uc3m | Universidad Carlos III de Madrid

University Degree in Biomedical Engineering

2020-2021

Bachelor Thesis

“Data Mining of Biological Macromolecules Annotations”

Sara Guillén Fernández-Micheltorena

Carlos Óscar Sorzano Sánchez (CNB-CSIC)

Mª Arrate Muñoz Barrutia (UC3M)

Leganés, 23 June

ABSTRACT

Bioinformatics refers to the set of tools that capture and interpret biological data, which has been in constant growth due to genome and sequencing projects. The introduction of computational methods to analyze the vast amount of biological information have contributed to significant discoveries in the biomedical research field regarding disease patterns. The combination of data mining techniques along with sophisticated machine learning algorithms is emerging as a potential way to extract useful information with applications in the vaccine development pipeline and cancer immunotherapy.

This method has been applied for the creation of an epitope predictor model with primary and secondary structure information on T-cell epitopes. The secondary structure information was obtained with a U-Net-based algorithm called ProteinUnet yielding fast and useful results. Natural language processing was implemented to understand protein language with the Bidirectional Encoder Representations from Transformers (BERT) architecture. The final epitope predictor model was a deep learning algorithm fed with features obtained from the language models with a validation accuracy of 70%. The proposed epitope predictor approach shows significant potential but further information and improvements could be implemented to increase the accuracy results.

Keywords: bioinformatics, machine learning, vaccine pipeline, cancer immunotherapy, T-cell epitopes, epitope predictor.

ACKNOWLEDGEMENTS

This thesis represents the culmination of four intense years of hard work that have helped me not only broaden my knowledge on the biomedical engineering field, but also grow as a person. I would like to thank all the people that have accompanied me through this process.

Carlos Óscar Sorzano Sánchez, for trusting me and my work and introducing me to the bioinformatics field. His constant support and endless help provided me with the confidence I needed to continue working on the project.

Arrate Muñoz Barrutia, for inspiring me to join this project. Her guidance and useful advices motivated me in the highs and lows of the process, and I am sure they will also do in the future.

Paola Núñez Hernández, for being so attentive and helping me every time I needed. Apart from contributing to the project with her work, she answered all of my questions and explained everything.

Members of the National Center of Biotechnology that have helped me when needed for accessing the computers.

My friends and family that have supported and encouraged me throughout the whole completion of the thesis. Specially, my roommates, who have enlightened my days making me laugh when I needed the most.

This project could not have been done without every single one of them. One of the greatest lessons I have learned is that the most important thing is to surround yourself with inspiring people who get the better from you. This is the key point for achieving great things.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AF:	Activation Function
ANN:	Artificial Neural Network
APC:	Antigen Presenting Cell
BERT:	Bidirectional Encoder Representations from Transformers
BRNN:	Bidirectional Recurrent Neural Networks
CD4+:	Helper T-cell
CD8+:	Cytotoxic T-cell
DL:	Deep Learning
FNN:	Feed-Forward Neural Network
HMM:	Hidden Markov Model
IEDB:	Immune Epitope Databas
LSTM:	Long Short-Term Memory
MHC:	Major Histocompatibility Complex
ML:	Machine Learning
MSA:	Multiple Sequence Alignments
NLP:	Natural Language Processing
PSSM:	Position-Specific Scoring Matrix
ReLU:	Rectified Linear Units
SVM:	Support Vector Machine

1. INTRODUCTION

Biological information has increased exponentially as a consequence of the genome sequences projects and advances in technology. This massive molecular data repository is constantly growing, promising to pave the way for precision medicine approaches. Bioinformatic tools are the only effective way to leverage the multi-dimensionality of the developed large data sets and convert the high-throughput data into relevant information [1].

The main focus of the biomedical research is nowadays mainly oriented towards knowledge discovery through biological data mining [2]. This typically involves the utilization of different tasks including associations rules, clustering, classification and regression analysis [3]. A big challenge arises when mining the huge amount of existing biological data, which entails the need for sophisticated machine learning techniques [4].

One of the main goals of data mining consists on the interpretation of the proteome information, which results in the promotion of health benefits in the vaccine design area [5]. In particular, epitope-based vaccine is an appealing notion that is being pursued successfully by many research groups. Epitopes are of special interest as they have the capacity to replace a pathogen in the vaccine. However, not all the epitopes are able to trigger the immune response [5]. Therefore, the requisite for designing epitope-driven vaccines would consist in discovering the right epitopes for assuring the correct performance of the vaccines.

The development of a bioinformatic tool for the prediction of epitopes that successfully trigger the immune response would have the potential of improving the epitope-based vaccine pipeline. The resulting vaccines would then have considerable more advantages over common vaccines, including a prolonged immunity and cost and time reduction [6]. In addition, these vaccines would serve as a new tool in the cancer immunotherapy field.

2. THEORETICAL BACKGROUND

2.1. Proteins

Proteins are biological macromolecules that perform highly sophisticated functions in all biological processes of the living systems. They are involved in transport and storage of molecules, mechanical support, immune protection, movement, transmission of nerve impulses, growth and differentiation [7]. Their wide variety of functional properties lays on their complex structure, which has been adjusted throughout evolution [8].

Structure of Proteins

Proteins are linear polymers composed by combinations of 20 amino acids, monomer units with different chemical properties [9]. The difference that characterizes each of the 20 amino acids is the side chain linked to the fourth valence [9]. The amino acids are linked by covalent peptide bonds in a specific arrangement defining the primary structure [8].

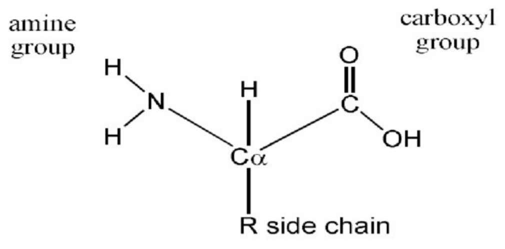


Fig. 2.1. Structure of an amino acid. Each amino acid has a central carbon atom with a hydrogen atom, an amino group and a carboxyl group attached. The side chain is the part that differentiates the 20 existing amino acids. *Source:* [10]

The unique organization of amino acids along with the physical properties and their interactions with the environment determine the folding pattern of the polypeptide chains. This localized organization is called secondary protein structure and there are two main conformations: alpha-helix and beta-sheet [8]. Both patterns are generated by hydrogen bonds established between the amino (N-H) and carboxyl (C=O) groups in the polypeptide chains. These two regular geometries are separated by regions of less organized loops called coils.

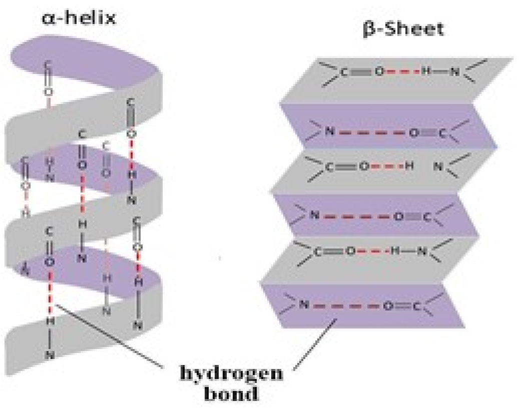


Fig. 2.2. The two common secondary structure conformations present in proteins: alpha-helix (left) and beta-sheet (right). The red discontinuous lines depict the hydrogen bonds between the amino (N-H) and carboxyl (C=O) groups of two amino acids. *Source:* [11]

The α-helix is a structure formed by right-handed rotations of the polypeptide chain with a particular angle that permits the generation of a strong hydrogen bond every four amino acids [7]. The resulting geometry is a regular helix which presents a complete turn every 3.6 amino acids [7].

