0 GB RAM.*

The above table 6 represents the time it took to run the training process for each epoch. As the batch size is reduced for each respective training pair, the average computational time increased. Hence, a negative correlation was observed. It is evident depending on the batch size and number of epochs, overfitting or underfitting could occur. Appendix A shows how the training and validation accuracy changed with the various phases DeepIce was trained to differentiate.

Furthermore, one could naively assume as the size of the training data set increases, the accuracy would also increase. But, this was observed to be counterproductive as when a concatenated data set was created comprising of various group member's data, the accuracy after the first epoch decreased substantially. This reduction in accuracy was caused by overfitting.

To further test for DeepIce's ability for pattern recognition, the neural network was trained to distinguish between hexagonal ice and water

and the accuracy was evaluated against the combination of cubic ice and water. As predicted, the accuracy was observed to be low at 51.2%. This proved that DeepIce was able to learn specific ice structures, hence, specific phases of ice could be identified but not ice in general when not trained for the specific structure.

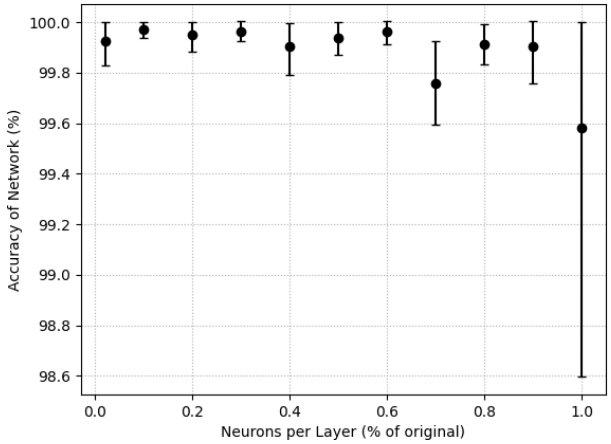


Figure 15: The accuracy of DeepIce's neural network is shown as a function of the size of the network. Note: The standard deviations error bars are capped at 100% as accuracy cannot be over 100%.

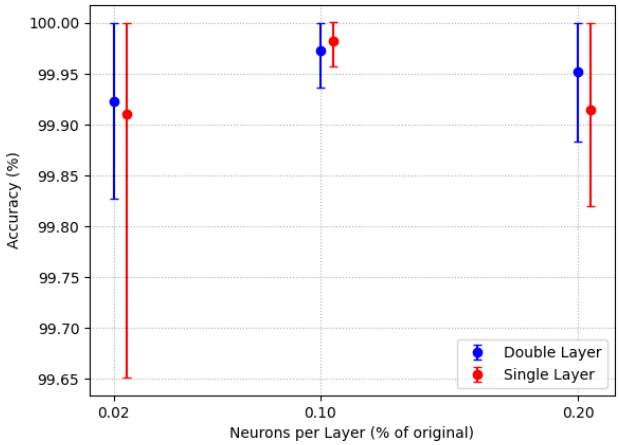


Figure 16: Accuracy as a function of the size of the neural network where single and double hidden layers within the neural net are compared. Note: The standard deviations error bars are capped at 100% as accuracy cannot be over 100%.

The size of the neural network was also varied to observe its effects on the accuracy of the model and computational time necessary for the neural network to complete training. Figure 15

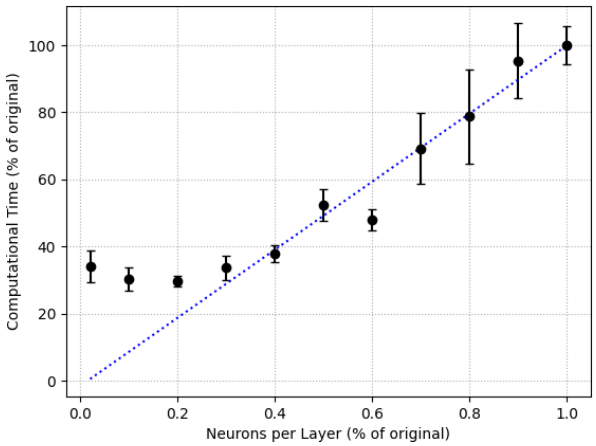


Figure 17: This graph shows how the computational time changes as a function of the neural network size.

