2020

University of Surrey

Department of Computing

Machine Learning Approaches For Classifying Animal Venom Proteins

Final Year Project

Student: Christelle Van Sebroeck
URN: 6418824

Supervisor: Dr. Alireza Tamaddoni-Nezhad

Declaration of Originality

Acknowledgements

I would like to take the opportunity to express the highest gratitude to my supervisor Dr. Alireza Tamaddoni-Nezhad, who has not only supported me throughout the whole year with this project, but has also been incredibly kind and patient in answering questions and concerns during the whole process. I am thankful for his honest insights and academic views in relation to the content in and surrounding my project topic.

Additionally I would like to thank my parents and friends for all their support throughout my last year of university, as well as all the previous years leading up to present.

Abstract

The research field of animal venom proteins carry a vast array of beneficial utilities, from medical applications, to genetic research, and evolutionary research. New animal species are constantly being discovered. Their biological information is often analysed to discover more about the species themselves, related species, and for new scientific development. Techniques and software used to identify biological data; amino acid sequences, specifically in this case, are an essential tool for scientists to be able to carry out their research more effectively. The faster and more accurately they are able to identify toxigenic proteins, the faster and more reliable their results and conclusions from carried research will be.

A variety of different machine learning techniques from LSTMs (Long Short-Term Memory) neural networks, to RNNs (Recurrent Neural Networks) have been explored, often resulting in software and web tools to aid scientists in their work.

This project investigates the use of CNNs (Convolutional Neural Networks) in the application of classifying animal amino acid sequences into either a ‘toxic’ or ‘atoxic’ class. The resulting CNN model is a small, but new contribution to the field of bioinformatics.

Table of Contents

Table of Figures

Acronyms

ARGM	Asymmetric Relative Minimal Generalisation
BLAST	Basic Local Alignment Search Tool
CNN	Convolutional Neural Network
CO_2	Carbon Dioxide
DNA	Deoxyribonucleic Acid
FASTA	FAST-All
GLM	Generalised Linear Models
GRU	Gated Recurrent Unit
HMM	Hidden Markov Models
HMMER	Hidden Markov Modelling
ILP	InductiveLogic Programming
LSTM	Long Short-Term Memory
MaxEnt	Maximum Entropy
NN	Neural Network
ReLU	Rectified Linear Unit
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
UniProtKB	UniProt Knowledgebase

Key Terms

Deep neural network	A neural network that has multiple hidden layers
Peptide	A biological molecule consisting of 2 or more amino acids

1 - Introduction

1.1 Problem Background

Many animals such as snakes, scorpions and spiders produce venoms as part of their survival mechanism towards predators and prey. These venoms have the ability of targeting the localised region of injection, as well as the nervous system and cardiovascular system. However, research into the components within deadly venoms have shown to have promising use in medicine. The same substances that cause the nervous system to malfunction, are able to make painkillers.

Using venoms for medicinal purposes is a practice that has been used by many ancient civilisations. Only recently there has been more effective development centered around creating targeted drugs from animal venom proteins, with currently around 20 different medications in approved use [1].

This is where machine learning comes in. By investing in the research to train machine learning models to classify animal venom proteins, more can be discovered about the biological compounds, the venomous animals themselves and most importantly it can aid in the medical development of useful targeted drugs for current health problems.

1.2 Project Description

This final year project endeavours to provide, primarily a Convolutional Neural Network (CNN) machine learning model approach to contribute to the research of classifying animal venom proteins.

The project will explain the biological aspects involved necessary for understanding the protein data that is analysed, as well as explaining the machine learning concepts of current research into this scientific area, and potential models that could be used for further development into the topic.

In addition, the project will demonstrate thoroughly the data pre-processing, model creation, model training, and analysis of the results obtained. Furthermore, evaluation of the results, limitations of the model, and comparisons to pre-existing models will be executed.

1.3 Project Aims and Objectives

The overall aim of the project is to explore and develop machine learning techniques to classify animal toxigenic proteins.

The main goals are outlined below:

1. Research, understand and summarise the necessary biological aspects in detail of animal venom proteins.
2. Research, understand and summarise machine learning models and techniques with current and potential use into solving the designated animal venom classification problem.
3. Understand in depth the current work done on the classification of animal venom proteins from a machine learning perspective.
4. Design, implement and improve a CNN machine learning model to classify animal venom proteins.
5. Critically evaluate the CNN model created with a focus on how it could be improved or done differently.
6. Critically evaluate the CNN model created with a focus on comparing it to current other existing machine learning models for classifying animal venom proteins in an attempt to contribute to scientific research.

1.4 Structure of the Report

Section	Description
1 - Introduction	This section describes the background of the topic, and the aims and objectives of the project.
2 - Literature Review	This section describes and explains in depth both the biological side and the machine learning side of the project. Touching on topics related to animal venom proteins, SVM's and Neural Networks.
3 - Problem Analysis & Data Design Choices	This section describes the reasoning behind strategies made for the project, the data that is used, and discussions on the data processing.
4 - Data Analysis	This section visualises the data processed in the previous section.
5 - Designing & Creating the CNN Model	This section describes and explains in depth the CNN model used, and the reasoning behind its particular architecture.
6 - Testing & Results	This section describes how testing of the created models are completed and talks about the results acquired from the model.
7 - Project Evaluation	This section evaluates the whole project, comparing with pre-existing implementations that took use of different machine learning methods.
8 - Future Developments	This section talks about potential future development and work in the field.
9 - Conclusion	This section is the overall conclusion of the project, also shedding some light on potential future works related to the research area.
10 - Statement of Ethics	This section describes and discusses in depth, the legal, ethical and social aspects of the project.

2 - Literature Review

2.1 - Animal Venom Proteins

2.1.1 - What Are Animal Venom Proteins?

The difference between the term toxins (poisons) and venoms lies in the method of delivery of such poisonous compounds, as opposed to its chemical structure.

The term ‘toxin’ refers to a singular compound with a specific binding site, which are usually passively encountered or ingested by another organism. The animal producing the toxin does not have a delivery mechanism to transfer these to another creature.

The term ‘venom’ refers to a blend of compounds, such as a combination of proteins, peptides and inorganic ions, that are able to target and affect various numbers of tissues. Venoms are poisonous substances that are composed and stored in specialised structures within the animal, such as venom ducts in snakes and serpents. They are associated with having the ability to deliver the poisonous compound to another organism, such as a Rattlesnake’s fang, or a scorpion’s tail, without having the recipient’s active participation [2].

2.1.2 - Uses of Venom Proteins

Venom proteins have demonstrated to have positive pharmacological effects on human diseases and disorders from anticancer activity, to neurodegenerative, analgesic, autoimmune and cardiovascular diseases [3].

For example, studies have shown that Alzeihemer’s disease has correlation with the solubility and quantity of one of the components in amyloid plaques called ‘amyloid β-protein’ (Aβ) [4]. Research has shown that metalloprotease from snake venoms can degrade Aβ in animal models [3]. Further development on the potential of this in humans are being undertaken.

The advantages of having natural venom peptides acting as drugs are that they have a high activity, high specificity, high biological diversity and can target a wide variety of areas. Some disadvantages include short serum half-life, potential immunogenicity, low membrane permeability and conformational instability during storage and transportation [5].

2.1.3 - Protein Structure

All multicellular organisms are composed of 2 types of nucleic acids: deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). Both of these are made by smaller components called nucleotides, which are composed of three smaller sections: a phosphate ion, a pentose sugar and a nitrogenous organic base.

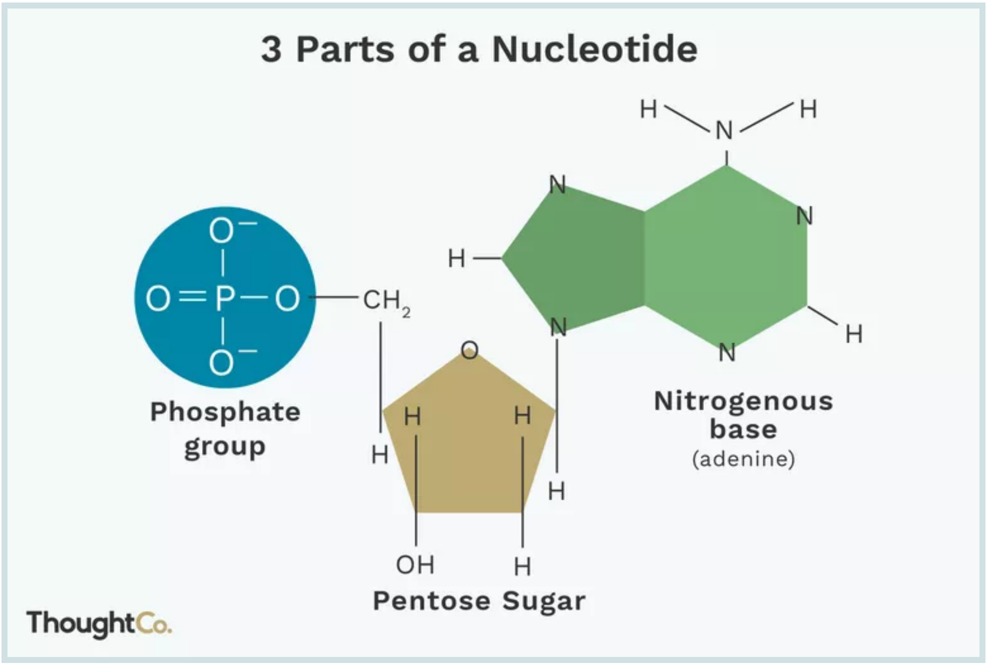


Figure 1 *- Three parts of a nucleotide [6]*

RNA is composed of only one strand of polynucleotides, allowing it to conclude various unique shapes, such as forming base pairs with itself and fold into numerous three dimensional shapes. It has three different versions of itself: messenger RNA (mRNA), ribosomal RNA (rRNA), and transfer RNA (tRNA), which all play a role in protein synthesis.

DNA is constructed by millions of the nucleotide units combining together via their phosphate ion groups and pentose sugar groups to form a polynucleotide chain and more importantly a ‘sugar phosphate backbone’ which protects the genetic information from damage. The double helix shape of the DNA is composed of two of these polynucleotide chains maintained by hydrogen bonds between the complementary organic nitrogenous bases. This results in DNA being able to undergo replication so each cell in the body will have the same genetic information inside its nucleus.

DNA carries the instructions for protein synthesis. It states what order the amino acids should be connected to one another to create polypeptide chains - this is known as the primary structure of a protein. These will then further fold into structures known as ‘alpha helices’ and ‘beta pleated sheets’ which make up the secondary protein structure. Tertiary protein structure occurs when the alpha helices and beta pleated sheets fold into three-dimensional structures. The final quaternary protein structure occurs when multiple three-dimensional structures of polypeptide chains come together to form macromolecules [7].

2.2 - Atchley Factors

Atchley factors are high dimensional data on the analysis of 494 attributes and properties of amino acids. A substantial issue in biological sequence data is called the “sequence metric problem”. This refers to using alphabetic letter codes, such as the latin alphabet to represent and identify different elements of a certain sequence.

Though inscribing letters or symbols to represent elements within a sequence offers simplicity, it also causes other issues to arise. Especially with regards to being able to compare the real world attributes and features of each sequence element with each other. For example, the amino acid leucine (L) is more similar to amino acid valine (V), as opposed to amino acid alanine (A), in regards to their physicochemistry, even though the “alphabetic distance” may subconsciously imply us otherwise. This method does not provide enough resolution for comparing complex substances such as amino acids.

Therefore, to solve this problem, a multivariate statistical analysis of a large number of amino acid attributes was proposed to be able to represent and gain further insight into the correlation patterns of properties between amino acids.

Of the total 494 amino acid attributes measured, a subset of 54 attributes were selected based on statistical distributional properties, relative ease of interpretation, relative magnitude of factor coefficient and structure importance.
Further analysis of these 54 attributes via factor analysis resulted in the 5 clusters shown below [8]:

1. Polarity	- Bonded energy versus non-bonded energy - Number of hydrogen bond donors - Polarity versus non-polarity - Hydrophilicity versus hydrophobicity
2. Secondary structure	The inverse relationship of relative inclination towards folding into a particular secondary structure configuration, such as a coil, or α-helix.
3. Molecular volume	Molecular size, volume, with factor coefficients for bulkiness, residual volume, average volume of a buried residue, side chain volume and molecular weight.
4. Relative composition	The number of codons coding for an amino acid, and amino acid composition.
5. Electrostatic charge	High coefficients on isoelectric point and net charge.

This study was able to not only provide extensive data into various dimensions of protein features, but also opened other possibilities of presenting the data in a more visual manner that ignites further understanding and insights into the correlation and association of amino acids.

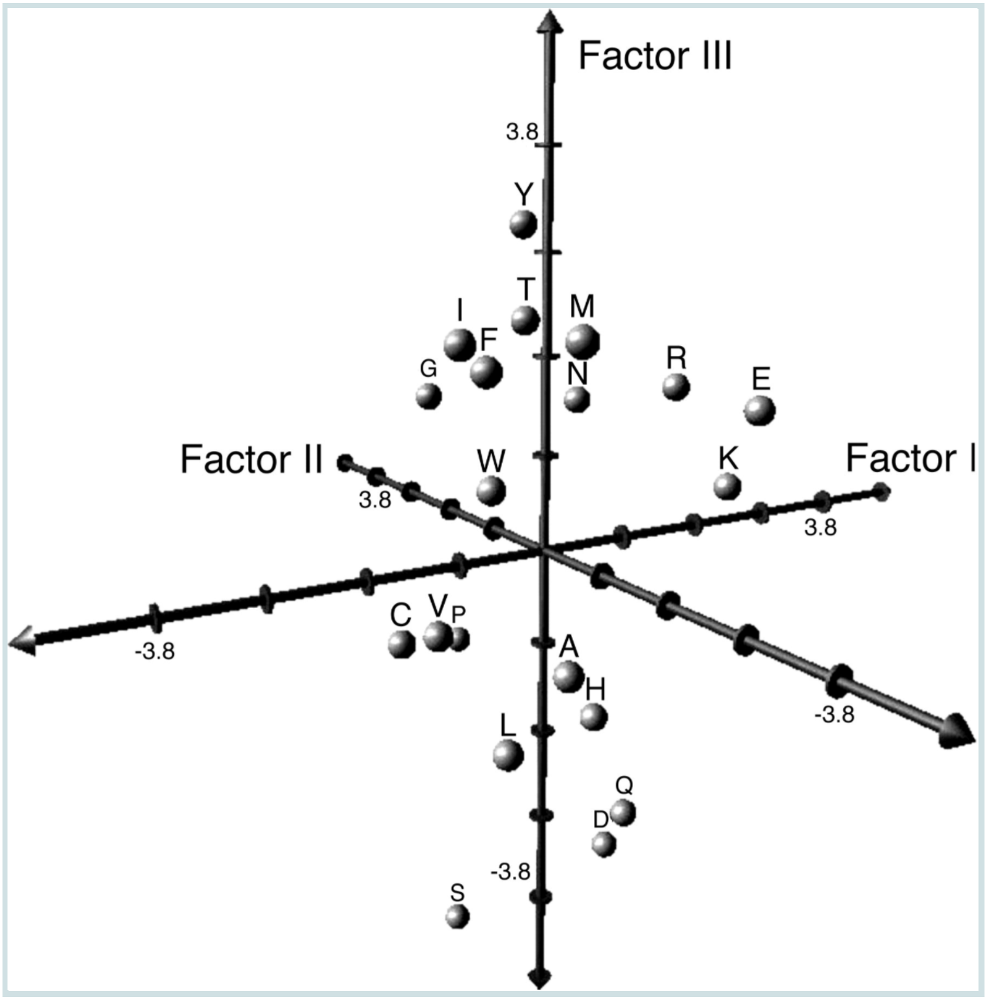


Figure 2 *- Spatial representational plot of scores of Atchley factors 1 to 3 for 20 amino acids [8]*

Overall, Atchley factors offer a simple way to quantify complex attributes and properties of amino acid sequence data, as well as providing a good solution to the “sequence metric problem”.

2.3 - Machine Learning Models and Technologies

This subsection describes and explains in depth the current machine learning models used to tackle venom protein classification, and potential other methods and models that could be used.

2.3.1 - Basic Neural Network Architecture

The basic idea of an artificial neural network is that it is modelled after a biological neural network in the brain. Neurons are interconnected with each other to form a network. There are variants to the types of neural networks, some of which include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and AutoEncoders.

2.3.1.1 - Neuron

Neurons are the most basic building blocks of the architecture of neural networks. Similar to a biological neuron, it takes an input, and delivers an output. In machine learning, this translates to the neuron taking an input, applying a mathematical function to it, and releasing the result as an output.

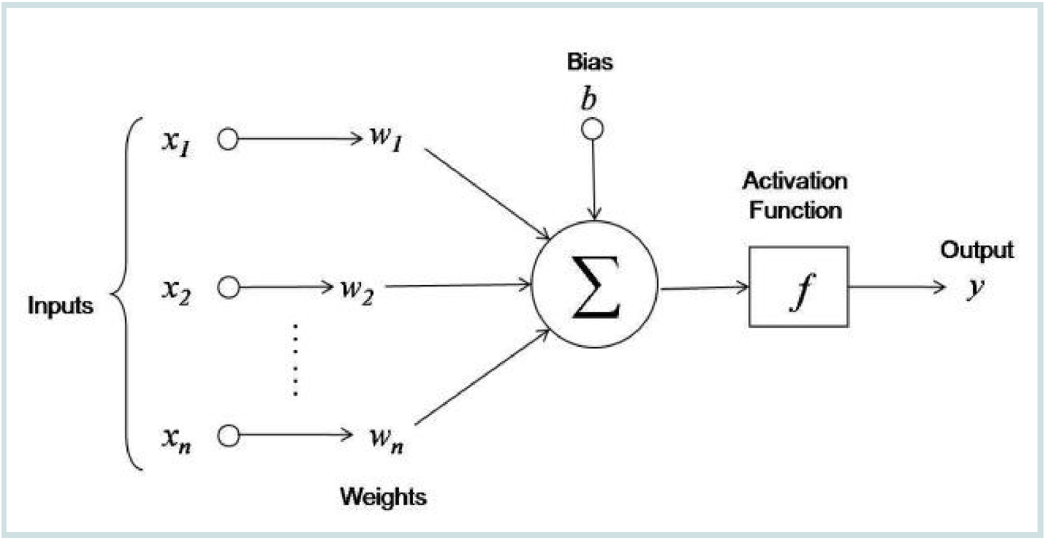


Figure 3 *- Operations done by an artificial neuron [9]*

A variable referred to as 'weight' usually represented as *'w'*, determines the connection between two neurons, and is the only value that is changed during the learning process of the neural network. The weight value is multiplied by the input value *'x'* coming into the neuron. A bias *'b'* is then added to this value. The bias is not related to the neuron, or the weights, but it's instead chosen before learning occurs as it and the activation function *'f(x)'* is applied to the result, before passing the output value *'y'* to the next layer [9].

2.3.1.2 - Bias
Bias is an additional parameter in a neural network. It mirrors the intercept in a linear equation. For example the intercept ‘c’ in ‘y = mx + c’. In a neural network it is used to adjust the output by adding a constant value to the weighted sum of inputs of the neuron. They are therefore not interconnected between layers [10].

