

# Predicting Protein Function by Machine Learning



**Swansea University**  
**Prifysgol Abertawe**

Department of Computer Science  
Swansea University

Submitted to Swansea University in fulfilment  
of the requirements for the Degree of Bachelor of Science


Vidhi Pala

978174

May 3, 2022

# Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed .....  ..... (candidate)

Date ..... 03/05/2022 .....

# Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed .....  ..... (candidate)

Date ..... 03/05/2022 .....

# Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed .....  ..... (candidate)

Date ..... 03/05/2022 .....

## Abstract

Protein-DNA binding affinity is the energy required to interlock the complex of the protein and DNA. Our project focuses on predicting the protein-DNA complex binding affinity using Neural Networks and One-Shot Learning. In the past few years there has been great discovery of proteins but not in their function. Traditional architectures can predict protein function in larger datasets relatively accurately however, the same cannot be said for smaller ones. This project focuses on using techniques designed for smaller datasets to improve the accuracy of the binding affinity results for each complex. We then compare our results to other architectures used on the same dataset. The evaluation for our neural network is not highly accurate however the actual predictions compared to the experimental results are fairly similar.

## Acknowledgements

I would like to thank my supervisor Dr Gary Tam for supporting me, providing useful tips and feedback throughout the project.

I would also like to thank my parents and my friends who have given me encouragement during the project.

# Contents

<b>List of Figures</b>	<b>VI</b>
<b>List of Tables</b>	<b>VII</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Background</b>	<b>3</b>
2.1 Biological Terminology . . . . .	3
2.2 Biological Database . . . . .	4
2.3 Machine learning . . . . .	5
2.3.1 Deep Learning . . . . .	5
2.4 Libraries . . . . .	7
<b>3 Related Work</b>	<b>7</b>
3.1 Protein-based work . . . . .	8
3.1.1 One-Shot/Few-Shot Learning . . . . .	8
3.1.2 Ensemble Based Learning . . . . .	9
3.1.3 Graph Based Learning . . . . .	9
3.2 Non-protein-based work . . . . .	11
3.2.1 Neural Networks . . . . .	11
3.2.2 One-Shot/Few-Shot Learning . . . . .	11
3.3 Comparing Methods . . . . .	13
<b>4 Methodology</b>	<b>13</b>
4.1 Dataset . . . . .	14
4.2 Fully Connected Neural Network . . . . .	15
4.3 One Shot Learning . . . . .	17
<b>5 Schedule</b>	<b>19</b>
5.1 Risk Analysis . . . . .	19
5.2 Risks Encountered . . . . .	21
5.3 Gantt Chart . . . . .	22
<b>6 Software Development</b>	<b>22</b>

<b>7 Evaluation</b>	<b>23</b>
7.1 Evaluation Measures . . . . .	23
7.1.1 Accuracy Scores . . . . .	24
7.2 Experimental Results . . . . .	25
<b>8 Reflection &amp; Challenges</b>	<b>28</b>
<b>9 Future Work</b>	<b>30</b>
<b>10 Conclusion</b>	<b>30</b>
<b>References</b>	<b>32</b>
<b>A Appendix</b>	<b>37</b>

# List of Figures

1	Growth in the number of entries in the UniProt databases over the last decade.	1
2	DNA sequence visualisation	4
3	Number of sequences deposited and experimentally validated in UniProtKB over the past decade. The drop observed between 2015 and 2016 is due to procedures deployed by curators to identify and remove redundant proteomes.	5
4	Machine learning vs Deep learning	6
5	Siamese Network visualisation	7
6	SigNet Architecture	8
7	Our Project's Pipeline	14
8	PreDBA Algorithm	15
9	Handcrafted Features from PreDBA	15
10	Neural Network model visualisation	16
11	Schedule followed for project	22
12	Predicted vs real binding affinities of Single Strand DNA complexes	25
13	Predicted vs real binding affinities of Double I DNA complexes	26
14	Predicted vs real binding affinities of Double II DNA complexes	26
15	Predicted vs real binding affinities of Double III DNA complexes	26
16	Predicted vs real binding affinities of Miscellaneous complexes	27
17	Predicted vs Biju's binding affinities of Single Strand DNA complexes	27
18	Predicted vs Biju's binding affinities of Double I DNA complexes	27
19	Predicted vs Biju's binding affinities of Double III DNA complexes	28
20	Predicted vs Biju's binding affinities of Miscellaneous complexes	28
21	Summary of SN model	37

## List of Tables

1	Comparing different ML methods discussed above. . . . .	13
2	Personal risks that are possible during the project. . . . .	19
3	Technical risks that are possible during the project. . . . .	20
4	Risks that occurred during the project. . . . .	21
5	Evaluation for PreDBA [1] . . . . .	24
6	Evaluation for Biju's 3-layer Stacking Ensemble [1] . . . . .	24
7	Evaluation for Neural Networks . . . . .	25



# 1 Introduction

Protein function prediction (PFP) methods are techniques that bioinformatic researchers and data scientists use to assign roles to proteins. This project applies regression to predict values of binding affinity of protein-DNA complexes [2]. We aim to use two different architectures to predict the binding affinity and compare both sets of results to other methods used on the same dataset [1] [2]. One technique we use is the machine learning (ML) method of neural networks. The other is One-Shot Learning (OSL) using Siamese Networks (SN), which is a form of deep learning (DL).

There is a swarming accumulation of biological data [3], for example, there are over 200 million proteins in the UniProtKB as of January 2022 [4]. The use of new technology has allowed us to discover new proteins, which can be visualised in Figure 1, however has left gaps of information for their functionality. Scientists are struggling to predict the role of each protein manually. This results in an increased demand for methods to decode the function of new protein discovery in a human-comprehensive way. Protein information is stored in multiple databases, including UniProt [5] and the Protein Databank [6]. This provides a freely accessible set of protein sequences with functional information. Due to the lack of techniques to find the function of new proteins, there are gaps of knowledge between the number of proteins discovered and their action purpose.

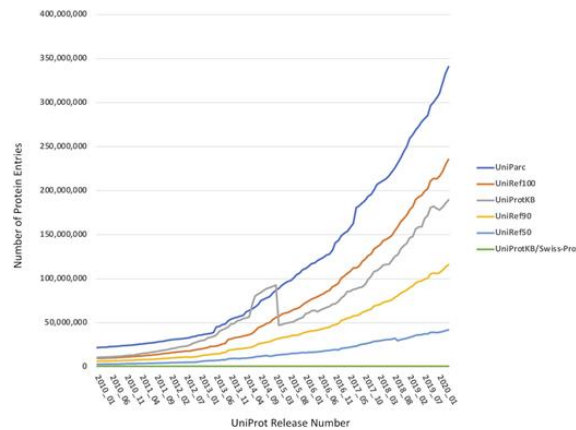


Figure 1: Growth in the number of entries in the UniProt databases over the last decade.

[7]

Researchers are inclined to use computational methods to predict protein function rather than laboratory methods. Using lab equipment is more costly and can be time-consuming.

There are many laboratory methods including, X-ray Diffraction [8] and Chromatin Immunoprecipitation [9]. Automated Function Prediction (AFP) deals with the algorithmic assignment of functional annotations to predict the proteins with unknown function by using proteins whose function has already been determined experimentally [10]. Computational methods for AFP methods can be sequence-based, network-based, kernel methods for structured output spaces or hierarchical ensemble-based [11].

Existing literature indicates there has been progress in PFP using machine learning. Researchers have developed architectures using DL for both classification and regression which produce high levels of accuracy [12] [13]. One limitation of these methods is they tend to be computationally expensive and require a large amount of RAM. There are also methods which are not as expensive however they are not used on biological datasets [14]. Many methods mention they have been trained on ‘small’ datasets, for example approximately 3000 training elements [14]. Alina Biju, a student from Swansea University, managed to improve the PreDBA algorithm [2] from implementing two-layer stacking ensemble methods to three-layer [1]. Biju managed to improve the accuracy of predicting the binding affinity however stated having a larger dataset would have shown better results.

Gaining access to a larger dataset requires expert biological knowledge. Our project intends to find a solution to predict more accurate regression results using deep learning rather than ensemble method learning. The dataset we will be using for our architecture consists of PDB files which contain the experimentally calculated protein-DNA complex binding affinity. The dataset is particularly small as it is difficult to curate datasets using binding affinity. The same files were used for the PreDBA algorithm [2] and Alina Biju’s [1] work. Due to the dataset being small, it restricts the application of traditional ML techniques. In this study, we want to explore DL to see if it is possible to improve the accuracy on smaller datasets by using OSL with SN.

My aims of this project are:

- Train the dataset, the PDB files that include the experimental binding affinities used for PreDBA and Biju’s code, on a fully connect neural network (NN).
- Create Siamese Network (SN) for One-Shot Learning (OSL) to train on the same PDB files as the NN.

- Compare PreDBA and Biju’s regression results with NN and SN architecture.

Section 2 includes the background terminology and use of resources needed to understand the project. In Section 3, we will look at related work to the project and make comparisons of state-of-the-art approaches. Section 4 entails how we implemented methods for this project. Section 5 looks at the risks that could have occurred in the project and the risks that did occur as well as the project timeline. Section 6 looks at the Software Development Life Cycle we followed. Section 7 discusses the evaluation measures we used to compare this project to PreDBA and Biju’s approach and results we achieved. In Section 8 we will look at the challenges faced and the reflection on the project. Section 9 involves looking at future work we would like to undertake after this project. Finally, section 10 summarises what we gained from this project.