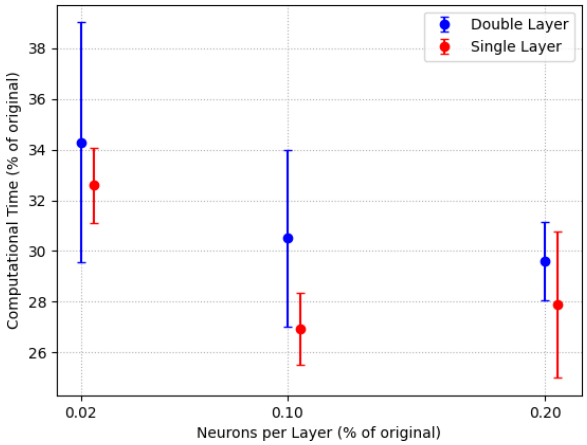


Figure 18: Computational time percentage of the original network as a function of the size of the neural network with both single and double hidden layers being shown.

and 17 shows that as the size of the neural network was reduced, the accuracy increased and the computational time decreased as less active nodes were utilised. Though, the computational time did plateau as the size of the neural network decreased. This implies the computational time is directly proportional to the size of the neural network until the minimum amount of computational time necessary for the neural network to perform is reached. Apart from the general size of the neural network, the number of hidden layers were also varied for all the neural networks used within DeepIce. In general, the double layer was more accurate than having a single layer except for when the network size was at 0.1%. But, the computational time

did increase with more hidden layers. This is attributed towards the idea that more processing needs to occur as the data is sifted through more artificial neurons: a longer neural network.

5.5 Ice Nucleation

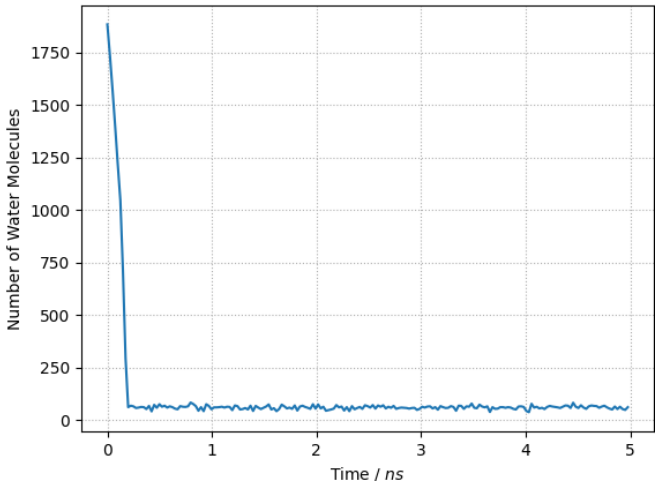


Figure 19: Graph showing how the number of water molecules vary as ice nucleation occurs.

DeepIce was subsequently also used to show the process of water crystallising. To achieve this, $\frac{2}{3}$ of the hexagonal ice slab at 260K with 2880 H_2O molecules were melted whilst keeping the rest as ice. This involved running the simulation for 100,000 steps at 260K and 200,000 steps at 400K to melt the $\frac{2}{3}$ of ice at 5fs per step. The water was then crystallised at 260K. Thereafter, the simulation ran at 260K for 5ns with 1,000,000 steps. VMD allowed the observation of water crystallising layer by layer. Figure 19 shows how the water number evolved over time as ice nucleation took place.

6 Discussion

DeepIce produced multiple results including the number of molecules in different ice slabs and their inferred QLL thicknesses. Overall, the number of water molecules compared at different temperatures were proven to be correct by intuition. Higher temperatures led to more water molecules being detected by DeepIce. Furthermore, the amount of fluctuations increased at higher temperatures. The order from highest to lowest number of water molecules was XIc, YPrism and ZBasal. The QLL thickness for each slab followed the same trends. Although there was an inversion between YPrism slab and the XIc slab at 270K.

Throughout the computational experiments, DeepIce was good at distinguishing between ice and water when the various slabs at different temperatures were fed in. Figures 9, 10 and 11 show visual examples of DeepIce's predictions using VMD to render the images for the three different slabs used to test against when compared to the predictions made using the TIP4P/Ice model on the slabs simulated using such a model. DeepIce visually produced similar results with water and ice being clearly distinguished but different quantities were observed. The different quantities can be attributed to the fact that different H_2O simulation models were used to produce the slabs. Hence, regardless of the training done, quantities such as the number of water molecules in the QLLs were different.