2.3.1.3 - Activation Functions
Activation functions are used to compress or normalise the output values from neurons between a specific range, such as between 0 and 1 before sending the value to the next layer.Depending on the nature of data and learning required, different kinds of activation functions will work more effectively for different kinds of tasks. Examples of widely activation functions are: sigmoid, tahn, ReLU (Rectified Linear Unit), and SoftMax [11].

Sigmoid functions have an output range of [0,1]. When the input value tends to negative infinity, the output value is 0. When the input value tends to infinity, the output is 1.
We use this function because it captures non-linearity in the data, which is necessary for the modelling process. It can be used with gradient descent and backpropagation since it is differentiable, and it is usually used for problems that require binary classification.
However, sigmoid functions, due to its small output range of [0,1] cause the problem of vanishing gradients [11]. The graph of a sigmoid function is shown below:

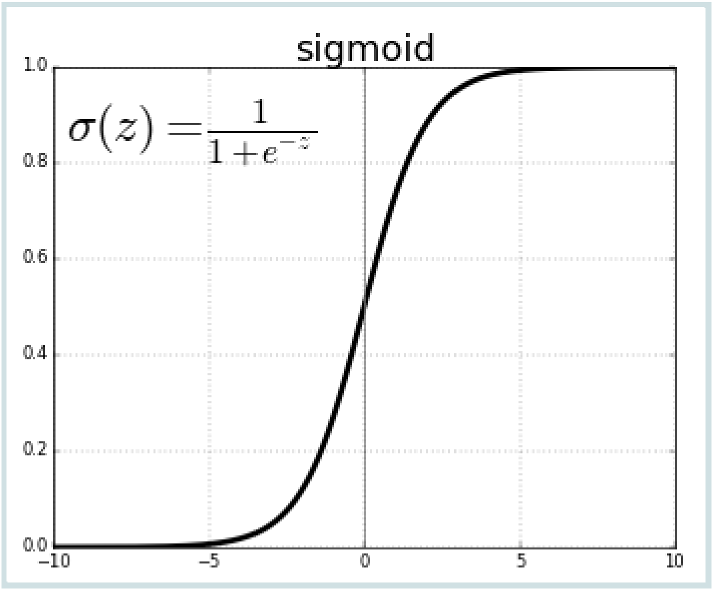


Figure 3 *- Sigmoid activation function [11]*

The Tahn function is a re-scaled version of sigmoid, where the output range is [-1,1] instead of [0,1]. This is useful to make the derivatives higher when working out the gradient, since the data is centered around 0, which can improve the learning rate of the model. This in shown in the graph below:

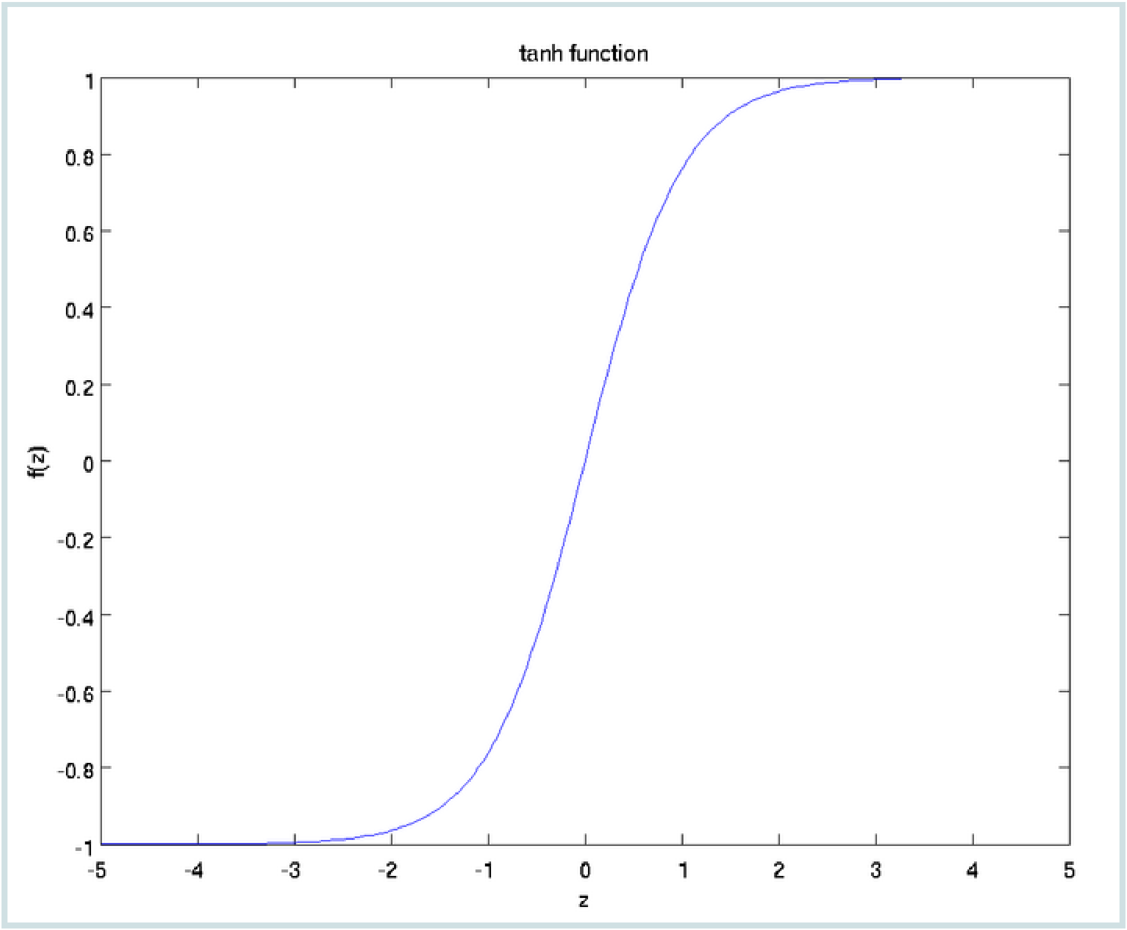


Figure 4 *- Tanh activation function [11]*

Softmax, also known as Maximum Entropy (MaxEnt) Classifier, is a logistic regression function that normalises inputs into a vector of values in which its probability distribution sums to 1. It outputs values with a range of [0,1], meaning it can accomodate for multi-class classification problems. This function is usually applied to the output layer of image classification neural networks. [12]

ReLU is commonly used in deep learning network models. If the input is negative, the function will return 0, but it will return the actual input value if it's positive. This is shown in the graph below show by the function R(z) = max(0, z):

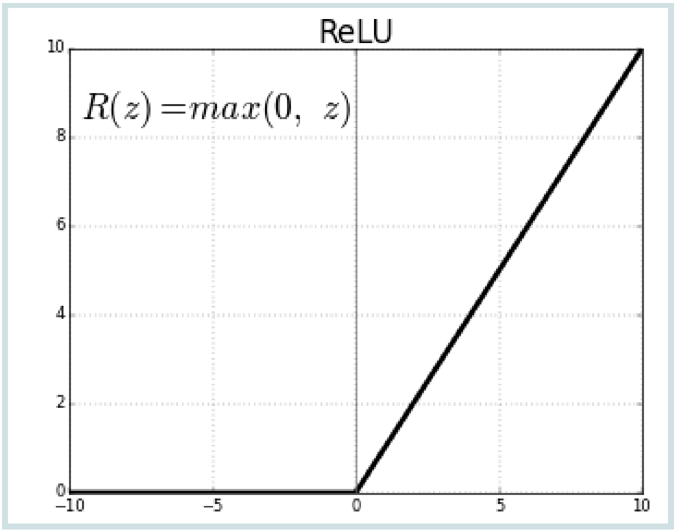


Figure 5 *- ReLU activation function [11]*

2.3.1.4 - Neural Networks

Traditionally an artificial neural network has 3 different types of layers: input layer, hidden layer, and output layer. A layer has a collection of neurons with attached weights. Each layer outputs a representation of the current data extracted and feeds it as input into the next layer.
In the 'input' layer, the neural network takes inputs directed from the usually pre-processed dataset. The 'output' layer is the last layer in the network, containing all the possible outputs.

The 'hidden' layer holds this name since its values are not observed in the training set. There can be multiple hidden layers in a neural network, and if this is the case, it is referred to as a 'deep neural network'. Each hidden layer may each have different activation functions depending on the problem the model is trying to solve.

2.3.2 - Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are similar to regular Neural Networks (NNs), however they are able to remember things learnt from previous inputs whilst at the same time generating outputs. These networks have been found to result in an effective way of leveraging between input items and surrounding neighbours.

In addition to RNNs being influenced by usual parameters such as weights applied to the input values in their neurons, they also have a 'hidden' state vector which is representative of the prior inputs and outputs [13].

2.3.3 - Convolutional Neural Networks

Convolutional neural networks or ‘ConvNet’s (CNNs) are deep learning algorithms that are renowned for their ability to learn spatial and temporal information, and hence is a method that is heavily applied in image classification problems.

The algorithm takes in an input image, assigns weights and biases to aspects of the image itself, which is how they have the ability to learn characteristics and filters over time, by firstly reducing the images into a form that's more feasibly processed, but without the loss of features to still achieve a high predictive accuracy in the end [14].

The traditional CNN input is 2-dimensional, ie. a 2D matrix, but can be changed to 1-dimensional inputs such as sequential data, allowing it to develop an internal representation of sequential relationships. Inputs can also expand higher beyond 3-dimensional tensors, meaning that CNNs offer good solutions to problems that require high-dimensionality modelling.

CNNs have a series of different layers used in their algorithms, which are composed of ‘convolutional layers’, ‘pooling’ layers and ‘fully connected’ layers.

In convolutional layers, the neural network essentially assigns a filter, of a smaller size than the image, for example, a 5 x 5 kernel, which gradually moves across the whole image, outputting a summary of firstly, the low-level features, such as colour and gradient orientation, of that particular image. This output then becomes the input into the next layer, which can be another convolution layer, as opposed to something different. The further along into the neural network, the higher the level of features of the image will be extracted, which overall gives an encompassed understanding of the images within the dataset.

In pooling layers, the spatial size of the convoluted features are reduced. This is responsible for one of the highest advantages of using CNNs, which is that they are fast at processing data, since they are able to decrease the amount of computational power necessary to process the dataset in question, via dimensionality reduction throughout its layers. There are two types of pooling layers: ‘Max pooling’, which returns the maximum value of the parts of the image covered by a single kernel, and ‘Average pooling’, which returns the average of all the values in question within the image kernel.

Fully connected layers are usually placed closer to the output layer, towards the final layers of the CNN. It is able to learn non-linear combinations of the high-level features provided by the previous layers in the network.

2.3.4 - CNN Case Study: LeNet-5

A very acclaimed and distinguished architecture for a CNN is ‘LeNet-5’. It was one of the first few CNNs to be developed and trained, in the year of 1994. The people involved in its creation were Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner [15].

The neural network itself comprises of a total of 7 layers (not including the input later) [16]:

1. Convolution - 2-dimensional
2. SubSampling/Pooling - average pooling
3. Convolutional - 2-dimensional
4. SubSampling/Pooling- average pooling
5. Fully connected
6. Fully connected
7. Output

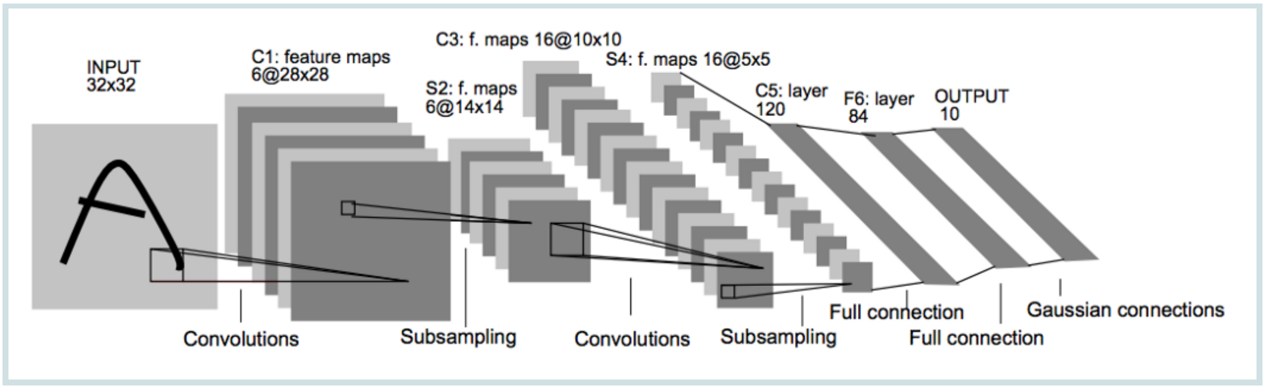


Figure 7 *- LeNet-5 architecture [15]*

This model, historically has been one of the highest achieving for classifying image data, therefore I decided to model my CNN in a similar manner to LeNet-5, especially because it was used for grayscale images with the MNIST dataset for handwritten digits.

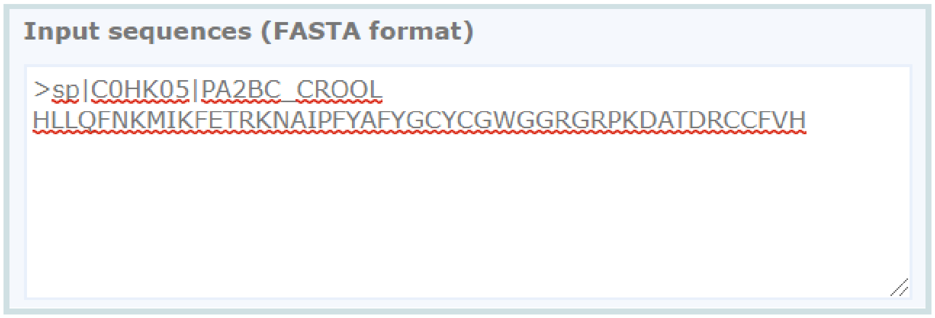
2.4 - Current Work in the Field of Animal Toxin Classification

2.4.1 - ClanTox: A classifier of short animal toxins

ClanTox is a toxic animal protein classifications software, specifically targeted towards short amino acid sequences. Its predictions result in the classification of ‘toxin or toxin-like proteins’, grading them with a mean score from -1 if they are not toxic, to +1 if they are found to be toxic. However, they only consider a prediction to be toxic if the mean is significantly greater than the standard deviation, to reduce any possible biases caused by false instances in the dataset [17].

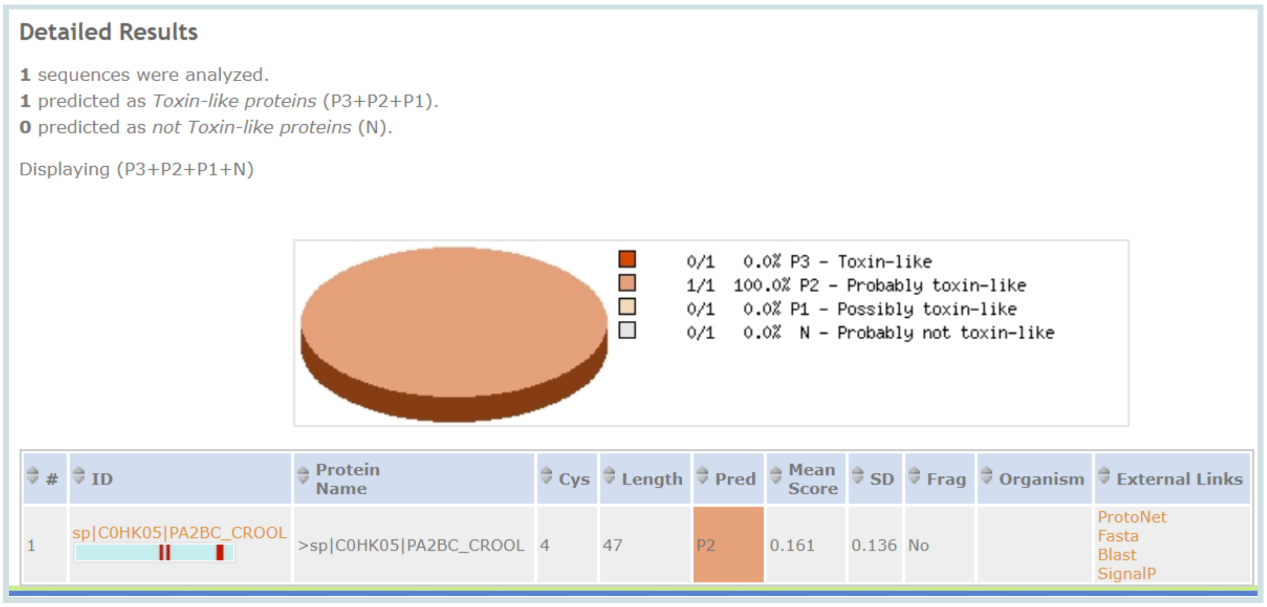
They have chosen to use a conjunction of 10 classification methods, in which their outputs were normalised to the highest positive prediction of each classifier to result in a ‘meta-classifier’, which is what they used for their online protein classification prediction tool [18].

2.4.1.1 - ClanTox online protein classifying tool



```
>sp|C0HK05|PA2BC_CROOL
HLLQFNKMIKFETRKNAIPFYAFYGCYCGWGGRGRPKDATDRCCFVH
```

Figure 8 *- Inputting a protein sequence to be analyzed by ClanTox [17]*



Detailed Results

1 sequences were analyzed.
1 predicted as *Toxin-like proteins* (P3+P2+P1).
0 predicted as *not Toxin-like proteins* (N).

Displaying (P3+P2+P1+N)

🟥	0/1	0.0%	P3 - Toxin-like
🟧	1/1	100.0%	P2 - Probably toxin-like
🟨	0/1	0.0%	P1 - Possibly toxin-like
⬜	0/1	0.0%	N - Probably not toxin-like

#	ID	Protein Name	Cys	Length	Pred	Mean Score	SD	Frag	Organism	External Links
1	sp|C0HK05|PA2BC_CROOL	>sp|C0HK05|PA2BC_CROOL	4	47	P2	0.161	0.136	No		ProtoNet Fasta Blast SignalP

Figure 9 *- ClanTox analysis output of the protein sequence [17]*

The classifier outputs 4 labels: N for negation toxic prediction, and P1 to P3, representing three different levels of positive prediction.

Overall their study provided significant insights on the development of a systematic scheme for toxic protein prediction, as well as revealing that some toxin types have a tendency for a particular unique prediction level (P1 - P3). For example, toxins from Viper snakes were found to be primarily detected at level P2, whereas toxins from Elapid snakes were primarily detected at level P1 [18].

2.4.2 - ToxClassifier: A Support Vector Machine (SVM) approach

“ToxClassifier” is a toxic amino acid prediction tool which uses various machine learning classifiers to evaluate and come to a final conclusion about the toxicity of a protein sequence, based on a sum of predictions of the chosen classifiers [19].

Their approach included Support Vector Machines (SVMs) and Generalized Linear Models (GLMs), which are an approach to merge together a variety of mathematical statistical methods such as Poisson Regression, Binary Logistic Regression, and the widely used Linear Regression [20].

2.4.2.1 - A note on SVMS

SVMs are a set of supervised learning methods used for classification, regression and outliers detection [21]. Their end goal is to find the number of features that distinctly classify data points.