## 2 Background

Section 2.1 covers biological terminology necessary to understand the purpose of the project. Section 2.2 explains the dataset used for the project. Section 2.3 is about the definition of Machine Learning (ML) and the different methods of ML we have used in the project. Finally, section 2.4 describes the libraries used in the project and how they were used.

### 2.1 Biological Terminology

**DNA Sequence and Structure:** DNA sequencing determines the order of the four chemical building blocks – the “bases” – that make up the DNA molecule [15]. The four bases are adenine (A), thymine (T), cytosine (C), and guanine (G) where A and T are always paired, and C and G are always paired. These are known as the ‘base pairs’. DNA is a two-stranded molecule, that appears twisted, giving it a unique shape referred to as the double helix [16].

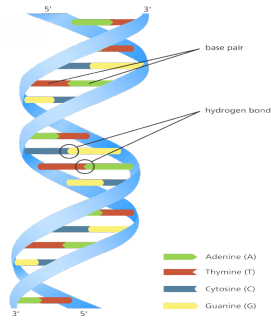


Figure 2: DNA sequence visualisation

[17]

**Protein Sequence and Structure:** Proteins are molecules found in living cells. All cells are made up mostly from protein and are needed to produce new cells for growth and repair damaged tissues. They are compounds that consist of amino acid molecules [18].

**Binding Site and Binding Affinity:** Binding sites are the areas where proteins and DNA bind to each other. The binding affinity of protein-DNA complex is the strength requires to for the binding site to occur between the protein and DNA.

## 2.2 Biological Database

In the past two decades, there has been a huge increase in the number of biological databases available to us. There is a variety of databases including the Protein Data Bank [6], as well as more complex databases such as WGS [19]. However, many of the databases have gaps of knowledge in them. Figure 3 shows the boost in the amount of data added into the UniProt database and how it compares to the much lower increase of data validated. This is due to the shortage of knowledge of protein function. This makes them less efficient to use for future research and unable to access to new knowledge. Deep learning has been successfully applied to solve many complex pattern recognition problems, which is why it is crucial to apply to biological databases [20]. This project uses a miniature dataset in comparison to most others. We aim to produce a model that is capable to input such a small dataset and not to affect the accuracy of the prediction.

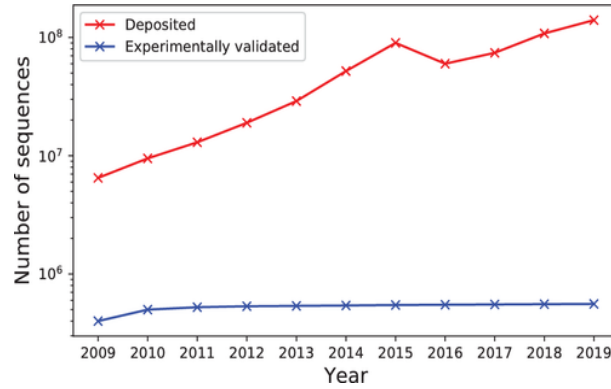


Figure 3: Number of sequences deposited and experimentally validated in UniProtKB over the past decade. The drop observed between 2015 and 2016 is due to procedures deployed by curators to identify and remove redundant proteomes.

[21]

## 2.3 Machine learning

Machine Learning (ML) is a form of artificial intelligence. ML is the analysis of different algorithms and models used to predict outcomes of data fed into models.

**Neural Network:** Neural Networks (NN) are a form of supervised learning. An NN is a series of algorithms that attempts to recognise underlying relationships in a set of data through a process that mimics the way the human brain operates [22]. An NN is characterised by its pattern of connected neurons, its method of determining the weights of the connections and its activation function. Each of these characteristics are included in each layer of the NN.

### 2.3.1 Deep Learning

Deep learning (DL) is a subfield of machine learning that use artificial neural networks that mimic how the brain works. DL uses a set of neurons organised in three layers including: the input layer, the output layer, and the hidden layers [23].

Figure 4 shows the difference between ML and DL is the DL neural network includes feature extraction in it whereas ML neural networks do not.

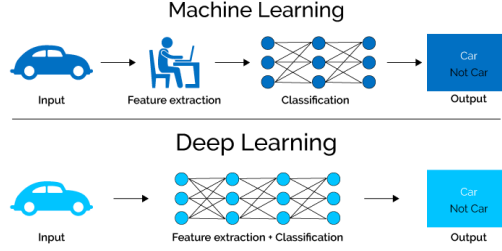


Figure 4: Machine learning vs Deep learning

[24]

**One-Shot/Few Shot Learning:** One-Shot Learning (OSL) and Few-Shot Learning (FSL) are a deep learning methods where the training dataset contains limited information. OSL is a special type of FSL problem, where the aim is to learn information about object categories from one training sample [25]. It has applications such as image classification, semantic segmentation, image generation, object detection and natural language processing [26].

There are a few different approaches of FSL. When there is prior knowledge of similarity, traditional ML models cannot discriminate towards classes that are not present in the training data [27]. FSL allows ML models to do so using different techniques such as using Siamese networks or Triplet networks in order to discriminate between two unseen classes. If there are multiple unseen networks, Matching networks, Prototypical networks or Relation networks can be used. To improve the accuracy of the predictions using FSL we can hyperparameter tune using Mega-Agnostic Meta-learning (MAML) or Reptile [27].

**Siamese Network:** A Siamese Network (SN) is an artificial neural network used as a one-shot classifier [28]. SN is a type of network which contains two or more identical subnetworks, such as having the same architecture, parameters, and weights [28]. These identical subnetworks then output an encoding to calculate the difference between the two inputs as shown in figure 5. The inputs of the SN are classified based on the ‘Similarity score’. This can be calculated using Binary cross-entropy, Contrastive function, or Triplet loss.

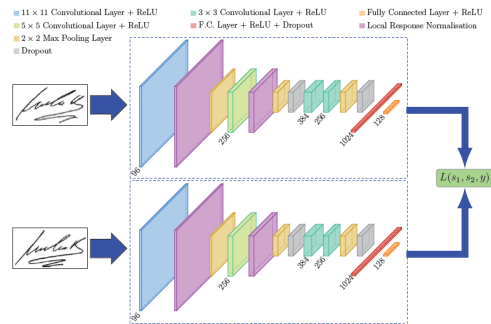


Figure 5: Siamese Network visualisation

29

## 2.4 Libraries

**Sckit-learn, Tensorflow and Keras:** The Python programming language is establishing itself as one of the most popular languages for scientific computing [30], as it has packages that can perform advanced numerical functions. Sckit-learn is used to implement many well-known machine learning algorithms, for example, Bayesian, SVMs and Linear Regression. Tensorflow is an open-source software library used for machine leaning and artificial intelligence. We used Tensorflow and Keras to build and train the different types of neural networks.

**NumPy and Pandas:** NumPy is the foundational package for scientific computing in Python. We used NumPy to input data in the form of arrays to input into the different machine learning models. NumPy as also used for mathematical calculations used in the project. Pandas was used to manipulate datasets using Pandas DataFrames.

## 3 Related Work

This section mentions work that motivated us to perform this project. Section 3.1 focuses on prediction protein function related work and the different ML methods that can be used. Subsection 3.1.1 shows OSL and FSL methods used, subsection 3.1.3 shows graph-based learning and finally 3.1.2 shows ensemble based learning. Section 3.2 summarises academic papers about non-protein related work to do with the project. Subsection 3.2.1 mentions papers about neural networks and subsection 3.2.2 concentrates on FSL and OSL papers.

## 3.1 Protein-based work

### 3.1.1 One-Shot/Few-Shot Learning

There is a large amount of data in bioinformatics and PFP is still an up-and-coming topic in the sector. This means there is still a scarcity and many gaps. Few-Shot Learning Dataset of Molecules (FS-MOL) is an algorithm used to implement FSL on carefully selected data from ChEMBL27 [31] to model early stages of drug discovery [12]. This algorithm managed to implement FSL successfully by querying the data, cleaning the data, splitting the data into the training dataset and the testing datasets and finally extracting the features. However, there were no strong evaluation measures carried out to compare the FS-MOL findings with others. FSL was used to classify data for FSL whereas this project use OSL for regression.

An OSL model created by Ahmed in 2017, is the study of protein-protein interactions (PPI) using protein signatures as feature vectors (SigNet) [13]. The architecture of SigNet is a Siamese convolutional neural network, as you can see below in figure 6. SigNet was compared to other start-of-the-art models including ProtVec [32] and LSTM [33]. Overall SigNet performed better compared to the other deep learning methods. SigNet shows that neural networks can be powerful for PPI and should be explore more. Thought SigNet produced good results the model is computationally expensive, and 32 GB of RAM was used. Our project aimed not to be as computationally xpensive.