The original paper which introduced DeepIce in 2019 used the TIP4P/Ice model to simulate the phases of H_2O compared to the mW potential used to produce this paper. This highlighted large differences between the two models such as the fact that the slabs produced using TIP4P/Ice seemed to have a thicker layer of water. For example, when the number of water molecules were measured for 270K cubic ice, the mW potential had over 1000 less water molecules than the TIP4P/Ice model. On the other hand, similarities are present between the trends observed. The relationship of rising temperature correlated with higher water number and QLL thickness was common between the two papers. Also, the increase in fluctuations with temperature are also shown in the earlier paper. One significant difference is how an inversion takes place between the water number graphs comparing the the ice surface QLLs. Hence, above 270K the basal plane has a higher water number than the primary prismatic plane. As shown in figure 13, the basal plane has significantly smaller water number meaning no inversion is seen regarding it.[38]

Further study applying DeepIce rather than testing its validity would be interesting. For example, simulating more complex systems such as inducing droplet shaped QLLs using hydrogen chloride and tracking their formation by using DeepIce would allow for more extensive research.[33] Essentially, any system that would requires distinguishing between water and ice would greatly benefit from DeepIce as it is its purpose.

As the largest constraint of further research was time, there are improvements that could be set out. A consideration of using mW potential to train DeepIce and using it to predict the phases of ice slabs produced using the TIP4P/Ice model and vice versa would be interesting. This would allow side by side comparisons between training DeepIce with the mW potential and TIP4P/Ice. Furthermore, due to lack of computational resources, larger data simulations would have been ideal to increase the accuracy of certain inferred quantities such as how the number of water molecules evolved over a long period of time. Moreover, as done with nucleation with a hexagonal ice crystallising over 5ns, other processes could be analysed using DeepIce. Also, an attempt at improving DeepIce's capability of differentiating between phases such as increasing the number of different arguments it can take during the prediction process would be compelling due to more ease of use. One last consideration that would have been interesting was to simply go through the process of training DeepIce with multiple different water models and comparing their respective qualities by predicting on a controlled constant slab of QLLs.

To reduce this constraint of time, implementing the pooling of computing resources by using clustering to pool computational resources or cloud computing could have been ideal. Using batch commands to generate more data would have increased accuracy and reliability of the simulation produced using LAMMPS in a shorter period of time. This would have enabled DeepIce to learn from a larger sample training set. In addition to this advantage, since the project was a collaborative, errors due to various members having differently configured systems would have been minimal.

7 Conclusion

The main goals of this project were to train the DeepIce neural network to distinguish between water and ice molecules in slabs containing QLLs at various temperatures, and to also analyse the thickness of the QLLs detected by DeepIce. In order to achieve these goals, the structure of ice needed to be understood, ice phase trajectories had to be simulated using the mW potential and prepared for it to become the DeepIce training data set, the simulations had to be crosschecked for the radial distribution function and diffusion coefficient, the number of water molecules needed to be calculated and, finally, the thickness of the QLLs were necessary to be calculated. These steps were achieved producing the results shown in figures 12-14.

This project enabled further insight into QLLs and better grasp the theory behind phases distinguishing algorithms such as DeepIce and how it was used to understand which molecules were ice and which were water. Through the use of different slabs, simulation tools, visualisation tools and DeepIce, multiple relationships were observed. The main relationships were how the increase in temperature consequently increased the number of water molecules, QLL thickness and the standard deviations of both quantities suggesting larger fluctuations. DeepIce with some python scripting to colour the molecules also allowed these relationships to be graphically represented using VMD. Correlations between varying slab ice phases led to the realisation that Zbasal, YPrism and XIc showed the smallest to the largest QLL thickness, respectively.

It was understood that manipulating the neural network itself could change the accuracy of the produced model and computational time required to produce such a model. The realisation that reducing the number of neurons whilst keeping the number of hidden layers constant

gave results with higher accuracy whilst lowering the computational time was fascinating. Furthermore, differing the batch number gave intuitive results such as the fact the time necessary for an epoch to run increased with a lower batch size. But, the accuracy deteriorated substantially for batch size 10 of hexagonal ice and water training.