Decision boundaries called “hyperplanes” are used to aid in the classification of data points. Since data can have multiple dimensions, hyper planes are able to be of higher dimensions than simply a 2D straight line [22].

Support vectors themselves are data points that influence the position and orientation of the hyperplane. They are the points which end up being the closest to the hyperplanes. By choosing different support vectors, the position of the hyperplane will be changed.

Data found placed on different sides of the hyperplane are able to be considered to pertain to different classes. Shown below, the green data points would be a different class compared to the red data points.

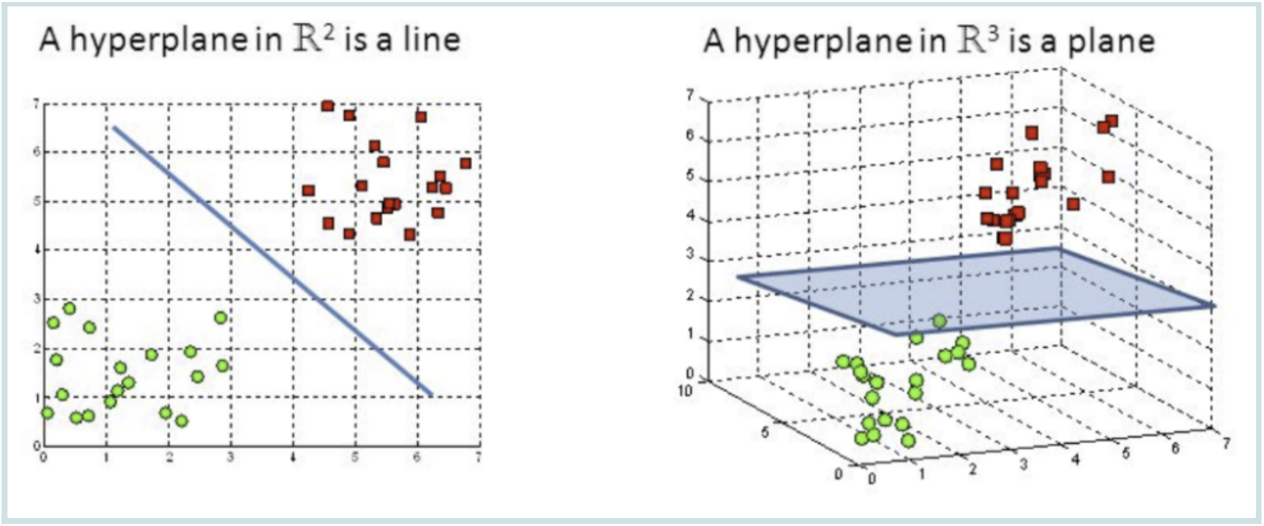


***Figure 10** - Hyperplanes in 2D and 3D feature spaces [22]*

Overall, SMVs are found to be able to produce significant accuracy with lower computational power.

ToxClassifer used both SVMs and GLMs in conjunction with Basic Local Alignment Search Tool (BLAST) and Hidden Markov Modelling (HMMER) methods to annotate toxins, which they further tested for accuracy, developed and calibrated.

2.4.2.2 - A note on BLAST

BLAST (Basic Local Alignment Search Tool) is an algorithm that allows for the comparison of primary biological sequence information, ie. amino acid chains, named ‘polypeptide chains’, as opposed to secondary structure sequences such as DNA and RNA sequences [23].

It is one of the most widely used tools in the field of bioinformatics for searching biological sequence data. It takes the approach of searching for only the most significant patterns in protein sequences, and it’s often used in conjunction with other searching algorithms.

2.4.2.3 - A note on HMMER

HMMER (Hidden Markov Modelling) is a software designed for similarity searches of protein sequences with the use of probability methods. It has developed from a fully Unix command line application to a more widely available web application, that is able to return most protein databases in a few seconds [24].

It utilises Hidden Markov Models to apply statistical functions for profiling proteins, searching protein sequences, searching DNA and RNA queries, and aligning protein sequences with an already identified profile [25].

2.4.2.4 - A note on Hidden Markov Models

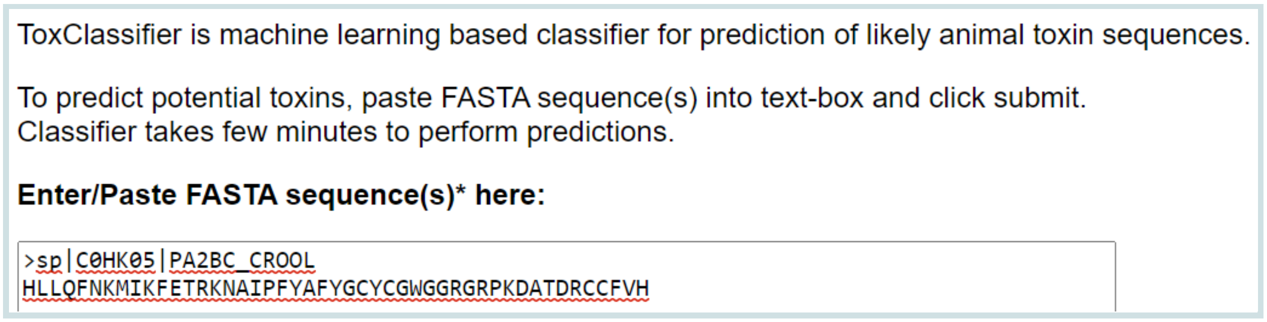
Hidden Markov models are a mathematical approach primarily used for solving sequential data problems. They are able to be applied to machine learning in both an supervised and unsupervised way.

They produce ‘rules’ which include two probabilities. One which relates to the fact that there will be a ‘certain observation’, and another in which relates to the fact that there will be a ‘certain state of transition, given the state of the model at a certain time’ [26].

2.4.2.5 - ToxClassifier software tool

They were able to provide both an online and offline tool, where a person is able to enter or paste a FASTA (sequence, upon which the result of it being toxic is shown.

Upon trying the online tool with a protein sequence used in the dataset for this project:



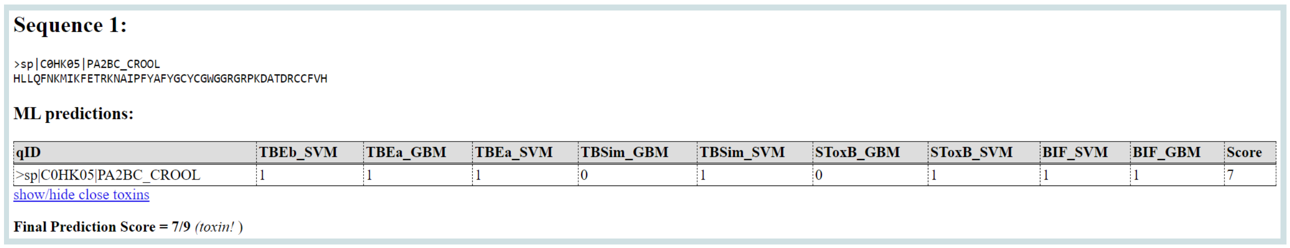
ToxClassifier is machine learning based classifier for prediction of likely animal toxin sequences.

To predict potential toxins, paste FASTA sequence(s) into text-box and click submit.
Classifier takes few minutes to perform predictions.

Enter/Paste FASTA sequence(s)* here:

```
>sp|C0HK05|PA2BC_CROOL
HLLQFNKMIKFETRKNAIPFYAFYGCYCGWGGRGRPKDATDRCCFVH
```

Figure 11 *- Pasting a FASTA sequence in ToxClassifier, version 1.0 [27]*



Sequence 1:

```
>sp|C0HK05|PA2BC_CROOL
HLLQFNKMIKFETRKNAIPFYAFYGCYCGWGGRGRPKDATDRCCFVH
```

ML predictions:

qID	TBEb_SVM	TBEa_GBM	TBEa_SVM	TBSim_GBM	TBSim_SVM	SToxB_GBM	SToxB_SVM	BIF_SVM	BIF_GBM	Score
>sp|C0HK05|PA2BC_CROOL	1	1	1	0	1	0	1	1	1	7

show/hide close toxins

Final Prediction Score = 7/9 *(toxin!)*

Figure 12 *- Results from pasting a FASTA amino acid sequence in ToxClassifier, 1.0 [27]*

2.4.3 - TOXIFY: An RNN Approach

"TOXIFY" is a software package that offers a deep learning machine learning approach to classifying animal venom proteins. It enables users to input a protein sequence, of which the output is the inferred probability of that particular sequence being either venomous or non-venomous [28].

Their methods included encoding proteins as numerical representations, by inscribing a 5x500 matrix with values corresponding to five Atchley factors (described in section 2.2) per amino acid in each protein.

RNNs with Gated Recurrent Units (GRUs) were used to classify the amino acid sequences, since they are able to specify hidden states that depend both on input as well as the previous hidden state. These are a gating mechanism for RNNs that behaves similarly, but at a higher speed compared to 'Long Short-Term Memory (LSTM). It contains a forget gate, but lacks an output gate [29].

This paper proposed the first animal toxin classifier that utilises deep learning, as opposed to other classifiers that used 'BLAST-based methods'.

Results show that TOXIFY had a poorer specificity (false positive rate) when compared to other methods. However, overall it achieves better accuracy, alongside better memory efficiency and training speed.

3 - Problem Analysis & Data Design Choices

Some more prominent limitations which I become aware of after concluding the literature review, include how the datasets used by TOXIFY, and ToxClassifier in particular, do not consider toxin generalisation over different species. The datasets are a single agglomeration of all toxic species together, and do not consider each species separately.

Limitations of this potentially may include lack of insight of what makes a particular species of animals contain toxins, especially since, by definition a 'species' refers to a group of animal populations which are freely able to interbreed among themselves. If species are not considered separately in different datasets, observations in regards to how venoms develop over a particular species evolution is likely to be overlooked.

On the other hand, generalising toxin by species, as the case studies have done, do provide a more general overview on what perhaps makes an amino acid sequence toxic.

Furthermore, another factor that has not been considered by the case studies, is the magnitude of strength of the toxins in each amino acid sequence. There are no labels included to differentiate between mildly toxic, and highly toxic protein sequences.

After reviewing the approaches of the most recent studies: 'Clantox', 'ToxClassifier' and 'TOXIFY' on classifying protein sequences I made the decision to further investigate a hybrid method to the classification of primary protein sequences. Using Atchley values as a tool to further investigate how the properties and attributes of each amino acid sequence relate to each other, I will be exploring and training a CNN to classify animal venom proteins, since this is a machine learning algorithm that has not been previously used within this problem scope.

I have decided to focus on the dataset used by 'Clantox', 'ToxClassifer' and 'TOXIFY' to be able to fairly compare results.

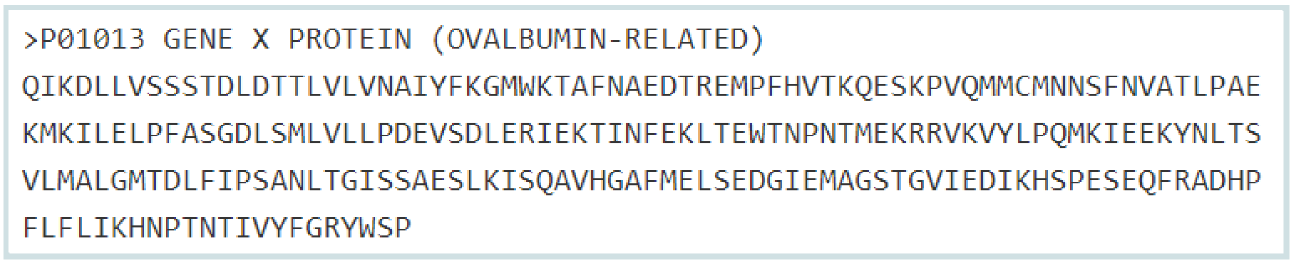
3.1 - Data structure

Due to the nature of CNNs, the data has to be fed into the neural network within a specific format. Therefore, a series of steps that involve data parsing, data cleaning and data processing had to be completed prior to feeding it into and training the model.

3.1.1 - FASTA sequence structure

The data for these models are processed in a 'FASTA' format. FASTA is a DNA and protein sequence alignment software package described by David J. Lipman and William R. Pearson. It stands for "FAST-All" as it works in any alphabet. Specifically, since the project is dealing with protein sequences, the extension of 'FAST-P' (protein) alignment tool will be used.

A FASTA sequence is split into 2 parts: The description line (defline), and the sequence data. The defline uses a greater-than symbol ('>') at the beginning to differentiate it from the rest of the protein sequence. Blank lines are not allowed in the middle of the sequence part of the data [19].

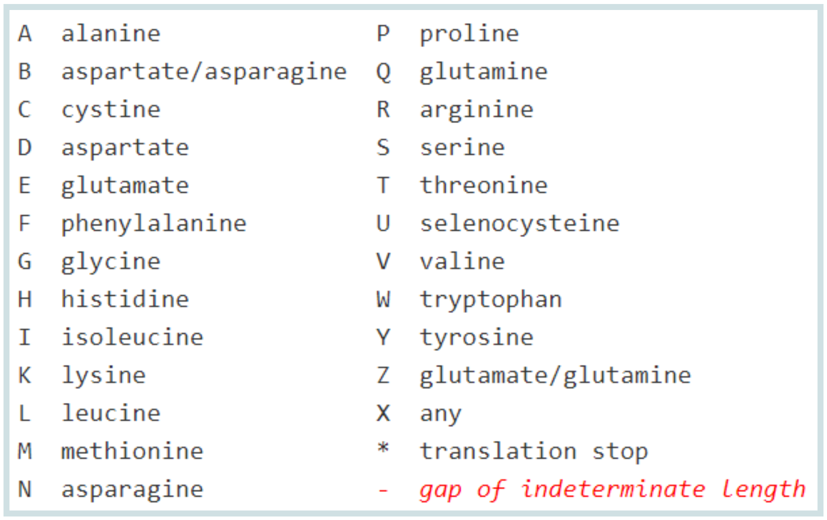


```
>P01013 GENE X PROTEIN (OVALBUMIN-RELATED)
QIKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNNSFNVATLPAE
KMKILELPFASGDLSMLVLLPDEVSDLERIEKTINFEKLTEWTNPNTMEKRRVKVYLPQMKIEEKYNLTS
VLMALGMTDLFIPSANLTGISSAESLKISQAVHGAFMELSEDGIEMAGSTGVIEDIKHSPESEQFRADHP
FLFLIKHNPTNTIVYFGRYWSP
```

Figure 13 *- A sample FASTA protein sequence [30]*

Sequences are presented in standard IUB/IUPAC (International Union of Biochemistry / International Union of Pure and Applied Chemistry with the exceptions that any lower-case letters are mapped into upper-case letters, and a dash ('-') is able to be used to represent a gap of intermediate length.

For amino acid sequences, the accepted codes are shown below:



A	alanine	P	proline
B	aspartate/asparagine	Q	glutamine
C	cystine	R	arginine
D	aspartate	S	serine
E	glutamate	T	threonine
F	phenylalanine	U	selenocysteine
G	glycine	V	valine
H	histidine	W	tryptophan
I	isoleucine	Y	tyrosine
K	lysine	Z	glutamate/glutamine
L	leucine	X	any
M	methionine	*	translation stop
N	asparagine	-	*gap of indeterminate length*

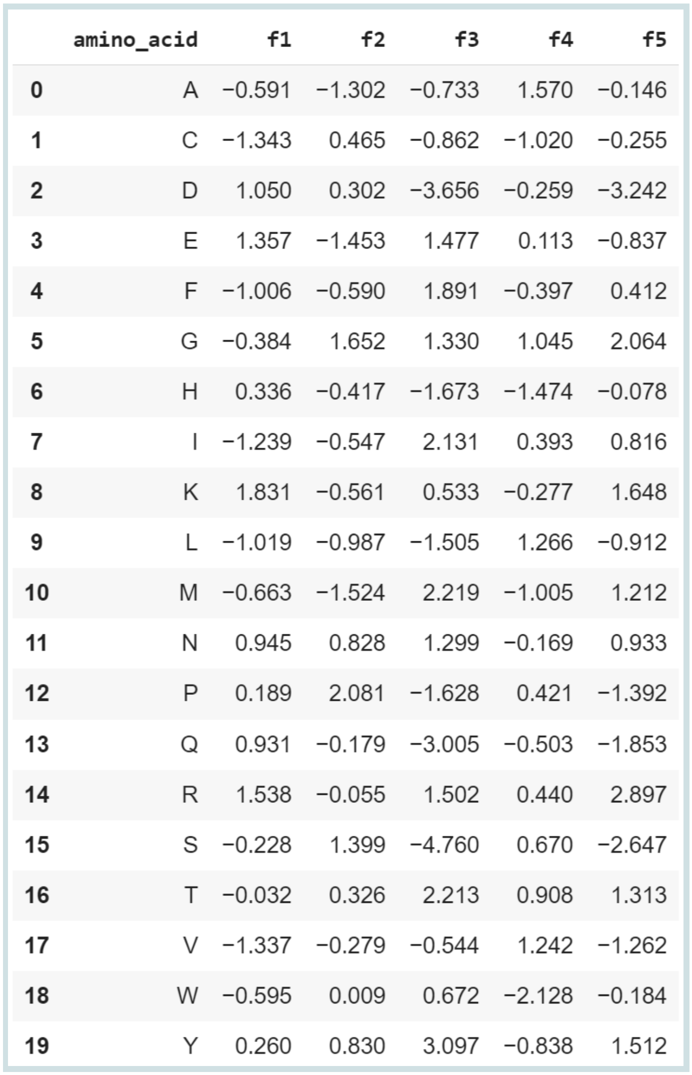
Figure 14 *- FASTA amino acid accepted codes [30]*

3.1.2 - Atchley Factors Values

I have decided to call each Atchley factors by the names below, with the letter ‘f’ standing for the word ‘factor’:

- f1 is ‘Polarity’
- f2 is ‘Secondary structure’
- f3 is ‘Molecular volume;
- f4 is ‘Relative composition’
- f5 is ‘Electrostatic charge’

Each amino acid base holds a value for each one of the five Atchley factors, as shown on the dataframe below:



	amino_acid	f1	f2	f3	f4	f5
0	A	−0.591	−1.302	−0.733	1.570	−0.146
1	C	−1.343	0.465	−0.862	−1.020	−0.255
2	D	1.050	0.302	−3.656	−0.259	−3.242
3	E	1.357	−1.453	1.477	0.113	−0.837
4	F	−1.006	−0.590	1.891	−0.397	0.412
5	G	−0.384	1.652	1.330	1.045	2.064
6	H	0.336	−0.417	−1.673	−1.474	−0.078
7	I	−1.239	−0.547	2.131	0.393	0.816
8	K	1.831	−0.561	0.533	−0.277	1.648
9	L	−1.019	−0.987	−1.505	1.266	−0.912
10	M	−0.663	−1.524	2.219	−1.005	1.212
11	N	0.945	0.828	1.299	−0.169	0.933
12	P	0.189	2.081	−1.628	0.421	−1.392
13	Q	0.931	−0.179	−3.005	−0.503	−1.853
14	R	1.538	−0.055	1.502	0.440	2.897
15	S	−0.228	1.399	−4.760	0.670	−2.647
16	T	−0.032	0.326	2.213	0.908	1.313
17	V	−1.337	−0.279	−0.544	1.242	−1.262
18	W	−0.595	0.009	0.672	−2.128	−0.184
19	Y	0.260	0.830	3.097	−0.838	1.512

Figure 15 *- Atchley Factors values for each amino acid base [31]*

I attained a .txt file containing the Atchley factors from a file in ‘vadimnazarov’s GitHub Repo [32].

