Linköping University | Department of Statistics and Machine learning
Master's thesis, 30 ECTS | Statistics
2023 | LIU-IDA/STAT-A--23/022--SE

Stochastic knowledge-based placement of water molecules in macromolecular structures

Prakhar

Supervisor : Amanda Olmin
Examiner : Krzysztof Bartoszek

External supervisor : Nicholas Pearce

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

Determination of water binding sites in protein structures is an area of extensive research with important applications in drug design and understanding protein functions. In this work a new stochastic Bayesian model is developed to predict water positions in protein structures referred to as Bayesian-CNN, a convolutional neural network that builds upon an existing deterministic model called GalaxyWater-CNN[11] by adding Bayesian convolutional layers. The training and test datasets are the same as those used for GalaxyWater-CNN. Comparison of the outputs of the two models indicates that the Bayesian-CNN can predict water positions with similar accuracy as that of GalaxyWater-CNN. Also the analysis of multiple stochastic predictions generated by Bayesian-CNN indicates that the variability in the predicted water positions mimics the uncertainty in the positions of individual water molecules to a small extent.

Acknowledgments

I would like to take this opportunity to thank my supervisor Amanda Olmin for being very helpful, and taking the time out of her busy schedule. Also for suggesting useful books and articles to help me understand different statistical methods.

I also extend my gratitude to my external supervisor Nicholas Pearce for his time and help with understanding domain specific concepts and the data, and providing useful background literature, and also for offering computing resources to train the neural network model.

A special thanks to my examiner Krzysztof Bartoszek and opponent Patrick Hiemsch for making some critical suggestions for improvements in the thesis draft.

Contents

List of Figures

1 Introduction

1.1 Background

Proteins are large molecules that are responsible for a variety of biological functions. Proteins are made up of long chains of amino acids [17] called polypeptides. A protein molecule can contain one or more such polypeptides. There are four structural levels of a protein : primary, secondary, tertiary and quaternary [26]. The primary structure 1.1 of a protein refers to the sequence of amino acids in the polypeptide chain.

Secondary structure refers to the repeating local structures within a polypeptide caused by Hydrogen bonding [22]. These are mainly of two types: α-helix and β-sheet as shown in the figure 1.2. Many such repeating structures can be present in a polypeptide.

Tertiary structure refers to the overall folding of a polypeptide resulting in a three dimensional structure of a single polypeptide chain as seen in figure 1.3.

Quaternary structure refers to the aggregation of two or more polypeptide chains to form a functional protein molecule as seen in figure 1.4.

The function of a protein is strongly dependent on its structure. If a protein loses its shape at any structural level, it may no longer be functional. Proteins are not static molecules as they have many degrees of freedom. The environment of the protein eg. the presence of water and other molecules can impact the activity of the protein. Small molecules like water adopt energetically favourable positions (bind) on the surface of proteins which are then subsequently displaced by or bind cooperatively with other molecules. These energetically favourable positions are determined by the structure of the protein and the interactions between the residues 1.2 in its polypeptide chains. Knowing the binding sites of water molecules can provide insight into the kinetics 1.2 of the protein and can help in the design of pharmaceuticals. There are various thermodynamic/kinetic 1.2 simulation and phylogenetics 1.2 based models avail-

Figure 1.1: Primary Structure of a small segment of a polypeptide showing the sequence of amino acids

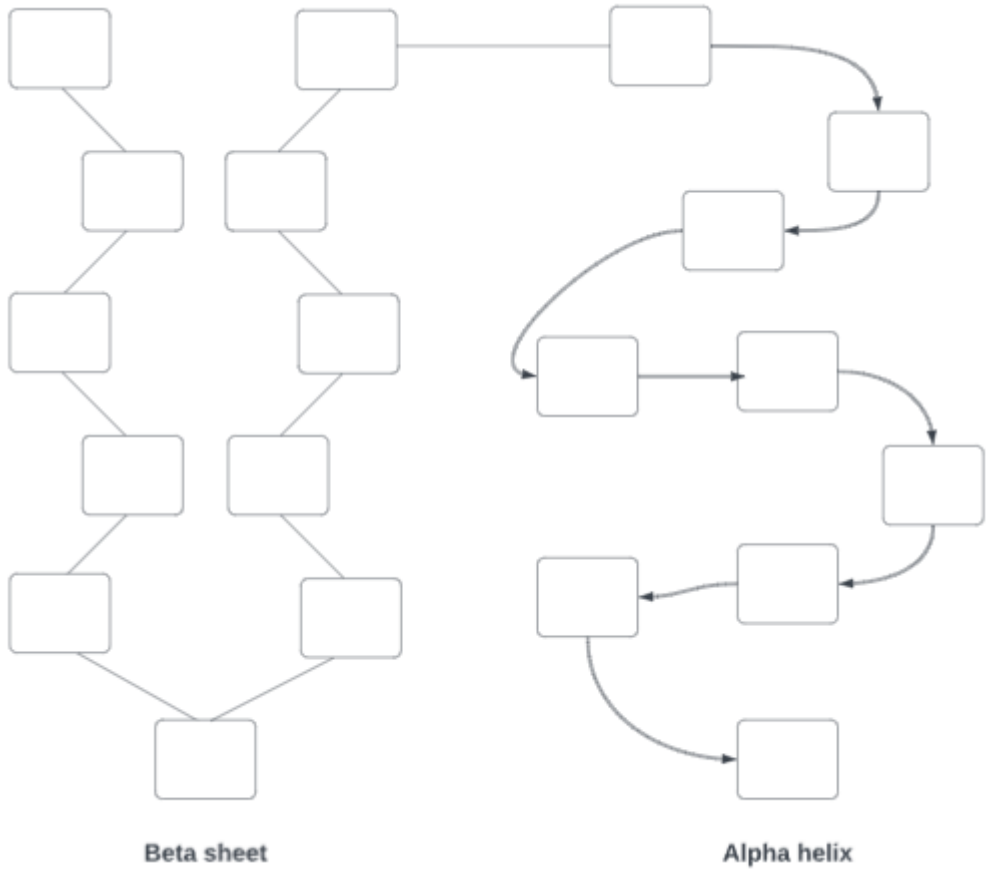


Figure 1.2: Secondary Structure of a segment of a polypeptide showing the two types of structures, α-helix and β-sheet

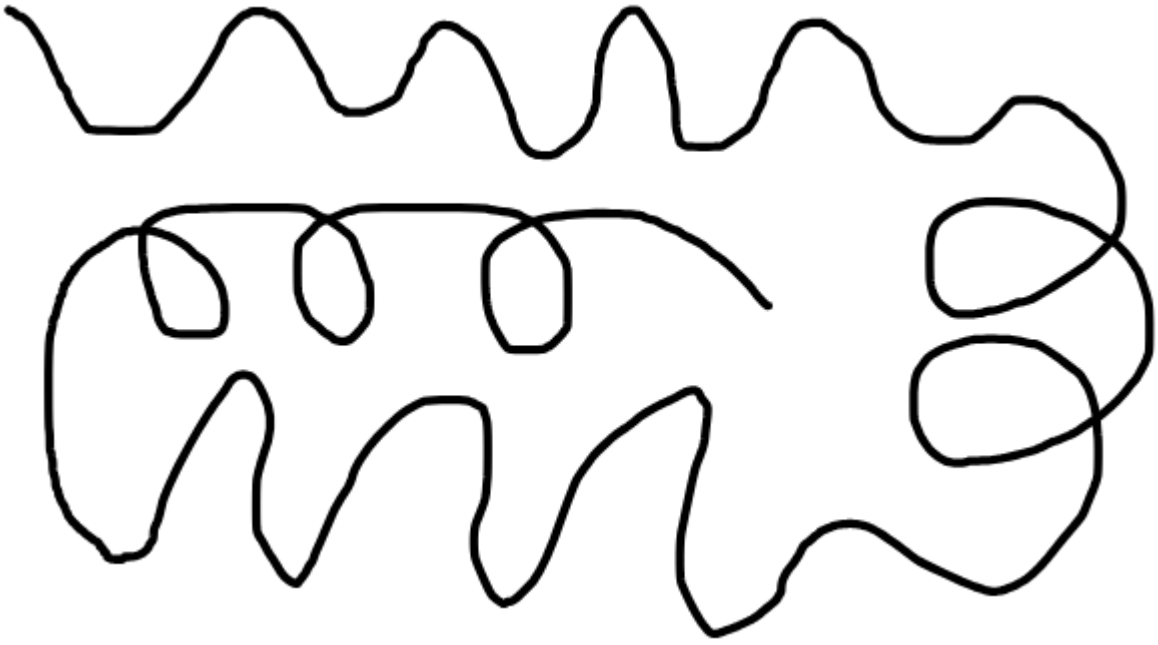


Figure 1.3: A simplified diagram showing the tertiary Structure of a polypeptide which results in its overall 3D shape

Figure 1.4: Quaternary Structure of protein 3RV1 showing the arrangement of the different polypeptide chains (Image source: https://www.rcsb.org/structure/3RV1 [15])

able that can predict protein structures with atomic accuracy, most notable among these being the AlhpaFold2 developed by DeepMind [8]. However there are currently very few machine learning models for modelling of water/solvent positions with atomic accuracy.

1.2 Definitions

In this section we provide the definitions of some terms related to protein structures and their study which are used frequently throughout this text.

Functional Group

A functional group is a group of atoms with a characteristic set chemical properties [21]. Molecules that contain a functional group share these chemical properties regardless of the rest of their compositions. Functional groups are usually charged and thus can attract other molecules and ions with opposite charge. The affinity of a water molecule can be affected by the presence of functional groups nearby.

Amino Acid/Residue

An amino acid is a molecule that contains two functional groups called amino and carboxyl. These functional groups impart the amino acids with their characteristic chemical properties. Out of over 500 naturally occurring amino acids, only 22 are found in proteins [17]. A protein

essentially consists of one or more long chains of amino acids called polypeptides. Within the context of a polypeptide, amino acids are also referred to as a residues.

Hetero-atoms

Hetero-atoms are the atoms that are not a part of the protein molecule, like water atoms, metal ions and other atoms.

B-factor

Protein molecules are not stationary but undergoing constant thermal motion which impacts the X-ray crystallography during the determination of protein structure. The degree of thermal motion varies for each molecule (and atom), and X-ray crystallography is better able to capture the position of atoms that are relatively more stable. B-factor is a measure of the mobility of an atom. Atoms with low B-factor (<10) are undergoing less thermal motion and therefore the crystallography technique is able to capture their positions better. On the other hand atoms with very large B-factors (>50) are undergoing so much thermal motion that they are poorly visible and thus are usually ignored and their positions are not documented [12].

Chemical Thermodynamics

A branch of chemistry that studies the interrelationship between heat and the extent of a chemical reaction. [19]

Chemical Kinetics

A branch of chemistry involving the study of rates of chemical reactions and how they are influenced by various experimental conditions. [18]

Phylogenetics

A branch of biology involving the study of evolutionary history of organisms and the evolutionary relationships between different organisms. [25]

Ligand

Ligand is any small molecule e.g. drugs or water molecules that can bind to proteins and modify their behaviour. [23]

Protein-Ligand Docking

It is a molecular modelling technique that predicts the positions and orientations of ligands bound to a protein.[27]

1.3 Related Work

There are various categories of protein-ligand docking1.2 methods that can find the binding sites of various kinds of ligands1.2 on a protein structure. Thermodynamic prediction methods, also called explicit methods because they treat every water molecule individually, use the thermodynamic interactions between protein residues and water molecules. For example Molecular Dynamic (MD) simulation methods like WATsite [5] use computer simulations to analyze the movements and trajectories of water molecules to find the binding sites. Structure Based Drug Design (SBDD) methods such as DiffSBDD[14] are used to design or generate new ligands that can bind to a given protein structure. However such explicit methods have

to take into account each water molecule individually and thus are computationally expensive.
Another category of methods are implicit methods [11] which treat water as a continuous medium and thus avoid a lot of computations involving the treatment of each water molecule separately, but still need to take into account the interactions between the protein atoms and the water medium. For example methods like HydraMap[9] and GalaxyWater-wKGB[4] that use statistical potential to predict water binding sites. Statistical potential is a scoring function derived by an analysis of known protein structures, that can be used to obtain the score for a water binding site [28].
In this research work we build a simple stochastic model for prediction of water molecules called Bayesian-CNN. For this we use a model called GalaxyWater-CNN[11], a 3D convolutional neural network that predicts the positions of water molecules bounded on the protein surfaces using the structure of the protein, as a baseline model. The method used by GalaxyWater-CNN is simpler because it ignores all the explicit and implicit interactions between the protein and its environment and predicts the water binding sites based solely on the structure of the protein. However it is a deterministic method that predicts fixed water positions for a given protein structure which does not accord with the reality as water binding sites always have some uncertainty associated with them. To build Bayesian-CNN architecture we replace some of the 3D convolution layers of the GalaxyWater-CNN with Bayesian 3D convolution layers and retrain on the same training data to obtain a stochastic model. We then compare the performance of GalaxyWater-CNN with our Bayesian modification. The reasons for adding Bayesian layers are further explained in the section 4.3.

1.4 Objectives

The main objectives include:

1. Does the Bayesian-CNN perform better on test data in terms of cross entropy loss and predicted water positions?
2. Is the Bayesian model able to capture the uncertainty in the positions of water molecules?

2 Theory

Here we present a brief introduction of Convolutional Neural Networks and Bayesian Neural Networks, which form the basis of the methods described in the next chapter.

2.1 Convolutional Neural Network

Convolutional neural networks are neural networks that are designed for grid structured data like 2D images or 3D volumetric data. The input is a grid which can be of one, two or more dimensions. For example, a 2D image can be thought of as a rectangular grid with each grid point representing a pixel of the image. Representing the structured data as a grid preserves the structure of the data and the spatial information contained in the data [10]. This is in contrast to ordinary neural networks that flatten the data into a vector. The grid input is then processed using one or many convolutional layers which are introduced briefly in the following subsection. The advantage of using the convolutional layers is that it maintains the dimensionality of the data which means that the output has the same number of dimensions as the input although it can be of different size.

Convolutional Layers

In the convolutional layer we have nodes, also called hidden units, arranged in the same way as the input grid. Unlike a dense layer where each node is connected to all the input features, in a convolutional layer a node is connected to only a small region of the input features, as shown in the figure 2.1 for 2D input. This region can be viewed as a small rectangular(in case of 2D input) or cubical(3D input) or higher dimensional(higher dimensional input) window that consists of the input features surrounding the node. Corresponding to each input feature within this window there is a weight(parameter). So for each node we have a set of weights called a filter as seen in figure 2.2, which shows a 2D(3x3) filter. There is a weight sharing among all the nodes of a convolutional layer, which means that each node in a convolutional layer has the same set of weights (filter), which reduces the number of weights significantly as compared to a dense layer. There can be more than one filters of same or different sizes in a convolutional layer, but all the filters are shared among all its nodes. The output of each node is a mapping of the input variables corresponding to the node and the filter, which results

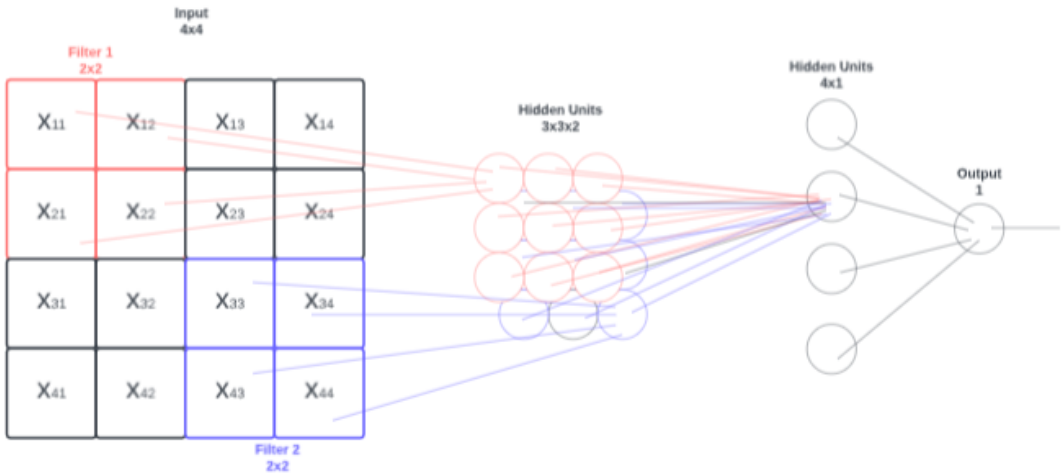


Figure 2.1: A CNN architecture showing a 2D input, a convolutional layer with 2 filters followed by a dense layer and output layer

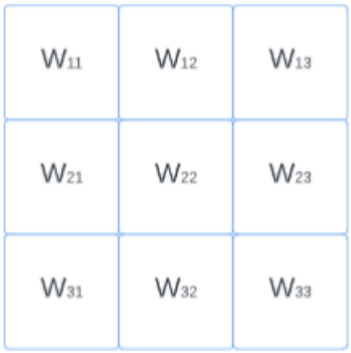


Figure 2.2: A 3x3 filter for single channel 2D input. The filter is shared by all the nodes in the convolutional layer

in a scalar output. This mapping is also called a convolution. The outputs of all the nodes together constitute a feature map, which has the same number of dimensions as the input.