In conclusion, DeepIce served as an insightful solution to ice water molecule distinguishing processes and ML. It also produced accurate results to analyse the quantifiable data from the mW potential synthesised QLLs.

8 Summary

The idea that **machine learning** can achieve pattern recognition, just as the human brain is able to, has allowed extracting information from simulations which would otherwise be an extremely tedious task to do by the human eye. **DeepIce** is an example of a machine learning algorithm. The algorithm aims to learn the structure of H_2O **phases** and, hence, distinguish the structures within systems which consist of multiple different phases.

DeepIce is built of four different **artificial neural networks**. Artificial neural networks is a method to simulate how the human brain learns. It receives an input of data to train with. This data is transformed in such a way that the individual networks are compatible. Each of these networks learns about the different quantities related to the phases of ice. All of these networks are connected together to create trained weights.[19]

To use DeepIce, different phases of ice are required to be simulated to produce data sets to train the machine learning model with. To achieve this, a model of water called the **mW potential** was used. This model boasts its low **computational time** for simulations and its accurate **reproducibility** of quantities such as the melting point and density of water.

The machine learning algorithm, DeepIce, was consequently used to differentiate between the phases of ice on **quasi-liquid layers (QLLs)**. QLLs are simply the thin layer of H_2O with melting temperatures below the main bulks melting point. Many properties of ice are attributed to its QLLs. For example, QLLs are responsible for the slippery surface of ice. This allows people to enjoy winter sports such as curling.

DeepIce was able to accurately distinguish between water-like and ice-like molecules. Therefore, the size of these QLLs could be calculated. The number of water molecules and the QLL thickness was calculated for the different QLLs used. Two main relationships with temperature were observed. Firstly, the water number and QLL thickness increased with temperature. Secondly, the water number and QLL thickness fluctuated more as the temperature increased. Both these trends agree with intuition. Furthermore, the QLL thickness for the different bulks were in the order: Basal < Primary Prism < Cubic for all the tested temperatures. The image above shows an example of prediction made by the machine learning algorithm outlining some error as well as clearly defining the QLLs.

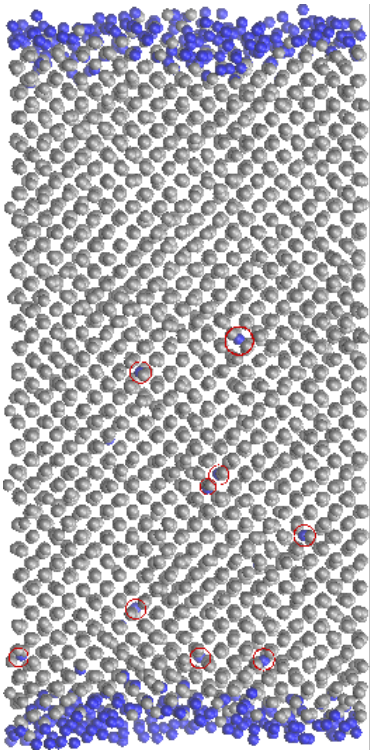


Figure 20: An image showing the prediction DeepIce produced on a cubic ice slab with QLLs. Blue is water and silver is ice. The red circles represent errors within the prediction.

As the progress in technology allows for more complex systems to be simulated, having powerful machine learning algorithms such as DeepIce to distinguish between H_2O phases would allow for more **accurate molecular analysis**.

References

[1] Batch size, epochs and their effect on learning. URL: https://www.srose.biz/wp-content/uploads/2020/08/Batch-Size-and-Epochs.html.

[2] Fixes. URL: https://docs.lammps.org/fixes.html.

[3] Ice phases - idc-online.com. URL: https://www.idc-online.com/technical_references/pdfs/chemical_engineering/Ice_phases.pdf.

[4] Radialdistribution. URL: https://docs.quantumatk.com/manual/Types/RadialDistribution/RadialDistribution.html.

[5] Run_style command. URL: https://docs.lammps.org/run_style.html.

[6] The velocity verlet algorithm. URL: https://www.ccp5.ac.uk/sites/www.ccp5.ac.uk/files/Democritus/Theory/verlet.html.