The β-sheets are very rigid and flat structures generated by hydrogen bonds between neighboring polypeptide chains. The β-strands can be orientated in the same or opposite direction, which define the parallel β-sheet and antiparallel β-sheet, respectively [7].

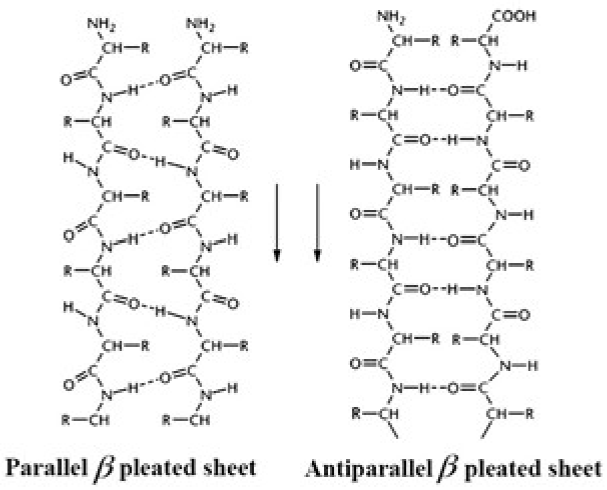


Fig. 2.3. The two directions in which beta-sheets are formed: parallel (left) and antiparallel (right). *Source:* [12].

The tertiary structure is the combination of secondary structure patterns linked by non-covalent bonds. These structures present conserved and functionally similar regions that will determine the functions related to the specific proteins [7].

2.2. Immunity and Immunotherapy

The immune system is a network of organs, cells and humoral components that hosts the defense mechanisms against pathogens. Immunity is divided into innate and adaptive responses. Although the innate response is faster than the adaptive response, this last one is more specific and precise [13]. Nevertheless, both systems act together, resulting in a highly effective immune response.

Innate Immunity

The innate or nonspecific immunity consists of a set of general defense mechanisms that serve as anatomic and physiologic barriers for a wide range of pathogens. The innate response is characterized by its rapidness when acting after pathogen exposure, providing an initial host reaction. In addition, it functions as an activating system for the adaptive immune response [14].

The first line of defense in the innate immune system is composed by the skin and mucous membranes that perform mechanical and chemical protection tasks. The second line of defense involves the action of interferons, complement proteins, natural killer cells, and phagocytes along with other processes such as inflammation and fever [15].

Adaptive Immunity

The acquired or specific immunity refers to the activation of T and B lymphocytes after encountering an antigen, a substance recognized as foreign by the immune system. The characteristics of the specific immune response are the specificity and memory. The resulting reaction can be cell-mediated, antibody-mediated or both [15].

The antibody-mediated immunity involves the destruction of antigens by antibodies generated by the B lymphocytes. On the other hand, the cell-mediated immunity performs the antigen destruction with the action of the T lymphocytes. These T-cells produce two types of cells: cytotoxic or CD8+ cells, which directly attack antigens, and helper or CD4+, which are involved in the co-stimulation of both T and B lymphocytes.

In order to ensure an adequate and safe immune response, the process of antigen recognition and presentation to T-cells is rather complex. This presentation is performed by the major histocompatibility complexes (MHC) located in the antigen-presenting cells (APCs).

The APCs are the molecules which recognize the specific antigens, process and presents them with the MHC complexes [16]. The MHC molecules are surface markers that bind to peptide fragments coming from pathogens [17]. Endogenous antigens are presented to the cytotoxic T-cells by the MHC-I markers, while exogenous particles are presented to the helper T-cells by the MHC-II molecules (see *Fig 2.4.*).

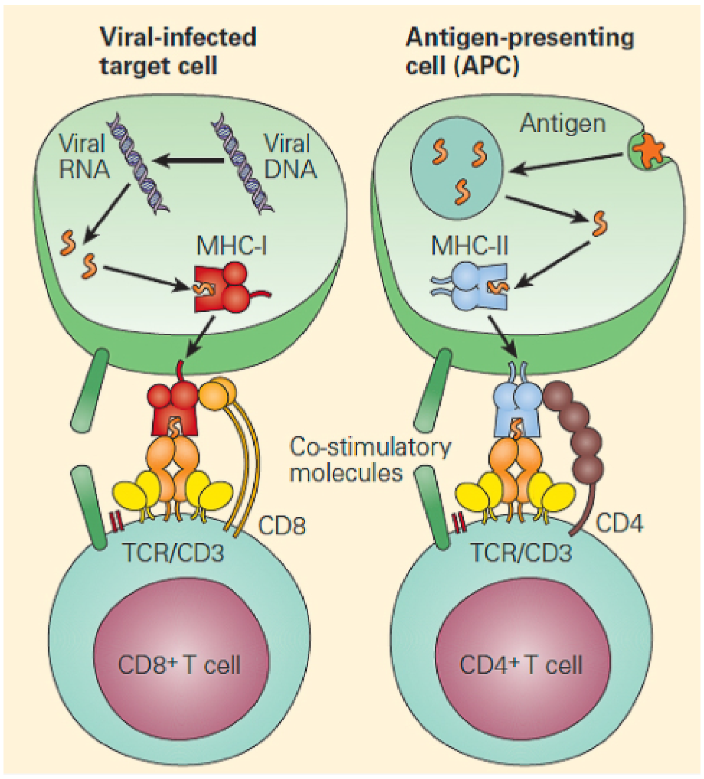


Fig. 2.4. There are two presentation pathways: endogenous antigens are presented to CD8+ T cells by class I major histocompatibility complex (MHC-I) molecules (left) and exogenous antigens are presented to CD4+ T cells by triggered by MHC class II (right). *Source:* [18]

The antigen presentation does not involve the whole antigen but the peptide fragments (see *Fig 2.5.*), called epitopes or antigenic determinants. These epitopes bind to the MHC molecules for their presentation to the CD4+ and CD8+ cells triggering immune responses [19].

Epitopes can be classified, depending on their structure, into continuous or linear and discontinuous or conformational epitopes. Continuous epitopes are usually helices recognized by T-cells, while discontinuous epitopes are recognized mostly by B-cells. Therefore, epitopes can also be grouped according to their receptors into T cell and B cell epitopes [20].

The capacity to discover T-cell epitopes that bind to MHC-I molecules is crucial in immunology. This results from the fact that class I MHC are the ones in charge of presenting endogenous antigenic peptides which define the requisite for the stimulation of CD8+ response, the principal mechanism against viral infections and tumors [21].

Inmunotherapy and Immunoinformatics

Immunotherapy is a therapy that consists of using substances to stimulate or suppress the immune system against cancer, infections or other diseases [22].