If there are N number of filters, we get N outputs for each node as each filter generates a different output for the same input. Thus we get N feature maps, also known as channels. If the input to the convolutional layer has N channels then every filter has a separate set of weights for each channel. For example if the filter size is (3x3) for one channel, it should be (Nx3x3) for N channel input.

For N channel 2D data and assuming Nx3x3 filter size and same number of output channels, the output for a given channel of the convolutional layer can be represented as:

$$y_{ijn} = (\sum_{k=1}^{F}\sum_{l=1}^{F}\sum_{m=1}^{N} x_{i+k-2,j+l-2,m} W_{k,l,m,n})$$

$y_{i,j,n}$: output corresponding to row i, column j and output channel n
$x_{i,j,m}$: input corresponding to row i, column j and input channel m
$W_{k,l,m,n}$: Weight corresponding to filter row k, filter column l, input channel m and output channel n (n^{th} filter)
F : the number of rows and columns in the input window (filter size)
N : the number of input channels

Zero Padding

Assuming a single channel 2D input with size IxI, and filter with size FxF, the output will be of size JxJ, such that:

$$J = I + 1 - F$$

I: Input size
J: Output size
F: Filter size

So convolution of the input with any filter having size greater than 1x1 will result in the output size being less than the input size as seen in figure 2.3.

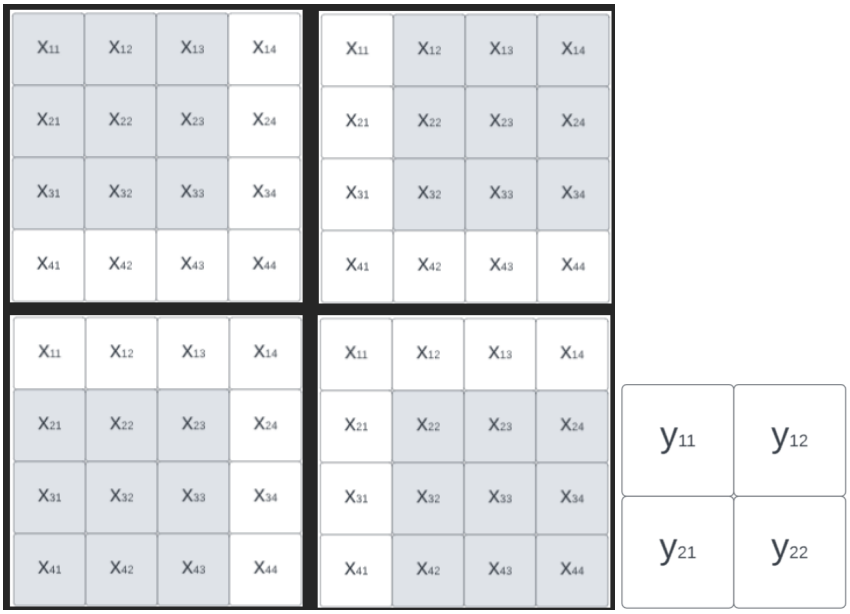


Figure 2.3: convolution of a 4x4 input with a 3x3 filter showing the filter moving across the input resulting in a 2x2 output

To keep the output size the same as the input, we need to introduce some padding around the features at the edges of the input grid as seen in the figure 2.4. This is done to keep the number of nodes in the convolutional layer equal to the number of features in the input, and thus the output size equal to the input size. This is usually done by adding zeroes around the features along the edges, which is called zero padding of the input. After including padding with width P around the input, the output size can be calculated as:

$$J = I + 2P + 1 - F$$

I: Input size
J: Output size
F: Filter size
P: Padding Width

Atrous Convolutions

Atrous convolutions, also called dilated convolutions are used to expand the window size for each node without increasing the number of weights in the filter. This is done by introducing gaps in the window by skipping some of the features surrounding the node as shown in the

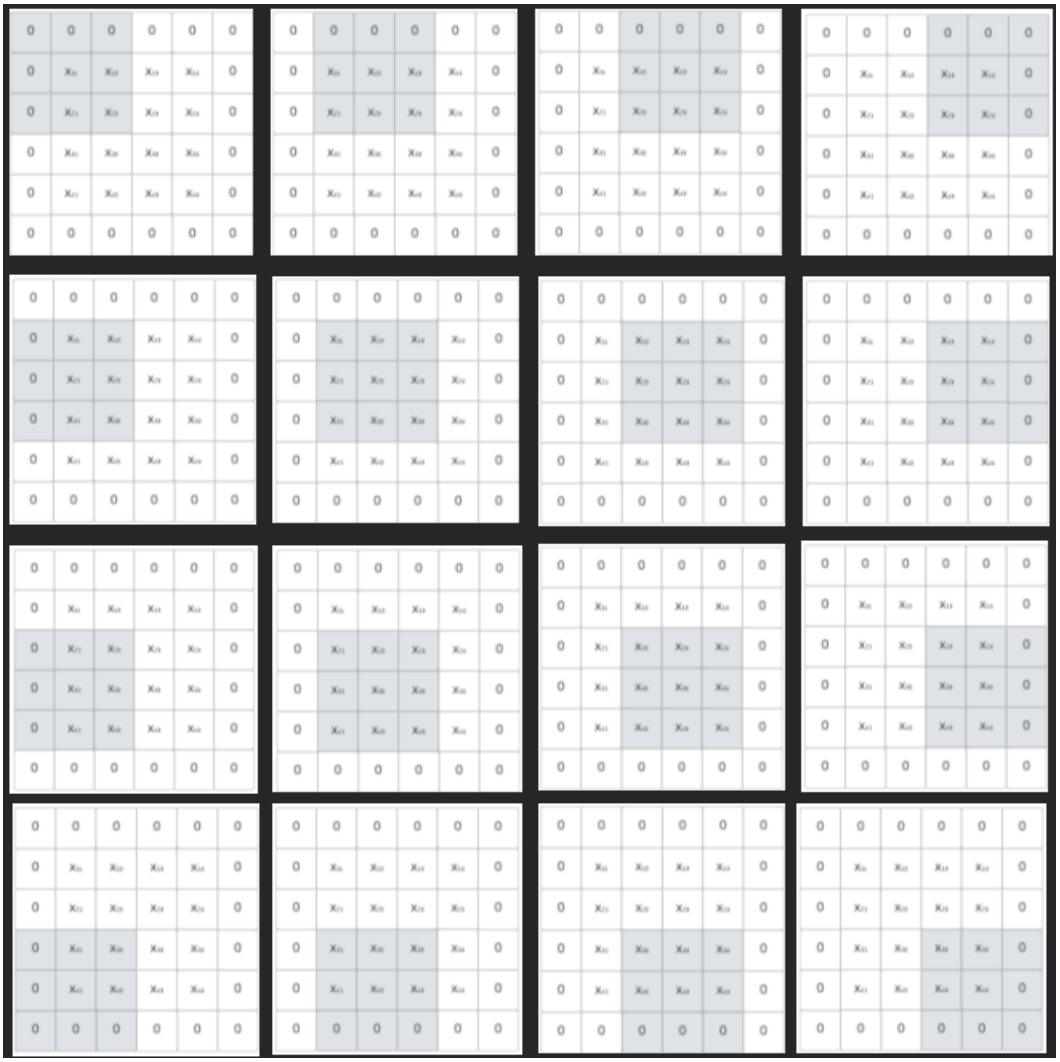


Figure 2.4: convolution of a 4x4 input with a 3x3 filter and zero padding of width 1, showing the filter moving across the input resulting in a 4x4 output

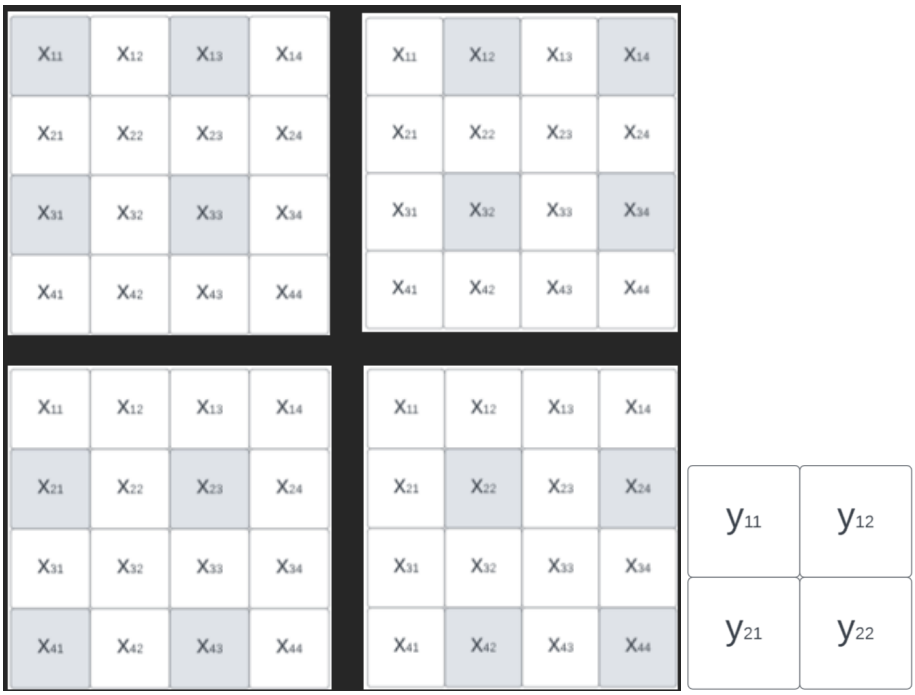


Figure 2.5: atrous convolution of a 4x4 input with a 2x2 filter with dilation 2, showing the filter moving across the input resulting in a 2x2 output

figure 2.5. This is done to include a larger region of the input for each node. The size of the gaps is described using dilation rate, with a dilation rate of 1 meaning no gap between the input features, dilation rate of 2 meaning introduction of 1 gap between input features, and so on. With atrous convolutions the output for a given channel of the convolutional layer can be represented as:

$$y_{ijn} = (\sum_{k=1}^{F}\sum_{l=1}^{F}\sum_{m=1}^{N} x_{i+(k-2)d,j+(l-2)d,m} W_{k,l,m,n})$$

$y_{i,j,n}$: output corresponding to row i, column j and output channel n
$x_{i,j,m}$: input corresponding to row i, column j and input channel m
$W_{k,l,m,n}$: Weight corresponding to filter row k, filter column l, input channel m and output channel n (n^{th} filter)
F : the number of rows and columns in the input window (filter size)
N : the number of input channels
d: dilation rate

2.2 Bayesian Neural Networks

Bayesian neural networks differ from standard neural networks in the sense that whereas a standard neural network learns the point estimates of the weights, a Bayesian neural network learns the distribution of the weights. In this section, we briefly explain the working of standard neural networks and then the differences between a standard and a Bayesian neural network. To this end, we use simple dense feedforward neural networks rather than convolutional neural networks, but the same ideas can be extended to the convolutional networks. A dense feedforward neural network is a simple neural network in which each node in a layer is connected to each node in the next layer. Thus the output of one layer is fed as the input to the next layer, resulting in a forward flow of information from the input layer to the output layer. This is in contrast to other types of neural networks in which the connections between

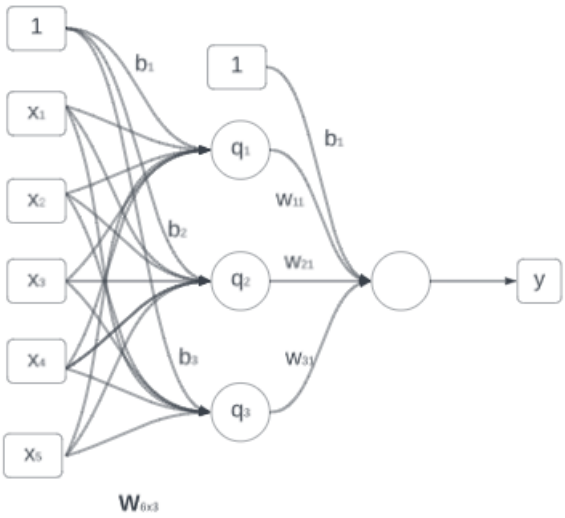


Figure 2.6: A simple feedforward neural network with 5 dimensional input and a single hidden layer with 3 nodes, and the output layer with 1 node. Some of the connections between the nodes of consecutive layers are labelled with the corresponding weights of the network. An additional input dimension with value 1 is added to the input layer and each hidden layer for the bias. More hidden layers can be added between the input and output layers to obtain a deep neural network.

nodes may form closed loops or cycles. Henceforth in this section, we use the term neural network to refer to a dense feedforward neural network unless stated otherwise.

A deep neural network is a neural network that consists of many layers known as hidden layers, between the input and the output layers, although there is no consensus on how many layers a network should have in order to qualify as a deep neural network[20]. Such deep neural networks have been shown to be able to approximate almost any complex input output relationship[10]. This is done by using multiple computational units called nodes working in parallel on the same input to form a layer, and stacking multiple such layers in such a way that the output of each layer acts as the input to the subsequent layer as shown in the figure 2.6.

Each node has a set of parameters including an offset parameter called bias, which are collectively called the weights of the node. It generates a scalar output which is a linear combination of the input components. A nonlinear function called activation function may be applied to the linear combination to obtain the final output of the node (eq. 2.1).

$$q = h(\sum_{i=1}^{n} x_i w_i + b) \tag{2.1}$$

q : output of the node
n : number of inputs to the node
x_i : i^{th} component of the input to the node
w_i : i^{th} weight of the node
b : an offset or bias added by the node
h : activation function

The output the neural network can be considered as a function of the input and the weights of the network as shown in the equation 2.2 where the the network is represented as a function F of the input parameterized by the weights **w**.

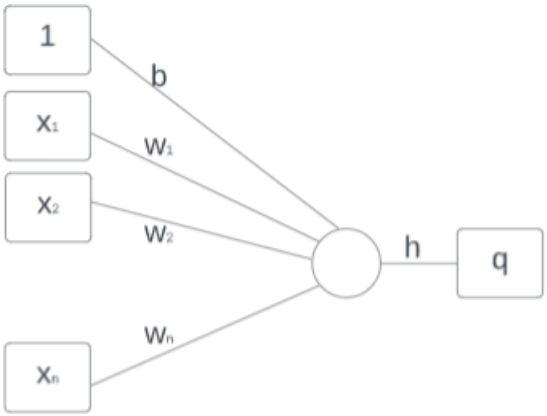


Figure 2.7: A single node in a neural network showing input components(x_i), weights(w_i, b), activation function(h) and output(q)

$$\hat{y} = F_{\mathbf{w}}(x) \tag{2.2}$$

As the neural network is only an approximation of the input output relationship, it cannot produce the exact same output for a given input so there is always some unaccounted error called random noise between the actual output and the prediction of the neural network, as shown in equation 2.3.

$$y = F_{\mathbf{w}}(x) + \epsilon \tag{2.3}$$

For convenience this random noise is usually assumed to be Gaussian random variable with 0 mean and some small variance σ^2 [7].

$$\epsilon \sim N(0, \sigma^2)$$

So the relationship between the actual output and the output of the neural network can be formulated as seen in equation 2.4, with the actual output being normally distributed around the output of the neural network, with a variance σ^2.

$$\implies y|x, \mathbf{w}, \sigma^2 \sim N(F_{\mathbf{w}}(x), \sigma^2) \tag{2.4}$$

x: input to the network
y: actual output
$\hat{y}$: output/prediction of the network
$\mathbf{w}$: set of all the weights in the network
F: function representing the network
ϵ: random noise in the actual outputs that cannot be accounted for by the model
σ^2: variance of the random noise

In a standard neural network the weights are fixed point estimates learned during the training of the network, thus the output of the network remains fixed for a given input. In other words a standard trained neural network can be viewed as a deterministic function of the input, always generating the same output for a given input.