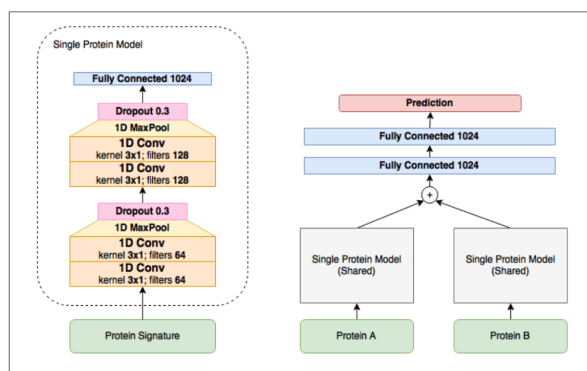


Figure 6: SigNet Architecture

[13]



### 3.1.2 Ensemble Based Learning

When it comes to ensemble methods, there are three types, bagging, boosting, and stacking, where bagging and boosting are homogenous ensemble methods, and stacking is heterogeneous. Ensemble methods are relatively appropriate when predicting protein function due to there being many gaps in biological datasets. Challenges such as CAFA [34] enable participants to fill in the gaps for protein function when the sequence is known. CAFA2 suggested using ensemble methods that integrate data from different sources to improve the prediction accuracy [35].

There has been an attempt to predict protein function using stacking ensemble methods. A heterogeneous ensemble approach for predicting protein-DNA binding affinity (PreDBA) [2] used stacking, with two-layer stacking. Early data integration was also applied to the dataset. This model was used to train PDB files, the same dataset used to train the models of this project, for the regression problem.

Last year, Alina Biju from Swansea University amended the PreDBA algorithm. Biju implemented a three-layer stacking ensemble as well as implementing the late data integration approach [1] compared to the original PreDBA model which applied early data integration. Biju's algorithm results showed higher accuracy rates for predicting the binding affinity with the three-layer ensemble. A challenge ensemble methods face is they require larger datasets to maximise accuracy rates. Our project aimed to see if deep learning would provide better results on the small dataset used for the PreDBA and Biju's algorithms.

### 3.1.3 Graph Based Learning

Another popular ML methods used for protein predictions is graph-based learning methods. Recently, several techniques using Graph Neural Networks (GNN) have been proposed for AFP including DeepFRI [36] and DeepGOA [37]. DeepFRI combines protein structure and pre-trained sequence embedding in a Graph Convolutional Network (GCN). DeepFRI extracts residual-level features using a recurrent neural network architecture with Long Short-Term Memory (LSTM). Then a GCN model is created using a deep architecture. This is used to distribute the residue-level features between residues that are proximal in the structure and create final protein-level feature representation. DeepFRI outperforms other state-of-the-art architectures such as BLAST [38] and DeepGO [39] where DeepFRI has a higher score in the robustness evaluation.

DeepGOA is an architecture using information encoded in the Gene Ontology (GO) hierarchy to create a GCN based model. This model predicts GO annotation of proteins. DeepGOA first studies the semantic representation of GO using GCN while optimising the representation of the protein sequence with CNN. DeepGOA then calculates the dot product of the GCN and CNN. This is to learn the mapping from the feature representation to the testing data using the cross-entropy loss function. DeepGOA was compared against popular methods including BLAST [38], DeepGO [39] and DeepGOPlus [40]. DeepGOA almost achieve better results than the other methods in all evaluation measure used such as AUC (area under the receiver operator characteristic curve) and AURPC (area under the precision-recall curve). AURPC was used as it is more sensitive to class-imbalance compared to AUC.

The two techniques mentioned above [36] [37] inspired You, Yao, Mamitsuka and Zhu to build the DeepGraphGO [41] model. DeepGraphGO has two inputs, the graph and nodes or weighted adjacency matrix representing the protein network and proteins respectively. Each node has a binary feature vector generated by InterProScan [42], indicating whether a protein is absent or not. The model then has a GCN layer, which show updates of each the representation vector of the node. Finally, the output layer predicts the score of the GO terms for each protein. DeepGraphGO algorithm overcomes the classification problem and outperforms similar approaches such as GeneMAINIA [43] and clusDCA [44]. The downside of these models is that they were created for larger datasets whereas our project uses a small dataset. This is because curating protein-DNA binding affinity datasets is difficult and not much data is available to us currently.

In January 2022, Kulmanov and Hoehndorf used zero-shot learning on GO terms on smaller datasets. This is because, as mentioned above, other methods [41] require a larger amount of training data and cannot make predictions on smaller amounts of data or when there are gaps of data. They developed an architecture known as DeepGOZero [45]. This combines PFP with a model-theoretic approach for embedding ontologies into a distributed space. Binary vectors were generated using InterPro as an input and then two layers of multi-layer perceptron (MLP) were implemented. This was used to generate an embedding of a protein. Finally, the binary cross-entropy loss was calculated between the DeepGOZero predictions and the labels. DeepZeroGO was then combined with the DiamondScore method uses the sequence similarity score taken from Diamond architecture [46]. These predictions and labels were

then optimised together using four normal losses from ELEmbedding [47]. Results showed that DeepGOZero with DiamondScore outperformed DeepGOPlus [40], DeepGOCNN [40], DeepGOZero [45] and DiamondScore individually in AUPR and AUC evaluation measures.

## 3.2 Non-protein-based work

### 3.2.1 Neural Networks

A relatively new algorithm founded in 2017, by Bataineh and Marler [48], uses neural network to train reduced datasets. The foundation of this architecture uses a basic radial-basis network (RBN). This revised version of the classic RBN includes four main components in the new artificial neural network (ANN). Firstly, training inputs are normalised by the Gaussian standardisation method. These values are then set for all basis-functions using root-mean-square deviation (RMSD). Next is setting the basis-function spread and selecting their centres using the new ordinary least squares (OLS) method. Finally, the optimal weight of the layers and the standard deviation are calculated but minimising the sum of square error (SSE) over all training cases.

The results of the new RBN design show an improvement for the regression problem when there is a reduced training dataset available. However, for more precise results, a larger dataset is still required. Even though this algorithm produced better results for the regression problem using deep learning methods, it was still computationally expensive.

### 3.2.2 One-Shot/Few-Shot Learning

One-Shot Learning (OSL) and Few-Shot Learning (FSL) are both relatively newfound architectures as they fall under the category deep learning. There have been several recent papers using different OSL and FSL models to find improved results for both classification and regression problems. In late 2020 Wetzel, Ryczko, Melko and Tamblyn used the model of a twin neural network (TNN), also known as Siamese network, on the regression problem [49].

The model takes a pair of inputs, where each input is connected to the fully connected neural network. The model is then optimised depending on the dataset used. The training object is to minimise the mean squared error on the training set. This paper concluded using TNN for regression outperformed other traditional regression methods such as random forests and xgboost. TNN and ensemble methods together improved the accuracy even further. This

paper shows using SNs for the regression problem can result in better accuracy compared to other traditional architectures. We chose to use this architecture on a protein dataset as it could lead to an advance in the world of bioinformatics.

Another SN based architecture is known as SiamReg (2020) [14]. SiamReg uses Long Short-Term Memory (LSTM) to build a regression network and combine the network with a Siamese network. This model was used on the ImageNet Video dataset 2015 [50] as the training set. The results using this architecture were average in terms of accuracy and other networks with more complex regression structure performed better. This could be due to the size of the dataset (3862 training snippets) or due to the fact the model was tested using images and videos. For our project, we would like to see if using a similar architecture on protein data would perform better.

SNs are a type of OSL. The well-known technique Mega-Agnostic Meta-Learning (MAML) can be used for FSL. The aim of meta learning is to train a model on a variety of learning tasks, such that it can solve new learning tasks using only a small number of training samples. In 2017, Finn, Abbeel and Levine tested MAML algorithms [51] to see if they could be used on supervised regression, classification, and reinforcement learning. For the regression problem, MAML took the input to regress to an output of a sine wave. To improve the MAML algorithm, gradient steps were added. This optimised the parameters of MAML and reduced overfitting. MAML was also used for classification on the Omniglot dataset [52] and MiniImageNet dataset [53]. The model outperformed some state-of-the-art approaches such as SNs and matching networks [54]. Our project is using SNs to see if they can outperform other well-known architectures for the regression problem.

### 3.3 Comparing Methods

Algorithm	Suitable for small datasets	Not computationally expensive	Neural Networks	Siamese Networks	Deep learning	Classification Problem	Regression Problem
FS-MOL	X	X			X	X	
SigNet	X		X	X	X		X
PreDBA	X	X					X
Alina Biju	X	X					X
DeepFRI			X		X	X	
DeepGOA			X		X	X	
DeepGraphGO			X		X	X	
DeepGOZero	X		X		X	X	
This Project	X	X	X	X	X	X	X

Table 1: Comparing different ML methods discussed above.

Table 1 shows us that some approaches of protein work use deep learning, such as DeepGOZero. There is still plenty to be discovered as DL is a relatively new concept. All the algorithms above trained on protein datasets have not used SNs, apart from SigNet, and used other methods of DL. A few of the methods discussed above are suitable for smaller datasets such as FS-MOL, which averages approximately 2000 data values. This project is focusing on only 100 protein values. This usually means the accuracy predicted will be lower however OSL is known to be suitable for small datasets and hopefully will predict accurate results.