I then created a Python script to parse these into a Pandas DataFrame shown above for simple visualisation purposes, and later converted into a dictionary for ease of use in other parts of the code.

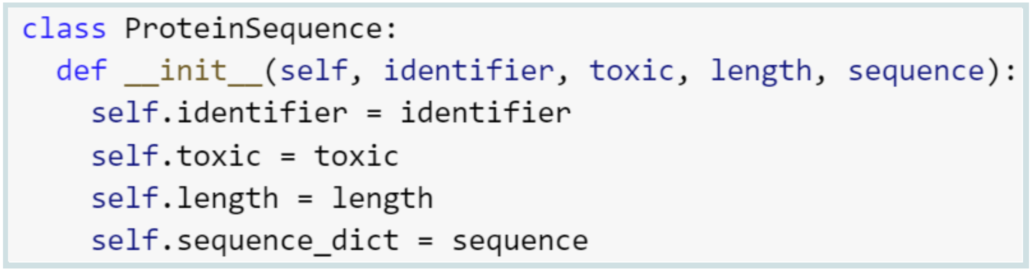
3.2 - Data Processing

3.2.1 - Fasta Sequences parsing

The dataset being used was acquired from TOXIFY’s github repository. The training data, and benchmark data as the training dataset. [33]

The term ‘toxic’ is used to describe venomous proteins, and the term ‘atoxic’ is used to describe non-venomous proteins when dealing with the data to facilitate shorter variable names.

Each .fasta file with protein sequences was parsed into a ‘ProteinSequence’ object for their corresponding dataset:

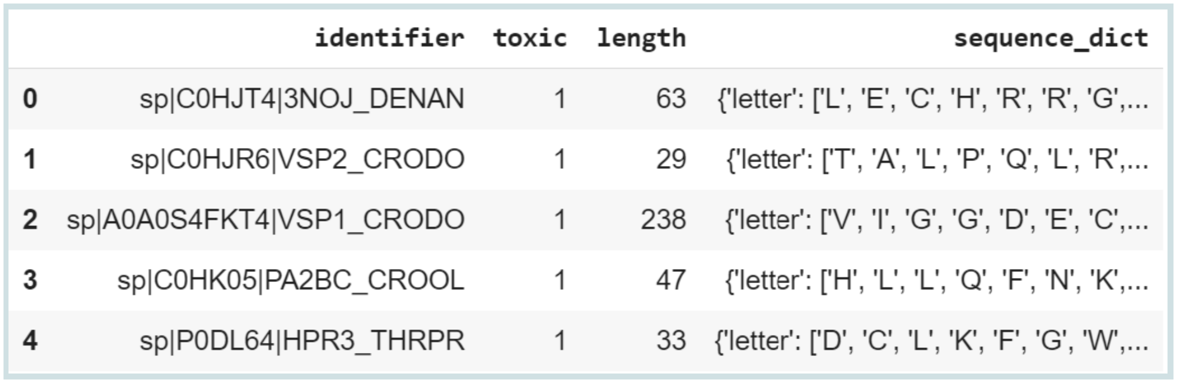


```
class ProteinSequence:
  def __init__(self, identifier, toxic, length, sequence):
    self.identifier = identifier
    self.toxic = toxic
    self.length = length
    self.sequence_dict = sequence
```

Figure 16 - *‘ProteinSequence’ class object*

The class variables being:

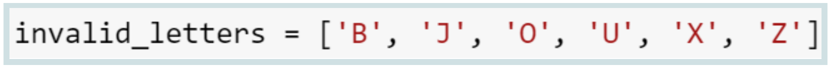
1. **‘identifier’**: (string) the description line (defline) of the sequence
2. **‘toxic’**: (int) binary classification of the toxicnicess of the protein. Value 1 representing ‘toxic’ and value 0 representing ‘atoxic’.
3. **‘length’**: (int) the total length of amino acids in that sequence
4. **‘sequence_dict’**: a dictionary of lists. After the initial parsing it contains one entry: ‘letter’, which is the letter representation of each amino acid in the sequence. After further processing the dictionary will be appended with each atchley values (‘f1’, ‘f2’, ‘f3’, ‘f4’ and ‘f5’) corresponding to each particular amino acid. After this the sequential change of each value is appended to the dictionary, overall containing 11 lists.



	identifier	toxic	length	sequence_dict
0	sp|C0HJT4|3NOJ_DENAN	1	63	{'letter': ['L', 'E', 'C', 'H', 'R', 'R', 'G',...
1	sp|C0HJR6|VSP2_CRODO	1	29	{'letter': ['T', 'A', 'L', 'P', 'Q', 'L', 'R',...
2	sp|A0A0S4FKT4|VSP1_CRODO	1	238	{'letter': ['V', 'I', 'G', 'G', 'D', 'E', 'C',...
3	sp|C0HK05|PA2BC_CROOL	1	47	{'letter': ['H', 'L', 'L', 'Q', 'F', 'N', 'K',...
4	sp|P0DL64|HPR3_THRPR	1	33	{'letter': ['D', 'C', 'L', 'K', 'F', 'G', 'W',...

Figure 17 *- Parsed .fasta file into ProteinSequence object (dataframe)*

Some protein sequences contained letters that were not included in the Atchley values:



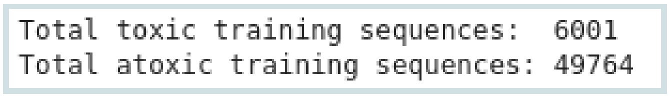
```
invalid_letters = ['B', 'J', 'O', 'U', 'X', 'Z']
```

Figure 18 *- Letters not in Atchley values, used for pre processing data*

The 'invalid_letter' list, holds the letters that are not used to represent amino acids. These are removed before being parsed into the 'ProteinSequence' object to make further processing of the data using Atchley values simpler.

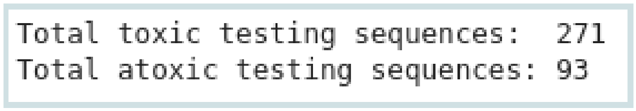
3.2.2 - Initial Data analysis

The training set contained in total 6,001 toxic protein sequences, and 49,764 atoxic protein sequences:



```
Total toxic training sequences:  6001
Total atoxic training sequences: 49764
```

Figure 19 *- Number of total protein sequences for each class in the training set*



```
Total toxic testing sequences:  271
Total atoxic testing sequences: 93
```

Figure 20 *- Number of total protein sequences for each class in the testing/benchmark set*

This is a significant difference between the classes of the training data that could potentially cause models some bias during training, and therefore affect results negatively. In addition, training on over 50,000 protein sequences would have required a large amount of computing power that I did not have at my disposal. Therefore I chose to downsample the toxic protein sequences to equal the number of toxic sequences.

However, firstly I needed to understand how the training data was distributed in regards to the number of amino acids in each sequence, so I created two distribution graphs to help visualise the lengths for both toxic and atoxic protein sequences:

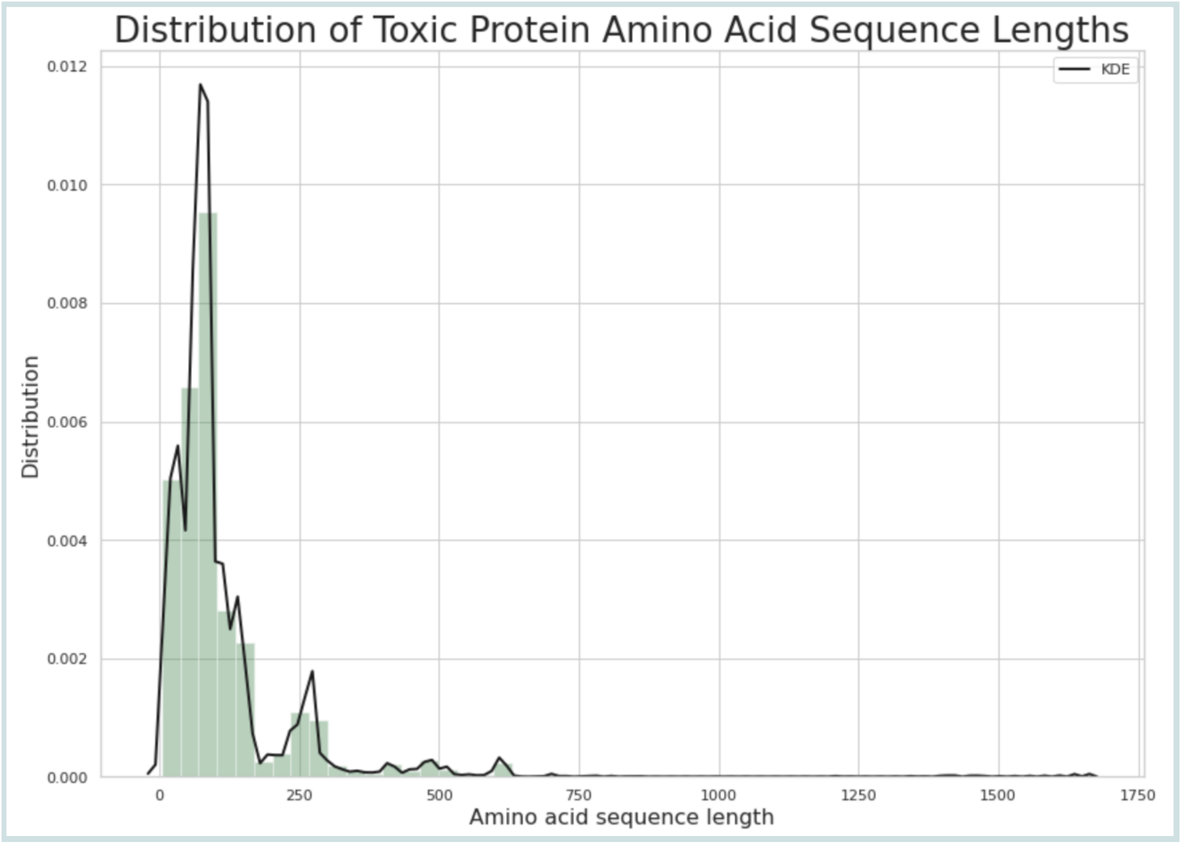


Figure 21 *- Distribution of lengths of toxic protein sequences*

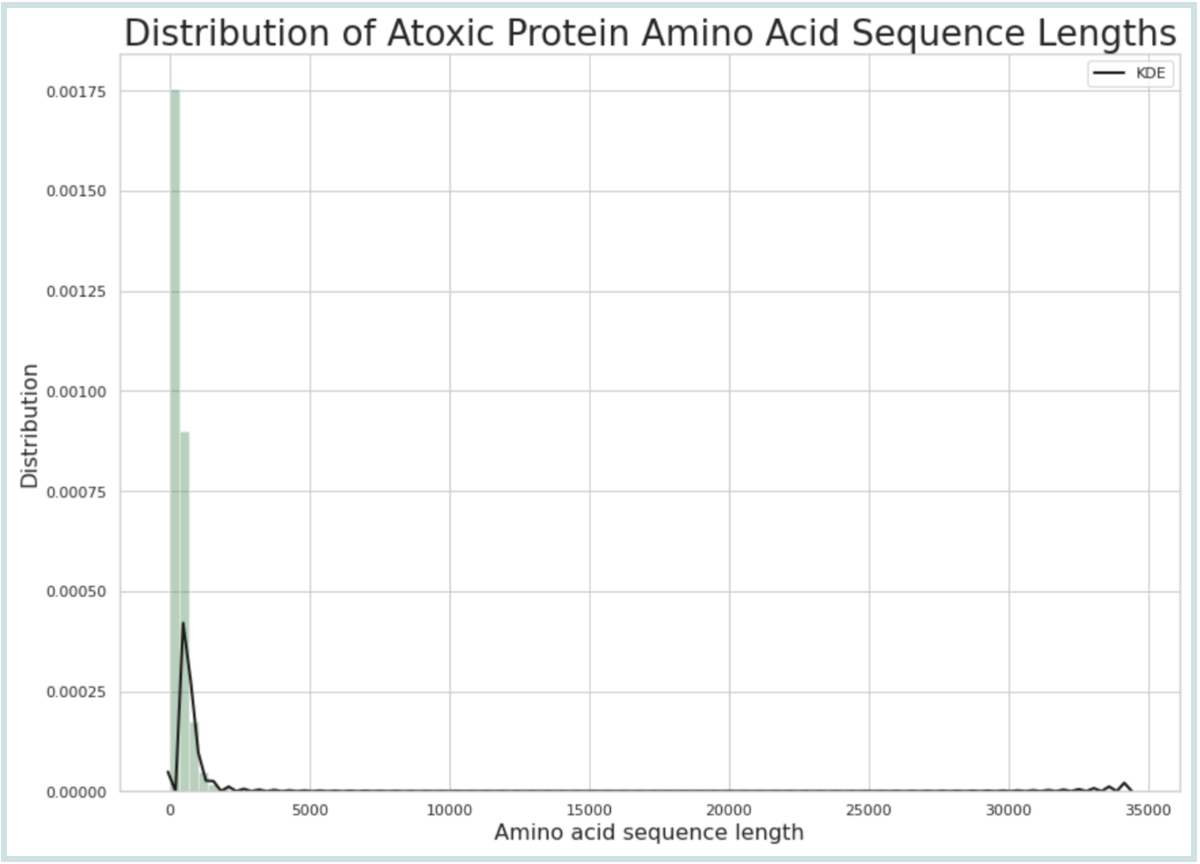


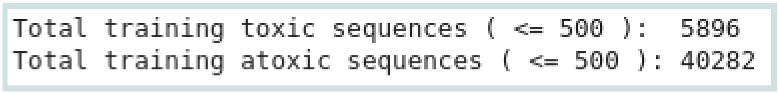
Figure 22 *- Distribution of lengths of atoxic protein sequences*

3.2.3 - Downsampling

Overall in the training dataset, there was a high difference in the number of sequences between toxic and atoxic proteins:

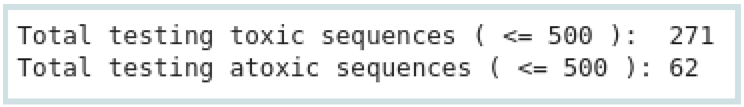
To prevent models from taking a significant amount of time to train, a 'MAX_SEQ_LEN' variable was created. This is a cut off point for removing any amino acid sequences that go beyond the chosen value.

Further analysis of the distribution graphs shown above, a 'MAX_SEQ_LEN' variable of 500 was chosen, as it would encompass the protein sequences, both toxic and atoxic, that are part of the highest proportion in the training dataset.



```
Total training toxic sequences ( <= 500 ):  5896
Total training atoxic sequences ( <= 500 ): 40282
```

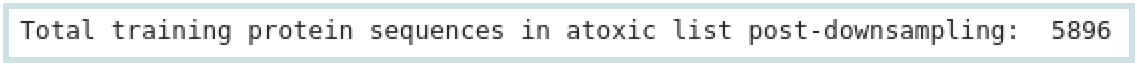
Figure 23 *- Number of training protein sequences less than 500 amino acids in length*



```
Total testing toxic sequences ( <= 500 ):  271
Total testing atoxic sequences ( <= 500 ): 62
```

Figure 24 *- Number of testing protein sequences less than 500 amino acids in length*

To account for this difference in class proportions themselves, the training atoxic protein sequences were randomly downsampled without replacement to equal the number of toxic protein sequences (5896 data points for both classes). This was done to ensure that metric scores such as ‘accuracy’ were not going to be misleading, since if the classifier model always favours the class with the majority number of sequences, the final prediction results would be misleading. [34]



```
Total training protein sequences in atoxic list post-downsampling:  5896
```

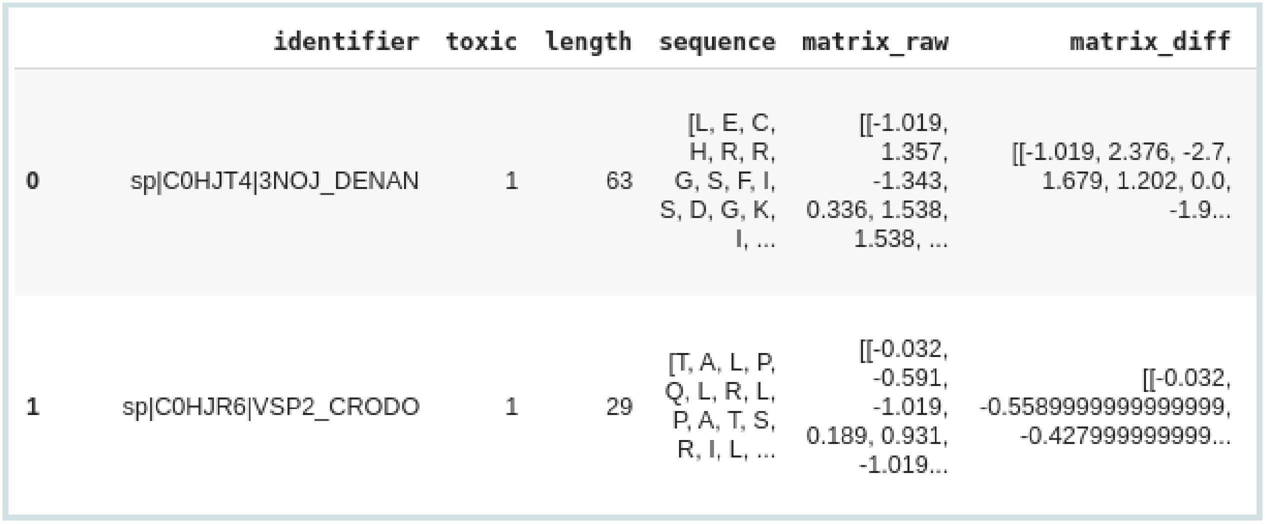
Figure 25 *- Total number of atoxic training protein sequences post downsampling*

The downsampling procedure was only executed on the training data set on the analysis of every resample and then again on the entire training set for when the final model is tested on the benchmark data, as it is important to propagate the effects of the downsampling procedure.

The testing dataset was left untouched in regards to making both toxic and atoxic classes equal, and has only been processed together with the training dataset for practicality, and to ensure that the final data shape for both training and testing datasets are exactly the same in regards on matrix dimensions to be fed into the CNN at the modelling stage.

3.2.5 - Atchley values feature extraction

After downsampling and splitting the dataset into training and validation sets, the atchley values for each protein in a sequence was appended to the existing sequence dictionary as show below:



	identifier	toxic	length	sequence	matrix_raw	matrix_diff
0	sp|C0HJT4|3NOJ_DENAN	1	63	[L, E, C, H, R, R, G, S, F, I, S, D, G, K, I, ...	[[-1.019, 1.357, -1.343, 0.336, 1.538, 1.538, ...	[[-1.019, 2.376, -2.7, 1.679, 1.202, 0.0, -1.9...
1	sp|C0HJR6|VSP2_CRODO	1	29	[T, A, L, P, Q, L, R, L, P, A, T, S, R, I, L, ...	[[-0.032, -0.591, -1.019, 0.189, 0.931, -1.019...	[[-0.032, -0.5589999999999999, -0.427999999999...