In the field of cancer immunotherapy, therapeutic cancer vaccines are used to teach the body how to defend itself from own damaged cells, which are the cancer cells [23]. These vaccines are designed to replicate the natural form of the pathogen without the associated pathogenic effects [24].

Vaccines based on epitopes would represent an alternative way for improving the immunity caused by the natural infection. The goal is to isolate T-cell epitopes which are the necessary components to trigger the response. The benefits that would be gained with this approach include the possibility of engineering epitopes as well as concentrating the immune responses on already known epitopes [24].

The development and production of vaccines is an expensive and long process [25]. One of the approaches to reduce them is the introduction of bioinformatics tools into the vaccine design area. Reverse vaccinology is the technique of using bioinformatics methods to identify cellular structures with the potential for inducing an immune response against a particular disease [25].

The epitope-based vaccines can be designed employing bioinformatics tools regarding the prediction of epitopes. There are several software packages for the identification of B-cell and T-cell epitopes [25]. These applications are based on a previous training using positive and negative epitope sequences.

The best and most accurate epitope predictor methods are the ones that combine sequence and structure analysis [20]. There are different machine learning techniques involved: position-specific scoring matrices (PSSMs), support vector machines (SVMs), hidden Markov models (HMMs) and artificial neural networks (ANNs). Each technique possesses different advantages and accuracy levels [20].

Immunoinformatics and the prediction of epitopes have a great impact in the fight against immune and infectious diseases. In this thesis, primary and secondary structure information of proteins will be used to create a bioinformatic tool for the prediction of epitopes using machine learning algorithms.

2.3. Machine Learning

Machine learning (ML) is focused on representing input data, learning patterns and generalizing for using them later in unknown data [26]. One important element in machine learning is feature engineering, which extracts features and representations of the input data.

Deep Learning (DL) algorithms are intended to develop complex and abstract data representations or features to solve complicated interpretation tasks [26]. Deep learning has shown promising results in many applications such as computer vision, image feature coding, information retrieval or analysis of molecules [27]. The architecture underneath deep learning algorithms is based on artificial neural networks (ANN).

An artificial neural network is a ML algorithm that contains nodes which communicate through connections in a way to simulate how human brains process information [27]. The basic components of an ANN are the nodes, network structure and learning rules [28].

Node Character

The node character controls how nodes process the signals in terms of the number of inputs and outputs of each node, the weights associated with these components and the activation function.

Each of these nodes receives several inputs with an associated weight. After the sum of the arriving weights has reached a certain value or threshold (T), a transfer function (f)

is applied to the signal which will be later transferred to adjacent nodes [28]. This advancing flow of information is called forward propagation and is illustrated in the following mathematical formula:

$$y = f\Big(\sum_{i=0}^{n} w_i x_i - T\Big) \tag{2.1}$$

being y the output of the node and w_i the weight of input x_i.

There are many transfer or activation functions (AF) which depend on the problem. Introducing activation functions of degrees higher than one (non-linear) allows the ANN to map input nodes to output nodes in a certain way in order to understand complicated, high dimensional and non-linear data [29]. The most common activation functions are Tanh, Rectified Linear Units (ReLU), Sigmoid and Linear.

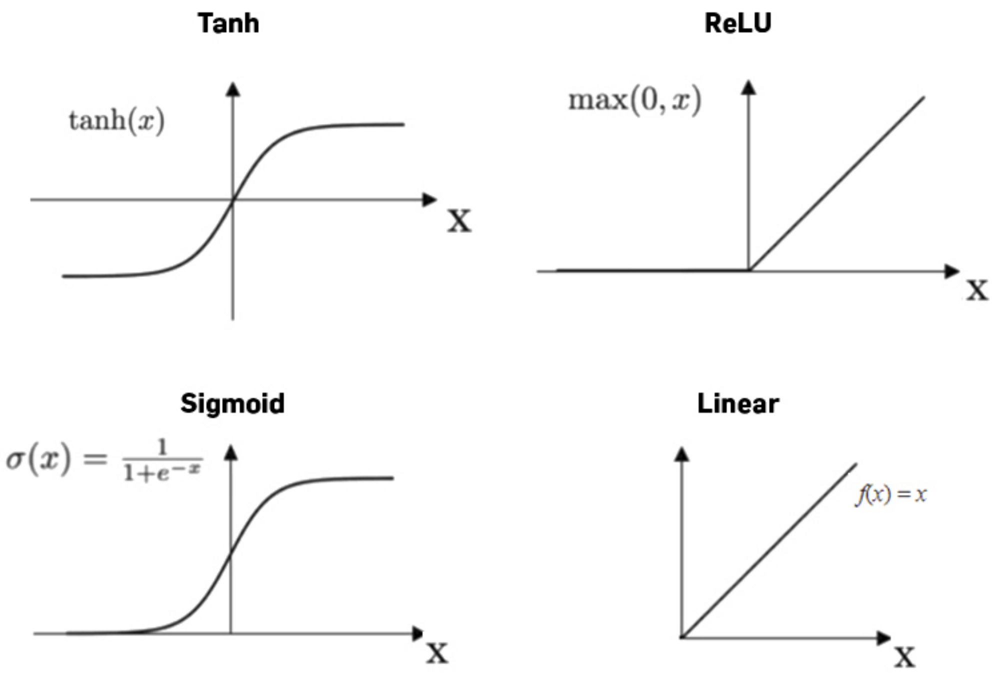


Fig. 2.5. The most common activation functions (Tanh, ReLU, Sigmoid and Linear) with their corresponding formulas and graphs. *Source:* [30]

Network Topology

The neurons that form neural networks are organized into input, output and hidden or dense layers. The most common layer is the dense layer, a regular deeply connected layer in which every node is connected to a node in the next layer [31].

The network topology defines the structure of the ANN in terms of the number of nodes in each layer, the number of layers and the direction of the connections among the nodes [28]. These factors are optimized by conducting different experiments. A classical ANN architecture is depicted in *Fig 2.7.*.

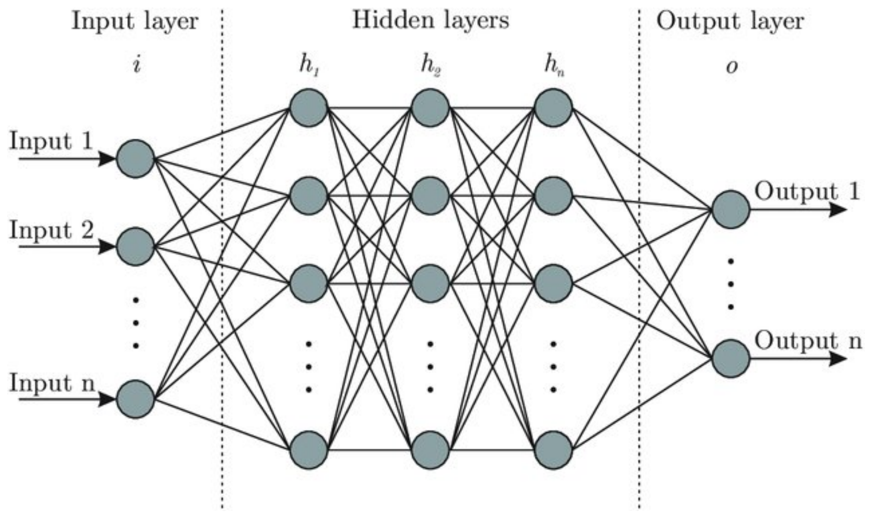


Fig. 2.6. Architecture of a classical ANN with n input layers (i), 3 hidden layers (h) and n output layers (o). *Source:* [32]

The connections between nodes can be one-way or loop-back defining two types of neural networks [28]:

- **Feedforward network**: The signal moves one-way meaning that there is one specific output for each input. This generates a static network.
- **Feedback network**: Dynamic network in which one input produces several outputs in a way to change the state of the network until equilibrium is reached. The most common example is the perceptron.