In a Bayesian neural network rather than point estimates, weight distributions are learned for each node during the training of the network as shown in the figure 2.8. So every weight of each node acts as a random variable. As a result each time a prediction is made for the same input, the network produces a different output depending on the sampled weights, making the Bayesian neural network a stochastic function of the input.

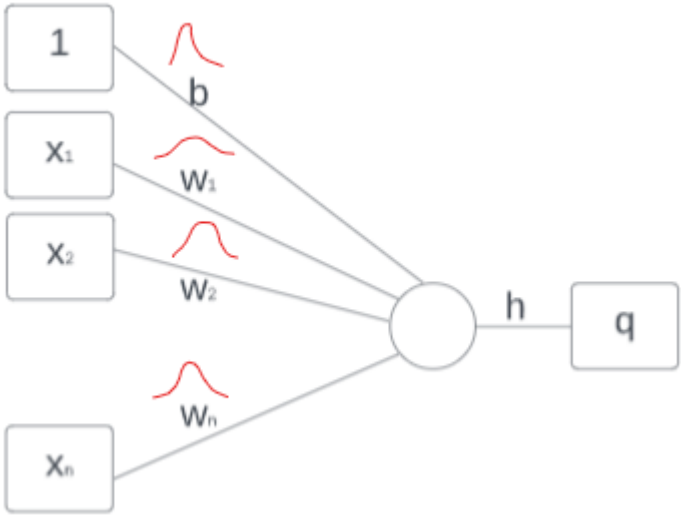


Figure 2.8: A single node in a bayesian neural network showing input components, weights distributions, activation function and the output

Training the neural networks

a) Standard Neural Network

Training a neural network involves learning the parameters of the network i.e. the weights(w) of all the nodes in the network using training data(D), which is a set of input-output pairs $\{x_i, y_i\}$. The weights are learned in such a way that the outputs generated by the network($\hat{y_i}$) on training data inputs(x_i) are as similar to the actual outputs(y_i) in the training data as possible. This similarity is measured using some appropriate similarity metric. A common approach is to learn the weights by maximizing the likelihood of the data. Assuming that all the data points are independent and we know the variance of the random noise σ^2, likelihood of the training data can be calculated as in the equation 2.5. Note that here we are explicitly expressing the likelihood as a function of the network weights(w) as the training data points $\{x_i, y_i\}$ are known and the variance of the noise σ^2 is also assumed to be known.

$$L(\mathbf{w}) = \prod_{i=1}^{n} p(y_i|x_i, \mathbf{w}, \sigma^2) \tag{2.5}$$

$$\mathbf{w} = \arg\max_{\mathbf{w}} L(\mathbf{w}) = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} p(y_i|x_i, \mathbf{w}, \sigma^2) \tag{2.6}$$

L: likelihood function of weights
n: number of data points in the training data D
$y_i : i^{th}$: output in D
$x_i : i^{th}$: input in D
$\mathbf{w}$: set of weights of the network
σ^2: variance of random noise

b) Bayesian Neural Networks

Training a Bayesian neural network involves learning the posterior distribution of the weights of the network conditional on the training data p($\mathbf{w}$ | D). For this we need to start

with some prior distribution of the weights p($\mathbf{w}$). We can apply the Bayes rule to find the posterior distribution of the weights (equation 2.7).

$$p(\mathbf{w}|D) = \frac{p(D|\mathbf{w})p(\mathbf{w})}{p(D)} = \frac{p(D|\mathbf{w})p(\mathbf{w})}{\int_{\mathbf{w}'} p(D|\mathbf{w}')p(\mathbf{w}')d\mathbf{w}'} \tag{2.7}$$

$\mathbf{w}$: a given set of weights of the Bayesian neural network
D: training data
$p(\mathbf{w})$: prior density of the given set of weights
$p(\mathbf{w}|D)$: posterior density of the given set of weights conditional on the training data
$p(D|\mathbf{w})$: likelihood of the training data conditional on the given set of weights
$p(D)$: marginal likelihood of the training data

Here by likelihood of the training data conditional on a given set of weights $p(D|\mathbf{w})$ we mean the likelihood of the outputs in the training data conditional on the inputs and the weights of the network. Assuming that all the observations in the training data are independent, it can be expressed as in equation 2.8.

$$p(D|\mathbf{w}) = \prod_{i=1}^{n} p(y_i|x_i, \mathbf{w}) \tag{2.8}$$

n: number of observations in the training data
$y_i : i^th$: output in D
$x_i : i^th$: input in D

Marginal likelihood of the training data is the expectation of the likelihood of the training data under prior distribution of the weights as shown in equation 2.9.

$$p(D) = \mathbb{E}_{p(\mathbf{w})} p(D|\mathbf{w}) = \int_{\mathbf{w}'} p(D|\mathbf{w}')p(\mathbf{w}')d\mathbf{w}' \tag{2.9}$$

Calculating the marginal likelihood of the training data analytically involves solving the integral in equation 2.9 which is not possible for most of the neural networks as it does not have a closed form solution [1]. Also numerical integration is intractable because of high dimensionality of the weight vector $\mathbf{w}$, with neural networks usually having thousands of weights. So generally some approximation methods like variational inference [29] are used to approximate the posterior distribution of the weights.

Making Inferences Using Bayesian Neural Networks

In a Bayesian neural network, the weights are random variables which are drawn randomly from the posterior distribution p($\mathbf{w}$ | D) every time a prediction is made. Consequently the output of a neural network for a given input is also a random variable with a probability distribution. This property of Bayesian neural networks makes them especially suitable for modelling stochastic processes i.e. processes which generate stochastic outputs for the same input [1]. We can use the learned posterior distribution of the weights p($\mathbf{w}$ | D) to obtain the posterior predictive distribution of the output of the neural network conditional on the given input p($\hat{y}|\mathbf{x}$, D). Posterior predictive density of a given output conditional on the input is the expected density of the output conditional on the input under the posterior distribution of the weights as seen in equation 2.10.

$$p(\hat{y}|\mathbf{x}, D) = \mathbb{E}_{p(\mathbf{w}|D)} p(\hat{y}|\mathbf{x}, \mathbf{w}) = \int_{w'} p(\hat{y}|\mathbf{x}, \mathbf{w}^{'}) p(\mathbf{w}^{'}|D) d\mathbf{w}^{'} \tag{2.10}$$

$p(\hat{y}|\mathbf{x}, D)$: Posterior predictive density of the prediction $\hat{y}$ conditional on the input $\mathbf{x}$
D: Training data

This way we can find the uncertainty associated with the neural network model's predictions. However like the marginal likelihood integral seen earlier, there usually exists no analytical solution of the posterior predictive integral and numerical integration is also intractable because of high dimensionality of the weight vector. This problem is usually overcome by approximating the posterior predictive distribution using various kinds of Monte Carlo sampling techniques [7].

3 Data

3.1 Data source

We use the same training and test sets as GalaxyWater-CNN [11] without any changes for better comparison of results with the Bayesian-CNN. Here we describe the datasets, which can be found at `https://github.com/seoklab/GalaxyWater-CNN`.

Data consists of 312 high resolution(>2Å) protein structures, having water molecules between 5-20% of the total protein molecules for uniformity, taken from RCSB Protein Data Bank[2] in PDB format for effective training of the model. Further in each protein structure only the high resolution water molecules i.e. those water molecules with a B-factor less than 50 were considered as relevant. Resolution is a measure of order in a protein structure, with higher resolution structures being more uniform and thus more clearly visible [12].

3.2 Data description

PDB [2] file contains structural information about a single protein crystal. Information is organized in separate lines called records. Records can be of different types like metadata records that contain information about the source of the protein, the authors, and various other kinds of metadata. It also has remark records that contain additional information and annotations from the authors, including experimental information. SEQRES records contain the amino acid1.2 sequence of each polypeptide [24] chain in the protein. There are also other different kinds of records providing other information about the protein structure.

The records of main interest in the PDB file for this project are the ATOM and HETATM records, that contain the information about each atom and hetero-atom1.2 respectively. Each such record contains the following information:

Atom number: A number that uniquely identifies the atom.
Atom type: The type of atom eg. Nitrogen, Carbon, Oxygen.
Residue name: Name of the amino acid residue which contains the atom.
Chain name: Name of the polypeptide chain which contains the atom's residue, usually coded A, B, C...
Residue number: A number that uniquely identifies the residue.

x-coordinate (in Å)
y-coordinate (in Å)
z-coordinate (in Å)
Occupancy: The relative frequency with which the atom appears in these protein crystals.
Temperature factor: Also called B-factor, a measure of the uncertainty in the atom's position due to thermal motion.
Element Name: Name of the element of the atom.

3.3 Data preprocessing done for GalaxyWater-CNN

Each PDB file was split into 2 separate PDB files, containing ATOM(protein atom) and HETATM(hetero-atom) records respectively. In the second file containing HETATM records, only those corresponding to water atoms were kept, and others were removed. In this way the protein atom coordinates for input and water coordinates for output could be obtained easily. Out of the 312 protein structures, 160 were used for training and the remaining for test. The coordinates and functional groups1.2 of each protein atom are used for input and those of each water atom for output.

4 Methods

The method used is Bayesian-CNN, a 3D convolutional Bayesian neural network trained to predict the positions of water molecules on protein surfaces stochastically. It is constructed using the architecture of GalaxyWater-CNN[11] by replacing some of the layers with Bayesian layers. We then trained the Bayesian-CNN on the same training data as the GalaxyWater-CNN to study the effect of introducing Bayesian inference on the prediction of water molecules.
In this following sections we will describe the architecture and other details about GalaxyWater-CNN, followed by Bayesian-CNN and finally the motivation for adding Bayesian layers.

4.1 GalaxyWater CNN

Architecture

The input is a 16 channel cubic 3D grid (16 x M x M x M), where each channel represents a specific atom type or a functional group:
1. Carbon (C) atoms
2. Nitrogen (N) atoms
3. Oxygen (O) atoms
4. Sulphur (S) atoms
5. Main chain amide C or N atoms
6. Side chain amide C or N atoms
7. Side chain HIS amine C or N atoms
8. Side chain phenol C or O atoms
9. Side chain carboxyl C or O atoms
10. Side chain thiol/sulfide C or S atoms
11. Main chain amide C or O atoms
12. Side chain LYS/ARG amine C or N atoms
13. Side chain TRP amine atoms C or N atoms
14. Side chain hydroxyl C or O atoms
15. Side chain carbonyl C or O atoms

16. Side chain aliphatic/aromatic C atoms

1
2

Each point in the grid is spaced 0.5Åand there are 32 grid points (M = 32) along each dimension during training because of memory constraints and 64 grid points (M = 64) during prediction. Each point in a grid represents the atomic distribution of the specific atom type corresponding to the channel. These atomic distributions are generated from the coordinates of the atoms in the protein structure using a shifted Gaussian kernel[11, Eq. 1]:

$$A(d,r) = \begin{cases} e^{-2d^2/r^2} & ;0 \leqslant d < r \\ \dfrac{4d^2}{e^2r^2} - \dfrac{12d}{e^2r} + \dfrac{9}{e^2} & ;r \leqslant d < 1.5r \\ 0 & ;d \geqslant 1.5r \end{cases}$$

d: distance of the grid point from the atom's centre
r: atomic radius

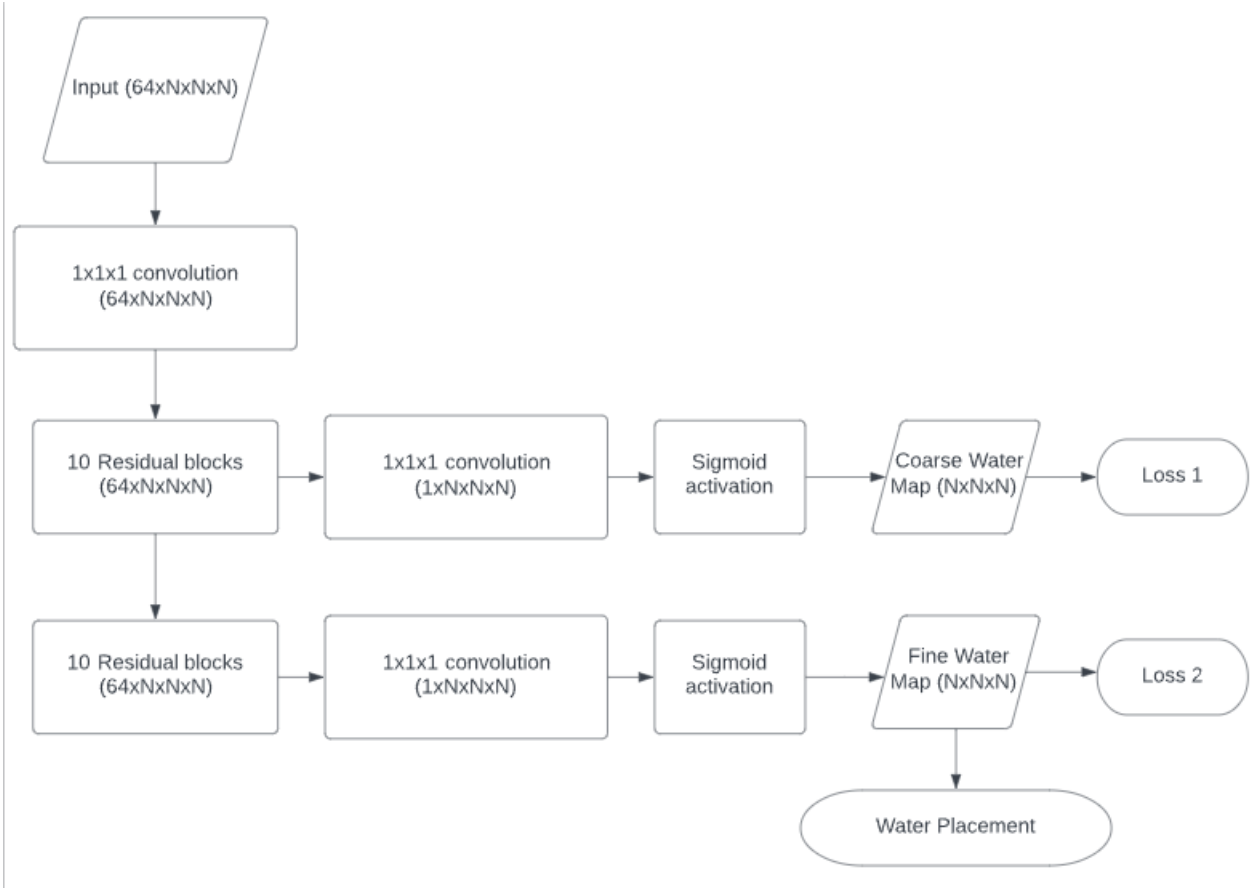


Figure 4.1: GalaxyWater CNN architecture

[1]amide, amine, phenol, carboxyl, thiol, sulfide, hydroxyl, carbonyl and aliphatic/aromatic Carbons are functional groups commonly found in the protein structure and can influence water affinity of the structure.
[2]HIS, LYS, ARG, TRP are common codes for specific amino acids.