Figure 26 *- Atchley values matrices appended to each protein sequence*

In addition to having lists for all 5 Atchley values that correspond to a sequence of amino acids. The sequential change between these values were calculated to result in 5 extra lists appended to the 'sequence_dict' variable: 'f1_d', 'f2_d', 'f3_d', 'f4_d', 'f5_d'. Where 'd' is used to represent that this is a list of the sequential change in Atchley values for each particular type.

4 - Data Analysis

For clarification and understanding purposes, data visualisation is an important aspect of any machine learning process, as it allows some insights into the data set, during and after data processing, which is often able to aid in the decision making throughout the whole machine learning pipeline, which may result in better modelling, and predictive accuracies at the end.

4.1 - Atchley Values Data Analysis

For clarity, I decided to further investigate the distribution of the Atchley values:

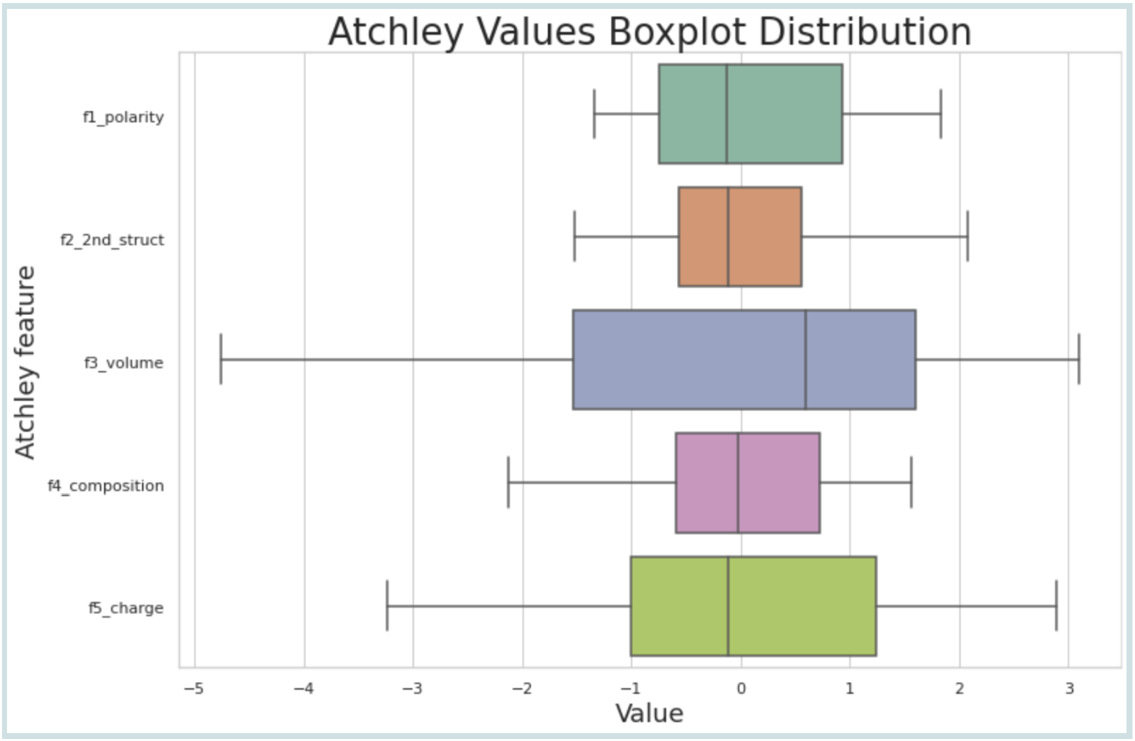
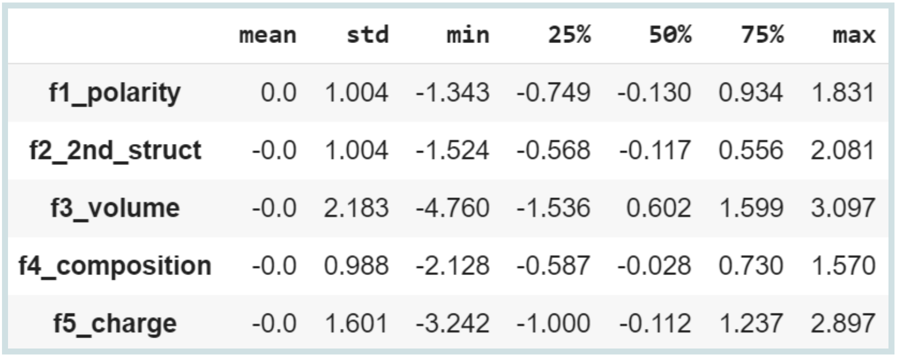


Figure 27 *- Distribution of Atchley values*



	mean	std	min	25%	50%	75%	max
f1_polarity	0.0	1.004	-1.343	-0.749	-0.130	0.934	1.831
f2_2nd_struct	-0.0	1.004	-1.524	-0.568	-0.117	0.556	2.081
f3_volume	-0.0	2.183	-4.760	-1.536	0.602	1.599	3.097
f4_composition	-0.0	0.988	-2.128	-0.587	-0.028	0.730	1.570
f5_charge	-0.0	1.601	-3.242	-1.000	-0.112	1.237	2.897

Figure 28 *- Numerical distribution of Atchley values*

Furthermore, post data processing where every single amino acid in the protein sequences was associated with each of the five Atchley values, I decided to further explore how these values change throughout a sequence's length. For example, I wanted to investigate how polarity (f1), or molecular volume (f3) of amino acid changes from the start of a protein sequence, compared to the end of a sequence, and if there are any peculiar variations occurring around the middle.

I settled on doing this in two stages: One taking into account just the raw magnitude of each Atchley value for the corresponding amino acids, and the other taking into account the sequential change in the Atchley values from one amino acid to the next. For example, if amino acid A, at position 'x' in a protein sequence had a polarity value of -0.591, and the next amino acid in the sequence at position 'x + 1' was amino acid K, with a polarity value of 1.831, the resulting change plotted would be 2.422. Calculated by doing 1.831 - - 0.591.

What is plotted for each Atchley factor is the average across all the amino acid sequences from their corresponding classes, 'toxic' and 'atoxic'.

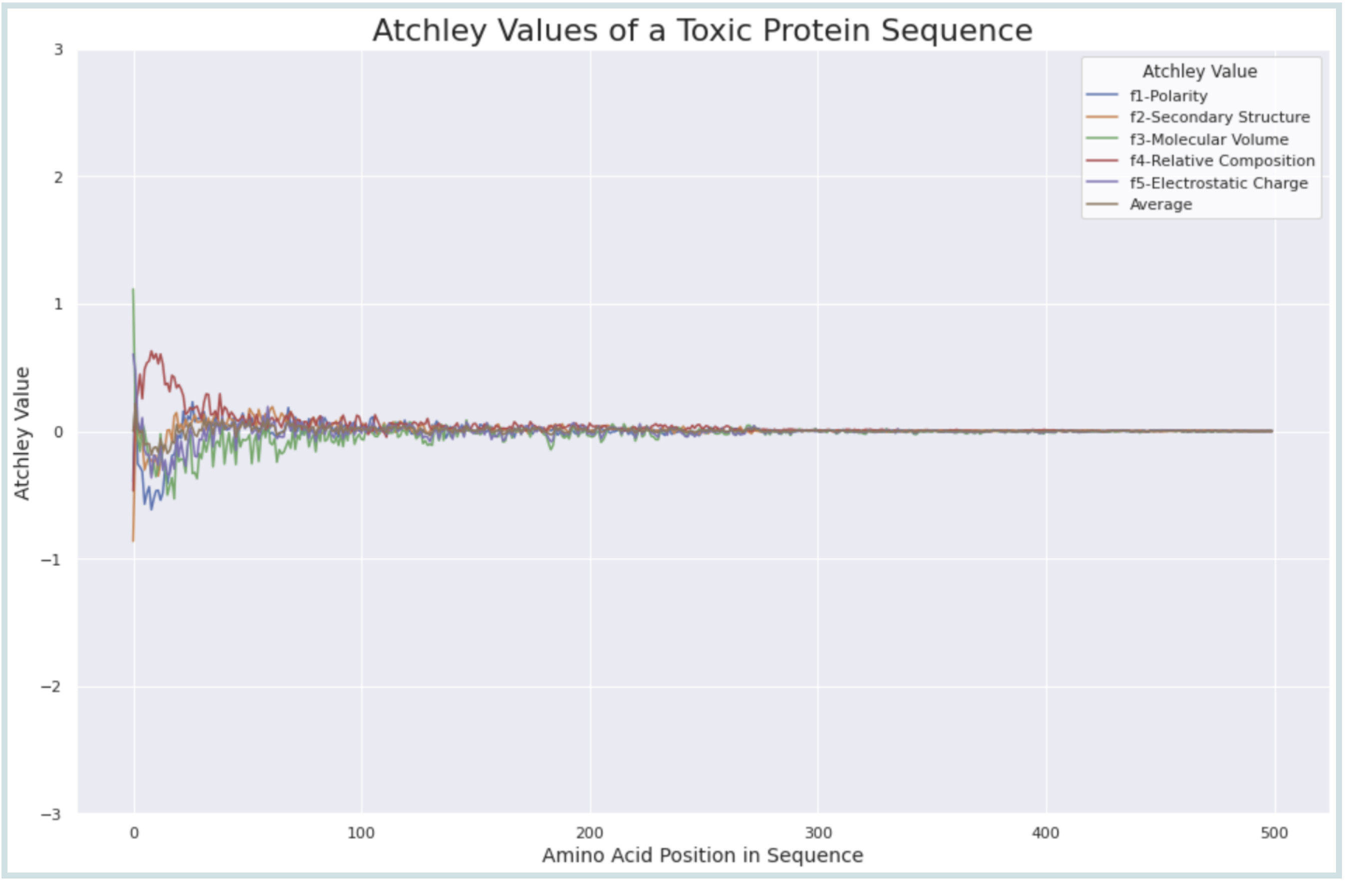


Figure 29 *- Average raw magnitude of the five Atchley values for toxic protein sequences*

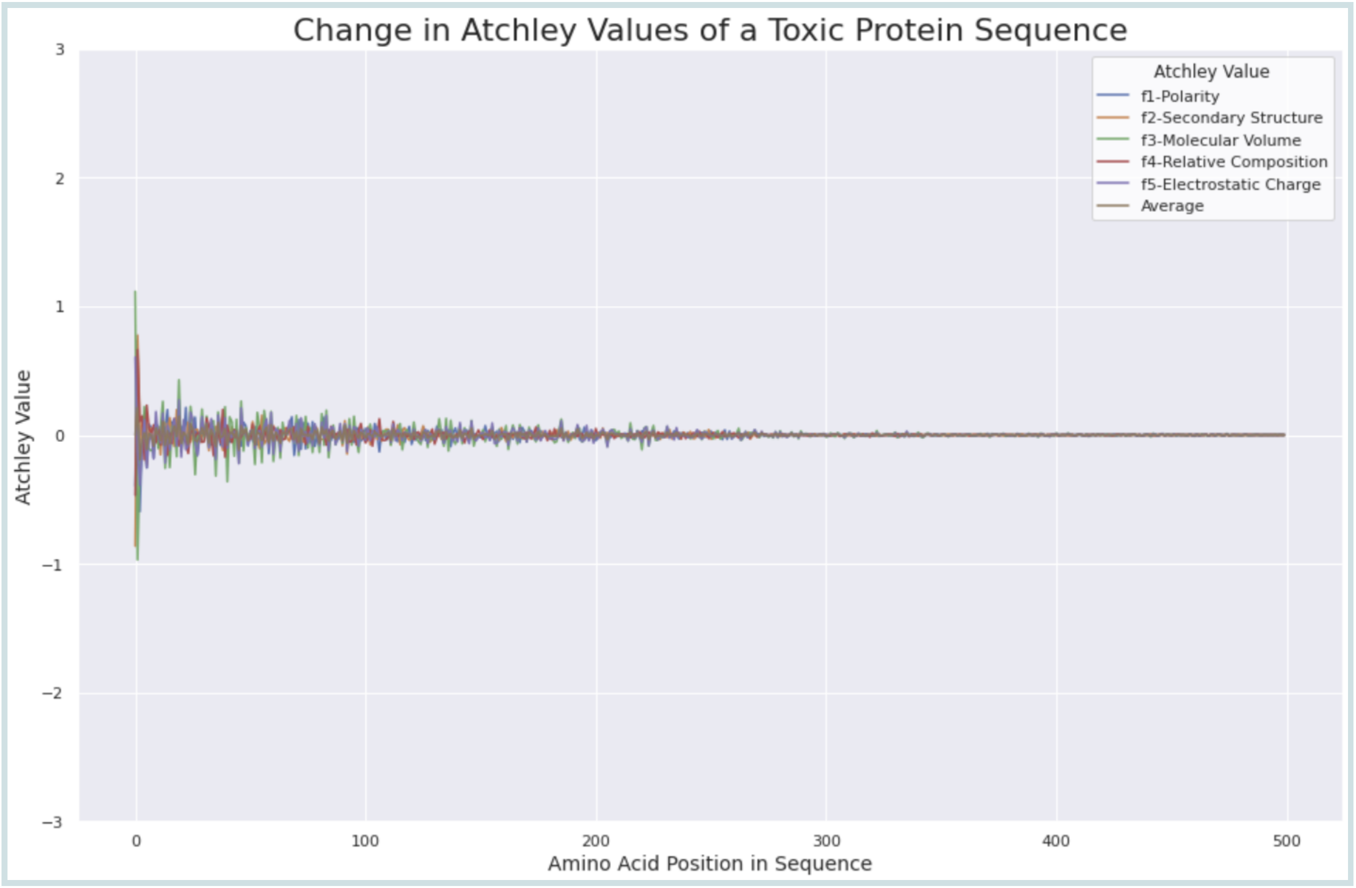


Figure 30 *- Average change in magnitude of the five Atchley values for toxic protein sequences*

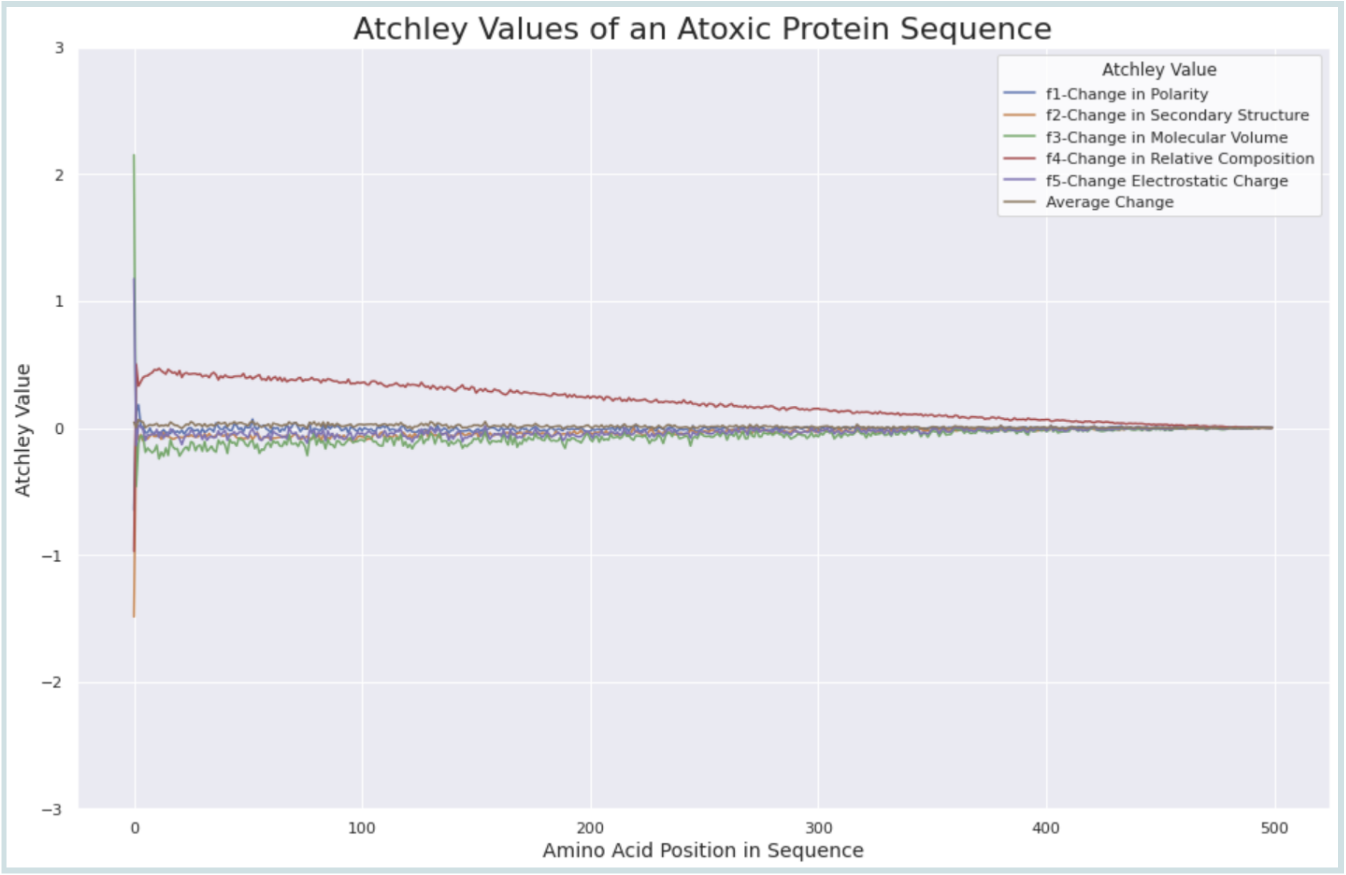


Figure 31 *- Average raw magnitude of the five Atchley values for atoxic protein sequences*

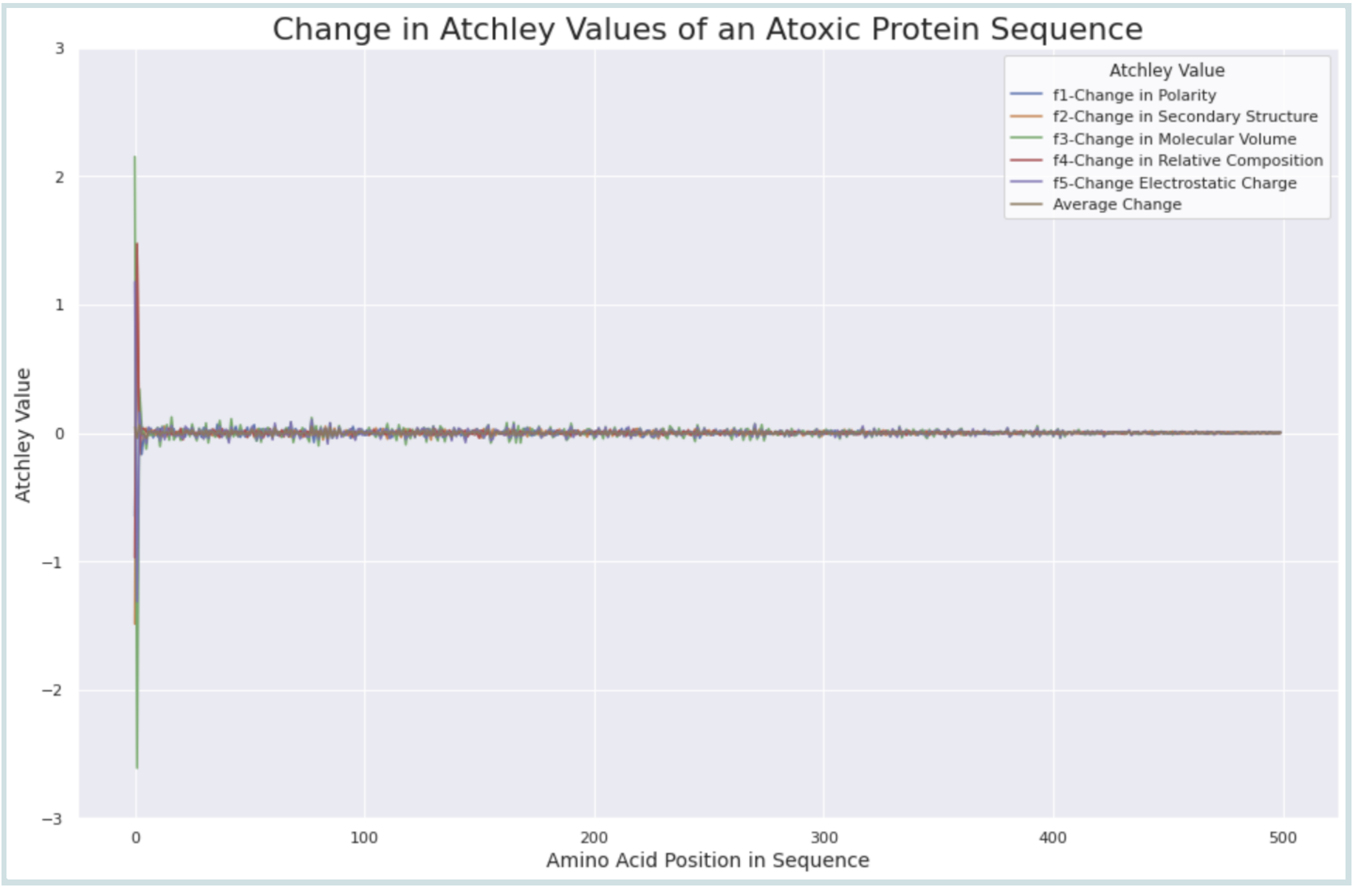


Figure 32 *- Average change in magnitude of the five Atchley values for atoxic protein sequences*

From the graphs plotted above we can imply that the general trend for both the raw magnitude and the change in magnitude for Atchley values for both toxic and atoxic protein sequences, is to start out large in the beginning of the sequence, and to slowly reduce as it reaches the end of a sequence.