Learning

A learning process is completed in order to initialize and adjust the weights to certain values, which is referred as the training of the ANN. There are two classes of learning [28]:

- **Supervised learning**: The target outputs corresponding to the training set are given. The objective is to minimize the error of the network output and its corresponding correct label. Therefore, the network is previously trained.

- **Unsupervised learning**: The target outputs are unknown so the network uses only the input data in order to attempt to discover what the trend is.

Many learning schemes have been devised in order to accomplish various learning objectives. The most commonly used one is computing the loss function calculated with the weights of previous layers by the backpropagation algorithm [33]. This algorithm measures the error through several paths from a node to the output. The error can be calculated with the difference between the output node $y_{k,n}$ and the target output y_k^* as it is depicted in the following mathematical formula [28]:

$$e_k = y_{k,n} - y_k^* \tag{2.2}$$

After getting the corresponding error, the new weight $w_{kj,n+1}$ is obtained for the input x_j calculated as follows [28]:

$$w_{kj,n+1} = w_{kj,n} - \lambda e_k x_j \tag{2.3}$$

where λ refers to the learning rate, which greatly impacts the rate of convergence.

Regularization

There are some practical issues associated to neural networks when reaching great levels of performance in the training process. The most important one is the overfitting or the poor generalization behavior [34]. This issue implies that a model that accurately fits to a given training set does not guarantee similar prediction results on test data.

A common strategy to control this problem is to introduce Dropout, a regularization technique that permits the elimination of complicated adaptations generated in the training data [35]. The process dropout consists of removing hidden or visible units and its corresponding connections from the network randomly [35].

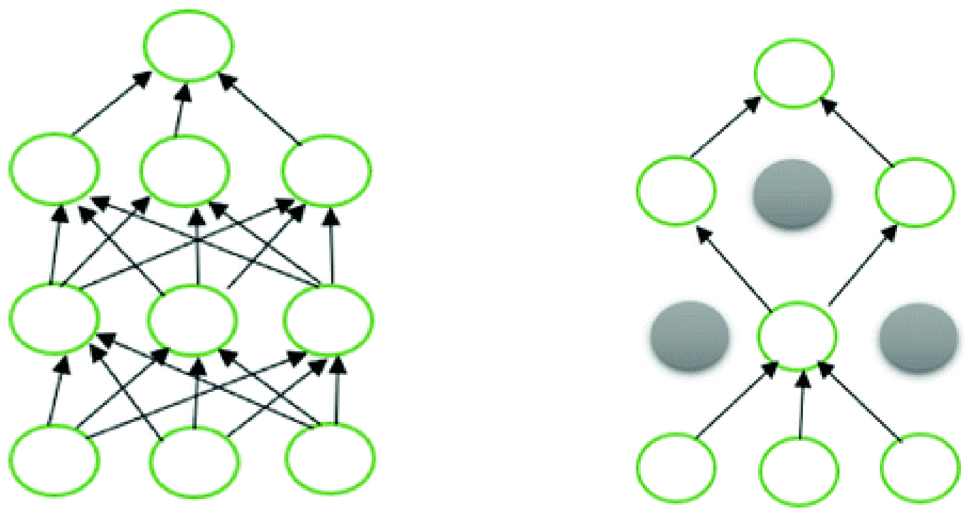


Fig. 2.7. Visualization of the dropout process consisting on removing units and their corresponding connections. Neural network without dropout (left) and neural network with dropout (right). *Source:* [35]

2.4. Secondary Structure Prediction

The amino acid sequences of proteins determine their three-dimensional structure, which is crucial to their functional mechanisms [36]. When compared to sequence determination, experimental structure determination is costly and time-consuming [37]. This entails the need for computational methods that can predict the secondary structure of a protein from the sequence of amino acids or primary structure.

Most of the methods used for predicting secondary structure of proteins rely on evolutionary data from multiple sequence alignments (MSA) [38], which are alignments of more than two sequences[39]. This is specially accurate for proteins that have known homologous sequences or sequences that have a common ancestor. Nevertheless, the majority of proteins either lack or contain a few homologous sequences [40] and thus, the prediction accuracy decreases. Besides, the increasing amount of known protein sequences leads to an increase in the time of computation of MSA.

The advances in deep neural networks have encouraged projects on the development of predictors with high accuracy values and without the requirement of finding homologous sequences. This is the case of SPIDER3-Single, a single-sequence-based model using long short-term memory (LSTM)-bidirectional recurrent neural networks (BRNNs)

[38]. A fast and similar approach is the ProteinUnet model [41], which provides a reduction in the amount of network parameters and decreases considerably the training and prediction times compared to SPIDER3-Single. ProteinUnet model will be used to obtain the secondary information in this thesis.

2.4.1. ProteinUnet

The ProteinUnet is a secondary structure prediction method based on the U-Net deep neural network. U-Net architecture is commonly used in segmentation of images. In this method, the predictions of 1D sequences would be comparable with the 2D image segmentation.

Model Architecture

The architecture of the ProteinUnet consists of blocks placed in a symmetric way with two paths: contractive (encoder) and expanding (decoder) [41]. These two paths create a U-shape, the characteristic form of the U-Net models.

- In the contractive part, the blocks are formed by three convolutional layers with a ReLU activation. The layers contain 64 filters in the first two blocks and 128 in the two last ones. The blocks are followed by an average pooling layer of kernel 2 for the downsampling.

- In the expanding path, the number of convolutional layers is reduced to two per block with a ReLU activation. There is a concatenation of each bock with its matched block from the contractive path. Then, there is an upsampling layer of kernel 2.

This peculiar architecture permits the extraction of high-level features from the contractive path and their propagation in expanding path layers which have a higher-resolution. This way, the local context is combined with the global information while increasing the output precision.