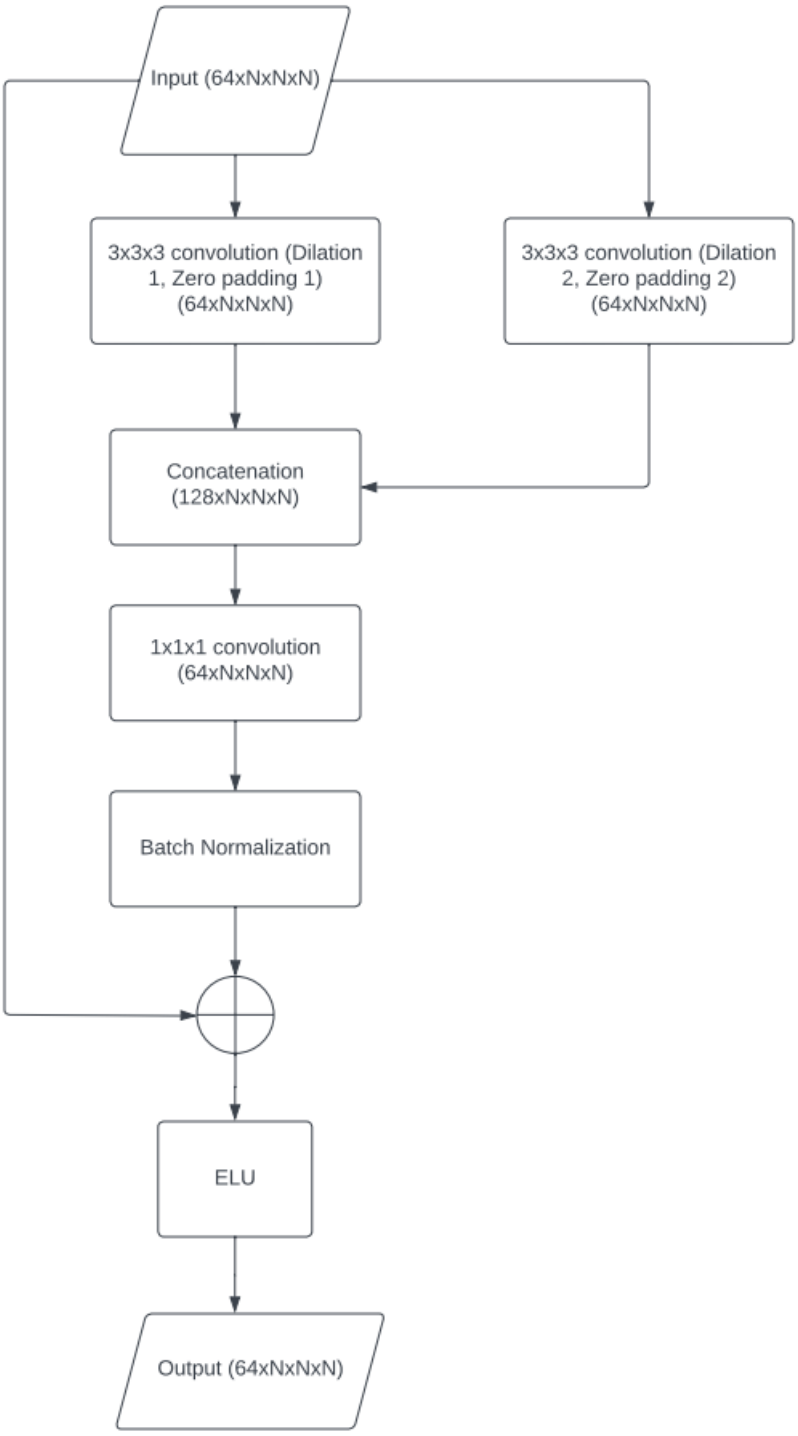


Figure 4.2: Architecture of each Residual Block

The 16xMxMxM input is first fed to a 3D convolution layer which increases the number of channels to 64 using 1x1x1 convolution. The output of the first layer is then passed through 10 residual blocks after which the network branches into two, the output of the first branch is a "coarse" water map using a cross entropy loss function Loss 1. It is called coarse as it uses a larger water radius, r_1 = 2.28Å(which is 1.5 times the van der Waals[3] radius of water molecules). The other branch uses additional 10 residual blocks to generate a "fine" water map using another cross entropy loss function Loss 2, and a smaller water radius r_2 = 1.52Å, which is the same as the van der Waals radius of water. These coarse and fine water maps are captured to show the overall distribution, and the regions of high probability of occurrence of water molecules respectively.

Structure of the residual blocks:

Each residual block is a 3D convolutional residual network using dilated convolutions to capture long range structural features. There are 3 layers in each residual block with 3x3x3 convolutions. The first two layers use different dilations and paddings of (1, 1) and (2, 2) respectively on the input. The third layer concatenates the outputs of these layers and runs another 3D convolution with 0 dilation and padding followed by a batch normalization so that the output is the same size as the input size of 64xMxMxM 4.2.
The final output of the network is converted into a water/probability score using the shifted Gaussian kernel described above. The probability scores represent the distribution of water molecules.

Placing Water Molecules

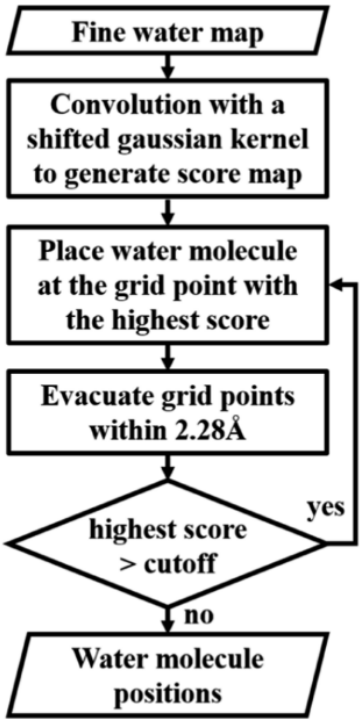


Figure 4.3: Placing water molecules using the fine water map [11, Fig. 3]

[3]van der Waals radius: A type of measurement of atomic radius

Water molecules are placed only on those grid points with probability scores higher than a cutoff. Starting from the point with the highest score and keeping the surrounding grid points vacant within 2.28Å, water molecules are successively placed on grid points in descending order of their probability scores as shown in figure 4.3.

4.2 Adding Bayesian Layers

In the last residual block before the fine water map, the 3D convolution layers were replaced with Bayesian 3D convolution layers with a Gaussian prior with mean 0.1 and variance 0.1, keeping the remaining architecture the same. No prior elicitation[4] was performed because of limited data, as that would require additional validation data. Addition of Bayesian layers resulted in a stochastic model which can generate different outputs for the same input.

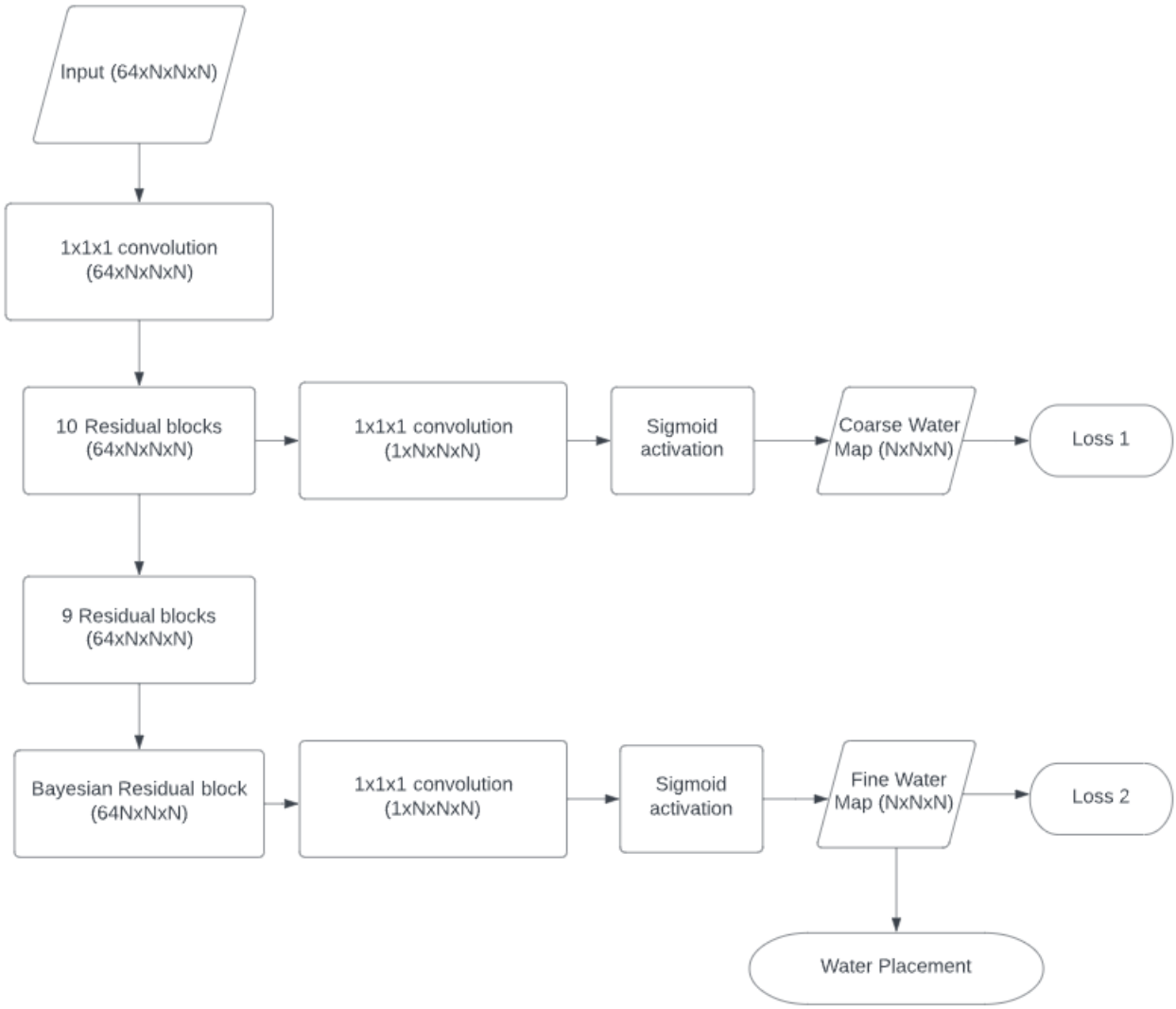


Figure 4.4: Architecture of the modified network after replacing the last residual block with a Bayesian residual block

[4]process of finding the most suitable prior distribution

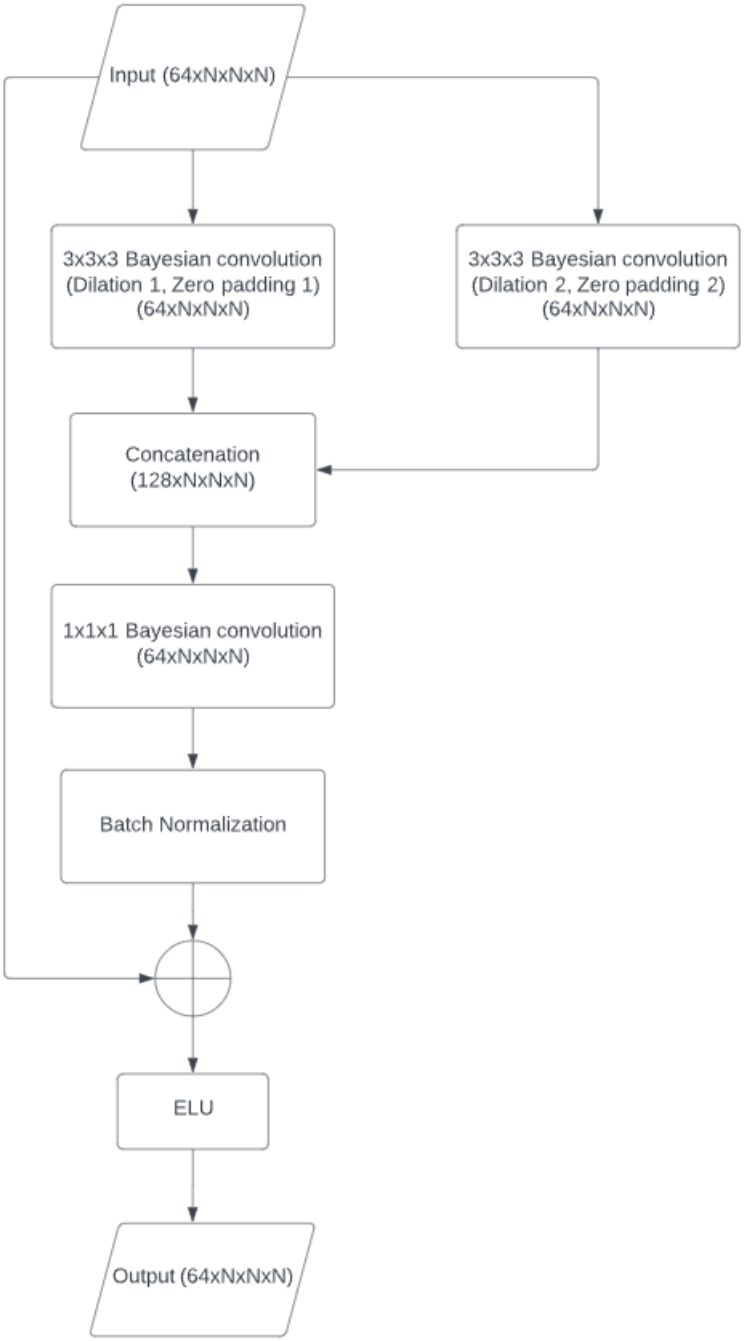


Figure 4.5: Architecture of the Bayesian Residual Block

4.3 Motivation for adding Bayesian layers

Bayesian neural network differs from an ordinary neural network in that it learns the distribution of the weights rather than point estimates. As a result it is harder to overfit and requires more epochs to learn the parameter distributions [7]. The output is also a distribution and the network samples from the output distribution every time it makes a new prediction. So each time it generates a different output for the same input which is appropriate for this problem as the bounded water positions are not supposed to be fixed but can deviate by small amounts.

As we have a small set of training data (160 protein structures), it is easy for a fairly deep neural network like GalaxyWater-CNN to overfit to the training data. By adding Bayesian layers we try to build a more robust model that requires much more epochs to converge and harder to overfit. This could be verified by comparing the training and test losses for both the models.

Another advantage of using a Bayesian network is that we can estimate the uncertainty of the new data as another measure of the model's performance.

$$p(y^*|x^*, \theta) = \int_W p(y^*|w, x^*, \theta).p(w)dw$$

x^*: new input
y^*: new output
W: set of all possible weights of Bayesian layers
w: random weights of the Bayesian layers
p(w): distribution of weights of Bayesian layers learned by the Bayesian-CNN
θ: fixed weights of the other layers of the network

Finding this integral is non trivial, so we approximate it by sampling T weight vectors from the learned distribution of weight vectors and find the density of the output corresponding to each weight vector. We can then use the mean of these densities as an approximation of output distribution.

$$p(y^*|x^*, \theta) \approx \frac{1}{T} \sum_{t=1}^{T} p(y|w^{(t)}, x^*, \theta).p(w^{(t)})$$

T: Number of samples drawn from the learned distribution of weights of Bayesian layers
$w^{(t)}$: t^{th} sample of weights of the Bayesian layers

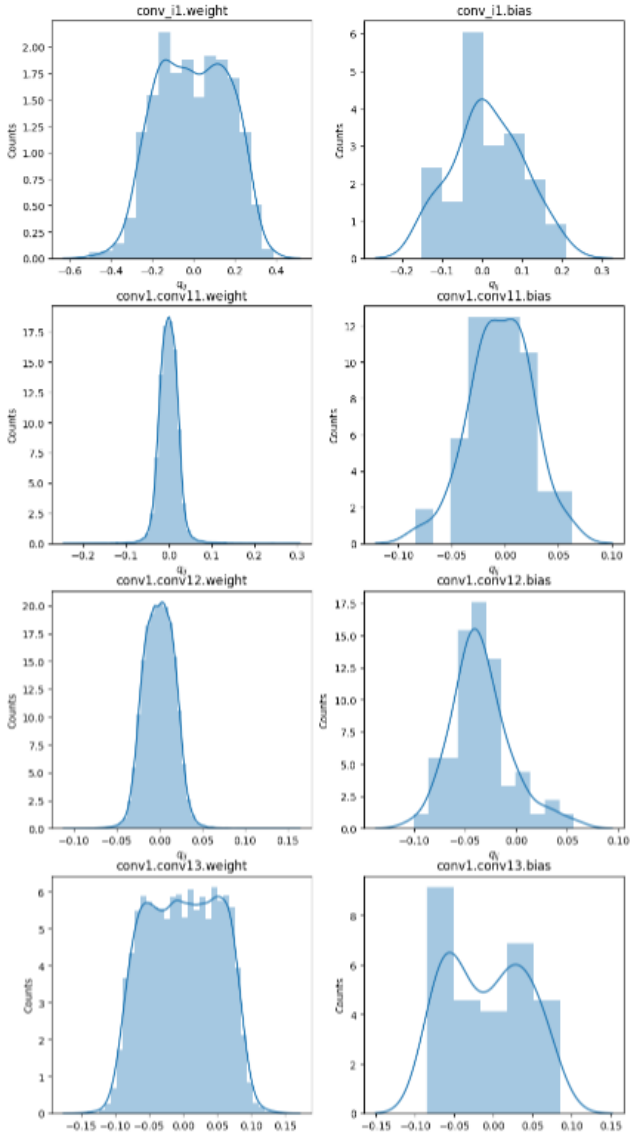


Figure 4.6: some parameter distributions learned by the network

5 Results

We start by comparing the performances of the GalaxyWater-CNN and the Bayesian-CNN using the training and test losses, the probability scores at corresponding grid points, and the distances of their predictions from the actual water positions. We then check whether the Bayesian-CNN is able to model the uncertainty in the positions of the water molecules. It is important to note here that to obtain all the results (except for training and test loss comparison) in this chapter a specific protein 3RV1[15] was chosen at random from the test data. However we obtain similar results for some other randomly chosen proteins in the test data and thus assume that the results obtained for 3RV1 are a good representation for other proteins in the test data as well. Also for time constraints, we generate only 10 different outputs using the Bayesian CNN and use their sample mean and variance as approximations of the true mean and variance of the stochastic output of the Bayesian CNN.