## 4 Methodology

This section explains the implementation of this project. Figure 7 shows a pipeline of the project starting with the dataset used explained in section 4.1. Part A of the figure discusses the feature extraction used for the NN in section 4.2 followed by building the model and hyperparameter tuning. Part B of the pipeline focuses on how we implemented OSL using a SNs. Section 4.3 explains how we prepared the data, created pairs to feed into the Siamese model and found the contrastive loss. This was used to predict the binding affinity.

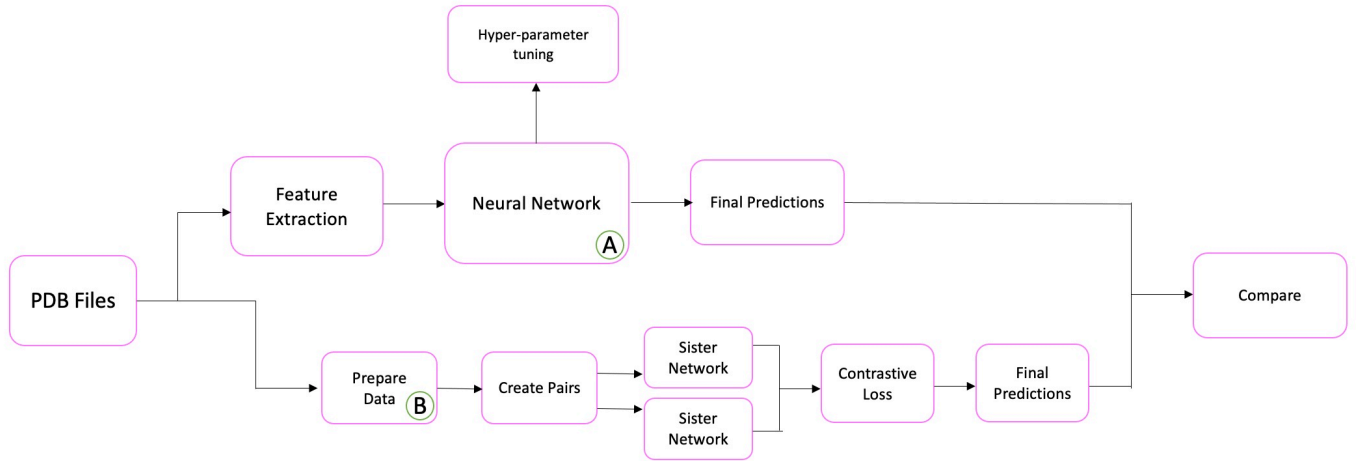


Figure 7: Our Project's Pipeline

## 4.1 Dataset

Experimental data on binding affinity data is reported abundantly [55]. However, there are not many datasets developed which are useful to train and test for protein-DNA binding affinity. We used the PreDBA [2] dataset in this project where PreDBA chose a set of 201 protein-DNA complexes by hand with experimentally determined binding affinity. These complexes were reduced to 100 as PreDBA only used proteins with sequence similarity more than 40% in the dataset.

Figure 8 below shows the pipeline for classification and feature extraction of the PreDBA algorithm. PreDBA then classified the data depending on their DNA complex type. This was based on the rule of the Nucleic Acid Database (NCB) [56] where the complexes were split into three categories: complexes with single stranded DNA, double stranded DNA complexes and miscellaneous complexes. The double stranded DNA category was then further split into three classes. Double I when the binding affinity residues is less than 10%, double II when residues are between 10% and 20% and double III when residues were greater than 20%.

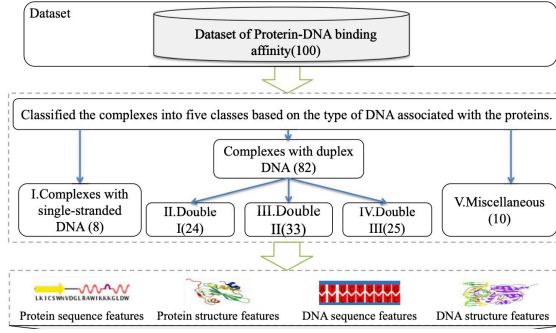


Figure 8: PreDBA Algorithm

## 4.2 Fully Connected Neural Network

### Feature Extraction/Feature Selection:

This project also used the handcrafted features created by the PreDBA algorithm [2]. Once the data was classified, the four feature types were found: protein sequence, protein structure, DNA sequence and DNA structure. The correlation coefficients were used to measure the relationship between each feature and binding affinity. These correlation coefficients were then put in descending order and the top 10 features were selected for each complex type. Figure 9 below shows the features selected for each protein-DNA compound.

Feature	Feature Type
The number of DNA nearest neighbour bases AA/TT	DNA Sequence
The number of DNA nearest neighbour bases CA/GT	DNA Sequence
The number of DNA nearest neighbour bases GA/CT	DNA Sequence
The number of DNA nearest neighbour bases GG/CC	DNA Sequence
The number of DNA nearest neighbour bases GC/CG	DNA Sequence
Percentage of Watson-Crick base pairs	DNA Structure
The amount of Watson-Crick base pairs	DNA Structure
The number of aromatic and positively charged residues in the protein	Protein Structure
Percentage of polar residues in the protein	Protein Structure
The amount of alpha helix in the protein	Protein Sequence
The amount of beta sheet in the protein	Protein Sequence
Molecular mass of the alpha helix	Protein Sequence
Molecular mass of beta sheet	Protein Sequence
The amount of hydrophilic residues	Protein Structure
Total number of hydrogen bonds of the protein fraction	Protein Structure

Figure 9: Handcrafted Features from PreDBA

### Neural Network Model:

First, we prepped the training dataset by taking in the excel sheet of the class, dependent on the complex type. After reading the file in, we filtered the sequence features and structure features and appended the features values of each protein to two lists. The final list had the expected values of the binding affinity. We then concatenated the structure feature and sequence feature lists and converted the list into an NumPy array.

We then created the sequential model which included the layers of the neural network. The input layer took 8 neurons and an input dimension of 8, due to the shape of the NumPy array. The input also had a normal kernel initialiser which initialises the kernel weights matrix and the activation is set to ReLU. ReLU (Rectified Linear Unit) activation function is a non-linear function that outputs directly if it is positive, otherwise zero [57]. We then set hidden layers to expand the shape to 10 neurons as you can see in the image below in figure 10. This layers kernel initialiser was also set to normal and the activation level was set to ReLU. The output layers shape was shrunk to one and the kernel initialiser was set to normal.

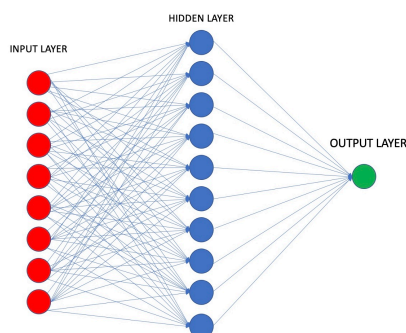


Figure 10: Neural Network model visualisation

### Hyperparameter Tuning:

Once we created the NN model, we attempted to hyperparameter tune to optimise our results. The parameters to optimise consisted of the batch size and epochs. We made a sequential model consisting of two layers, one with 12 neurons as the units, the input dimensionality as 6 and the activation ReLU. The other with one neuron and both layers with normal kernel initialisers. The optimal parameter results as the batch size being size to 40 and the number of epochs run set to 200. However, as our training data is miniature, so hyperparameter tuning did not improve our results in predicting the binding affinity.



### 4.3 One Shot Learning

#### Preparing Data & Creating Pairs:

Unlike the neural network that takes handcrafted features, OSL processes the raw data and computes features itself. We prepared the data to feed into the model by getting the data values from the supplementary table provided by PreDBA [2]. We did this by using the pandas library to read in the excel sheet, with the PDB files information, and take the relevant data into and put it in a data frame. One of the titles for the table was “PDB IB” which consisted of information about the complex protein chain and DNA chain. We then split the information to separate the chains from each other. These values were then appended into a 2D NumPy array which we used as the training data for the model. We also extracted the experimental protein-DNA complex binding affinities from the data frame and appended them into a separate 1D NumPy array. These values were used as training labels for our SN. We needed a way to find similarities between each protein-DNA complex. Using the data prepared, we generated random pairs to input into the SN. This was done using the random range function from the random package.

#### Siamese Network:

The Siamese network consisted of two identical subnetworks, as mentioned earlier. We created a function for making the base mode which took in the input shape as a parameter as this was the shape of the training data. We used Convolutional Neural Networks (CNN) as the base model this architecture. The difference between FCNN and CNN is unlike CNN’s, FCNN can take an input of an arbitrary size. The sequential model consists of six layers: two 1-dimensional convolutional layers, two max pooling layers, a flatten layer and three dense layers. The convolution layers are the core of the CNN as this is where most the computations happen. They are used to create feature maps that summarise the present features in the input [58].

The first convolutional layer, the input layer, consisted of 4 filters, which is the dimensionality of the output space. The kernel size specifies the length of the 1D convolutional window. This was 1 due to the scarce amount of data we had, the activation function was set to ReLU and finally the input shape was the size of the training data. After a convolutional layer usually comes a pooling layer. This is because the feature maps generated from the convolutional layer are too sensitive to the location of the features in the data [59]. Therefore, we used pooling to down-sample the feature map. We used max pooling layers which took a

pool size of 1, no strides and valid padding, which means no padding.