In the final part, the classification is added by fully connected layers with ReLU activation of 128 and 64 nodes. Then, the output layer presents two possible activations:

softmax for classification model and sigmoid for regression results. The overall architecture is depicted in *Fig 2.8.*

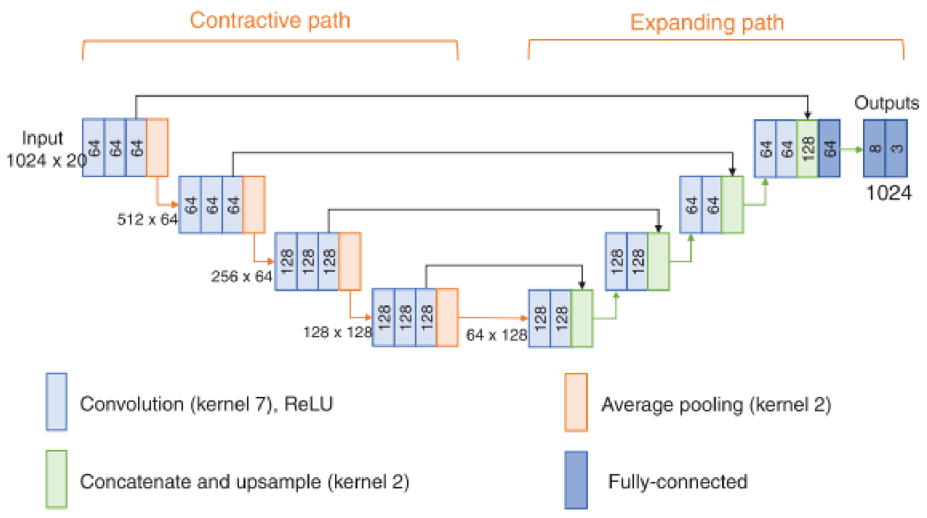


Fig. 2.8. Architecture of the ProteinUnet deep neural network. The left part represents the contractive path (encoder) and the right part refers to the expanding path (decoder) *Source:* [41]

The training of the model was done in 10 subsets of the same datasets used to train the SPIDER3-Single method. Finally, the 10 models were ensembled forming the final prediction model.

Predicted Outputs

There are two categories of outputs obtained from the ProteinUnet model: classification and regression [41]. The classification is at the same time divided into eight and three state classification.

For the eight-state classification, there are three helix conformations (310-helix, alpha-helix and pi-helix), two strand foldings (beta-bridge and beta-strand) and three coil states (high curvature loop, beta-turn and coil). The eight states are converted into a three-state classification by grouping the helix, strand and coil foldings resulting in alpha-helix (H), beta-strand (E) and coil (E). The regression outputs consist of 12 parameters including the accessible surface area (ASA), angles (ϕ, θ, ψ, ρ), half sphere exposure (HSE) separated into up and down, and contact number (CN). This is summarized in *Fig. 2.10.*

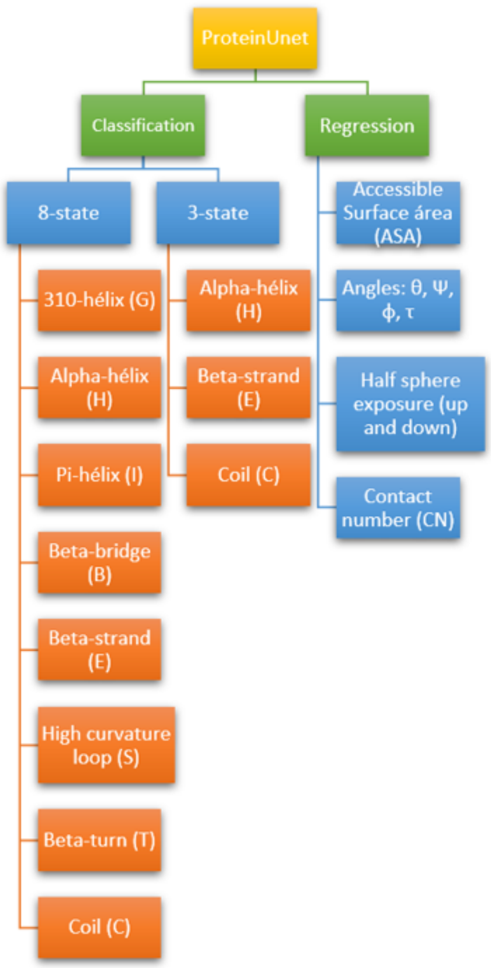


Fig. 2.9. ProteinUnet output divided into two categories: classification and regression. The classification is at the same time divided into 8-state and 3-state classification, which results from grouping helix, strand and coil states. The regression output consists of 12 parameters.

The accuracy levels reached 73.53% in ProteinUnet for 3-state classification compared to 73.18% of the SPIDER3-Single approach [41] with special precision in shorter sequences. Apart from achieving comparable results, the model is several times faster having less parameters.

2.5. Natural Language Processing

Natural Language Processing (NLP) is a machine learning field which aims to use computers to understand and manipulate human language in order to perform useful tasks

[42]. NLP combines computational linguistics, computing science and artificial intelligence. In recent years, NLP techniques have acquired great importance in the biomedical field for its applications.

The pre-training of language models has increased the effectiveness of many natural processing applications [43]. A recent introduction of NLP into the biomedical field is the BioBERT (Bidirectional Encoder Representations from Transformers for Biomedical Text Mining), a language representation model pre-trained on biomedical articles [44].

In this thesis, we will use the BERT model to pre-train language models regarding protein structure. Proteins viewed as a sequence of amino acids in their primary structure can be understood as a language. Besides, proteins in their secondary structure conformation (alpha-helix, beta-strands and coils) can be studied as a language. Therefore, they can be modeled by neural architectures developed for natural language [45].

2.5.1. Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) is a model of language representation designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers [46].

Model Architecture

The model architecture of the BERT model consists of a multi-layer bidirectional Transformer encoder [46]. The Transformer relays on attention mechanisms in order to create global dependencies between the input and the output [47]. In particular, it makes use of a self-attention mechanism, which relates positions in a sequence to compute the corresponding representation of that specific sequence.

The attention function consists of mapping a query and a collection of key-value pairs to the output [47]. This output is calculated by summing the values which have an associated weight computed by the query and its corresponding key.

Our attention function is the Scaled Dot Product Attention, where the inputs and values have dimensions of d_k, and d_v, respectively. The final result will be the dot product of the query with the keys divided by $\sqrt{d_k}$. Then, the weights are obtained by applying a Softmax function (see *Fig 2.12*). The Softmax function transforms a vector of real values into another vector of the same length and with real values that add up to 1 [48].

In practice, the attention function is computed on the matrices of queries (Q), keys (K) and values (V). The matrix of outputs is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.4}$$

For an increased usefulness, the Multi-Head Attention function was introduced, which consists of running several attention functions in parallel (see *Fig 2.12*). The queries, keys and values are linearly projected and then, the outputs resulting from the attention functions are concatenated [47]. This process allows giving attention to information coming from different representation subspaces at different points.

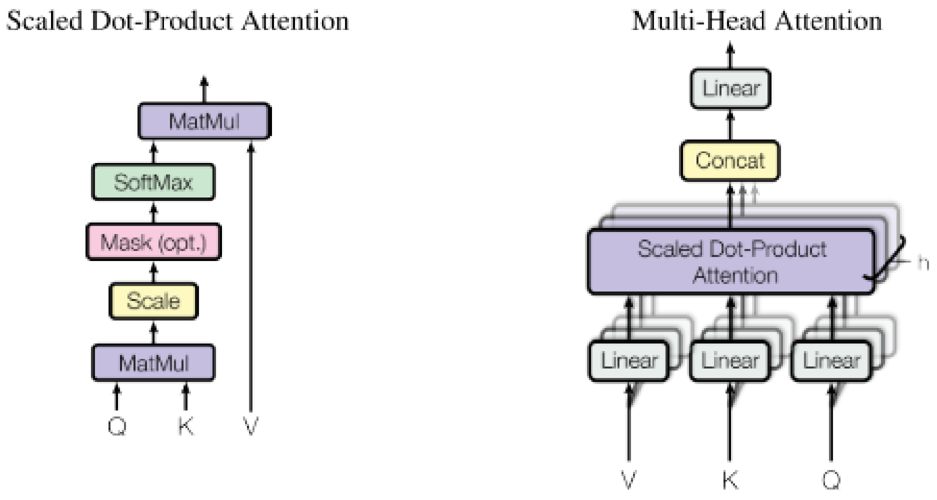


Fig. 2.10. Self-attention mechanisms used in the encoder part of the Transformer. The Scaled Dot Product Attention (left) is the basic attention function. The Multi-Head Attention function (right) runs several attention functions in parallel. *Source:* [47]

As it was stated above, the transformer includes an encoder and a decoder. First, the encoder creates a continuous representation of the input sequence of symbols. Then, the decoder produces an output sequences of symbols element by element.