[7] What are neural networks?, Aug 2020. URL: https://www.ibm.com/uk-en/cloud/learn/neural-networks.

[8] Lecture 2: Pair distribution function (pdf) and radial distribution function (rdf), Jan 2021. URL: https://www.youtube.com/watch?v=uB_blBtUOKo&ab_channel=MatSciBrain.

[9] What is overfitting?, Mar 2021. URL: https://www.ibm.com/cloud/learn/overfitting.

[10] What is underfitting?, Mar 2021. URL: https://www.ibm.com/cloud/learn/underfitting.

[11] JLF Abascal, E Sanz, R García Fernández, and C Vega. A potential model for the study of ices and amorphous water: Tip4p/ice. *The Journal of chemical physics*, 122(23):234511, 2005.

[12] Jason Brownlee. Loss and loss functions for training deep learning neural networks, Oct 2019. URL: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.

[13] Jason Brownlee. Softmax activation function with python, Jun 2020. URL: https://machinelearningmastery.com/softmax-activation-function-with-python/.

[14] Martin Chaplin. Cubic ice (ice ic and ice xic), 2021. Last accessed 1 March 2022. URL: https://water.lsbu.ac.uk/water/cubic_ice.html.

[15] Martin Chaplin. Hexagonal ice (ice ih), 2021. Last accessed 1 March 2022. URL: https://water.lsbu.ac.uk/water/hexagonal_ice.html.

[16] MM Conde, C Vega, and A Patrykiejew. The thickness of a liquid layer on the free surface of ice as obtained from computer simulation. *The Journal of chemical physics*, 129(1):014702, 2008.

[17] Ivo D Dinov. Black box machine-learning methods: Neural networks and support vector machines. In *Data science and predictive analytics*, pages 383–422. Springer, 2018.

[18] Alaina EO Emmanuel. *Molecular Simulation of Ice Growth Inhibition by Biomimetic Antifreeze Macromolecules*. PhD thesis, University of Warwick England, 2015.

[19] Maxwell Fulford, Matteo Salvalaglio, and Carla Molteni. Deepice: a deep neural network approach to identify ice and water molecules. *Journal of Chemical Information and Modeling*, 59(5):2141–2149, 2019.

[20] Philipp Geiger and Christoph Dellago. Neural networks for local structure detection in polymorphic systems. *The Journal of chemical physics*, 139(16):164105, 2013.

[21] Toni Giorgino. Computing diffusion coefficients in macromolecular simulations: the diffusion coefficient tool for vmd. *Journal of Open Source Software*, 4(41):1698, 2019.

[22] Thomas C Hansen. The everlasting hunt for new ice phases. *Nature Communications*, 12(1):1–3, 2021.

[23] Alexander Jung. *Machine learning: The basics*. Springer Nature, 2022.

[24] QHwan Kim, Joon-Hyuk Ko, Sunghoon Kim, and Wonho Jhe. Gcicenet: a graph convolutional network for accurate classification of water phases. *Physical Chemistry Chemical Physics*, 22(45):26340–26350, 2020.

[25] Wolfgang Lechner and Christoph Dellago. Accurate determination of crystal structures based on averaged local bond order parameters. *The Journal of chemical physics*, 129(11):114707, 2008.

[26] Yimin Li and Gabor A Somorjai. Surface premelting of ice. *The Journal of Physical Chemistry C*, 111(27):9631–9637, 2007.

[27] Kenneth G Libbrecht. Snow crystals. *arXiv preprint arXiv:1910.06389*, 2019.

[28] Patrick B Louden and J Daniel Gezelter. Why is ice slippery? simulations of shear viscosity of the quasi-liquid layer on ice. *The journal of physical chemistry letters*, 9(13):3686–3691, 2018.

[29] Walter Mickel, Sebastian C Kapfer, Gerd E Schröder-Turk, and Klaus Mecke. Shortcomings of the bond orientational order parameters for the analysis of disordered particulate matter. *The Journal of Chemical Physics*, 138(4):044501, 2013.

[30] Tom M Mitchell et al. Machine learning, 1997.

[31] Valeria Molinero and Emily B Moore. Water modeled as an intermediate element between carbon and silicon. *The Journal of Physical Chemistry B*, 113(13):4008–4016, 2009.