However, it appears that the Atchley values for atoxic proteins appear to be much larger in the beginning of the sequence in comparison to toxic proteins.

Furthermore, another difference appears to be that toxic proteins have more volatile changes throughout the whole sequence itself, whereas atoxic proteins seem to have a more stable change from the beginning until the end of a sequence.

5 - Designing & Creating the CNN Model

Though reporting about the machine learning steps of data-processing, data-analysis and data modelling imply that they are carried out sequentially; in practice this is far from reality. To reach a stage where I was able to input the data into any of the models, I had to take an iterative approach. Whilst doing data analyses, realisations of how to make data processing better for this particular dataset would arise, and so I would naturally go back to the previous step and improve it, before doing more data analyses. A similar approach was done with the data modelling.

5.1 - Modelling steps

The steps towards modelling the data involved, first involved an iterative approach throughout the pre-processing stage. The target features were to incorporate all five Atchley values in an appropriate format to be fed through a neural network.

The strategy I concluded in taking was to investigate the differences in accuracies between using the raw magnitude, versus the sequential change of Atchley values. I was inspired to use a similar methodology to TOXIFY. For each group, the raw values, and the change in values, each protein was inscribed a 5 x 500 matrix. One dimension being 5, due to having 5 Atchley values, and the other dimension being 500, since this was the decided cut off number to be used for the maximum length of an amino acid sequence to be included in the dataset. If the protein sequence was shorter than 500 amino acids in length, the matrix was zero padded.

I decided to explore how a CNN (Convolutional Neural Network) would perform on this dataset. CNNs are primarily used for image data (both grayscale and coloured), however due to the physical data structure of the feature data being in a 5 x 500 matrix, this would be suitable to feed into a CNN. The Atchley values inscribed matrix would essentially provide an ‘image’ of each amino acid sequence. Rather than only being able to consider each separate Atchley value, such as polarity or molecular volume in a separate sequential manner, we would be able to combine them all to create a final picture that could be trained to classify whether that protein sequence is toxic or atoxic.

5.1.1 - A note on zero padding

Zero-padding is a method that allows the original input size to be conserved. They are often used when CNNs are involved, because their input requires it to be shaped in a specified standard way [35].

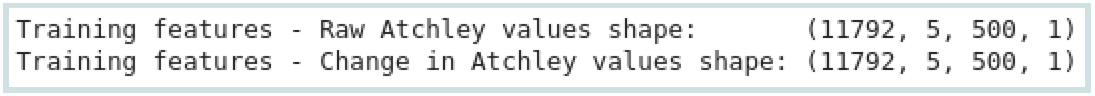
5.1.2 - Reshaping data for modelling

Naturally the next step after transforming the Atchley values into a 5 x 500 matrix, was to split the training dataset into feature data and label data. Since I decided to compare the difference in performance of the raw magnitude matrix of Atchley values, to the sequential change matrix of Atchley values, I have split the training dataset features into two different classes, to be able to measure and analyse their results in a separate manner.

I then had to reshape the training features to be able to feed it into the CNN. The first number '11792' is representative of the total number of amino acid sequences to be fed into the model. The next two numbers, '5' and '500' represent the shape of the 5 x 500 matrix feature data.

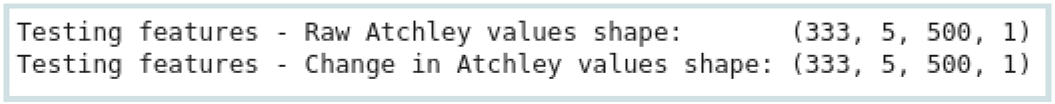
Finally the last number '1' indicates the number of 'channels' the CNN is required to handle. Since these types of the neural networks primarily work with image data, if the images being fed into it are coloured, then that means they would have '3' channels correlating to 'RBG', the red, green and blue colour channels. With this case, the CNN input would essentially be 3 matrices of the defined pixel dimensions. However, the same is true for grayscale images, of which the only difference is that rather than containing 3 colour channels, they only have 1 colour channel, and therefore only 1 matrix of the specified dimension is fed into the neural network.

This is a very similar case of what we have below, as substantially, we are feeding an 'image' of the Atchley values for each amino acid sequence:



```
Training features - Raw Atchley values shape:       (11792, 5, 500, 1)
Training features - Change in Atchley values shape: (11792, 5, 500, 1)
```

Figure 33 *- Training features data shape*

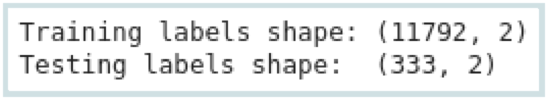


```
Testing features - Raw Atchley values shape:       (333, 5, 500, 1)
Testing features - Change in Atchley values shape: (333, 5, 500, 1)
```

Figure 34 *- Testing features data shape*

Similarly with the training labels 'toxic' and 'atoxic', we also were required to reshape the data into a specified format for the CNN to be able to understand and process it into a coherent model.

In this case, the number '11792', exactly the same as the training features shown above, represent the total number of amino acid sequences to be fed into the neural network to train it. The number '2' equates to the number of classes which is required to classify the data, in this case, a 'binomial' classification problem.



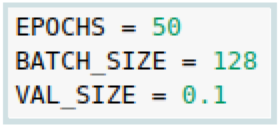
```
Training labels shape: (11792, 2)
Testing labels shape:  (333, 2)
```

Figure 35 *- Training and testing labels shape*

5.1.3 - Training vs Validation Set Splitting

The training data was further split into a smaller training set (90%), and validation (10%) set using k-fold stratified cross validation, so the same class proportions are kept within the training and validation data.

Following in the steps of 'TOXIFY' which have trained their RNN over 50 epochs, to be able to keep results comparable, I also decided to set my CNN training epochs to 50:



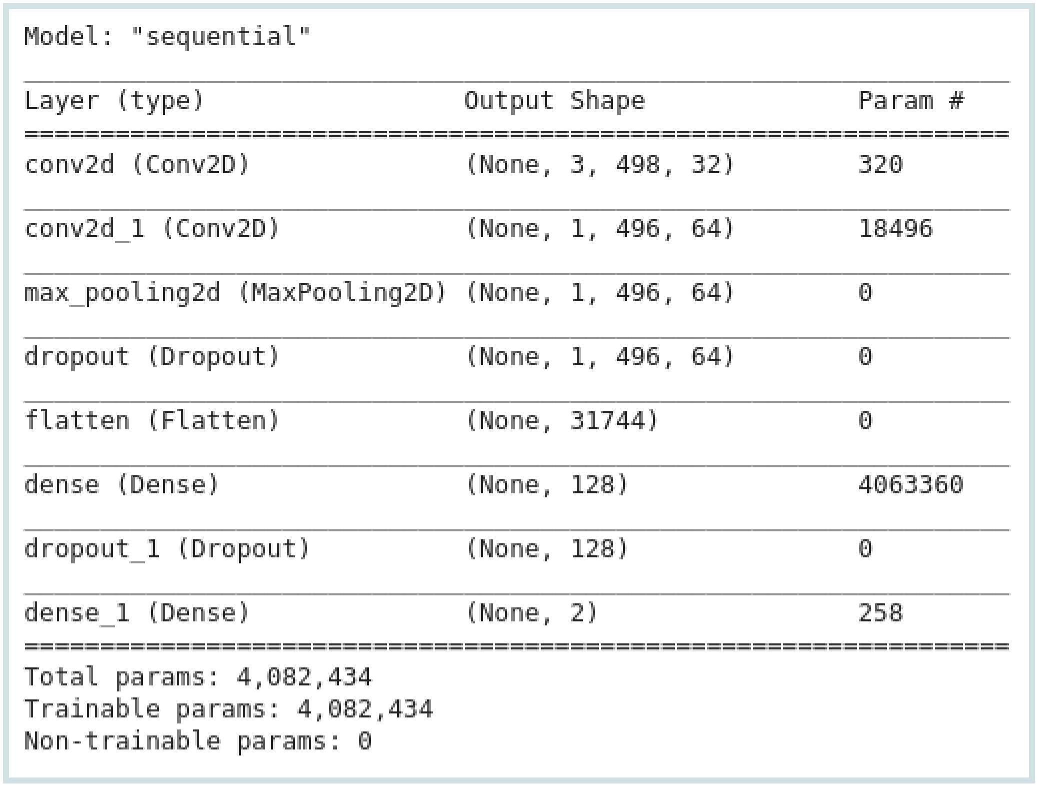
```
EPOCHS = 50
BATCH_SIZE = 128
VAL_SIZE = 0.1
```

Figure 36 *- Chosen parameters for modelling*

5.2 - CNN Modelling

To be able to keep my results as comparable as possible to previous development in the field provided by the projects ‘ToxClassifier’ and ‘TOXIFY’ (mentioned and explored in section 2.4 of this report), I used the same dataset they have used, which are protein sequences from UniProtKB (UniProt Knowledgebage), which is the central hub collection of functional protein information, with included names, descriptions and other annotations [36].

As previously mentioned in section ‘2.3.4 - CNN Case Study: LeNet-5’, I decided to model my CNN after the ‘LeNet-5’ architecture, as this has been proven to work on a series of grayscale image datasets from MNIST. I did however have to change a few parameters, such as the activation function, input size, and opmisers to better suit the nature of the dataset. I have decided to use Keras [37] and Tensorflow [38] libraries due to their practicality and robustness at offering straightforward routes on creating neural network models. Below is the overall summary of the CNN architecture used:



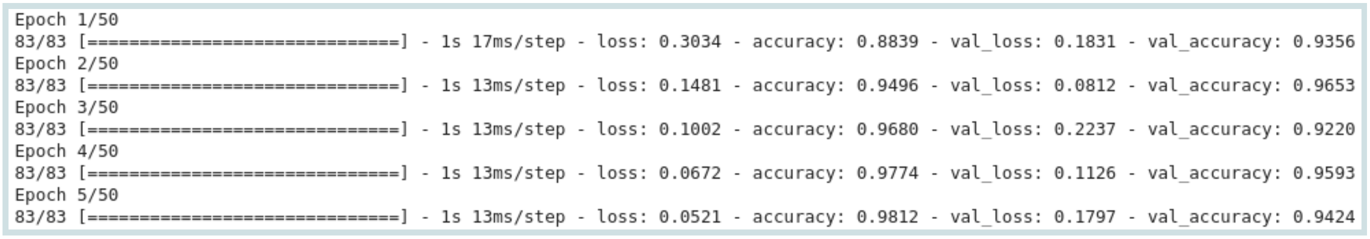
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 3, 498, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 1, 496, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 1, 496, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 1, 496, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 31744)             0
_________________________________________________________________
dense (Dense)                (None, 128)               4063360
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 4,082,434
Trainable params: 4,082,434
Non-trainable params: 0
```

Figure 37 *- CNN model summary*

I also included two ‘Dropout’ layers, as these are a regularisation technique of preventing neural networks from overfitting the validation set during model training.

5.3 - Training the CNN

I decided to explore what the difference in the predictive accuracies for two different types of features within the amino acid sequences: ‘Atchley raw magnitude’ and ‘Atchley sequential change’ would be. To test train both of these features, and be able to test results in a comparable manner, I have used the exact same CNN model architecture as shown in the previous section above.