5.1 Comparing the GalaxyWater-CNN and Bayesian-CNN in terms of outputs

Training and Test losses

We can plot the cross entropy losses for training and test data in each training epoch of both the networks to find out if the Bayesian CNN performs better in terms of test loss.

Here if we compare the training and test losses for the GalaxyWater-CNN and our Bayesian modification, we observe that the training losses are slightly higher for the Bayesian model, but the test losses are lower than the training losses (figure 5.2). The reason for this discrepancy is that the Bayesian-CNN produces training loss for each batch in an epoch, however the test loss is calculated at the end of the epoch. We have plotted the mean of training losses of all the batches against the test loss for each epoch. As the weights of the network are optimized progressively with each batch we get higher training losses for the initial batches in an epoch and lower training losses for the later batches affecting their mean. However as the test loss is calculated at the end of the epoch with better optimized weights, it is lower than the mean of the training losses for the epoch. We do not observe any sign of overfitting in the GalaxyWater-CNN in 5.1, however the test losses are comparable to training losses and not lower. We suspect that some processing has been done on the test losses of the GalaxyWater-CNN to get a better comparison with the training losses. One such strategy

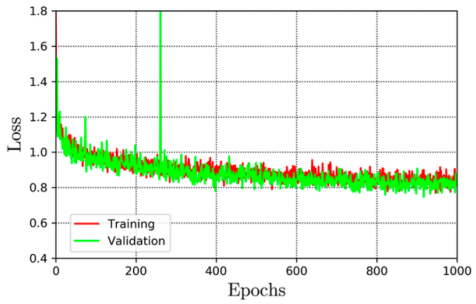


Figure 5.1: training and test losses of GalaxyWater-CNN [11, Fig. 4] (validation loss refers to test loss)

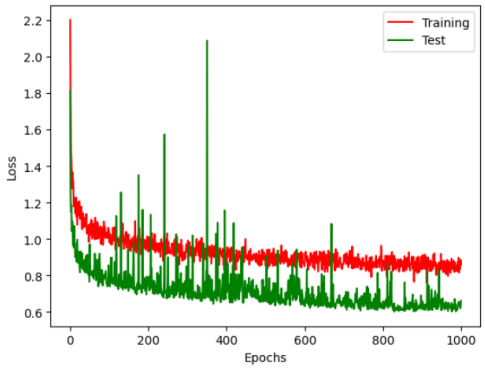


Figure 5.2: training and test losses of the Bayesian CNN

could be to measure the test loss after each batch of the epoch and take their mean as a measure of test loss for the epoch, similar to training losses. We also observe some spikes in the test losses in some epochs for the Bayesian model. This could be due to the model generating water molecules stochastically and thus producing a different output in each epoch. In some of the epochs this stochastic output could be such that test loss for the epoch becomes abruptly high.

Comparing the models in terms of probability scores

As we know the outputs of both the neural networks are 3D arrays called probability scores where each index of the array represents a grid point in the 3D structure of the protein and the value at the index represents the probability of occurrence of a water molecule at that grid point. We compare the deterministic output of the GalaxyWater-CNN with the mean output of the Bayesian-CNN in two ways as described below.

Comparison of the probability scores at corresponding grid points

We can determine the similarity of the outputs of the two models at the corresponding grid points with the help of a scatterplot. We expect small differences in the corresponding outputs as Bayesian-CNN is a stochastic model, however if the outputs differ too much it could indicate incorrect inference by the Bayesian-CNN which could be due to underfitting. The probability scores of the GalaxyWater-CNN model are plotted against the mean of probability scores of the Bayesian-CNN model for each grid point and a linear regression model is fitted, as seen in figure 5.3. From visual inspection we can say that the fitted line resembles the identity function, which is confirmed by the R-squared value which is found to be 0.95. We can further check for the similarity of the outputs of the two models at the grid points nearest in terms of Euclidean distance to the actual water positions as seen in figure 5.4. Again we observe a linear trend similar to the identity function with an R-squared value of 0.91.

Comparison of histograms of the probability scores

We can check if the original model is overconfident in its predictions, i.e. if it predicts probability scores close to 1 for most of the grid points, and whether the Bayesian model improves on this, which is one of the main reasons of using a Bayesian neural network 4.3. For this we plot the histograms of the probability scores of the two neural networks as seen in figure 5.5. From the plot we do not observe any major difference in the distributions of the outputs. We also observe that both the outputs have probability scores close to 0 for most of the grid points. We further plot the probability score histograms for only those grid points that are

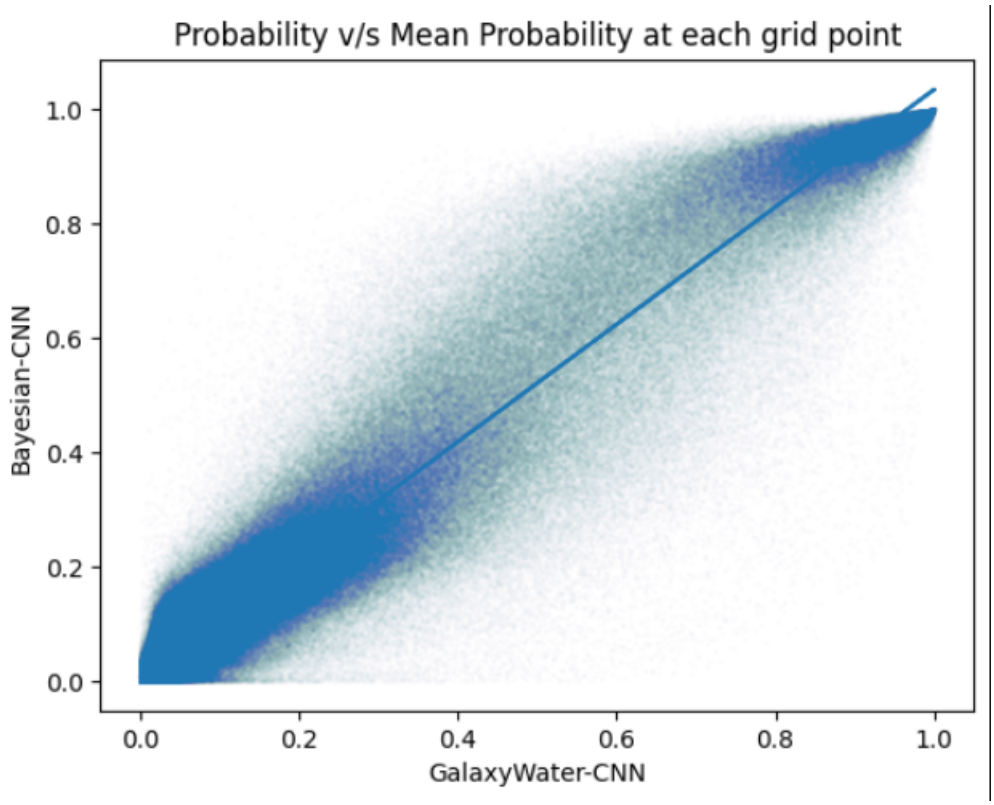


Figure 5.3: Probability score of the GalaxyWater-CNN against the mean probability score of the Bayesian-CNN at each grid point for the protein 3RV1

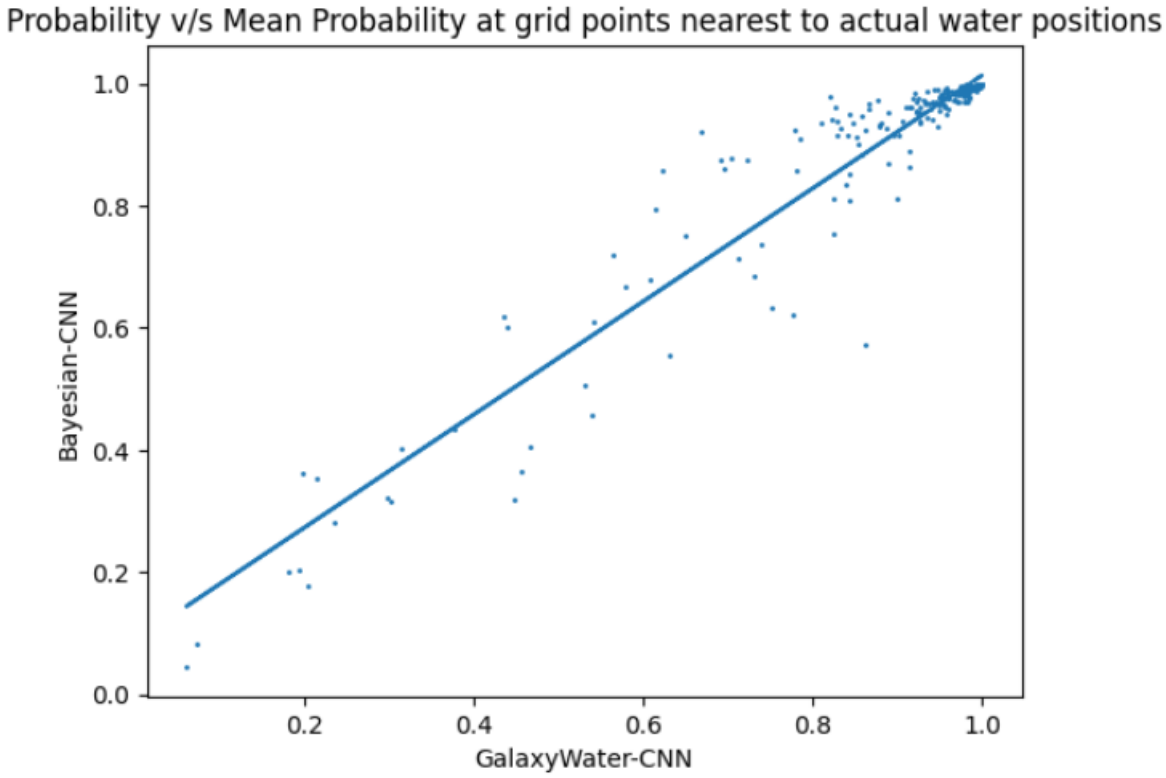


Figure 5.4: Probability score of the GalaxyWater-CNN against the mean probability score of the Bayesian-CNN at grid points nearest to actual water positions for the protein 3RV1

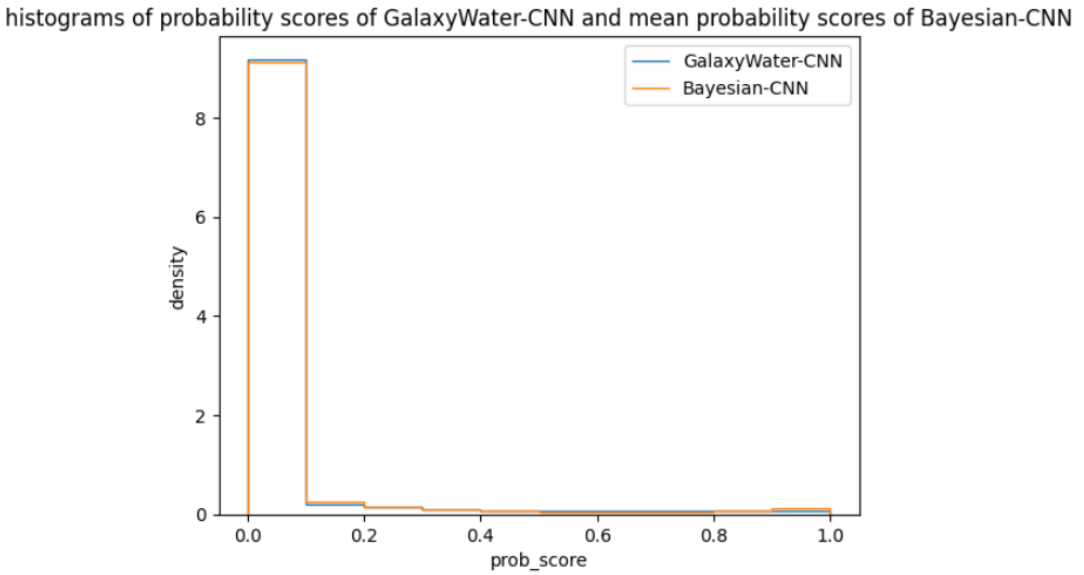


Figure 5.5: Histograms of the probability score of the GalaxyWater-CNN and the mean probability score of the Bayesian-CNN for the protein 3RV1

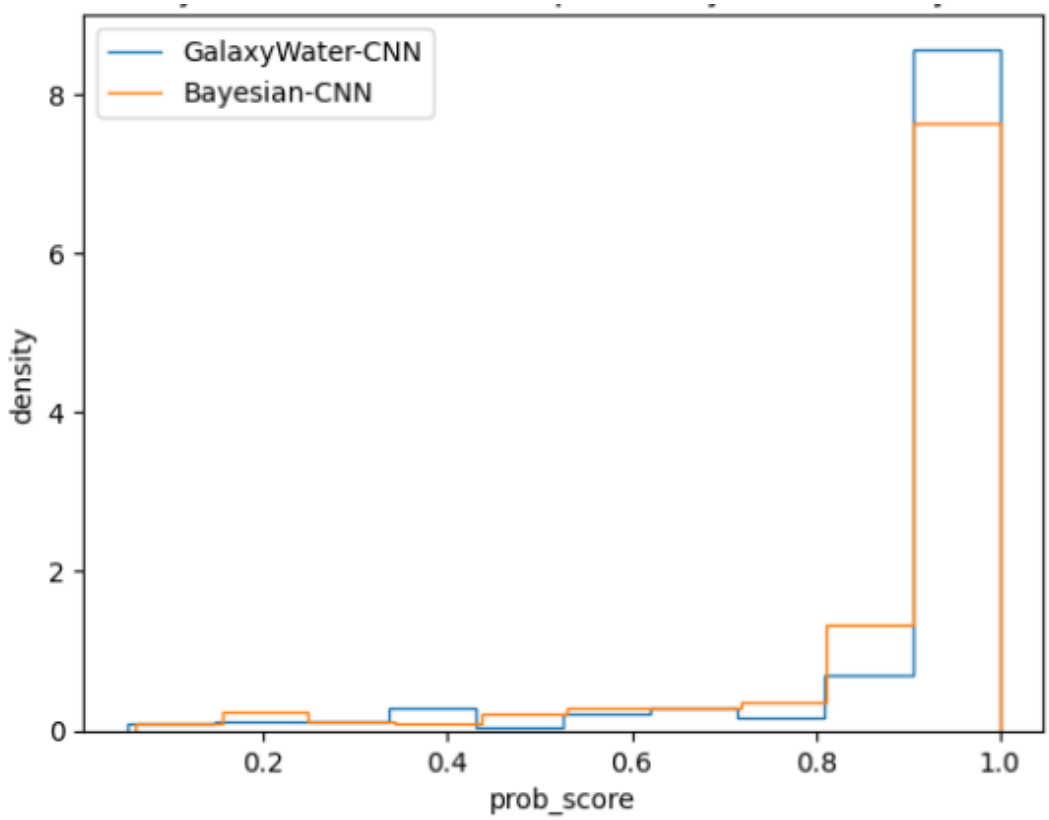


Figure 5.6: Histograms of the probability score of the GalaxyWater-CNN and the mean probability score of the Bayesian-CNN for the protein 3RV1 corresponding to actual water positions