For BERT, the only part of the transformer that is needed is the encoder component, as the objective is to create a language model. The encoder part is formed by a set of 6 layers, each of them having two sublayers. The first one is a multi-head attention layer and the second one a feed-forward neural network[1] (FFN). After each of these sublayers, the outputs are passed through a residual connection and a layer of normalization. Therefore, the resulting output from each sublayer is *LayerNorm(x + Sublayer(x))*, being *Sublayer(x)* the function of the actual sublayer [47].

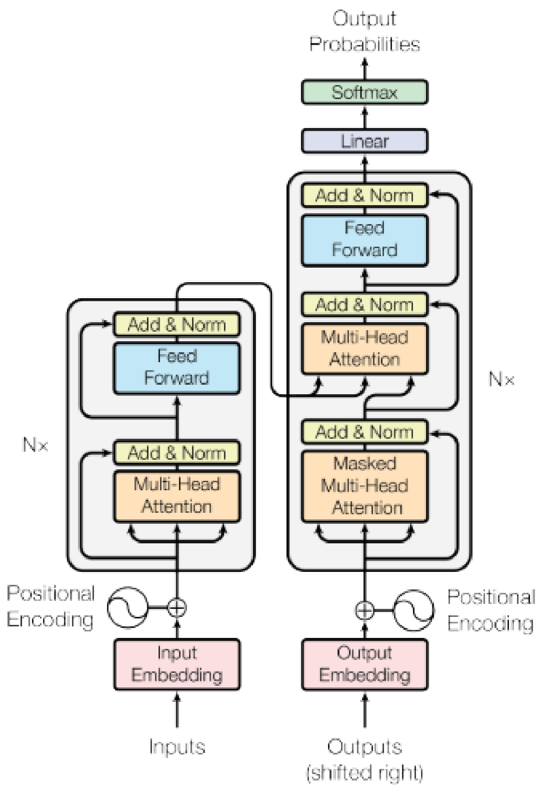


Fig. 2.11. The overall process of the Transformer with the corresponding encoder (left) and decoder (right). In the case of BERT, the only part required is the encoder component. *Source:* [47]

Input Representation

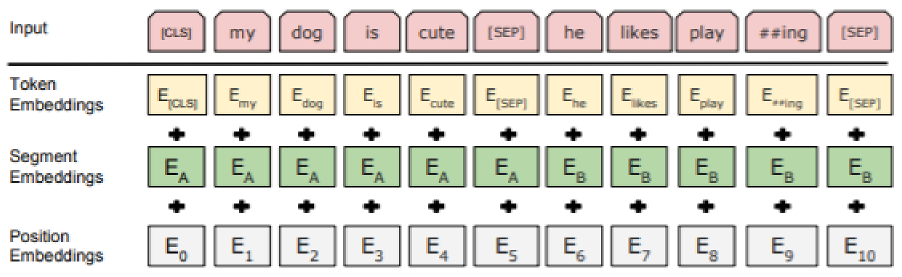
The first step before initiating the pre-training is to represent the input data as the addition of three embeddings: token, segment and position embeddings. Embeddings refer to the representation of words and sentences as vectors.

The token embeddings represent the input text as vectors. For this purpose, the sentences that form the input data are firstly separated into smaller parts (words or characters), a process referred to as tokenization [46]. Some extra tokens are added to each sentence.

[1]A feed-forward neural network (FNN) corresponds to a simple neural network in which the information is processed through hidden nodes in only one direction [49].

The token ([CLS]), which is added at the beggining of each sentence, corresponds to the special classification token, whose final hidden layer serves as the sequences' representation for conducting classification tasks [46]. For sentence pairs, a second special token is used to separate them ([SEP]).

Besides, a segment embedding is added to specify the sentence the token belongs to in the case of having more than one sentence. Finally, a last embedding is introduced to indicate the position of the token. The overall input representation is depicted in *Fig. 2.13*.



Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Fig. 2.12. BERT input representation visualized as the sum of token, segment and position embeddings. *Source:* [46]

Pre-training

BERT is pre-trained on the previously tokenized input data with two unsupervised tasks, maskel LM and next sentence prediction (NSP).

1. **Masked LM**: This procedure involves randomly masking part of the input token sequence and later predicting the masked tokens [50].

2. **Next sentence prediction (NSP)**: This task allows the model to understand sentence relationships for Question Answering (QA) or Natural Language Interference (NLI). It consists of taking two sequences and estimating if the second one follows the first sentence in the original data [47].

There exist two approaches for the application of these pre-trained models: Feature-based and fine-tuning [46]. The first one, the feature-based, refers to the pre-trained representations as additional features using task-specific architectures. On the other hand, the fine-tuning strategy is trained by fine-tuning the obtained pre-trained parameters.

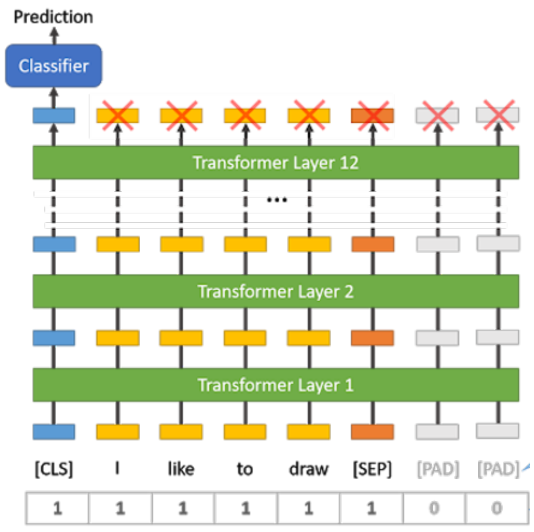
In this thesis, we will focus on the feature-based approach as we will train a neural network with the obtained features from the pre-trained BERT models.

Feature-based Approach with BERT

Depending on the task, it can be beneficial to obtain the pre-trained contextual embeddings. The term embedding is essentially referred to as the output of the final layer of the Transformer [51]. These contextualized representations consists of representing each token by a vector containing information of the context.

Pre-trained BERT models include 12 hidden layers, each with 768 hidden units or features [46]. Most of the times, classification tasks are performed using the [CLS] token, a special classification token which appears at the beginning of each sentence.

oken is the one used for classification purposes.]Hidden state of BERT with the 12 Transformer layers. The [CLS] token is the one used for classification purposes. *Source:*



[47]

Fig. 2.13. t

For this thesis, two different BERT models will be pre-trained, one on protein primary structure information and the other one on protein secondary structure information. Then, the feature-based approach will be implemented with the objective of obtaining the word-contextualized vectors from the protein structure pre-trained models to feed a neural network. The final trained model will represent the epitope predictor.