The second convolution layer and max pooling layer used the same parameters as the first. After pooling, the next layer of the model was flattening the pooled feature map as this is used as the data to be inserted into the ANN. We use the same model we created in the FCNN in section 4.2 with three dense layers which are the input, hidden and output layer. The summary of the SN model can be found in Appendix A 21. However, the one difference is the output layer in the SN uses the activation function ‘sigmoid’ as it is one of the best normalised functions and gives us a clear prediction between 0 and 1. Below is the formula for the sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}}$$

### Calculating Distance and Loss:

Once we get the outputs of the SN for the protein-DNA complex, it is important to see how similar they are to one another. We did this by calculating the distance between each complex’s feature vectors found in the SN. We found the distance using the Euclidean distance equation:

$$d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

where  $p$  and  $q$  are two points in Euclidean  $n$ -space,  $q_i$  and  $p_i$  are the Euclidean vectors starting at the initial point and  $n$  represents  $n$  space.

We created a contrastive loss function to calculate the loss of similarity between each complex defined as follows:

$$L(p, q) = (1 - Y) \frac{1}{1} (d(p, q))^2 + Y \frac{1}{2} (\max(0, m - d(p, q)))^2$$

This takes the distance equation above but uses it as loss for each complex. If both feature vectors are part of the same complex, the loss is small for this part of the equation. The first part of the equation is calculating the hinge loss. This is when A and B are different protein-complexes, called a negative pair. For  $p$  and  $q$ , we wanted a distance more than  $m$  to make the function more efficient as it would disregard negative pairs that have already been found. Together, this equation is the contrastive loss and shows us if A and B are the same complex. If so, label  $y$  will be equal to 1 and if they are different label  $y$  will be 0.

## 5 Schedule

This section covers the potential risks that could have occurred in the project in section 5.1. Followed by the risks that we did encounter during the project in section 5.2. Finally, section 5.3 shows the schedule we followed throughout the project.

### 5.1 Risk Analysis

Risk	Likelihood (1-5)	Consequence (1-5)	Risk Indicator	Mitigation	Management
Time Management	Moderate (3)	Significant (3)	9	Refer to Gantt Chart and achieve tasks and milestones set. Make a daily to do list with priorities at first. Make a weekly plan including events and modules to study for which are not part of the project.	Discuss weekly plan with supervisor in each meeting so you know what your aim is for the week and have it completed for the next meeting.
Hardware Failure	Unlikely (2)	Major (5)	10	Add all files to Git-Hub and continue to upload newer versions when updated so they can be accessed on other device if necessary.	Contact supervisor and department to see what the most suitable solution is, i.e., have access to a PC lab room or borrow a laptop.
Health	Moderate (3)	Major (4)	12	Be wary that COVID-19 is still an issue and follow guidelines the university have offered. Take care of yourself and give yourself some downtime once you've reached a milestone. Aim to finish 2 weeks earlier in case of getting ill.	Communicate with the university and mentor. Use final 2 weeks (if not ill) to proof read and contact supervisor for additional pointers.

Table 2: Personal risks that are possible during the project.

Risk	Likelihood (1-5)	Consequence (1-5)	Risk Indicator	Mitigation	Management
Dataset too small	Unlikely (2)	Major (4)	8	Look for papers which have methods that are small data set friendly. Use datasets that those previous papers have use to see if you can find bigger ones.	Ask colleagues in department to see if they have any bigger datasets you could use.
Errors in implementing algorithms	Moderate (3)	Major (4)	12	Find websites with templates on how to implement the algorithms you're going to use and look out for mistakes while coding.	Debug code as written. If you're at a standstill ask supervisor to look over code to see if they can stop any mistakes.
Lack of experience with ML	Moderate (3)	Significant (3)	9	Look at many papers and compare the FSL algorithms which will work best with your dataset and give encouraging results	Ask supervisor and other peers for advice
Unable to stick to the Software Development Life Cycle (Agile Scrum)	Unlikely (2)	Major (3)	6	Weigh the advantages and disadvantages of the SDLC. Look at papers which have followed the same SDLC and see approaches those projects took in order to follow it.	Analyse with the methodology chosen and ask for advice from supervisor on what to do next.

Table 3: Technical risks that are possible during the project.

## 5.2 Risks Encountered

Risk	Likelihood (1-5)	Consequence (1-5)	Risk Indicator	Mitigation	Management
Time Management	Moderate (3)	Significant (3)	9	Refer to Gantt Chart and achieve tasks and milestones set. Make a daily to do list with priorities at first. Make a weekly plan including events and modules to study for which are not part of the project.	Discuss weekly plan with supervisor in each meeting so you know what your aim is for the week and have it completed for the next meeting.
Hardware Failure	Unlikely (2)	Major (5)	10	Add all files to Git-Hub and continue to upload newer versions when updated so they can be accessed on other device if necessary.	Contact supervisor and department to see what the most suitable solution is, i.e., have access to a PC lab room or borrow a laptop.
Health	Moderate (3)	Major (4)	12	Be wary that COVID-19 is still an issue and follow guidelines the university have offered. Take care of yourself and give yourself some downtime once you've reached a milestone. Aim to finish 2 weeks earlier in case of getting ill.	Communicate with the university and mentor. Use final 2 weeks (if not ill) to proof read and contact supervisor for additional pointers.
Lack of experience with ML	Moderate (3)	Significant (3)	9	Look at many papers and compare the FSL algorithms which will work best with your dataset and give encouraging results	Ask supervisor and other peers for advice

Table 4: Risks that occurred during the project.

As well as the project, there were also other deadlines to meet for other modules. Some assignments took longer to complete than others. Due to some deadlines being earlier, I had to prioritise them over the project. I had to proceed with the mitigation and set myself internal deadlines of what should be completed for the deadline every couple of weeks. In January 2022, my laptop got wet. Luckily all the project progress was backed up on my drive therefore not being a major issue in the project.

I tested positive for COVID-19 in February 2022. This was mitigated by being slightly ahead schedule and allowing 2 weeks at the end of the project if I was to fall behind. Finally, as I had very little experience in Machine Learning (ML), it was difficult to grasp the idea of some algorithms such as FSL. This made it difficult to implement, as I had to do extensive research. I spoke to the project supervisor who helped me solidify ideas and gave me some new ones to move forward with the project.

### 5.3 Gantt Chart

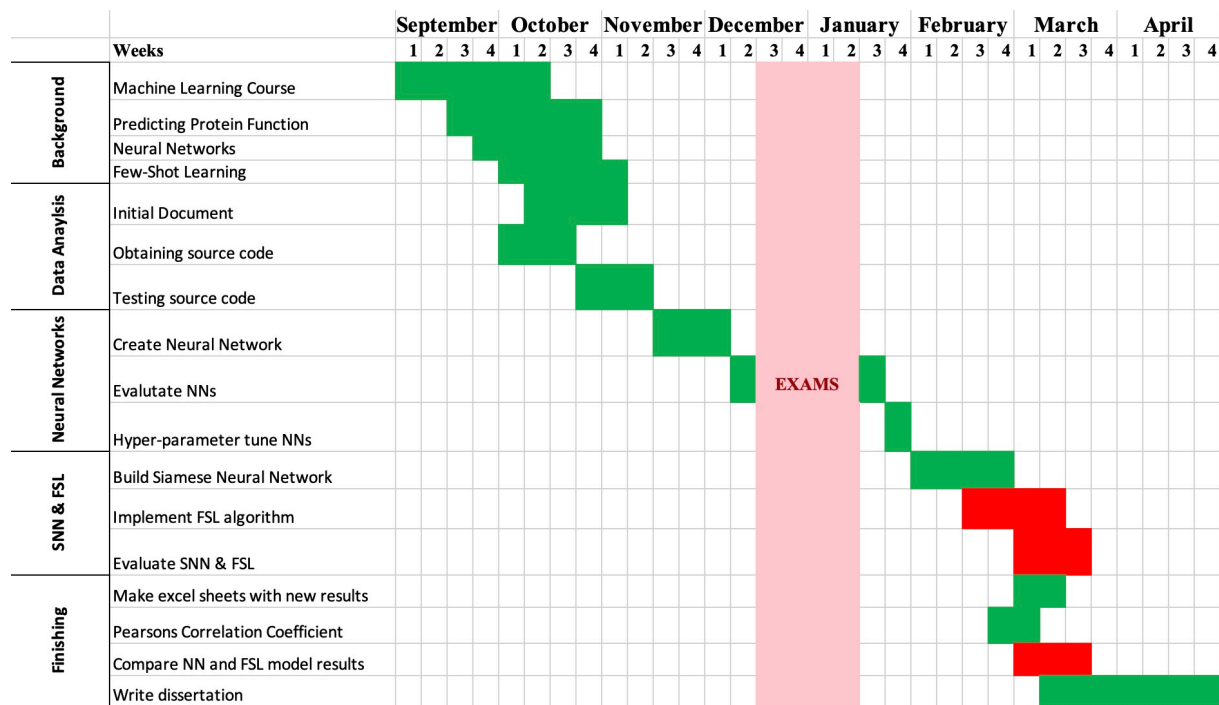


Figure 11: Schedule followed for project

## 6 Software Development

For the project we decided to use the Agile Scrum methodology. Agile is a method where continuous iterations and testing occur throughout the entire Software Development Life Cycle (SDLC) of a product. The steps of the Agile Scrum Methodology are what you proceed with each step until you (or the client) are content with the outcome. This methodology was the most efficient for the project as Agile Scrum enabled us to make flexible changes, which was necessary as it was easy to change algorithms in the implementation. Another benefit of this SDLC is the reduction of risks and that functionality can be added in later stages. This

was helpful as we could backtrack once advice was received, and we could change parts of the code where necessary. Despite all the advantages, for the SDLC to work, it was vital we stuck strictly to the routine. This was tedious and frustrating as it was an iterative process.