```
Epoch 1/50
83/83 [==============================] - 1s 17ms/step - loss: 0.3034 - accuracy: 0.8839 - val_loss: 0.1831 - val_accuracy: 0.9356
Epoch 2/50
83/83 [==============================] - 1s 13ms/step - loss: 0.1481 - accuracy: 0.9496 - val_loss: 0.0812 - val_accuracy: 0.9653
Epoch 3/50
83/83 [==============================] - 1s 13ms/step - loss: 0.1002 - accuracy: 0.9680 - val_loss: 0.2237 - val_accuracy: 0.9220
Epoch 4/50
83/83 [==============================] - 1s 13ms/step - loss: 0.0672 - accuracy: 0.9774 - val_loss: 0.1126 - val_accuracy: 0.9593
Epoch 5/50
83/83 [==============================] - 1s 13ms/step - loss: 0.0521 - accuracy: 0.9812 - val_loss: 0.1797 - val_accuracy: 0.9424
```

Figure 38 *- Example of the CNN training over epochs (screenshot cut off at epoch 5)*

In most machine learning algorithms, we tend to have a particular focus on examining the ‘loss’ and ‘accuracy’ variables. ‘Loss’ is a reference to the ‘loss function’, which is a manner of predicting the error of a neural network.

For this particular problem, ‘binary cross entropy’ loss was required, since this is a binary classification problem, i.e. we are classifying a protein into either a ‘toxic’ or ‘atoxic’ class. This also meant that it was required to use a ‘sigmoid’ activation function to be able to convert the final output in the ranges between ‘0’ and ‘1’, which goes hand in hand with the nature of a binary classification problem [39].

On the other hand, the ‘accuracy’ variable relates to the ratio between the number of correct predictions, to the total number of input samples [40]. In its simplest form, it equates to how many data points, in this case amino acid sequences, were classified correctly to either ‘toxic’ or ‘atoxic’.

5.3.1 - Training Feature 1: Atchley Raw Magnitude

The graph below shows the training loss of using a 5 x 500 matrix with all of the raw Atchley values inscribed for each amino acid sequence. It was trained over 50 epochs. Overall, we are able to see a gradual decrease in the errors caused within the CNN by the blue line in the graph. The orange line, labelled 'val' shows the loss when it is predicted in the validation set. The graph shows some peaks for this around epochs 32 and 34.

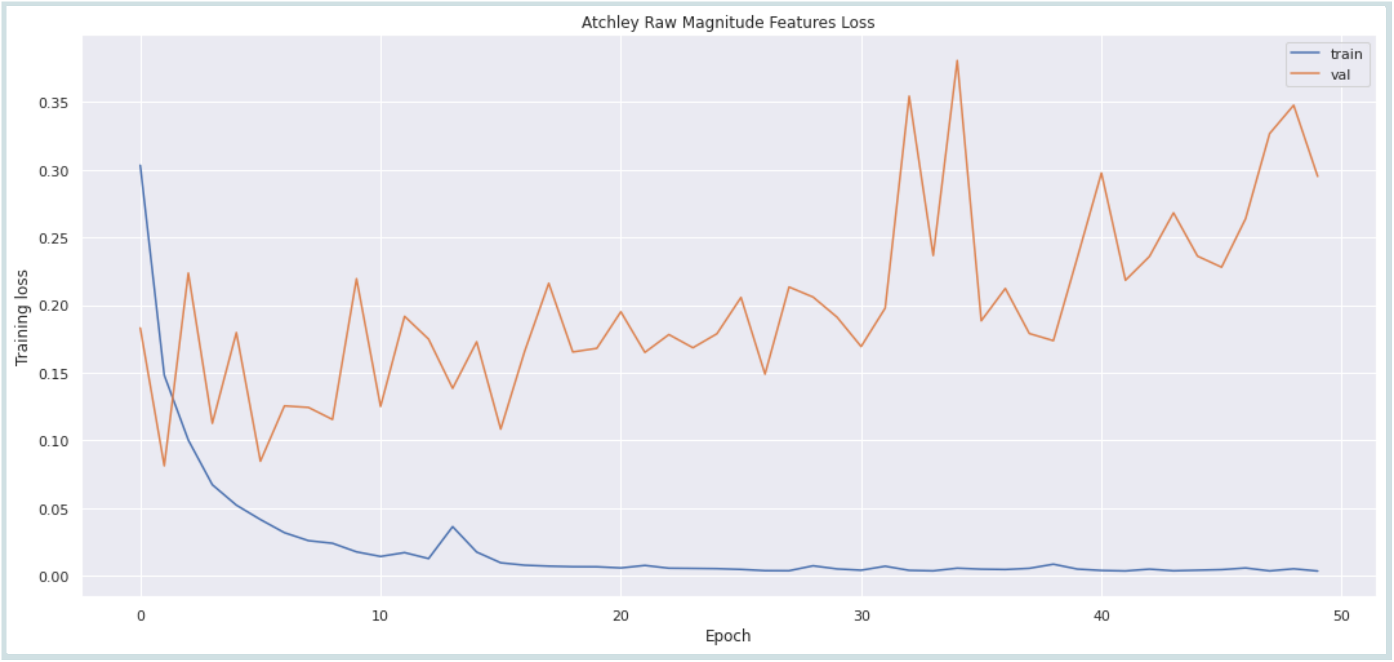


Figure 39 *- Atchley raw magnitude feature values loss over epochs*

The graph below shows the training accuracy of using a 5 x 500 matrix with all of the raw Atchley values inscribed for each amino acid sequence. It was trained over 50 epochs. We are able to see that over time, the training accuracy decreases, even though at certain points, around epochs 32 and 34, the prediction on the validation set shows a lower accuracy than previous and later epochs.

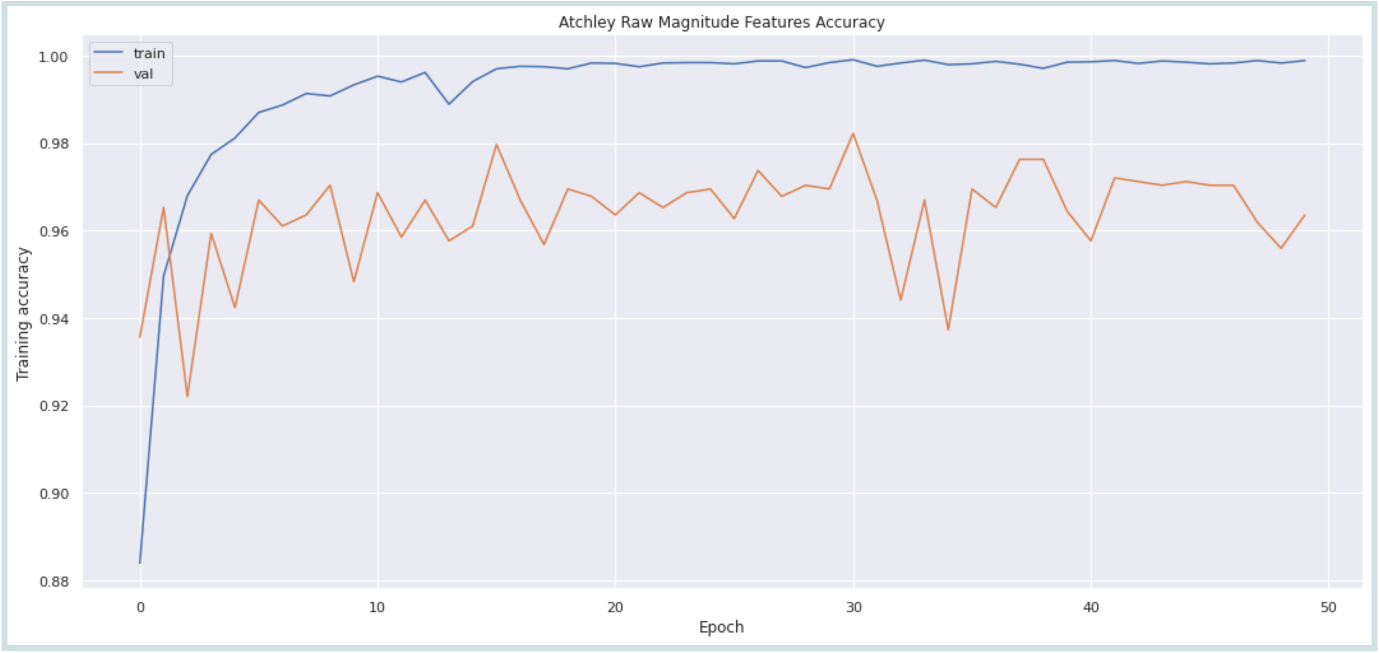


Figure 40 *- Atchley raw magnitude feature values accuracy over epochs*

5.3.2 - Training feature 2: Atchley Sequential Change

The graph below shows the training loss of feature 2 using a 5 x 500 matrix with all of the sequential change in Atchley values inscribed for each amino acid sequence. Like the previous training feature, it was trained over 50 epochs, so results could be kept comparable. Overall, we are able to see a significantly more gradual decrease in the errors caused within the CNN by the blue line in the graph. It is almost a flatline in comparison to training feature 1, which had a higher gradient. The orange line, labelled ‘val’ shows the loss when it is predicted in the validation set. This line for training feature 2 appears to be a lot more volatile than for training feature 1.

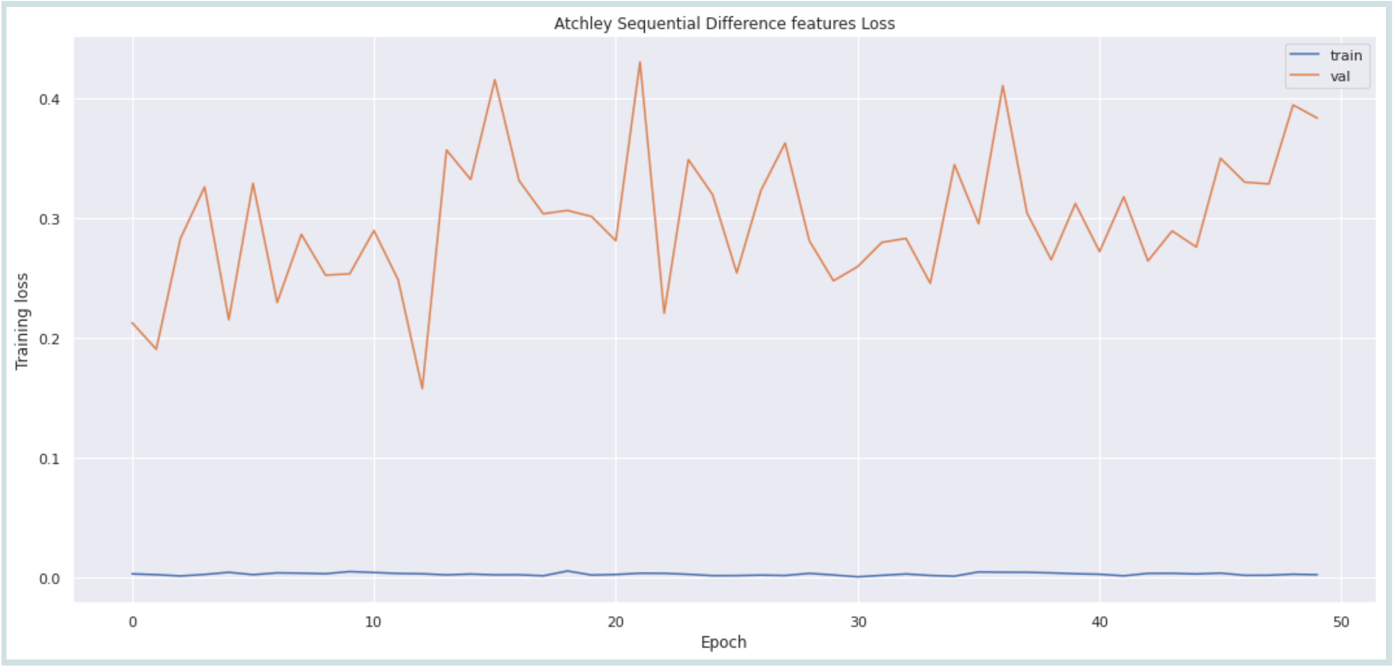


Figure 41 *- Atchley Sequential Difference feature values loss over epochs*

The graph below shows the training accuracy of feature 2, using a 5 x 500 matrix with all of the raw Atchley values inscribed for each amino acid sequence. It was trained over 50 epochs. We are able to see that over time, the training accuracy remains fairly stable, with no distinct positive or negative gradient. The predictive accuracy on the validation set (orange line in the graph) appears to be significantly more volatile than for feature one, throughout all of the training epochs.

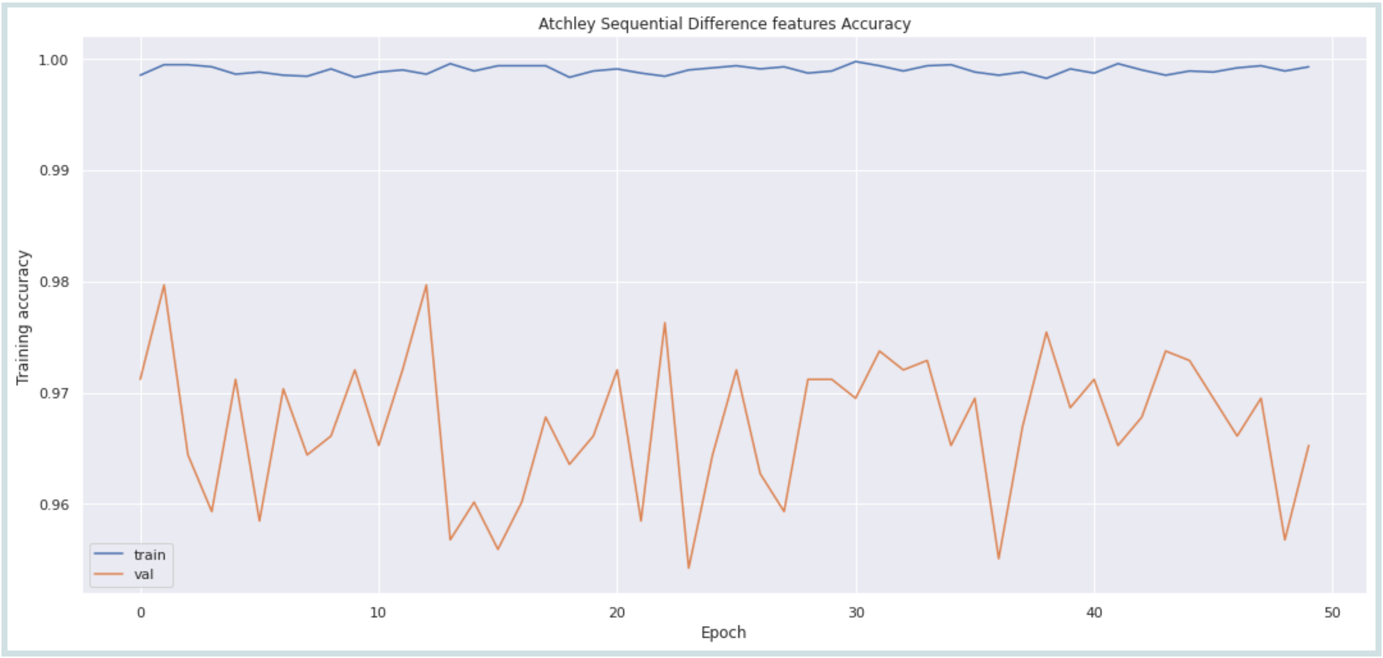


Figure 42 *- Atchley Sequential Difference feature values accuracy over epochs*

After tweaking some of the model parameters, such as optimisers ‘adam’ or ‘nadam’, which are included in the Keras library, I have settled that the models described above for both features have been the best performing out of the ones attempted throughout the modelling process.

Overall, at this stage, training feature 1 ‘Atchley raw magnitude’ appears to have outperformed training feature 2 ‘Atchley sequential change’, due to the fact that in feature 1, there is some gradient to show the loss function decreasing over the epochs, and the accuracy metrics gradually increasing throughout the epochs. In the other hand, in feature 2, both loss and accuracy metrics present a plateau, with significantly more volatile predictions in the validation set.

6 - Testing & Results

This chapter discusses the differences in the results between the two features I have chosen to explore through a CNN model, the ‘Atchley raw magnitude’ and ‘Atchley sequential change’. Considerations on how my highest predictive accuracy out of these two features compare with previous development in the field, such as ‘TOXIFY’s RNN based approach.

6.1 - Predicting on benchmark set

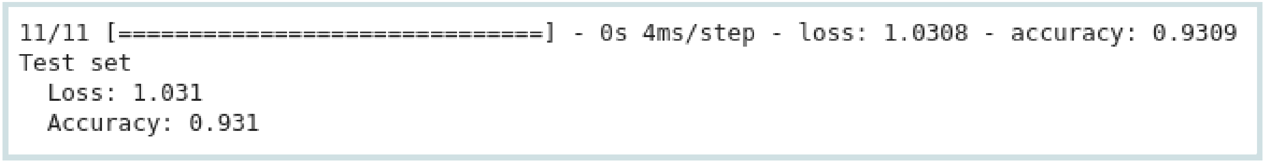
For testing my machine learning models, I have used the same ‘benchmark’ dataset that both ‘TOXIFY’ and ‘ToxClassifier’ have used in their testing stage, so that any results obtained from my work would be fairly comparable to their research. The testing data, like the training data has also been acquired from UniProtKB.

In addition, to prevent any unwanted bias to have infiltrated the decisions I took on modelling, I did not analyse the data from the benchmark set. However, due practicability and architectural reasons of the CNNs required input and output, I have pre-processed the benchmark dataset in exactly the same way as I have pre-processed the training dataset.

To test both ‘Atchley raw magnitude’ and ‘Atchley sequential change’ features, I have used the exact same CNN model architecture, and only changed the training input features accordingly.

6.1.1 - Feature 1: Atchley raw magnitude feature prediction

After evaluating the benchmark dataset with the Atchley raw magnitude, an overall loss value of ‘1.013’, and an overall prediction accuracy of ‘0.931’ was achieved.



```
11/11 [==============================] - 0s 4ms/step - loss: 1.0308 - accuracy: 0.9309
Test set
  Loss: 1.031
  Accuracy: 0.931
```

Figure 43 *- Final resulting predictive accuracy for Atchley raw magnitude features*

6.1.2 - Feature 2: Atchley sequential change feature prediction

After evaluating the benchmark dataset with the Atchley sequential change, an overall loss value of ‘2.153’, and an overall prediction accuracy of ‘0.820 was achieved.

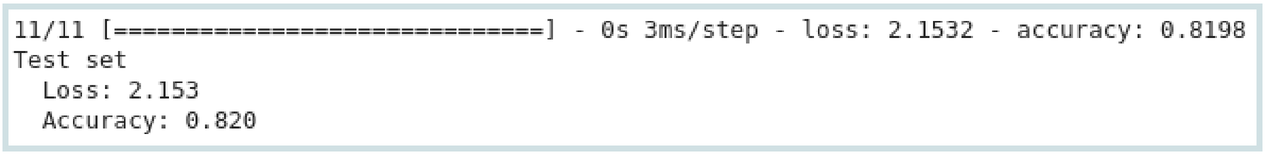