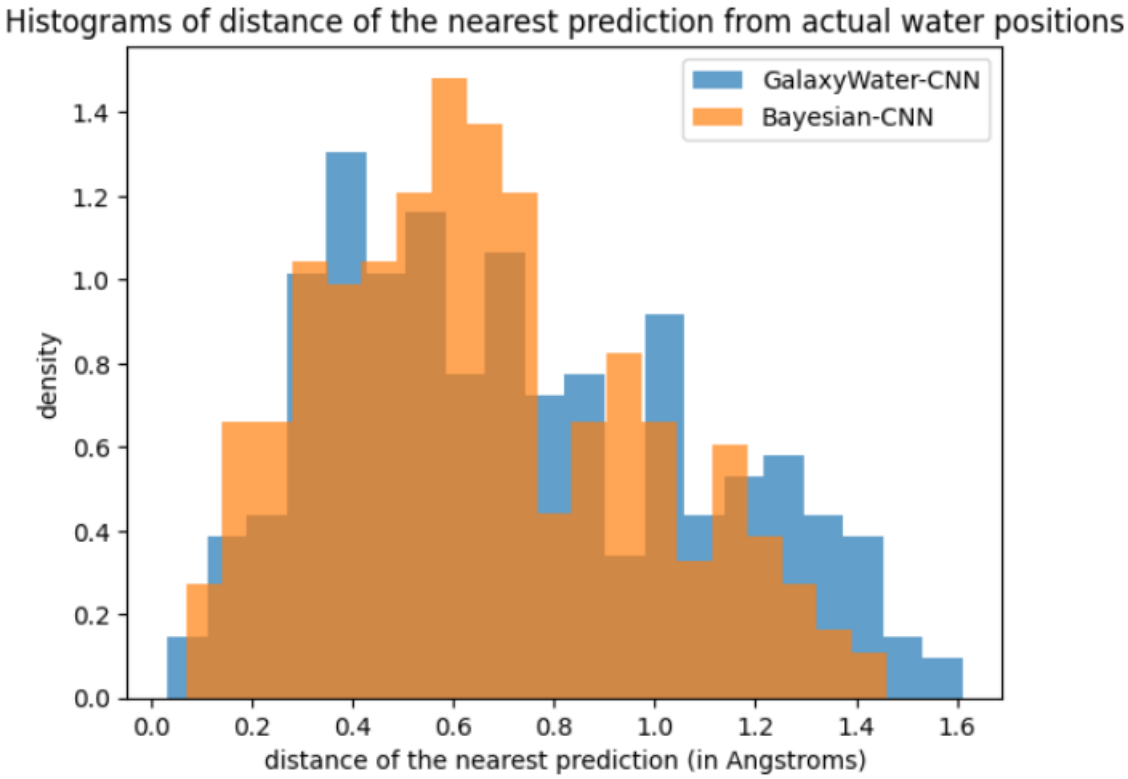


Figure 5.7: Histogram of distances of the nearest predictions (mean of the nearest predictions in case of Bayesian CNN)

nearest to actual water positions in terms of Euclidean distance to see if the output distributions of the models differs at these grid points5.6. We observe that both the models predict probability scores close to 1 at most of the grid points which are close to actual positions of the water molecules. However Bayesian-CNN has slightly lower density at values close to 1 indicating that the model could be less overconfident in its predictions, although the distribution of mean probability scores obtained for Bayesian-CNN could vary, so we cannot conclude it in general.

Comparing the models in terms of Euclidean distances of the predictions from actual water positions

Another way of comparing the two models is by measuring the Euclidean distances of the nearest predictions from the actual water positions. For evaluation purpose, here we are assuming that the nearest predicted position is the best representation of the actual water molecule. To this end we use the outputs of both the models to predict the water positions as described in section 4.1, for the protein 3RV1 which has 262 actual water molecules. As GalaxyWater-CNN is a deterministic model, it is used to make prediction once, and we find the distances of the nearest predicted positions from each actual water molecule. Bayesian-CNN on the other hand generates different predictions each time, so we make 10 different predictions, and for each water molecule take the mean of the nearest predicted positions as an approximation of the nearest prediction. Mean position is calculated as seen in equation 5.1.

$$\bar{\mathbf{p}} = \frac{\sum_{i=1}^{n} \mathbf{p_i}}{n} \tag{5.1}$$

n: number of outputs generated by Bayesian-CNN (10 in this case)
$\mathbf{p_i}$: position vector of the nearest prediction in the i^{th} output
$\bar{\mathbf{p}}$: mean of the position vectors

Then we calculate the distances of the mean of the predicted positions from each actual water position and plot the histograms of the distances (figure 5.7).

From the histograms we observe similarity in predictions of the Bayesian-CNN and GalaxyWater-CNN in terms of distances of the nearest predicted positions with the support of the distributions being roughly the interval [0 Å, 1.6 Å]. This is also confirmed by the mean distance of the nearest prediction from actual water position being 0.654 Åfor Bayesian-CNN and 0.724 Åfor GalaxyWater-CNN. Although it is important to note here that the distribution we get for the Bayesian-CNN is somewhat arbitrary and depends on the sample of outputs generated by the model. Repeated sampling can be done to get a better estimate. Also the purpose of using Bayesian-CNN is not to make accurate predictions of the water positions like GalaxyWater-CNN but rather to model the variability in the positions of the water molecules in the protein structure while still being fairly accurate.

5.2 Evaluating the Bayesian-CNN in terms of capability to model the uncertainty in water positions

As discussed in section 4.3 the main objective of using the Bayesian-CNN is to model the uncertainty in the positions of water molecules as these are not fixed in a protein structure. A good measure of the variability in the position of an atom in a protein structure is its B-factor as described in section 1.2, which is determined experimentally and documented in the PDB file[30]. We can compare the B-factor of a water molecule to the output at the nearest grid point to check whether the model is able to capture the variability in the position of that molecule. We use 3 different metrics to accomplish this as described in the following sections. Here again we assume that the output at the nearest grid point is suitable to estimate the uncertainty in the position of a water molecule.

B factors vs Mean Probabilities

With higher B factor there is greater variation in the location of the water molecule, which means that the probability of its presence at the nearest(in terms of Euclidean distance) grid point should be lower. For each actual water position we determine the mean probability score at the closest grid point over 10 different outputs for the Bayesian-CNN. Also for comparison with the GalaxyWater-CNN, we determine the probability score generated by it at those grid points. We then plot these probabilities against the B factors of the water molecules(figures 5.8 and 5.9). We also fit linear regression models to measure the proportion of variance in the probability score that can be explained by B-factor using R-squared value. With increasing B factor the probability scores decrease for many of the grid points as expected, but for a large number of grid points these remain close to 1 even at high values of B factor. From visual inspection we can see that the linear regression models do not fit very well to the data which is confirmed by the R^2 values that are quite low, 0.14 and 0.15 for the GalaxyWater-CNN and the Bayesian-CNN respectively.

We suspect that a possible reason for this anomaly in the models' outputs could be because of relative B-factors. Relative B-factor of a water molecule is its B-factor relative to that of its environment, and can be calculated as:

$$\beta_{rel}(w) = \frac{\beta(w)}{\beta_r(env)}$$

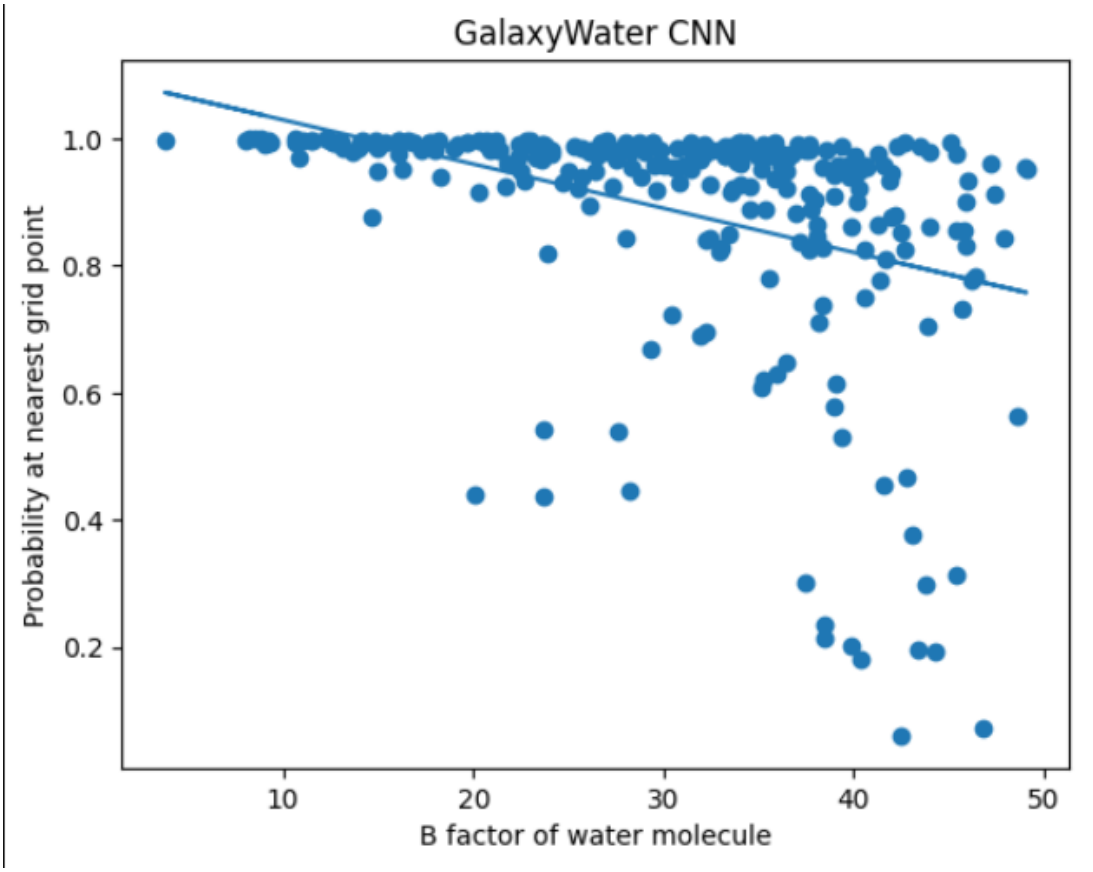


Figure 5.8: B factors vs the probability scores of GalaxyWater-CNN for each water molecule

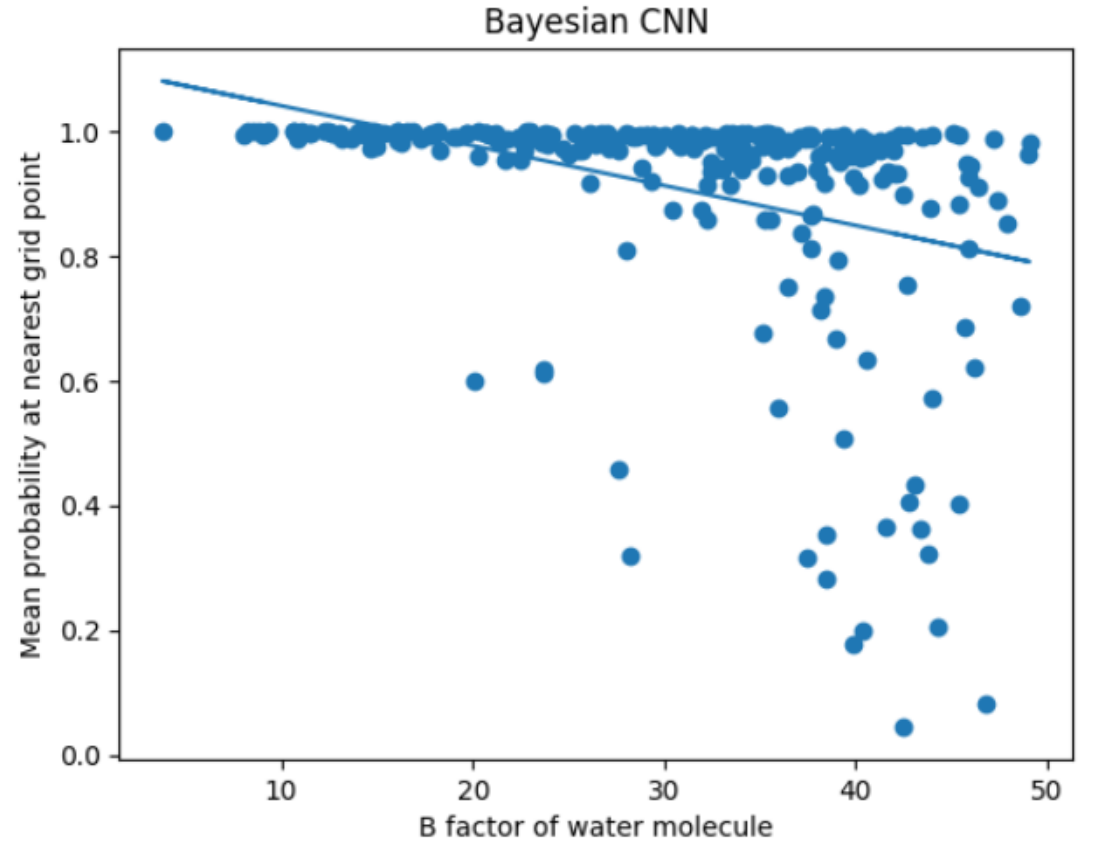


Figure 5.9: B factors vs the mean probability scores in 10 different predictions of Bayesian-CNN for each water molecule

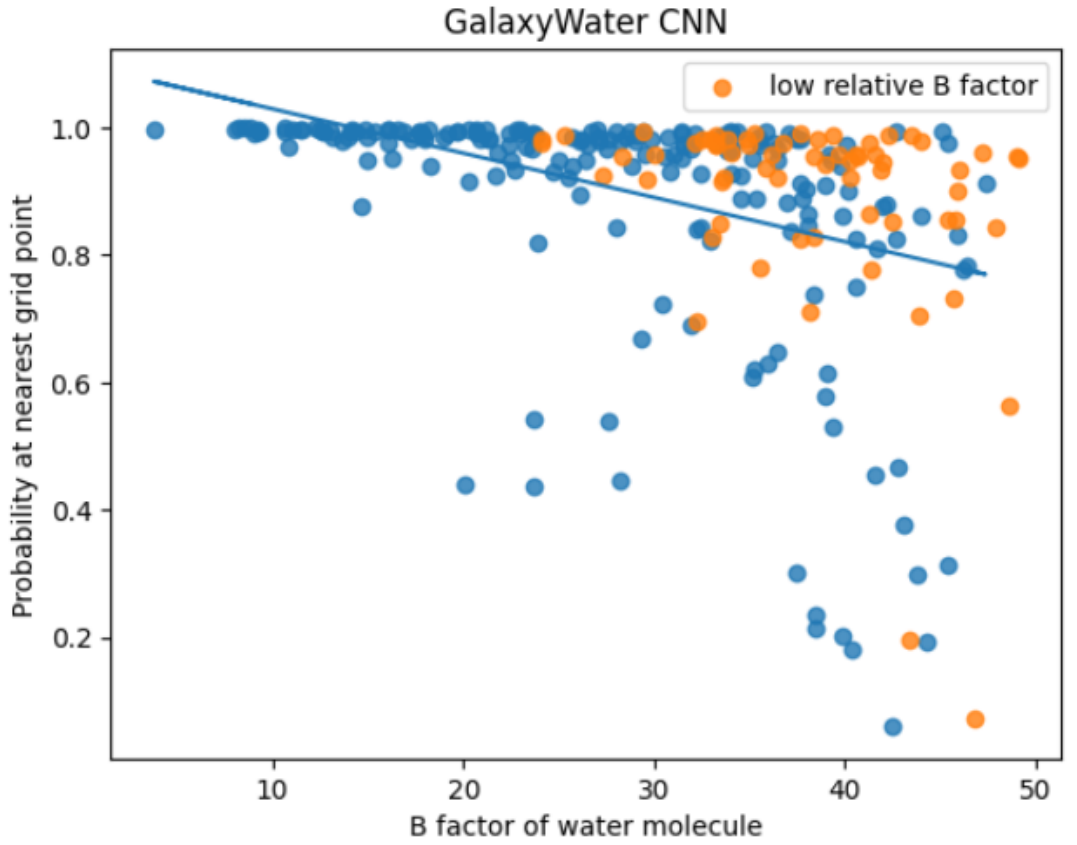


Figure 5.10: B factors vs the probability scores of GalaxyWater-CNN for each water molecule with water molecules having relative B-factors < 1 indicated