## 7 Evaluation

This section covers how we evaluated the project carried out. Section 7.1 explains the evaluation measures used to compare other architectures to the project. Section 7.1.1 shows the accuracy scores and section 7.2 shows the experimental results we achieved in the project.

### 7.1 Evaluation Measures

#### Mean Absolute Error (MAE):

MAE is a model evaluation metric used with regression models. The MAE represents the average of the absolute difference between the actual and predicted values in the dataset [60]. Even though MAE is less biased than other evaluation metrics such as MSE, it is still sensitive towards higher values [61]. The formula below shows how MAE is calculated where  $\hat{y}$  is the predicted value and  $\bar{y}$  is the mean value. The closer MAE is to zero the better the regression model is.

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}|$$

#### Coefficient of determination (R2):

R2 is used to analyse how differences in one variable can be explained by a difference in the second variable [62]. The range of results is from 0 to 1, where 1 is the best result. The formula below shows how to calculate R2.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n(\sum x^2 - (\sum x)^2)]} \sqrt{[n(\sum y^2 - (\sum y)^2)]}}$$

#### Pearson Correlation Coefficient:

Pearson correlation coefficient measures strength between different variables and their relationships. The coefficient returns a value between -1 and 1 where if the correlation coefficient

is -1, a strong negative relation is indicated. If the correlation coefficient is 0, there is no relationship. If the correlation coefficient is 1, there is a strong positive relationship between the variables, hence we want to get the value as close to one as possible [63]. The PreDBA algorithm uses this evaluation method which benefited me to compare to mine and Biju’s results.

The Pearson correlation coefficient  $r$  is defined as follows:

$$r = \frac{\sum_{i=1}^n (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^n (a_i - \bar{a})^2} \sqrt{\sum_{i=1}^n (b_i - \bar{b})^2}}$$

where  $n$  represents the number of samples  $a_i$ ,  $b_i$  are the  $i$ th sample, and  $\bar{a}$  and  $\bar{b}$  represent the mean of the samples, i.e  $\bar{a} = \frac{1}{n} \sum_{i=1}^n a_i$  and  $\bar{b} = \frac{1}{n} \sum_{i=1}^n b_i$ .

### 7.1.1 Accuracy Scores

	<b>Correlation Coefficient (r)</b>	<b>Mean Absolute Error (MAE)</b>	<b>Coefficient of determination (R2)</b>
<b>Single</b>	0.94	0.64	0.71
<b>Double I</b>	0.83	1.17	0.67
<b>Double II</b>	0.84	0.82	0.70
<b>Miscellaneous</b>	0.83	0.43	-0.62

Table 5: Evaluation for PreDBA [1]

	<b>Correlation Coefficient (r)</b>	<b>Mean Absolute Error (MAE)</b>	<b>Coefficient of determination (R2)</b>
<b>Single</b>	0.89	0.67	0.65
<b>Double I</b>	0.99	0.14	0.98
<b>Double II</b>	0.995	0.14	0.98
<b>Miscellaneous</b>	0.85	0.42	0.68

Table 6: Evaluation for Biju’s 3-layer Stacking Ensemble [1]



	Correlation Coefficient (r)	Mean Absolute Error (MAE)	Coefficient of determination (R2)
Single	0.99	2.33	0.19
Double I	1.0	3.50	-0.45
Double II	1.0	5.36	-0.80
Double III	0.99	5.93	-2.13
Miscellaneous	1.0	3.02	-0.47

Table 7: Evaluation for Neural Networks

Table 7 shows that the neural network performs poorly compared to both PreDBA and Biju’s architectures. Even though the correlation coefficient shows a strong positive, the MAE results of our architecture are quite high, especially for the Double II and Double III classes. The coefficient of determination is best when result is as close to 1 as possible. Our results, especially for Double III. These results could be indicating that a bigger dataset is required to improve the accuracy score and a small dataset was not sufficient.