[32] Andrew Murphy. Batch size (machine learning), May 2019. URL: https://radiopaedia.org/articles/batch-size-machine-learning?lang=gb#:~:text=Batch%20size%20is%20a%20term,examples%20utilised%20in%20one%20iteration.

[33] Ken Nagashima, Gen Sazaki, Tetsuya Hama, Harutoshi Asakawa, Ken-ichiro Murata, and Yoshinori Furukawa. Direct visualization of quasi-liquid layers on ice crystal surfaces induced by hydrogen chloride gas. *Crystal Growth & Design*, 16(4):2225–2230, 2016.

[34] Yuki Nagata, Tetsuya Hama, Ellen HG Backus, Markus Mezger, Daniel Bonn, Mischa Bonn, and Gen Sazaki. The surface of ice under equilibrium and nonequilibrium conditions. *Accounts of chemical research*, 52(4):1006–1015, 2019.

[35] Ignacio Pickering, Martin Paleico, Yamila A Perez Sirkin, Damian A Scherlis, and Matias H Factorovich. Grand canonical investigation of the quasi liquid layer of ice: Is it liquid? *The Journal of Physical Chemistry B*, 122(18):4880–4890, 2018.

[36] Prashanth Saravanan. Understanding loss functions in machine learning, Feb 2021. URL: https://www.section.io/engineering-education/understanding-loss-functions-in-machine-learning/#introduction.

[37] Avinash V Sharma. Understanding activation functions in neural networks, Mar 2017. URL: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

[38] Jihong Shi, Maxwell Fulford, Hui Li, Mariam Marzook, Maryam Reisjalali, Matteo Salvalaglio, and Carla Molteni. Investigating the quasi-liquid layer on ice surfaces: a comparison of order parameters. *Pre-print*, 2022.

[39] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[40] Ben Slater and Angelos Michaelides. Surface premelting of water ice. *Nature Reviews Chemistry*, 3(3):172–188, 2019.

[41] Alan M Turing and J Haugeland. Computing machinery and intelligence. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, pages 29–56, 1950.

[42] Neelam Tyagi. 7 popular applications of machine learning in daily life, Apr 2020. URL: https://www.analyticssteps.com/blogs/7-popular-applications-machine-learning-daily-life.

[43] Eric Wang. Turnitin tech talk: Artificial intelligence and machine learning at turnitin, Jan 2022. URL: https://www.turnitin.com/blog/artificial-intelligence-and-machine-learning-at-turnitin.

[44] Chris Woodford. How neural networks work - a simple introduction, Aug 2021. URL: https://www.explainthatstuff.com/introduction-to-neural-networks.html.

[45] Chongqin Zhu, Yurui Gao, Weiduo Zhu, Yuan Liu, Joseph S Francisco, and Xiao Cheng Zeng. Computational prediction of novel ice phases: A perspective. *The Journal of Physical Chemistry Letters*, 11(17):7449–7461, 2020.

Appendix A

Ic_wlh

batch10						batch30				
epoch	time / s	accuracy	loss	validation accuracy	validation loss	time / s	accuracy	loss	validation accuracy	validation loss
1	1109	0.9969	0.0298	0.9875	0.2015	366	0.9986	0.0088	0.9888	0.0162
2	1117	0.9968	0.0517	0.9985	0.024	400	0.9948	0.0597	0.9277	0.4515
3	1071	0.9965	0.0569	0.9976	0.0394	409	0.9908	0.1459	0.9999	0.0013
4	1091	0.9916	0.1348	0.9975	0.0405	411	0.9985	0.0246	0.9974	0.0415
5	1147	0.9871	0.2083	0.9945	0.0884	388	0.9987	0.0208	0.9974	0.0415
6	1080	0.9929	0.1148	0.9945	0.0884	393	0.9987	0.0208	0.9974	0.0415
7	1129	0.9929	0.1148	0.9945	0.0884	388	0.9987	0.0208	0.9974	0.0415
8	1141	0.9929	0.1148	0.9945	0.0884	387	0.9987	0.0208	0.9974	0.0415
9	1089	0.9929	0.1148	0.9945	0.0884	403	0.9987	0.0208	0.9974	0.0415
10	1129	0.9929	0.1148	0.9945	0.0884	391	0.9987	0.0208	0.9974	0.0415
11	1115	0.9929	0.1148	0.9945	0.0884	387	0.9987	0.0208	0.9974	0.0415
12	1117	0.9929	0.1148	0.9945	0.0884	404	0.9987	0.0208	0.9974	0.0415
13	1118	0.9929	0.1148	0.9945	0.0884	405	0.9987	0.0208	0.9974	0.0415
14	1104	0.9929	0.1148	0.9945	0.0884	402	0.9987	0.0208	0.9974	0.0415
15	1102	0.9929	0.1148	0.9945	0.0884	403	0.9987	0.0208	0.9974	0.0415
16	1113	0.9929	0.1148	0.9945	0.0884	398	0.9987	0.0208	0.9974	0.0415
17	1103	0.9929	0.1148	0.9945	0.0884	421	0.9987	0.0208	0.9974	0.0415
18	1111	0.9929	0.1148	0.9945	0.0884	418	0.9987	0.0208	0.9974	0.0415
19	1107	0.9929	0.1148	0.9945	0.0884	417	0.9987	0.0208	0.9974	0.0415
20	1101	0.9929	0.1148	0.9945	0.0884	417	0.9987	0.0208	0.9974	0.0415