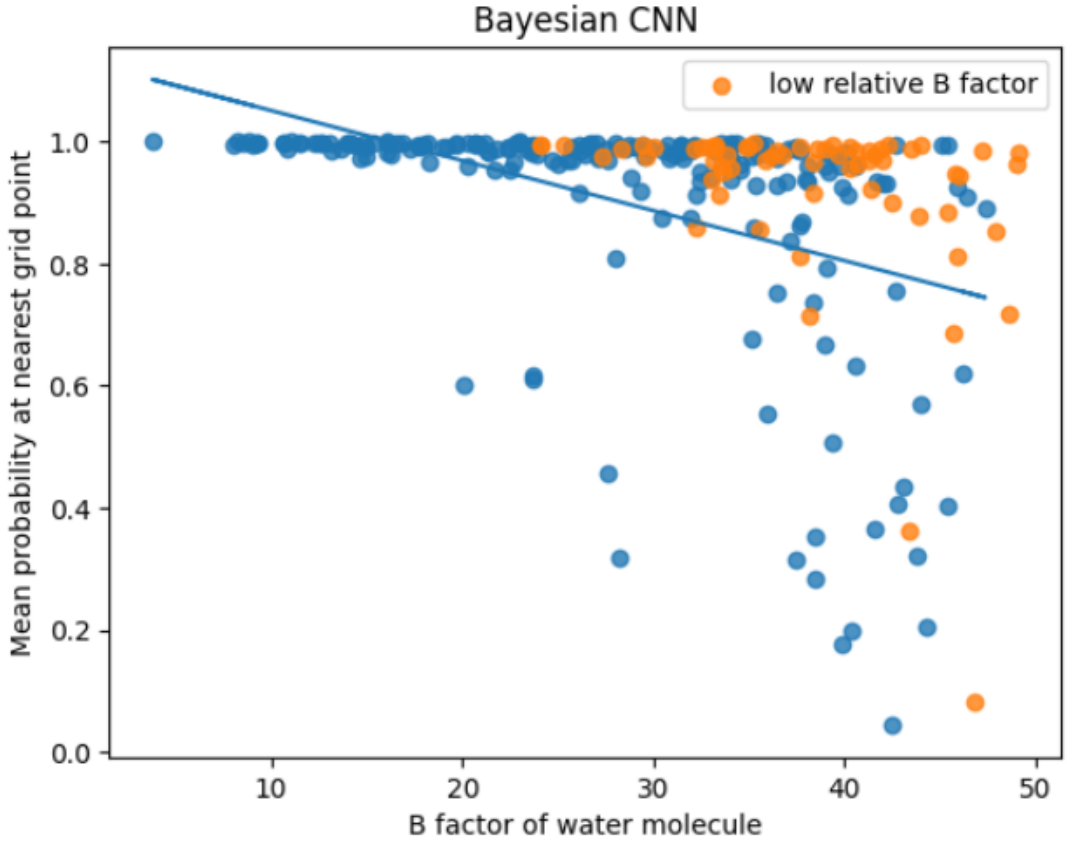


Figure 5.11: B factors vs the mean probability scores in 10 different predictions of Bayesian-CNN for each water molecule with water molecules having relative B-factors < 1 indicated

$$\beta_r(env) = \frac{\sum_{i=1}^{n} \beta(p_i)}{n}$$

$\beta(w)$: B-factor of a water molecule
$\beta_{rel}(w)$: relative B-factor of the water molecule
env: the environment of the water molecule which is a sphere centered at the molecule
r: radius of the sphere/environment
n: the number of protein molecules in the environment of the water molecule
$\beta(p_i)$: B-factor of i^{th} protein molecule in the environment of the water molecule
$\beta_r(env)$: B-factor of the environment of the water molecule with radius r i.e. the average B-factor of all protein molecules in the environment

Here we treating a particular region of the protein structure, a sphere of radius r centered at a water molecule, as having a B-factor although B-factors are measured for atoms and not regions. The basis for this reasoning is that usually the B-factor of an atom does not differ much from that of its surrounding atoms [2]. This is because there are some regions of protein structures like side chains 1.1 that are highly mobile (atoms have high B-factors) and some that are more stationary. So there may be some variations in the B-factors of individual atoms, but usually these do not deviate too much from the average B-factor of the atoms in the region, which we can call the B-factor of the region.

Thus we calculated the B-factors of all the water molecules relative to their respective environments with radius 8 Å. As the data is unbalanced with most of the waters having low B-factors as described in section 3.1, we exclude those data points that have relative B-factors lower than 1. These threshold values for the environment radius and relative B-factors were suitable as discussed with our domain expert. We fitted linear regression models again after excluding the anomalous data points with low relative B-factors as shown in figures 5.10 and 5.11, with the anomalous data points labelled in orange, for the GalaxyWater-CNN and Bayesian-CNN respectively. We can see that of the points having high B-factors and still assigned high probability scores by the models, most have low relative B-factors. We also observe improved R^2 values of 0.196 and 0.205 for the GalaxyWater-CNN and Bayesian-CNN respectively.

B factors vs Probability Variance

With lower B-factor there is less variation in the location of the water molecule indicating a more stationary water molecule. So if the Bayesian-CNN was able to model this variability in the position we should expect that the variance in the probability scores at the nearest grid point should be low along with a higher mean. Similarly for higher B-factor, the nearest grid point should have higher variance in probability scores.

For each actual water position we determine the nearest(in terms of Euclidean distance) grid point and the variance of the probability score for that grid point over 10 different predictions of the Bayesian-CNN. We then plot B-factors of the water molecules against the variance of the probability score at the nearest grid point as seen in figure 5.12, along with a fitted linear regression model. We observe that even at higher B-factors, the variance in the probability scores is very low for most of the water molecules indicating that there does not seem to be a strong correlation between B-factors and variance of probability score at the nearest grid point, which is confirmed by a low R^2 value of 0.107.

However this anomaly can again be explained using relative B-factors as in the previous section. We fitted another linear regression model excluding the water molecules having relative B-factors less than 1 as shown in figure 5.13, with the excluded data points labelled in orange color. We observe that of the water molecules with high B-factors and still assigned

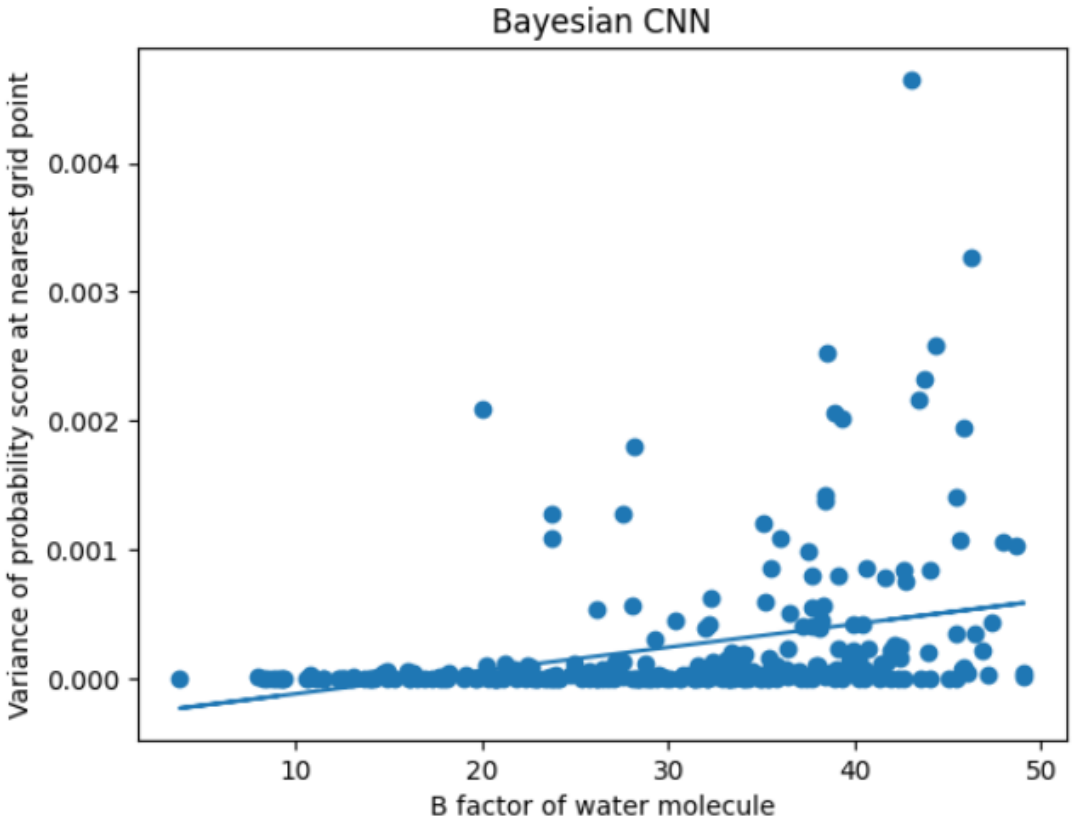


Figure 5.12: B factors vs the variance of probability scores at nearest grid points in 10 different predictions of Bayesian CNN

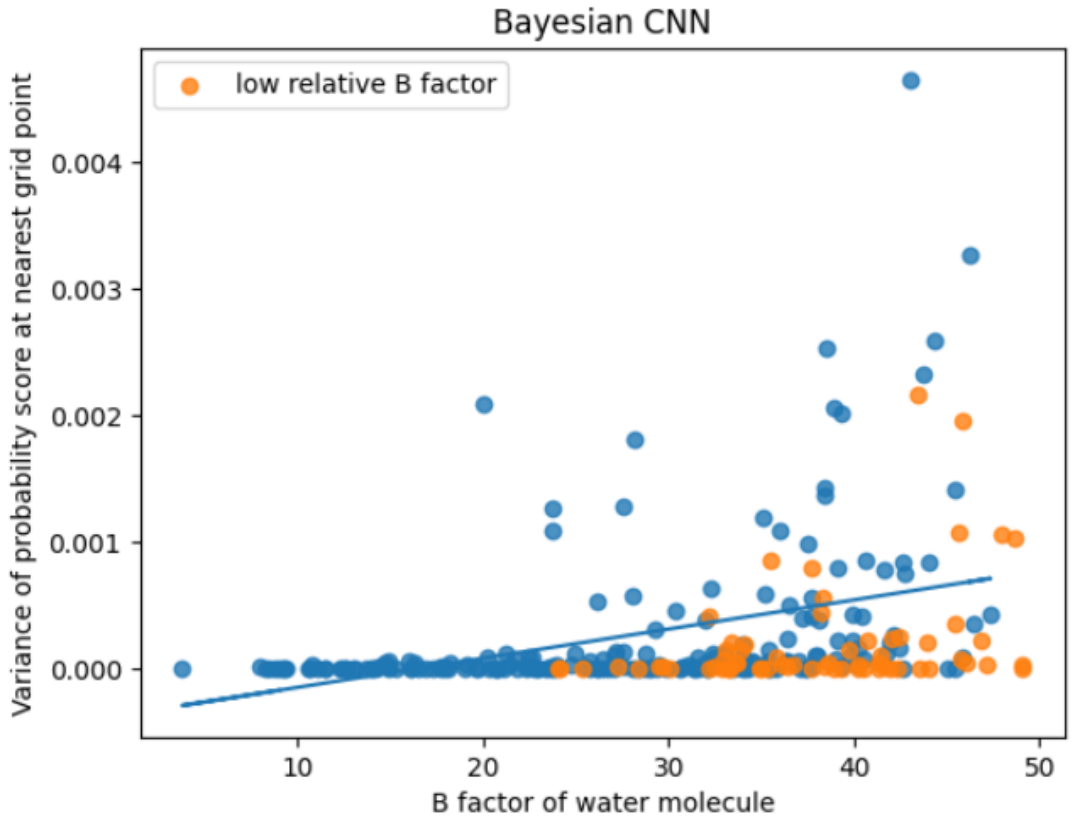


Figure 5.13: B factors vs the variance of probability scores at nearest grid points in 10 different predictions of Bayesian CNN with water molecules having low relative B-factors indicated

very low variances by the model, most have low relative B-factors. So excluding these points results in a better linear model with increased R^2 value of 0.149.

B factors vs Mean Squared distance

As discussed in section 4.3 the positions of individual molecules and atoms are not fixed in a protein structure and can vary. Thus we can view the position of a water molecule as a 3 dimensional random variable ($\mathbf{P} = [x \quad y \quad z]^T$), where x, y and z are the coordinates of the water molecule. Then B-factor can be mathematically interpreted as a linear function of the variance of this random variable[12]. Also if we consider the actual(experimentally determined) position of a water molecule as the mean of the random variable and a predicted water position as a random observation from the distribution, then the Euclidean distance between the predicted and actual water positions can be interpreted as a measure of deviation of the observation from the mean. If we take a sample of predicted water positions of size n(10 in our case) we can calculate the sample variance as :

$$Var(\mathbf{P}) = \frac{\sum_{i=1}^{n} d_i^2}{n}$$

n: number of predictions
d_i: distance of the i^{th} predicted water position from the actual water position

Further we can use the sample variance as an approximation of the population variance or the B-factor and compare the two. We expect a higher B-factor should result in a larger mean squared distance of the predicted positions, which means a positive correlation between the two. Thus we can verify whether the Bayesian neural network is able to model this correlation. For each actual water position we find the nearest(in terms of Euclidean distance) predicted water positions in each of the 10 different predictions. Here we are assuming that the nearest prediction represents a random observation from the distribution. We then take the mean squared distances of the nearest predictions for each actual water position, and plot it against the B-factor of that water position as shown in figure 5.14. From the figure we can observe a positive correlation between B-factors of the water molecules and the mean squared distances of the predicted positions, which indicates that the model is able to model the variability in the water positions to some extent. To measure the extent we fitted a linear regression model and the R^2 value was found to be 0.146 indicating that only 14.6% of the variation in mean squared distance of the predictions can be explained using the B-factors.

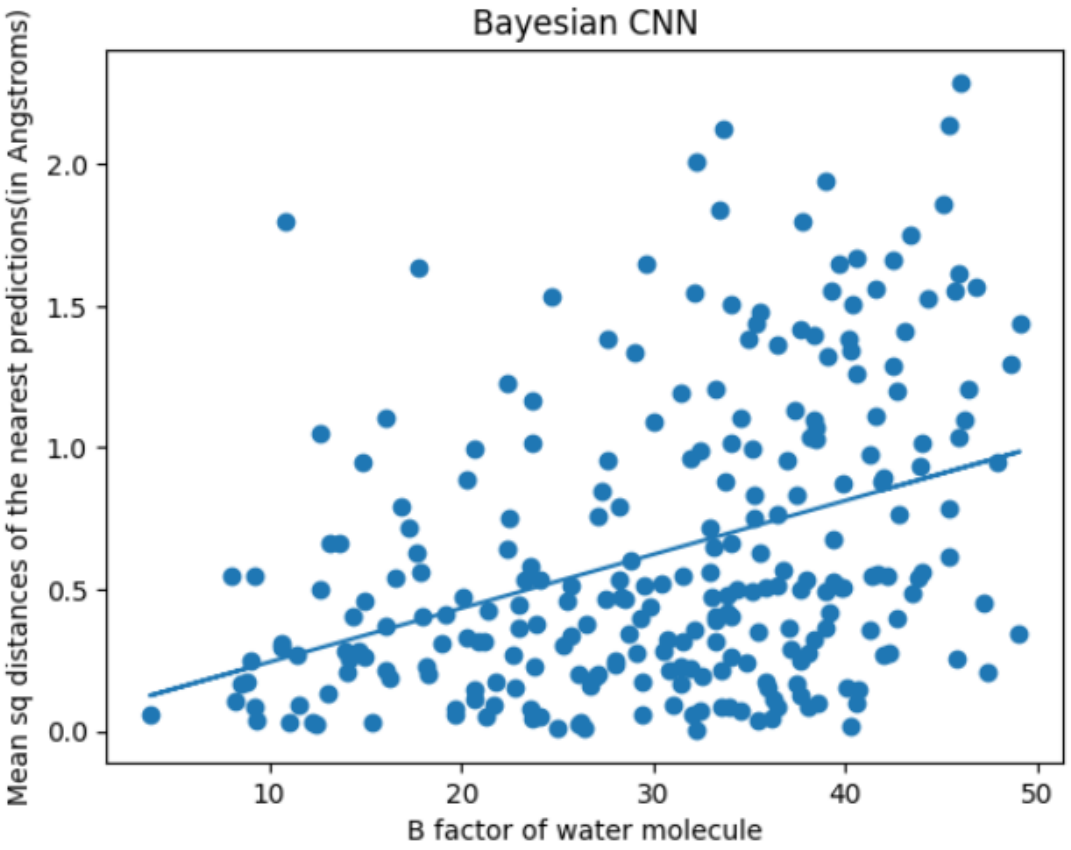


Figure 5.14: B factor vs the mean squared distance of the nearest predictions of Bayesian CNN for each water molecule

6 Discussion

Here we discuss the results obtained and some limitations of the methods and the data used, along with some suggested improvements for future work.