## 7.2 Experimental Results

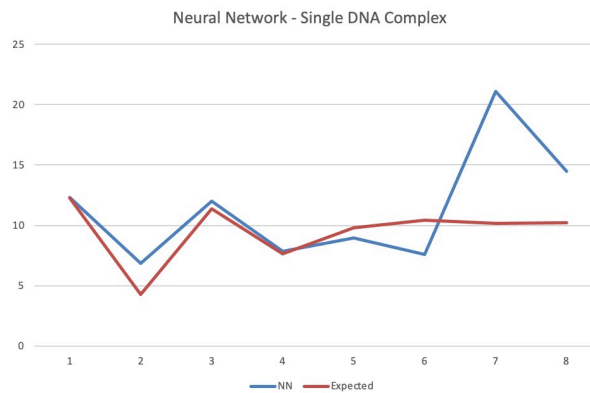


Figure 12: Predicted vs real binding affinities of Single Strand DNA complexes

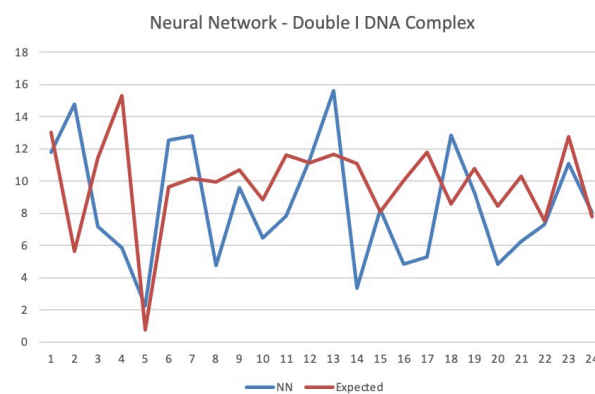


Figure 13: Predicted vs real binding affinities of Double I DNA complexes

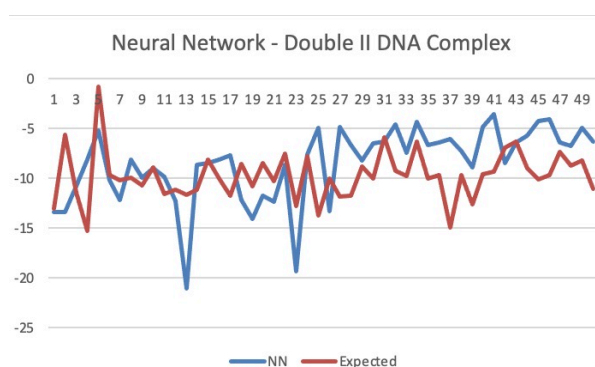


Figure 14: Predicted vs real binding affinities of Double II DNA complexes

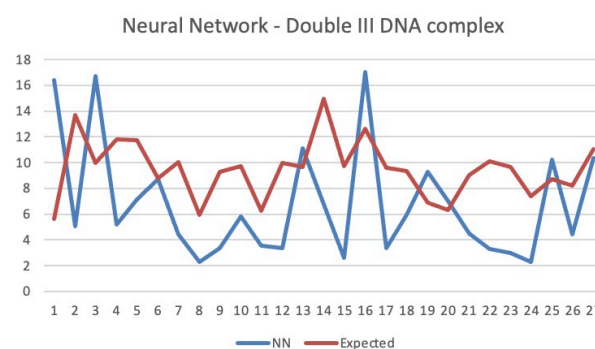


Figure 15: Predicted vs real binding affinities of Double III DNA complexes

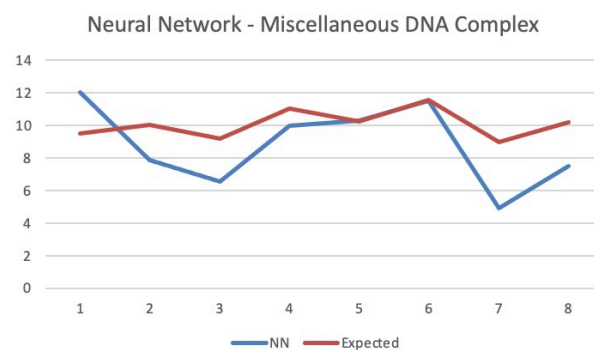


Figure 16: Predicted vs real binding affinities of Miscellaneous complexes

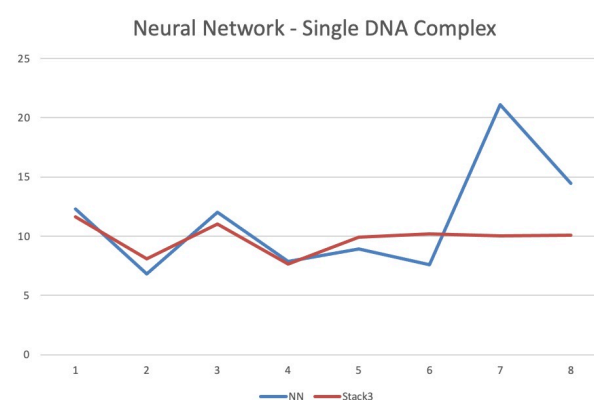


Figure 17: Predicted vs Biju's binding affinities of Single Strand DNA complexes

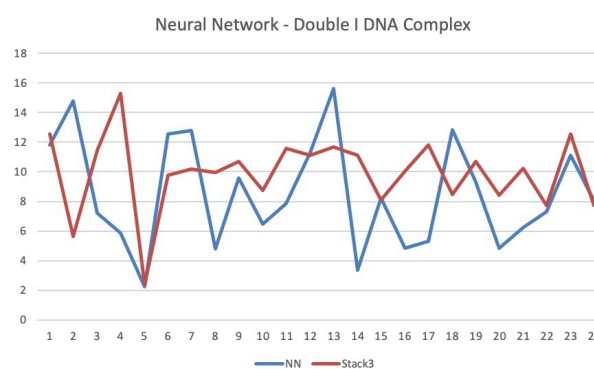


Figure 18: Predicted vs Biju's binding affinities of Double I DNA complexes

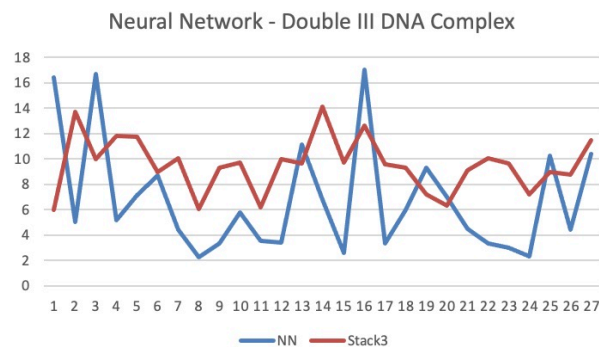


Figure 19: Predicted vs Biju's binding affinities of Double III DNA complexes

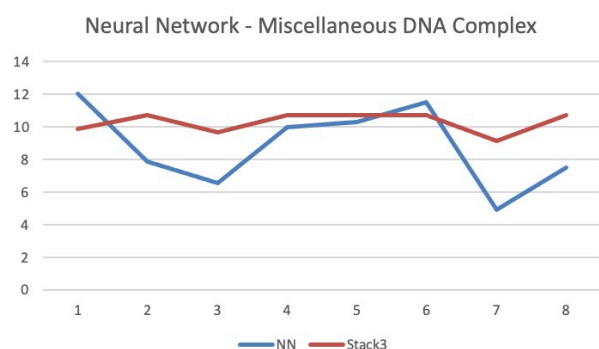


Figure 20: Predicted vs Biju's binding affinities of Miscellaneous complexes

Figures [12](#) - [20](#) show that the neural network model is capable of predicting fairly accurate results. However, the graphs are inconsistent, and it cannot be guaranteed that they will always predict accurately, as you can see especially in figures [13](#) and [18](#) for the double I DNA complex class.

## 8 Reflection & Challenges

This section will discuss challenges that were time consuming during the length of this project. The project presented many challenges to me including understanding and editing someone else's code. Using the PreDBA and Biju's algorithms source code was challenging as the code did not have many comments. I struggled to comprehend the code at the start of the project, due to not knowing much about ML and the different algorithms. However, after doing much research the architecture started to make more sense. This enabled us to edit parts of the source code which benefited this project for example, prepping the complex

type datasets for the neural network. Machine learning, more precisely deep learning, is a relatively daunting area in computer science. Having only started learning ML in September 2021, when our project code presented an error, I did not know how to fix it straight away. With the help of the project supervisor and sources online, I was able to find solutions to errors I was having and prevent myself from making the same mistakes again.

One of my main challenges was using the CNN in the SN base model. This is because, when I was researching how to implement them, most articles used the SN for image similarity recognition [64]. This means that the dimensions used for the layers were in 2D which I tried to use for this project. However, I then realised I was using the wrong dimension and changed the dimension of the layers from 2D to 1D which allowed the CNN model to be built.

Overall, the project went relatively well for the aim of building a fully connected neural network model. We managed to predict the binding affinity for all five classes using the NN model. We expected the accuracy results to be quite low due to the low quantity dataset. However, some of the NN results were quite similar to the experimental results. If the data set was larger, we might have got better accuracy scores but obtaining biological datasets is more difficult than we had expected. This is out of our reach to acquire as computer scientists and would require someone with more biological background knowledge to help us.

The project must be worked on during the whole academic year which became quite overwhelming. This is because we have so many other tasks to complete for other modules such as assignments and revising for exams. This meant that we had to plan our project strategically and sometimes prioritise unrelated project tasks when there was a deadline. If I was to do this project again, I would start researching earlier. This is because I did not have much knowledge about ML, and it took me a while to grasp the concepts of OSL and FSL especially. This made it difficult for me to start implementation earlier hence why I was not able to meet the aim of implementing the Siamese network. This could have been avoided with better time management and more research prior.

In conclusion, I was able to meet two out of the three aims of the project. These were train the dataset on a neural network model and comparing the results to PreDBA and Biju's results. This project has taught me a lot and is an experience I will never forget. I was not a confident programmer, but this project has showed me I am capable of dealing with

source code written by others and understanding it. I have also learnt how important it is to stay task orientated when carrying out a project. In the future, when handling a bigger project, I know I will have self-discipline and time management to ensure I meet all aims of the project.

## 9 Future Work

If we had more time, we would prepare the dataset correctly to feed into the OSL model. This would allow us to create the correct pairs and be able to predict the results and evaluate the SN model. This would help us to see if using OSL on a smaller dataset would improve the accuracy of predicting the binding affinity. Once we have the base model of the SN working, we would be able to change parameters in the network and optimise them. This includes changing the weights of the layers, bias initialisers and changing activation functions.

Once we have met the original aims of this project, we would be able to explore other directions. By getting a larger dataset for binding affinity, with help from specialists of the field, we could train out data on larger datasets. This would enable us to evaluate how much the accuracy scores would be affected and see if OSL is suitable for large datasets as well.

Another idea is implementing another deep learning algorithm on the dataset such as Zero-Shot Learning. This is another popular method used for small datasets that we came across in our research. Zero-Shot Learning (ZSL) could also lead to more accurate predictions for smaller datasets. ZSL can classify data the model has not even seen before [65]. As there are many biological datasets with gaps of knowledge, this architecture would be beneficial to society and the healthcare sector, especially dealing with small datasets.

## 10 Conclusion

In conclusion, the project was relatively successful. We managed to create a neural network model to predict the protein-DNA complex binding affinity in the PreDBA dataset. We also compared our project results to both the PreDBA algorithm and Biju's work. The comparisons show us that our architecture did not outperform the two prior architectures due to the lack of data in the dataset. However, our model could potentially outperform the stacking ensemble methods used if a bigger binding affinity dataset was available to us.

Unfortunately, due to the lack of time we were not able to get the OSL architecture working. If we had more time or I had more experience with DL from the start of the project, we could have got the OSL model to also predict each complex's binding affinity. This might have outperformed the other algorithms as OSL is better suited for smaller datasets. The OSL architecture has the potential to make a difference to scientists and researchers in the bioinformatics industry. It would allow smaller datasets to be predicted more accurately compared to other architectures.

## References

- [1] A. Biju, “Protein functions prediction (pfp) using machine learning,” 2021.
- [2] W. Yang and L. Deng, “Predba: A heterogeneous ensemble approach for predicting protein-dna binding affinity,” *Scientific Reports*, vol. 10, no. 1, 2020.
- [3] A. Vikram, A. Singh, A. Tiwari, and S. Tripathi, “Function prediction of human proteins using machine learning algorithms,” 2019.
- [4] <https://www.ebi.ac.uk/uniprot/TrEMBLstats>, journal=Ebi.ac.uk, 2022.
- [5] UniProt, “Uniprot.” [online] Available at: <https://www.uniprot.org/>, 2021.
- [6] W. P. D. Bank, “wwpdb: Worldwide protein data bank.” [online] Available at: <https://www.wwpdb.org/>, 2021.
- [7] T. U. Consortium, “Uniprot: the universal protein knowledgebase in 2021,” *Nucleic Acids Research*, vol. 49, pp. D480–D489, 11 2020.
- [8] L. F. Leijôto, T. Assis De Oliveira Rodrigues, L. E. Záratey, and C. N. Nobre, “A genetic algorithm for the selection of features used in the prediction of protein function,” 2014.
- [9] M. J. Buck and J. D. Lieb, “Chip-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments,” 2004.
- [10] S. Makrodimitris, R. C. H. J. van Ham, and M. J. T. Reinders, “Automatic Gene Function Prediction in the 2020’s,” *Genes*, vol. 11, Nov. 2020.
- [11] G. Valentini, “Hierarchical ensemble methods for protein function prediction,” *ISRN Bioinformatics*, vol. 2014, pp. 1–34, 2014.
- [12] M. Stanley, J. F. Bronskill, K. Maziarz, H. Misztela, J. Lanini, M. Segler, N. Schneider, and M. Brockschmidt, “FS-mol: A few-shot learning dataset of molecules.” <https://openreview.net/forum?id=701FtuyLlAd>, 2021.
- [13] M. S. Ahmed, “Signet: A neural network architecture for predicting protein-protein interactions,” 2017.
- [14] Y. GE, R. CHEN, Y. TONG, X. CAO, and R. LIANG, “Combining siamese network and regression network for visual tracking,” *IEICE Transactions on Information and Systems*, vol. E103.D, no. 8, pp. 1924–1927, 2020.