Table 7: The table showing the time it took for each epoch with batch sizes 10 and 30 when training DeepIce with cubic ice and 260K supercooled water. The table also shows the training accuracy, training loss, validation accuracy and validation loss. *Note: Investigation was conducted on a Desktop with a AMD Ryzen 5 3600 6-Core Processor and 16.0 GB RAM.*

Ih_wlh

batch10						batch30				
epoch	time / s	accuracy	loss	validation accuracy	validation loss	time / s	accuracy	loss	validation accuracy	validation loss
1	808	0.9985	0.0074	0.9923	0.1229	403	0.9988	0.0131	0.9996	0.0015
2	806	0.7152	4.5888	0	16.1181	409	0.9995	0.002	0.9993	0.0052
3	827	0.5556	7.1636	0	16.1181	413	0.9996	0.002	0.9998	0.0012
4	820	0.5556	7.1636	0	16.1181	406	0.948	0.8338	0.9995	0.0043
5	819	0.5556	7.1636	0	16.1181	398	0.9995	0.0026	0.9998	0.0013
6	800	0.5556	7.1636	0	16.1181	398	0.9997	0.0015	0.9999	0.0011
7	786	0.5556	7.1636	0	16.1181	395	0.9773	0.363	0.9998	0.0013
8	800	0.5556	7.1636	0	16.1181	404	0.9997	0.0016	0.9999	0.0009137
9	771	0.5556	7.1636	0	16.1181	421	0.9998	0.0012	0.9998	0.0018
10	776	0.5556	7.1636	0	16.1181	400	0.9997	0.0016	0.9998	0.0024
11	806	0.5556	7.1636	0	16.1181	419	0.9998	0.0016	0.9997	0.0023
12	817	0.5556	7.1636	0	16.1181	413	0.9998	0.0022	0.9993	0.0077
13	809	0.5556	7.1636	0	16.1181	412	0.9998	0.0012	0.9998	0.0016
14	808	0.5556	7.1636	0	16.1181	412	0.9997	0.0016	0.9999	0.0015
15	801	0.5556	7.1636	0	16.1181	395	0.9998	0.0023	0.9995	0.0072
16	791	0.5556	7.1636	0	16.1181	440	0.9998	0.0024	0.9998	0.0014
17	815	0.5556	7.1636	0	16.1181	434	0.9999	0.00097439	0.9998	0.0018
18	815	0.5556	7.1636	0	16.1181	430	0.9998	0.0015	0.9989	0.0147
19	825	0.5556	7.1636	0	16.1181	430	0.9998	0.0014	0.9996	0.0055
20	812	0.5556	7.1636	0	16.1181	309	0.6767	5.2108	0	16.1181

Table 8: The table showing the time it took for each epoch with batch sizes 10 and 30 when training DeepIce with hexagonal ice and 260K supercooled water. The table also shows the training accuracy, training loss, validation accuracy and validation loss. *Note: Investigation was conducted on a Desktop with a AMD Ryzen 5 3600 6-Core Processor and 16.0 GB RAM.*