```
11/11 [==============================] - 0s 3ms/step - loss: 2.1532 - accuracy: 0.8198
Test set
  Loss: 2.153
  Accuracy: 0.820
```

Figure 44 *- Final resulting predictive accuracy for Atchley sequential change features*

6.2 - Results exploration

Overall, for the same CNN model architecture, feature 1 ‘Atchley raw magnitude’ outperformed feature 2 ‘Atchley sequential change’. It had a lower loss value, and a higher predictive accuracy score.

6.2.1 - Results considerations

The results above may suggest that, on a biological level, the actual raw Atchley values, in a sequential manner, influence more on identifying whether a protein is toxic or atoxic, as opposed the sequential change in the Atchley values themselves.

6.2.2 - Results comparison with previous studies

Since the study ‘TOXIFY’ is the most recent out of the ones explored in section 2.4 of this project, as well as the fact they have investigated the use of RNNs within the application of classifying animal venom proteins, as opposed to ‘ToxClassifier’ using LSTMs, I will be focusing the comparison of my results primarily to ‘TOXIFY’ and secondarily and tertiarily to ‘ToxClassifier’ and ‘ClanTox’ respectively.

During the RNN training, ‘TOXIFY’ achieved below the overall loss and accuracy metrics:

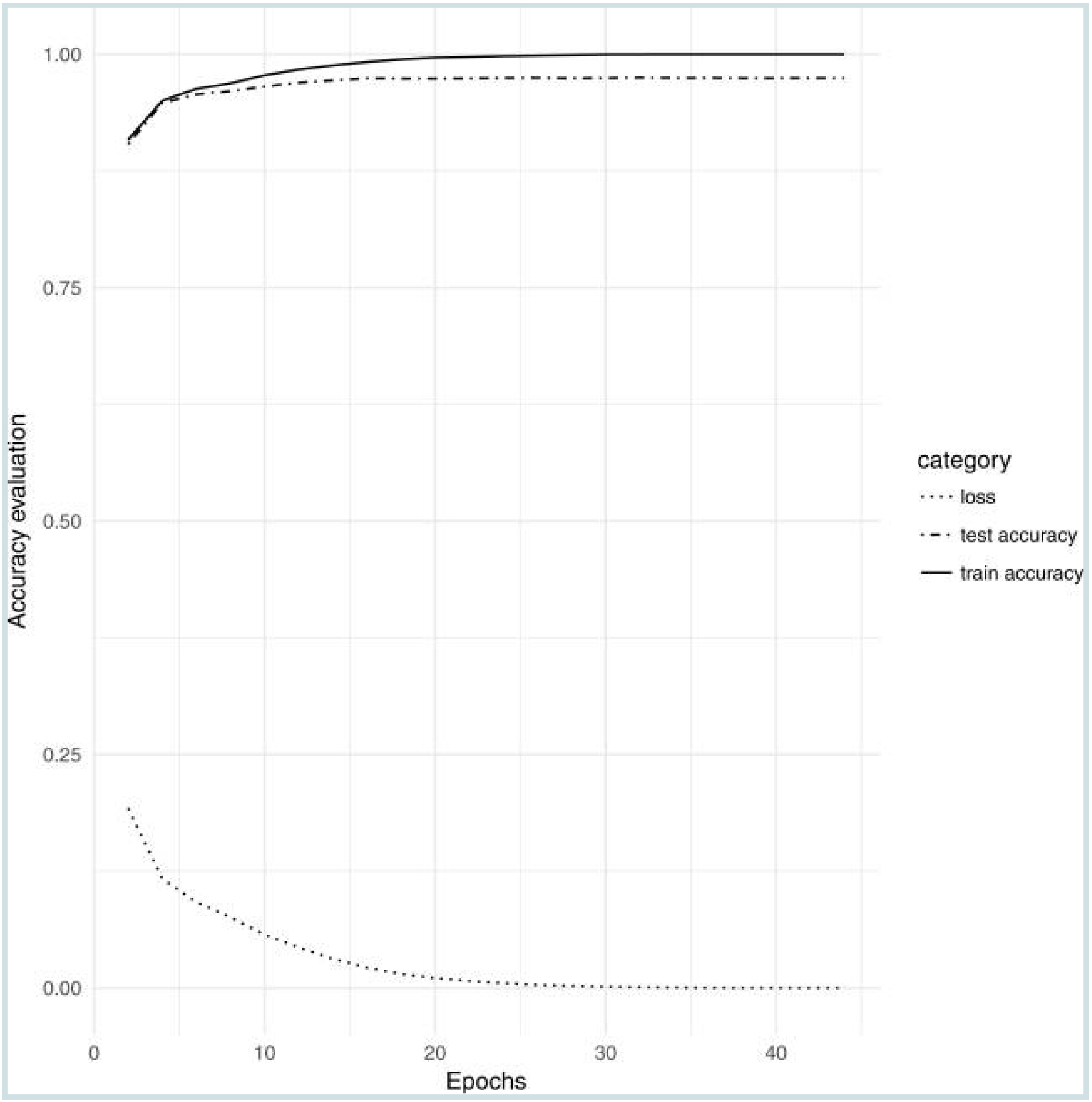


Figure 45 *- ‘Accuracy progression for RNN as training progressed for both training and test datasets [28]*

The final accuracy for TOXIFY on the benchmark dataset was 96.0%, and for ToxClassifier it was 99.7%. My CNN model, with feature 1 ‘Atchley Raw Magnitude’ achieved a predictive accuracy of 93.1% on the benchmark dataset. Overall, even if my CNN machine learning model scored lower than the RNN and LSTM models, it is a comparable result, and new exploration into the field that has not previously been tried before.

Potentially, by changing some parameters, reiterating again through the data processing stages, experimenting with different ratios for training vs validation set splitting, a higher predictive accuracy of the benchmark testing dataset could have been achieved.

In addition, possibly training and exploring other more expansive datasets, with a larger number and variety of both toxic and atoxic amino acid sequences, in conjunction with higher computational power available, could provide a deeper insight into and higher achieving accuracy results into classifying animal venom proteins.

7 - Project Evaluation

Overall, I believe this project has achieved its aims and objectives. For goal 1 and goal 2, sections in the literature review have explained and summarised the necessary biological concepts in detail that regard amino acid sequences, as well as machine learning techniques used today for a plethora of different problems, and how they in turn can be applied to the task presented for this final year project.

For achieving goal 3, in chapter 2.4, previous research and case studies 'TOXIFY', 'ToxClassifier', and 'ClanTox', were explored, analysed, and discussed in a thorough manner, considering both the biological and machine learning sides, as well as critically considering their modelling strategies.

For achieving goal 4, section 5 of this report shows how a CNN machine learning model was implemented to classify animal venom proteins, justifying the architectural reasoning behind the neural network.

Finally, for achieving goal 5 and goal 6, chapter 6 of this report explores the final results between two different features 'Atchley raw magnitude' and 'Atchley sequential change' I have explored myself by training them through a CNN model, as well as critically comparing their outcomes to other machine learning methods such as RNNs and LSTMs used in previous studies in the field.

However, even if the goals for this projects' scope have been achieved, I believe there is significantly more to explore not only within the field of classifying the toxicity of animal amino acid sequences, but within bioinformatics itself. Scopes can be increased or decreased to further research and explore scientific discoveries between the fields of biology and machine learning.

For the scope of this final year project, I would have liked to firstly, further explore how to improve a CNN model to predict non-image related data, but rather sequential data, which can be mimicked into an image format, as we have done by inscribing the Atchley values in a 5 x 500 matrix, but perhaps with larger or smaller matrices to achieve a larger exploration of the solution search space.

In addition, I believe I could have provided a higher variety of more visual graphs to further explore data insights, to not only support my own iterative development into using a CNN model for protein classification, but to provide a more visualisation heavy mode of communication to the readers.

Finally, I would have liked to produce a similar web tool like 'ToxClassifer' or 'ClanTox' to make the predictive classifical of animal venom proteins more accessible to the public, as I believe providing exposure and tools for other people to use and build upon their research projects and hobbies is significantly important to scientific discovery.

8 - Future Developments

The current issue with the recent machine learning techniques for classifying animal venom proteins provided by 'ClanTox', 'ToxClassifier', 'TOXIFY', and my own exploration into CNN models to tackle this problem, is that, even if LSTMs, RNNs and CNNs have the potential to offer high predictive accuracies, they do not offer insight to the real reasons as to why those proteins are toxic. The models take in no biological background information, and therefore cannot provide any rules as to what could potentially make those biological compounds toxic.

A potential solution to this is Inductive Logic Programming (ILP), which brings the opportunity to offer some background knowledge which is able to clarify as to what is happening behind the scenes with complex biological problems such as this.

8.1 - Inductive Logic Programming

Inductive Logic Programming (ILP) is a research area found at the intersection of Machine Learning and Logic Programming. Its main feature is that it uses background knowledge and final descriptions about the particular dataset as logic programs for classification. Primarily uses Prolog, which is a declarative language. The programmer is required to only express what is known about the problem domain. Potential benefits of this approach include the requirement for less amount of data to achieve the same accuracy of results when comparing to other methods and models [42] , as well as seamlessly working with relational databases.

8.1.1 - ProGolem Algorithm

ProGolem is an ILP system based on Asymmetric Relative Minimal Generalisation (ARMG). It repeatedly constructs clauses from a set of best ARMGs, and uses negative examples to reduce (and therefore generalise) these clauses [43]. It combines a bottom up search of rules, implemented with a new ILP system, that is able to learn long, non-determinate clauses, starting theory that is overly specific and is generalised to cover more examples of the dataset.

A few limitations to ILP would be that due to the nature of working with high-dimensional data, there will always be a trade off between accuracy and high computation and time requirements. Even with very powerful hardware used, for accurate and coherent results to be achieved.

Another limitation is that a lot of the software and development languages required to use and understand how to work with algorithms such as ProGolem is not as popular today, and so many people in the field end up overlooking, or not considering an inductive logic programming based approach.

Furthermore, an intriguing potential future research within this field would be to see if it is possible to track the evolutionary development of animal venom proteins for each specific species of animals. However, this would need significant consideration as how to approach the task, due to the significantly higher number of dimensions to be taken into account.

9 - Conclusion

Even though this project has offered a small contribution by exploring the capabilities of CNN machine learning models to classify animal venom proteins, there is still significant improvement and development that could be made in regards to using CNNs for classifying sequential biological data.

This project has achieved its defined goals, and hopefully has offered a small stepping stone into further research that professionals who are more ingrained everyday in the fields of biology, machine learning, and bioinformatics, are able to take on, and explore further with a higher amount of appropriate knowledge and computational power.

Bioinformatics is a field with opportunities for a vast amount of scientific research, which hopefully with more exposure, will become more known and interesting for people, both researchers and those who take a personal interest alike, to make infinite curious discoveries.

10 - Statement of Ethics

Considerations of ethics are an essential part, of not only research projects, but of ordinary everyday situations. Discussing the moral set of principles regarding the legal, social, privacy concerns, copyright concerns, and most importantly the implications of how conducted research, and developed software affect society, is severely important. Taking responsibility to ensure that all actions undertaken in any situation do not cause harm to society, including the environment in which we live in, is an essential component that ensures we are going towards the right direction in evolution.

In this section, since this project involves biological animal data information, the legal, social and ethical implications will be considered, in order to establish that any research and development have been conducted professionally and in accord with legal and ethical principles.

10.1 - Legal considerations

The original Data Protection Act (DPA) [44] is a legal document which dictates how personal data can be held, processed, and used by government, businesses, organisations, and any individuals required to hold people's personal data.
Strict rules called 'data protection principles' ensure that information must be: [45]

- 'Used fairly, lawfully and transparently.'
- 'Used for specified, explicit purposes.'
- 'Used in a way that is adequate, relevant and limited to only what is necessary.'
- 'Accurate and, where necessary, kept up to date.'
- 'Kept for no longer than necessary.'
- 'Handled in a way that ensures appropriate security, including protection against unlawful or unauthorised processing, access, loss, destruction or damage.'

There are stricter legal protection for information which is even more sensitive such as: race, ethnic background, political opinions, religious beliefs, trade union membership, genetics, biometrics, health and sex life or orientation.

Due to the nature of this project, it does not hold, store, or make the use of any sensitive human data. However, it does include the use of genetic animal data, which any implications must be further considered.

10.2 - Animal cruelty considerations

Even if on the data handling and machine learning side there is no direct harm caused to animals, we must also consider the moral implications of acquiring their biological data, since extracting any form of DNA will require handling of animals, and perhaps intrusive methods of in this case, toxin withdrawal.

Various research organisations will have different methods of keeping, caring and conducting research to a diversified number of different species, each with their required and specific procedures needed for protein extraction. For example, the methods in which venoms from a snake are extracted, will significantly differ on the methods of extracting toxins from a toad. In most cases, for venomous creatures, such as snakes, this would include sedating them in carbon dioxide (CO_2) gas first.

In addition to this. Research institutions will go a further step to consider all the elements that are used in their surrounding methods to conduct research in an ethically environmental manner as well, such as considering atmospheric emissions, solids and effluent disposal [47].

However, unfortunately, where the data for this project and the pre-decessing related research explored in the literature review use data from the UnitProt database, which do not offer information on the biological matter extraction of each particular animal in its database. If researchers are not directly involved within a research institution that partakes in the handling of animals, the reality is that they will have no direct control on how the animal is treated for the period of time they are there.

On the other hand, UnitProt does offer a privacy policy on their handling of their data, even if that does not include the biological procedures of extracting animal data. They consider the ethical implications of keeping personal data of people who are using their service, and offer contacts of all the institutions, referred to as the 'UniProt Consortium', which are their joint data controllers [47].

10.3 - Social considerations

A very important aspect of any project, business, or software created, is to ensure that it does not harm the public in any way. We must consider, not only the intention of research, but also how any discoveries, new information, and tools generated in the process could be used to cause harm. In reality there is no way to fully supervise how each individual who comes across a tool, or information will use them. However, our responsibility lies in keeping any project undertakings to legal law, and also importantly considering any potential for harm on what we have created or discovered.

In this project, since it regards venomous proteins, the implications of any further knowledge gained from researching these must be considered, since these biological compounds could potentially be used for harm intentionally, or if not handled properly, unintentionally.
On the other hand, research into venomous animal proteins can aid the development of new medical research. For example, anti-venoms of snakes are created by venoms of each corresponding snake themselves. In addition to this, there are a vastful number of diseases where substances developed with animal venom proteins have been shown to help ease the

effects, and often reverse them. Moreover, in the day-to-day laboratory research scenario, tools that help identify whether a protein is venomous or not, can seriously improve the speed at which research is conducted, offering a more effective base for discoveries to potentially occur, as well as minimising harm within laboratory practices. For example, if a biological compound is identified to be venomous earlier on, a different procedure in handling such compound would differ to the procedure of handling an non-harmful compound.

10.4 - Project conduction

As a member of the BCS, The Chartered Institute for IT [48], I have followed the Code of Conduct and Code of Ethics to ensure I have undertaken this project in an ethical and legal capacity. Throughout the project, I have referenced and given credits to all other works in the field which I have discussed, and well as any supporting knowledge included for clarification. Any code, and software tools have been referenced, and can be found in the references section of this project.

Albeit not many human related ethical principles needed to be followed, due to the content handling animal data, even if I did not have control of how the data used for training and testing models was acquired or handled, I attempted to be conscious of the legal, and moral implications of its potential resulting uses, with the sole intention of contributing positively to society and the environment.

References

[1] K. Nightingale, “The bite that cures: how we’re turning venom into medicine”, Science Focus, 5th Jul. 2019. [Online] Available: https://www.sciencefocus.com/nature/the-bite-that-cures-how-were-turning-venom-into-medicine/ [Accessed: March 2020]

[2] N. Tait and R. C. Vogt, “The Encyclopedia of Reptiles Amphibians & Invertebrates”, pp. 20, 60-81, 2006.

[3] E. D. Brodie III, “Toxins and Venoms”, *Current Biology*, Vol 19, Issue 20, 03 Nov. 2009. [Online] Available: https://www.cell.com/current-biology/comments/S0960-9822(09)01541-3 [Accessed March 2020].

[4] Chen, N., Xu, S., Zhang, Y. *et al.* “Animal protein toxins: origins and therapeutic applications”, *Biophysics Reports* 4, 233–242, 11 Oct. 2018. [Online] Available: https://link.springer.com/article/10.1007%2Fs41048-018-0067-x [Accessed March 2020].

[5] Murphy, M. P., & LeVine, H., 3rd (2010). Alzheimer's disease and the amyloid-beta peptide. *Journal of Alzheimer's disease : JAD*, *19*(1), 311–323, 29 Jan. 2010. [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2813509/ [Accessed March 2020].

[6] Helmenstine, A. Marie, “What Are the 3 parts of a Nucleotide? How Are They Connected?”, *ThoughtCo*, 11 Feb. 2020. [Online] Available: https://www.thoughtco.com/what-are-the-parts-of-nucleotide-606385 [Accessed March 2020]

[7] Cooper GM. “Protein Folding and Processing”, in *The Cell: A Molecular Approach*. 2nd ed. Sunderland (MA): Sinauer Associates; 2000. [Online] Available: https://www.ncbi.nlm.nih.gov/books/NBK9843/ [Accessed March 2020].

[8] W. R. Atchley, J. Zhao, A. D. Fernandes, T. Drüke, “Solving the protein sequence metric problem”, *Proceedings of the National Academy of Sciences of the United States of America*. [Online] Available: https://www.pnas.org/content/102/18/6395 [Accessed: March 2020]

[9] A. Arnx, “First neural network for beginners explained (with code)”, *towards data science*, 13 Jan, 2019. [Online] Available: https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf [Accessed: March 2020].

[10] H. S. Bisht, “Effect of Bias in Neural Network”, *GeeksforGeeks*. [Online] Available: https://www.geeksforgeeks.org/effect-of-bias-in-neural-network/ [Accessed: March 2020].

[11] V. Nigam, “Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning”, *towards data science*, 11 Sep. 2018. [Online] Available: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90 [Accessed: March 2020].

[12] H. Mahmood, “The Softmax Function, Simplified”, *towards data science*, 26 Nov. 2018. [Online] Available: https://towardsdatascience.com/softmax-function-simplified-714068bf8156 [Accessed: March 2020].

[13] M. Venkatachalam, “Recurrent Neural Networks”, *towards data science*, 1 Mar. 2019. [Online] Available: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce [Accessed: March 2020].

[14] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way”, *towards data science*, 15 Dec. 2018. [Online] Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [Accessed: March 2020]

[15] M. Rizwan, “LeNet-5 - A Classic CNN Architecture”, *engMRK*, 30 Sept. 2018. [Online] Available: https://engmrk.com/lenet-5-a-classic-cnn-architecture/#:~:text=Yann%20LeCun%2C%20Leon%20Bottou%2C%20Yosuha,which%20they%20called%20LeNet%2D5. [Accessed: April 2020]

[16] M. Gazar, “LeNet-5 in 0 lines of code using Keras”, *medium*, 26 Nov. 2018. [Online] Available: https://medium.com/@mgazar/lenet-5-in-9-lines-of-code-using-keras-ac99294c8086 [Accessed April 2020]

[17] ClanTox, “A Classifier of Animal Toxins”, *The Hebrew University of Jerusalem*, 21 Jan. 2009. [Online] Available: http://www.clantox.cs.huji.ac.il/ [Accessed: May 2020]

[18] G. Naamati, M. Askenazi, M. Linial, “ClanTox: a classifier of short animal toxins”, Nucleic Acids Research, 1 Jul. 2009. [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2703885/ [Accessed: May 2020]

[19] R. Gacesa, P. Long, “ToxClassifier”, *Chemical Biology Group, King’s College London*, 12 May. 2016. [Online] Available: http://bioserv7.bioinfo.pbf.hr/ToxClassifier/toxClassInstructions.jsp [Accessed May 2020]

[20] “6.1 - Introduction to Generalized Linear Models”, *PennSate Eberly College of Science*. [Online] Available: https://online.stat.psu.edu/stat504/node/216/ [Accessed: June 2020]

[21] “1.4. Support Vector Machines”, *scikit learn*. [Online] Available: https://scikit-learn.org/stable/modules/svm.html [Accessed: June 2020]

[22] R. Gandhi, “Support Vector Machine - Introduction to Machine Learning Algorithms, SVM model from scratch”, *towards data science*, 7 Jun. 2018. [Online] Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [Accessed: June 2020]

[23] Basic Local Alignment Search Tool, *BLAST*, 18 Jun. 2020. [Online] Available: https://blast.ncbi.nlm.nih.gov/Blast.cgi [Accessed: May 2020]

[24] HMMER, “HMMER: biosequence analysis using profile hidden Markov models”, *HMMER*. [Online] Available: http://hmmer.org/ [Accessed: May 2020]

[25] R. D. Finn, J. Clements, S. R. Eddy, “HMMR web server: interactive sequence similarity searching”, *Nucleic acids Research*, 18 May. 2011. [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3125773/ [Accessed: June 2020]

[26] “Hidden Markov Model”, *Bioinformatics Organization*, 4 Sept. 2009. [Online] Available: https://www.bioinformatics.org/wiki/Hidden_Markov_Model [Accessed: June 2020]

[27] R. Gacesa, P. Long, “ToxClassifier”, *Chemical Biology Group, King’s College London*, 12 May. 2016. [Online] Available: http://bioserv7.bioinfo.pbf.hr/ToxClassifier/[Accessed May 2020]

[28] T. J. Cole, M. S. Brewer, “TOXIFY: a deep learning approach to classify animal venom proteins”, *PeerJ Life & Environment*. [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6601600/ [Accessed: November 2020]

[29] S. Kostadinov, “Understanding GRU Networks”, *towards data science*, 16 Dec. 2017. [Online] Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be [Accessed: March 2020].

[30] “BLAST topics”, National Institutes of Health [Online] Available: https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp [Accessed: April 2020]

[31] Atchley Factors Data. [Online] Available: https://github.com/vadimnazarov/kidera-atchley/blob/master/atchley.txt [Accessed: April 2020]

[32] vadimnazarov, “atchley.txt”, *kidera-atchley*, 9 Jun. 2016. [Online] Available: https://github.com/christellevs/kidera-atchley/blob/master/atchley.txt [Accessed: Jan 2020]

[33] Toxify. [Online] Available: https://github.com/tijeco/toxify [Accessed: May 2020]

[34] “How to Handle Imbalanced Classes in Machine Learning”, *Elite Data Science*. [Online] Available: https://elitedatascience.com/imbalanced-classes [Accessed: May 2020]

[35] “Zero Padding In Convolutional Neural Networks”, *deep lizard*, [Online] Available: https://deeplizard.com/learn/video/qSTv_m-KFk0 [Accessed: August 2020]

[36] UniProtKB, 10 Apr. 2018. [Online] Available: https://www.uniprot.org/help/uniprotkb [Accessed: Jan 2020]

[37] Keras. [Online] Available: https://keras.io/] [Accessed: May 2020]

[38] TensorFlow [Online] Available: https://www.tensorflow.org/ [Accessed: May 2020]

[39] S. Verma, “Understanding different Loss Functions for Neural Networks”, *towards data science*, 20 Jun. 2019. [Online] Available: https://towardsdatascience.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718 [Accessed: August 2020]

[40] A. Mishra, “Metrics to Evaluate your Machine Learning Algorithm”, *towards data science,* 24 Feb. 2018. [Online] Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234 [Accessed: August 2020]

[41] S. H. Muggleton. “Inductive Logic Programming”. [Online] Available: https://www.doc.ic.ac.uk/~shm/ilp.html [Accessed: August 2020]

[42] J. C. A. Santos, H. Nassif, D. Page, S. H. Muggleton, M. J. E. Sternberg, “Automated identification of protein-ligand interaction features using Inductive Logic Programming: a hexose binding case study”, *BMC Bioinformatics*, 2012 [Online] Available: https://bmcbioinformatics.biomedcentral.com/track/pdf/10.1186/1471-2105-13-162 [Accessed: August 2020]

[43] S. H. Muggleton, J. Santos, A. Tamaddoni-Nezhad, “ProGolem: a system based on relative minimal generalisation”, in *Proceedings of the 19th International Conference on Inductive Logic Programming*, LNC5 5989, pages 131-148, Springer-Verlag, 2010. Available: https://www.doc.ic.ac.uk/~shm/Papers/progolem.pdf

[44] Data Protection Act 2018, UK public General Acts c. 12. [Online] Available: https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted [Accessed: August 2020]

[45] “Data Protection”, *Gov UK*. [Online] Available: https://www.gov.uk/data-protection [Accessed: August 2020]

[46] “Sustentabilidade”, *Instituto Butantan*. [Online] Available: http://www.butantan.gov.br/sustentabilidade [Accessed: August 2020]

[47] “UniProt privacy notice”, *UniProt*, 8 Mar. 2019. [Online] Available: https://www.uniprot.org/help/privacy [Accessed: August 2020]

[48] BSC the Chartered Institute for IT. [Online] Available: https://www.bcs.org/ [Accessed: August 2020]

Appendix

SAGE Form

SAGE form is included at the the end of the report and also included in the submission folder of this project under the name “SAGE_form_6418824_ChristelleVanSebroeck”.