- [15] N. H. G. R. Institute, “Dna sequencing fact sheet.” [online] Available at: [www.genome.gov](http://www.genome.gov), 2021.
- [16] T. Newman, “Dna explained: Structure and function,” 2022.
- [17] [online] Available at: <https://www.pngkey.com/maxpic/u2w7q8u20e6e6o0/>, journal=pngkey.com, 2022.
- [18] “Protein.” [online] Available at: <https://www.bbc.co.uk/bitesize/topics/zf339j6/articles/zh2r97h#zm33f820>, 2022.
- [19] C. Xu and S. A. Jackson, “Machine learning and complex biological data,” *Genome Biology*, vol. 20, no. 1, 2019.
- [20] M. Mahmud, M. S. Kaiser, T. M. McGinnity, and A. Hussain, “Deep learning in mining biological data,” *Cognitive Computation*, vol. 13, no. 1, pp. 1–33, 2021.
- [21] R. Bonetta and G. Valentino, “Machine learning techniques for protein function prediction,” 2019.
- [22] J. Chen, “Neural network definition,” 2022.
- [23] A. Gupta, “Deep learning vs machine learning for regression - analytics vidhya,” 2022.
- [24] E. SPERLING, “Deep learning spreads,” 2022.
- [25] “Neural Network Definition,” May 2022. [Online; accessed 1. May 2022].
- [26] I. A. Ozsubasi, “Few-shot learning (fsl): What it is & its applications.” [online] Available at: <https://research.aimultiple.com/few-shot-learning/>, 2021.
- [27] C. Dilmgani, “What is few-shot learning? methods & applications in 2022,” 2022.
- [28] R. Khandelwal, “One-shot learning with siamese network,” 2022.
- [29] S. Benhur, “A friendly introduction to siamese networks,” 2022.
- [30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, “Scikit-learn: Machine learning in python..”

- [31] D. Mendez, A. Gaulton, A. P. Bento, J. Chambers, M. De Veij, E. Félix, M. P. Magariños, J. F. Mosquera, P. Mutowo, and M. e. a. Nowotka, “ChEMBL: towards direct deposition of bioassay data,” *Nucleic Acids Research*, vol. 47, no. D1, pp. D930–D940, 2018.
- [32] E. Asgari and M. R. K. Mofrad, “Continuous distributed representation of biological sequences for deep proteomics and genomics,” *PLOS ONE*, vol. 10, no. 11, p. e0141287, 2015.
- [33] J. Brownlee, “A gentle introduction to long short-term memory networks by the experts,” 2022.
- [34] “Bio function prediction.” [online] Available at: <https://www.biofunctionprediction.org/cafa/>, 2021.
- [35] L. Wang, J. Law, S. D. Kale, T. M. Murali, and G. Pandey, “Large-scale protein function prediction using heterogeneous ensembles,” *F1000Research*, vol. 7, p. 1577, 2018.
- [36] V. Gligorijević, P. D. Renfrew, T. Kosciółek, J. K. Leman, D. Berenberg, T. Vatanen, C. Chandler, B. C. Taylor, I. M. Fisk, and H. e. a. Vlamakis, “Structure-based protein function prediction using graph convolutional networks,” *Nature Communications*, vol. 12, no. 1, 2021.
- [37] G. Zhou, J. Wang, X. Zhang, and G. Yu, “Deepgoa: Predicting gene ontology annotations of proteins via graph convolutional network,” in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1836–1841, 2019.
- [38] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, and A. e. a. Ben-Hur, “A large-scale evaluation of computational protein function prediction,” *Nature Methods*, vol. 10, no. 3, pp. 221–227, 2013.
- [39] M. Kulmanov, M. A. Khan, and R. Hoehndorf, “DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier,” *Bioinformatics*, vol. 34, pp. 660–668, 10 2017.
- [40] M. Kulmanov and R. Hoehndorf, “Deepgoplus: improved protein function prediction from sequence,” *Bioinformatics*, 2019.

- [41] R. You, S. Yao, H. Mamitsuka, and S. Zhu, “Deepgraphgo: graph neural network for large-scale, multispecies protein function prediction,” *Bioinformatics*, vol. 37, no. Supplement<sub>1</sub>, pp. i262–i271, 2021.
- [42] P. Jones, D. Binns, H.-Y. Chang, M. Fraser, W. Li, C. McAnulla, H. McWilliam, J. Maslen, A. Mitchell, G. Nuka, *et al.*, “Interproscan 5: genome-scale protein function classification,” *Bioinformatics*, vol. 30, no. 9, pp. 1236–1240, 2014.
- [43] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, and Q. Morris, “Genemania: a real-time multiple association network integration algorithm for predicting gene function,” *Genome Biology*, vol. 9, no. Suppl 1, p. S4, 2008.
- [44] S. Wang, H. Cho, C. Zhai, B. Berger, and J. Peng, “Exploiting ontology graph for predicting sparsely annotated gene function,” *Bioinformatics*, vol. 31, pp. i357–i364, 06 2015.
- [45] M. Kulmanov and R. Hoehndorf, “DeepGOZero: Improving protein function prediction from sequence and zero-shot learning based on ontology axioms,” *bioRxiv*, p. 2022.01.14.476325, Jan. 2022.
- [46] B. Buchfink, C. Xie, and D. H. Huson, “Fast and sensitive protein alignment using DIAMOND,” *Nat. Methods*, vol. 12, pp. 59–60, Jan. 2015.
- [47] M. Kulmanov, W. Liu-Wei, Y. Yan, and R. Hoehndorf, “EL Embeddings: Geometric Construction of Models for the Description Logic EL++,” 2019. [Online; accessed 29. Apr. 2022].
- [48] M. Bataineh and T. Marler, “Neural network for regression problems with reduced training sets,” *Neural Networks*, vol. 95, pp. 1–9, 2017.
- [49] S. J. Wetzel, K. Ryczko, R. G. Melko, and I. Tamblyn, “Twin neural network regression,” 2022.
- [50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. e. a. Bernstein, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [51] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” 2017.

- [52] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum, “One shot learning of simple visual concepts,” *Cognitive Science*, vol. 33, 2011.
- [53] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in *ICLR*, 2017.
- [54] O. Vinyals, C. Blundell, T. P. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” in *NIPS*, 2016.
- [55] K. Harini, A. Srivastava, A. Kulandaisamy, and M. M. Gromiha, “ProNAB: database for binding affinities of protein–nucleic acid complexes and their mutants,” *Nucleic Acids Res.*, vol. 50, p. D1528, Jan. 2022.
- [56] B. C. Narayanan, J. Westbrook, S. Ghosh, A. I. Petrov, B. Sweeney, C. L. Zirbel, N. B. Leontis, and H. M. Berman, “The Nucleic Acid Database: new features and capabilities,” *Nucleic Acids Res.*, vol. 42, p. D114, Jan. 2014.
- [57] V. Praharsha, “ReLU (Rectified Linear Unit) Activation Function,” *OpenGenus IQ: Computing Expertise & Legacy*, Jan. 2022.
- [58] “A Gentle Introduction to Pooling Layers for Convolutional Neural Networks,” July 2019. [Online; accessed 1. May 2022].
- [59] “Max Pooling in Python - A Brief Introduction - JournalDev,” Oct. 2020. [Online; accessed 1. May 2022].
- [60] A. Chugh, “Mae, mse, rmse, coefficient of determination, adjusted r squared which metric is better?,” 2022.
- [61] [online] Available at: <https://akhilendra.com/evaluation-metrics-regression-mae-mse-rmse-rmsle/>, journal=Akhilendra.com, 2022.
- [62] S. Glen, “Coefficient of determination (r squared),” 2022.
- [63] M. Thakur, “Pearson correlation coefficient.” [online] Available at: <https://www.wallstreetmojo.com/pearson-correlation-coefficient/>, journal=WallStreetMojo, 2021.
- [64] G. R. Koch, “Siamese neural networks for one-shot image recognition,” *undefined*, 2015.
- [65] E. Tiu, “Understanding Zero-Shot Learning — Making ML More Human,” *Medium*, Jan. 2022.

# A Appendix

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 2, 4)	8
max_pooling1d (MaxPooling1D)	(None, 2, 4)	0
conv1d_1 (Conv1D)	(None, 2, 10)	50
max_pooling1d_1 (MaxPooling1D)	(None, 2, 10)	0
flatten (Flatten)	(None, 20)	0
dense (Dense)	(None, 8)	168
dense_1 (Dense)	(None, 10)	90
dense_2 (Dense)	(None, 1)	11
Total params: 327		
Trainable params: 327		
Non-trainable params: 0		

Figure 21: Summary of SN model