6.1 Results

Training and Test losses

We observed that the training losses of the Bayesian-CNN are slightly higher than that of GalaxyWater-CNN (figures 5.1 and 5.2). This could be attributed to the fact that Bayesian neural networks are slower to converge and need more epochs to train than standard neural networks because of larger number of parameters needed to model the distribution of the weights rather than just point estimates[6].

We also observe that the training losses of both the CNNs seem to be following a slight downward trend and could reduce further if the networks were trained for more epochs.

On the other hand we do not have an effective comparison of the training and test losses for the Bayesian-CNN. Both the CNNs were implemented using PyTorch[13] library of Python[16]. Neural network implementations in PyTorch split the training data into batches and the optimization algorithms optimize the weights of the network progressively using one batch at a time, and output the training loss for each batch separately. When all the batches are processed, which marks the end of an epoch, the test loss is evaluated on the test data. This results in the test loss being lower than the mean of the training losses for each epoch. Consequently our strategy of comparing the mean of training losses of an epoch with the test loss of the epoch is not effective as the training losses are not evaluated using a single set of weights, but with different progressively optimizing sets. A better approach could be to change the implementation of the Bayesian-CNN such that it evaluates and outputs test loss after each batch like the training loss, and compare the mean of training and test losses for each epoch, but for that we would have to train the Bayesian-CNN again. As the test losses for GalaxyWater-CNN are almost equal to the training losses as seen in the figure 5.1, we believe that there must have been some adjustment of either training or test losses done for the GalaxyWater-CNN to get a more effective comparison of the two. As we do not have the actual training and test losses for the GalaxyWater-CNN and there is no mention of the

adjustments done to obtain the specific training and test losses that they get in the paper[11], we have no means of comparing the two figures for test losses.

Comparison of the outputs of the CNNs

In section 5.1 the outputs of the two CNNs are compared in terms of the probability scores at the corresponding grid points and also the distributions of the probability scores using histograms. The main objective of doing this comparison is to verify the credibility of the outputs of the Bayesian-CNN. As we know that the Bayesian-CNN is designed to generate stochastic outputs that are supposed to reflect the variability in the water positions of different water molecules depending on their B-factors. So we expect the output of the Bayesian-CNN at any grid point to differ from that of the GalaxyWater-CNN at the same grid point, however this difference should not be too large for most of the grid points. If the probability scores of the Bayesian-CNN differ too much at a large number of grid points then the placement algorithm4.1 could predict water molecules at arbitrary positions. From figures 5.3 and 5.4 we observe that the mean probability scores of the Bayesian-CNN over 10 different predictions are quite similar to the probability scores of the GalaxyWater-CNN at most of the grid points including the ones that are nearest to the actual water positions in terms of Euclidean distance. This confirms that the Bayesian-CNN adds some variability to the outputs while still generating reasonable outputs with mean close to the output of the GalaxyWater-CNN at the corresponding grid point. We also compared the histograms of the output scores of the two CNNs to verify if the output distribution of Bayesian-CNN is similar to that of the GalaxyWater-CNN. From figures 5.5 and 5.6 that the distributions of the outputs of the two CNNs are also very similar.

Comparison of water positions predicted by the CNNs in terms of their distances from the actual water positions

We also compared the distribution of the Euclidean distance of the nearest prediction from actual water position of the two CNNs as seen in 5.7. The histograms obtained for the Bayesian-CNN and GalaxyWater-CNN are similar which indicates that the water positions predicted by Bayesian-CNN have the same distance on average from the actual water positions as GalaxyWater-CNN.

B-factor vs probability mean and variance

As we argued in the section 5.2, as the B-factor of a water molecule increases its position becomes more variable, and the probability of occurrence at any specific position decreases. We verified this by choosing the nearest(in terms of Euclidean distance) grid point as that specific position and plotting the B-factor against the probability of occurrence of water molecule at the grid point as determined using the outputs of the CNNs. The rationale behind choosing the grid point nearest from the actual water position being that if we take a grid point further away, the probability of occurrence would decrease and become close to 0 irrespective of the B-factor because the distance between adjacent grid points is particularly large (0.5 Å). We observed that the Bayesian-CNN performed marginally better than GalaxyWater-CNN but still poorly with R^2 values being 0.20 and 0.19 respectively. This could be attributed to the fact that the data is imbalanced as only those protein structures were chosen where the water and protein atoms have low B-factors, with the restriction for water molecules being especially strict (B-factor < 50). As a result most of the water molecules have either low B-factor(<30) or low relative B-factor as evident from the figures 5.10 and 5.11, which could cause the models to be biased and predict higher probability scores at the nearest grid points. We see similar potential bias of Bayesian-CNN model from the plot of B-factors against the variance of the probability scores at the nearest grid point5.13. We expect the model to have higher variance

in the output at the nearest grid point if the B-factor of the water molecule is large. However because of the bias we observe that the variances for most of the data points are very low.

B-factor vs Mean Squared distance

In section 6.1 we plotted the B-factor of each water molecule against the mean squared distance of the nearest predictions over 10 different outputs generated by the Bayesian-CNN. We also described rough mathematical interpretations of B-factor as a linear function of the variance of the position of water molecule and mean squared distance of the predictions from the water molecule as the sample variance. Thus the mean squared distance can be viewed as an estimate of the B-factor, and their plot should be roughly linear. However as seen in figure 5.14 the plot is not linear although there seems to be a positive correlation between the B-factor and the mean squared distance, as observed from the fitted linear regression model with R^2 value 0.146. We suspect that the sample size of 10 is too small for the mean squared distance to be a good estimate of the B-factor and that increasing the sample size should make the plot slightly more linear. However we do not expect to see a large improvement as the model is trained on a very imbalanced data where most of the water molecules have either low B-factors or low relative B-factors as discussed in the previous section. Also it is important to note here that the plot can never be completely linear even if we used balanced data as the predicted positions and thus the mean squared distances can possess only discrete values for a given water position because of the grid structure of the output.

6.2 Methods

The main limitation with the method could be the large grid spacing of 0.5 Åwhich limits the number of positions where water molecules can be predicted by the model. Another limitation is with the placing of the water molecules using the probability maps. The placing is done sequentially starting from the highest probability grid point to the lowest, stopping when a probability threshold is reached. However, we could try a more stochastic approach for water placement to obtain a more realistic method. One such approach could be to normalize the probability scores at all the grid points and then randomly sample a set of grid points using the normalized scores as a probability distribution to predict the water positions.

6.3 Data

As discussed earlier because of unbalanced data where most of the water molecules have low B-factors(or relative B-factors), there seems to be a bias introduced in the Bayesian-CNN towards such data points. As a result its ability to model the uncertainty in the positions of water molecules with high B-factors could be reduced. So including new protein structures in the training data with larger variability in the positions of water molecules could potentially improve the ability of the model to produce outputs that better reflect the uncertainty in the positions.

6.4 Scope for future work

Here are some suggestions based on the above discussions that could improve the outputs generated by Bayesian-CNN.

i. Use a smaller grid spacing.

ii. Train on a more balanced dataset with protein structures having both low and high B-factor water molecules.

iii. Take a larger sample of predictions to better approximate the output distribution.

7 Conclusion

The goal of this work was to devise a stochastic model that can predict the positions of water binding sites in a protein structure and identify the regions with low and high variability in the positions of water molecules. For this GalaxyWater-CNN, a deterministic model that can predict the positions of water molecules in a protein structure, was used as a base model. Some of the layers of this CNN were replaced by Bayesian layers to obtain a stochastic model referred to as Bayesian-CNN. Bayesian-CNN was trained using the same training data for the same number of epochs and evaluated on the same test data as the GalaxyWater-CNN. A comparative analysis of the two models was done to verify the quality of stochastic outputs generated by the Bayesian-CNN using GalaxyWater-CNN as a benchmark. Bayesian-CNN was further analysed to check whether the stochastic outputs are completely random or reflect the uncertainty in the positions of actual water molecules in the test data protein structures. Based on the results we can conclude our research questions as follows:

1. Does the Bayesian-CNN perform better on test data in terms of cross entropy loss and predicted water positions?

Training losses of the Bayesian-CNN were found to be slightly higher than that of the GalaxyWater-CNN indicating that the it needed some more epochs of training to get training losses comparable to that of the original CNN. Unfortunately we do not have an effective comparison of the test losses of the two CNNs because of the reasons mentioned in the section 6.1.

However in terms of the probability score outputs and predicted water positions, the results obtained for the two models were very similar. Thus we can conclude that in the same number of epochs of training, the Bayesian-CNNs ability to predict water binding sites is on par with that of GalaxyWater-CNN.

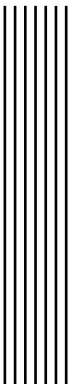
2. Is the Bayesian model able to capture the uncertainty in the positions of water molecules?

Based on the plots obtained for B-factors against mean squared distances of the predictions, and mean and variance of the probability scores at the nearest grid points, we can conclude that the outputs of the Bayesian-CNN are able to reflect the uncertainty in the positions of water molecules only to a small extent. In all the plots we were able to account for

less than 20% of the variability using B-factors, which can be attributed to the imbalance in the training data among other factors as discussed in section 6.1.

8 Ethical considerations

The data is obtained from the Research Collaboratory for Structural Bioinformatics Protein Database [2], a publicly available repository of protein and crystal structures maintained by Worldwide Protein Data Bank [3] which has well defined set of rules for ethical collection of data. The aim of this thesis, that is to find improved methods of determining water binding sites in protein structures, could have beneficial results in pharmaceutical design and does not entail any ethical violations to the best of my knowledge.



Bibliography

[1] *Bayesian Neural Networks*. URL: https://www.cs.toronto.edu/~duvenaud/distill_bayes_net/public/.

[2] H. M. Berman. "The Protein Data Bank". In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 235–242. DOI: 10.1093/nar/28.1.235. URL: https://doi.org/10.1093/nar/28.1.235.

[3] Helen M. Berman, Kim Henrick, and Haruki Nakamura. "Announcing the worldwide Protein Data Bank". In: *Nature Structural Molecular Biology* 10.12 (Dec. 2003), p. 980. DOI: 10.1038/nsb1203-980. URL: https://doi.org/10.1038/nsb1203-980.

[4] Lim Heo, Sangwoo Park, and Chaok Seok. "GalaxyWater-wKGB: Prediction of Water Positions on Protein Structure Using wKGB Statistical Potential". In: *Journal of Chemical Information and Modeling* 61.5 (May 2021), pp. 2283–2293. DOI: 10.1021/acs.jcim.0c01434. URL: https://doi.org/10.1021/acs.jcim.0c01434.

[5] Bingjie Hu and Markus A. Lill. "WATsite: Hydration site prediction program with PyMOL interface". In: *Journal of Computational Chemistry* 35.16 (Apr. 2014), pp. 1255–1260. DOI: 10.1002/jcc.23616. URL: https://doi.org/10.1002/jcc.23616.

[6] Mike Johnson. *Understanding a Bayesian neural Network: a tutorial*. Dec. 2022. URL: https://nnart.org/understanding-a-bayesian-neural-network-a-tutorial/#:~:text=There%20are%20also%20some%20cons,and%20often%20require%20more%20data..

[7] Laurent Valentin Jospin, Wray L. Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. "Hands-On Bayesian Neural Networks—A tutorial for deep learning users". In: *IEEE Computational Intelligence Magazine* 17.2 (May 2022), pp. 29–48. DOI: 10.1109/mci.2022.3155327. URL: https://doi.org/10.1109/mci.2022.3155327.

[8] John M. Jumper, Richard Evans, Alexander Pritzel, Timothy J. Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russell Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew M. Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, R. D. Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michał Zieliński, Johannes Söding, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and

Demis Hassabis. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (July 2021), pp. 583–589. DOI: 10.1038/s41586-021-03819-2. URL: https://doi.org/10.1038/s41586-021-03819-2.

[9] Yan Li, Ying-Duo Gao, M. Katharine Holloway, and Renxiao Wang. "Prediction of the Favorable Hydration Sites in a Protein Binding Pocket and Its Application to Scoring Function Formulation". In: *Journal of Chemical Information and Modeling* 60.9 (May 2020), pp. 4359–4375. DOI: 10.1021/acs.jcim.9b00619. URL: https://doi.org/10.1021/acs.jcim.9b00619.

[10] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022. URL: https://smlbook.org.

[11] Sangwoo Park and Chaok Seok. "GalaxyWater-CNN: Prediction of Water Positions on the Protein Structure by a 3D-Convolutional Neural Network". In: *Journal of Chemical Information and Modeling* 62.13 (2022). PMID: 35749367, pp. 3157–3168. DOI: 10.1021/acs.jcim.2c00306. eprint: https://doi.org/10.1021/acs.jcim.2c00306. URL: https://doi.org/10.1021/acs.jcim.2c00306.

[12] *PDB101: Learn: Guide to Understanding PDB Data: Dealing with Coordinates*. URL: https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/dealing-with-coordinates.

[13] *PyTorch*. URL: https://pytorch.org/.

[14] Arne Schneuing, Yuanqi Du, Charles Harris, Arian Jamasb, Ilia Igashov, Weitao Du, Tom Blundell, Pietro Liò, Carla Gomes, Max Welling, Michael Bronstein, and Bruno Correia. "Structure-based Drug Design with Equivariant Diffusion Models". In: *arXiv (Cornell University)* (Oct. 2022). DOI: 10.48550/arxiv.2210.13695. URL: http://arxiv.org/abs/2210.13695.

[15] David H. Weinberg, Kotaro Nakanishi, Dinshaw J. Patel, and David P. Bartel. "The Inside-Out Mechanism of Dicers from Budding Yeasts". In: *Cell* 146.2 (July 2011), pp. 262–276. DOI: 10.1016/j.cell.2011.06.021. URL: https://doi.org/10.1016/j.cell.2011.06.021.

[16] *Welcome to Python.org*. Aug. 2023. URL: https://www.python.org/.

[17] Wikipedia contributors. *Amino acid — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-May-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Amino_acid&oldid=1155066635.

[18] Wikipedia contributors. *Chemical kinetics — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Chemical_kinetics&oldid=1155381654.

[19] Wikipedia contributors. *Chemical thermodynamics — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Chemical_thermodynamics&oldid=1138469078.

[20] Wikipedia contributors. *Deep learning — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Deep_learning&oldid=1166742951. [Online; accessed 24-July-2023]. 2023.

[21] Wikipedia contributors. *Functional group — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Functional_group&oldid=1161063788.

[22] Wikipedia contributors. *Hydrogen bond — Wikipedia, The Free Encyclopedia*. [Online; accessed 26-June-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Hydrogen_bond&oldid=1161624960.

[23] Wikipedia contributors. *Ligand (biochemistry) — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Ligand_(biochemistry)&oldid=1157487466.

[24] Wikipedia contributors. *Peptide — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-May-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Peptide&oldid=1155430771.

[25] Wikipedia contributors. *Phylogenetics — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Phylogenetics&oldid=1160973909.

[26] Wikipedia contributors. *Protein — Wikipedia, The Free Encyclopedia*. [Online; accessed 24-May-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Protein&oldid=1154039670.

[27] Wikipedia contributors. *Protein–ligand docking — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2023]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Protein%E2%80%93ligand_docking&oldid=1126921361.

[28] Wikipedia contributors. *Statistical potential — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-June-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Statistical_potential&oldid=1142602019.

[29] Wikipedia contributors. *Variational Bayesian methods — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Variational_Bayesian_methods&oldid=1162569495. [Online; accessed 28-July-2023]. 2023.

[30] Christine Zardecki, Shuchismita Dutta, David S. Goodsell, Robert Lowe, Maria Voigt, and Stephen K. Burley. "<scp>PDB</scp> -101: Educational resources supporting molecular explorations through biology and medicine". In: *Protein Science* 31.1 (Oct. 2021), pp. 129–140. DOI: 10.1002/pro.4200. URL: https://doi.org/10.1002/pro.4200.