3. OBJECTIVES

This thesis has been developed at the Biocomputing Unit of the National Center of Biotechnology (CSIC). It is part of an ongoing project with the aim of creating a bioinformatic tool for predicting epitopes using data mining techniques. Part of this process was executed with the help of Paola Núñez Hernández, a Biomedical Engineering graduate student.

In this work, information regarding primary and secondary structure information of proteins is used to train a neural network that will serve as the epitope predictor. The development of an epitope predictor would facilitate the epitope-based vaccine design pipeline with many applications in cancer immunotherapy.

The first goal consisted of implementing a secondary structure prediction model to obtain the secondary structure information from the protein sequences. This way, each protein sequence would be defined by both its primary and secondary structure information. The model used was ProteinUnet, which provides fast and accurate results.

The next objective was to obtain two language models that would process and understand the protein primary and secondary structure information using the BERT architecture. After that, the word contextualized vectors would be obtained from both BERT pre-trained models to serve as features for the machine learning algorithm.

The ultimate goal was to design and train a neural network using the vectors of features corresponding to protein primary and secondary structure information. This model would constitute an epitope predictor for new protein sequences.

4. MATERIALS

4.1. Databases

The UniProt databases provide with the set of all known protein sequences along with its corresponding experimental and predicted functional information. This database was used to obtain the human proteome[2] as well as the parent protein sequences of the epitopes.

The database for obtaining the epitopes was the Immune Epitope Database (IEDB). It is a public resource that includes the catalog of antibody and T cell epitopes from humans and other animal species. This database offers 1 million peptidic epitopes from more than 4 thousand of organisms [53].

4.2. ProteinUnet

The scripts for implementing the ProteinUnet model for protein secondary structure prediction were downloaded from `https://codeocean.com/capsule/2521196/tree/v1` [41] and was implemented in Google Colab for obtaining the predictions for the all the protein sequences.

The files obtained included the two ensemble models along with the running code for accessing them and conducting the prediction. However, this code was altered for obtaining the desired outputs.

4.3. Bidirectional Encoder Representations from Transformers

The BERT code used in this thesis was downloaded from Github `https://github.com/google-research/bert`. This code contains the files for TensorFlow for performing the pre-training of the language model and the embedding for extracting the desired features from our dataset.

[2]The human proteome refers to the set of protein sequences that can be obtained by translating all the human reference genome's protein-coding genes [52].

For this thesis, the specific BERT programs used were *create_pretraining_data.py* and *run_pretraining.py* for the pretraining step and a modified version of *extract_features.py* for creating the embeddings.

4.4. Python

The programming language used in this thesis was Python. The design of the neural network model was done with Keras, an open-source python library. This tool provides simple APIs, reduces the user actions and shows clear error messages. Keras is built on top of Tensorflow 2.0 and it covers important machine learning steps, from data management to training `https://keras.io/`.

5. METHODS

The workflow including all the steps leading to the creation of the epitope predictor is depicted in *Fig. 5.1.* After obtaining the protein primary structure information, a secondary structure prediction model was implemented for obtaining that information. Then, two language models were generated. After that, the embedding vectors were calculated, which finally fed a neural network. The resulting model was the epitope predictor.

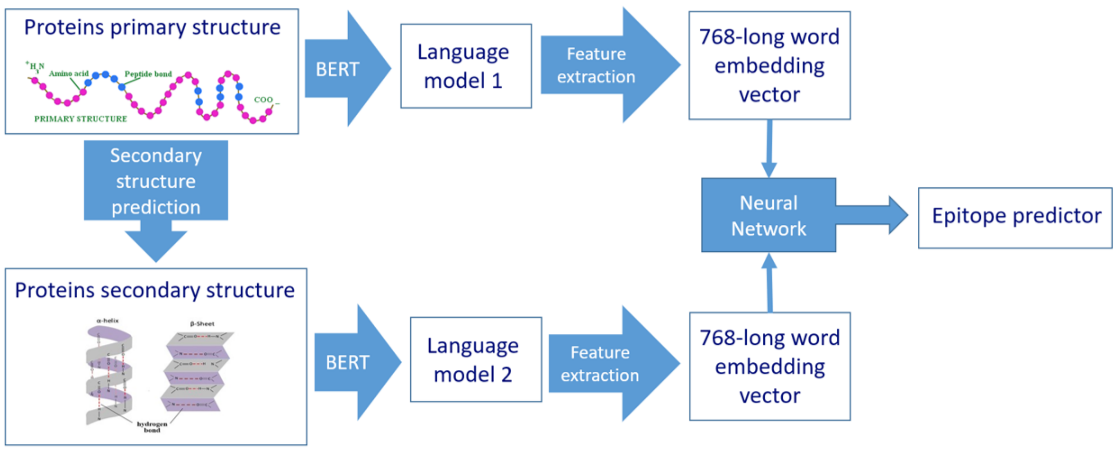


Fig. 5.1. Workflow diagram of the steps followed in this thesis which ultimately result in an epitope predictor model.

The primary structure language model as well as the corresponding feature extraction was performed by Paola Núñez. In addition, she was part of the data extraction and preprocessing steps, creating the dataset corresponding to the primary structure information.

5.1. Data Extraction

The data to conduct the experiments in this thesis was obtained using two databases: Uniprot and IEDB.

The research on human epitopes was conducted in the IEDB database. It was restricted to the parameters shown in *TABLE 5.1.*. The research was focused on MHC-I which are the molecules involved triggering of the immune response due to endogenous antigens.

TABLE 5.1. PARAMETERS FOR EPITOPE RESEARCH IN IEDB

Epitope	Linear epitope
Epitope source	Homo sapiens
Host	Human
Assay	T cell and MHC analysis
Outcome	Positive
MHC restriction	Class I
Disease	Any

The resulting set contained a set of 473,115 positive MHC-I epitopes from which we obtained the epitope and its corresponding parent protein sequence from Uniprot. The set of epitopes ranged from 7 to 24 amino acids of length, with the distribution illustrated in *Fig 5.2.*

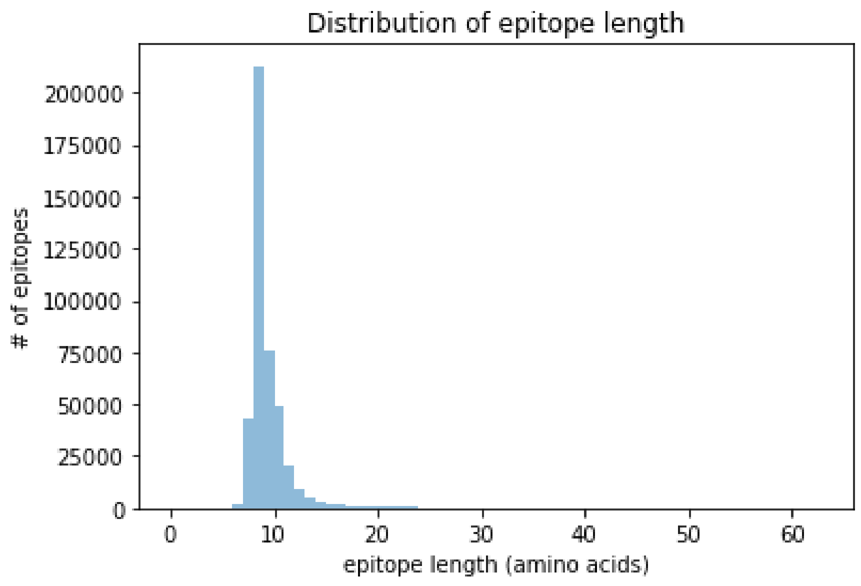


Fig. 5.2. Distribution of epitopes according to their length. Most of the epitopes have 9 amino acids.

The dataset containing the parent proteins from the obtained epitopes was divided in subsequences of 30 amino acids of length, given that the maximum epitope length was of 26 amino acids. This way, all the epitopes would be contained in one subsequence at a given point. These subsequences were obtained by passing a sliding window of 30 amino acids of length through the epitope-containing proteins creating a final set of 12 million subsequences of 30 amino acids.

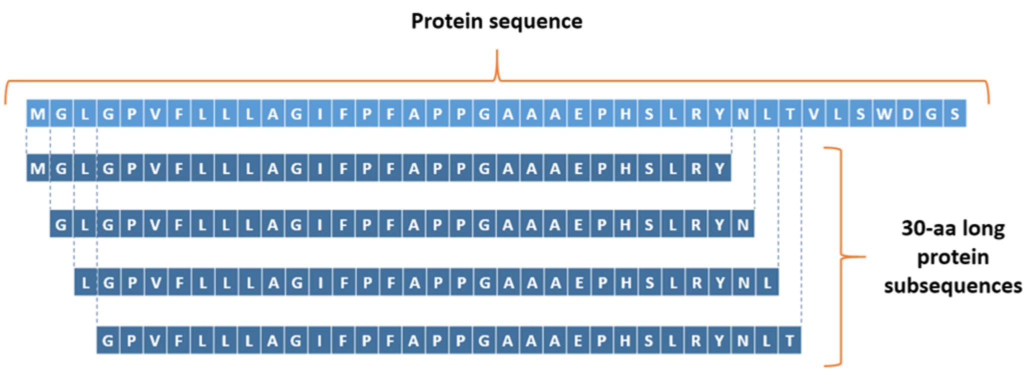

Protein sequence
M G L G P V F L L L A G I F P F A P P G A A A E P H S L R Y N L T V L S W D G S
M G L G P V F L L L A G I F P F A P P G A A A E P H S L R Y
G L G P V F L L L A G I F P F A P P G A A A E P H S L R Y N
L G P V F L L L A G I F P F A P P G A A A E P H S L R Y N L
G P V F L L L A G I F P F A P P G A A A E P H S L R Y N L T
30-aa long protein subsequences

Fig. 5.3. Sliding window procedure example in the MHC class I polypeptide-related sequence A. The light blue sequence represents the whole protein sequence while the dark blue regions refer to the 30-long subsequences resulting from passing the sliding window.

The features extracted from these subsequences were later used to train the neural network, a supervised learning. Therefore, they needed to be labeled. The subsequences were labeled negative (non-epitope) or positive (epitope) checking if one of the following conditions was fulfilled by each of the sequences:

- An epitope is entirely contained in the subsequence.
- 50% or more of the subsequences is composed by one or more epitope regions.

The final set contained 2.803.003 epitope sequences and 9.196.997 non-epitope sequences.

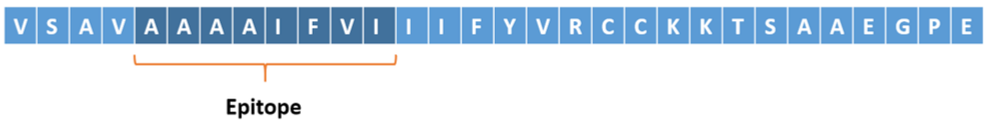


Fig. 5.4. Example of a positive epitope 30-long subsequence of protein MHC class I polypeptide-related sequence A and the epitope AAAAIFVI (dark blue).

After that, the complete human proteome was extracted from Uniprot. This time, the resulting set was divided in subsequences of 30 consecutive amino acids, as it is depicted in *Fig 5.4.* The length of the subsequences was set again to 30 amino acids, corresponding to the length of the previously obtained subsequences.

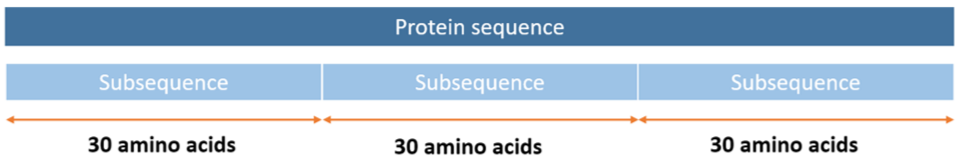


Fig. 5.5. The human proteome obtained sequences were divided into consecutive sequences of 30 amino acids.

The final set was later used to pre-train the language model of the primary structure and, after obtaining the secondary structure, to pre-train the other language model for the secondary structure. As the language model is unsupervised learning, these subsequences were not labeled.

5.2. Secondary Structure Prediction

The secondary structure prediction of the obtained protein sequences was implemented using the ProteinUnet model, a method based on the U-Net architecture. This model was chosen as it had open-source and easy to implement scripts, the prediction time was low and, it had shown great accuracy prediction results.

The ProteinUnet model provides two output categories: classification and regression. The model predicts for classification both eight and three states. On the other hand, the regression output includes twelve other parameters. For simplicity of the problem, the 3-state classification was chosen to be the required output. Therefore, each amino acid was classified into alpha-helix (H), beta-strand (E) or coil (C).

A Python code was written in a way that the only output of the prediction was the 3-state classification. This reduced considerably the time for obtaining the secondary structure information. In addition, this code analyzed each of the input sequences to ensure that they were composed of upper case single-letter amino acids belonging to the set of 20 amino acids. As some sequences coming from the human proteome included unknown letters, those were excluded from the dataset.

The accuracy of the obtained results was checked with Uniprot information on secondary structure by comparing the results obtained from ten epitope-containing proteins. The comparison was based on calculating the number of successes (S) and errors (E) of the predicted structure with respect to the known regions provided by the Uniprot database and dividing by the length of those regions (L). Then, the accuracy was obtained as the average of the calculated accuracy for each of the proteins:

$$accuracy = \frac{\sum_{n=1}^{n=10} \frac{S}{L}}{n} \tag{5.1}$$

This result represented an approximated accuracy of the prediction using the Protein-Unet model.

With this model, the secondary structure of the protein sequences of the two datasets (whole human proteome and epitope/non-epitope sequences) was predicted. This resulted in two additional datasets with the corresponding secondary structure information. Special care was taken to ensure that each protein subsequence in one dataset corresponded to the correct secondary structure information in the other dataset.

5.3. Bidirectional Encoder Representations from Transformers

The protein language modeling and feature extraction were performed using BERT. The necessary scripts for performing the pre-training of the language models and the feature extraction were downloaded from GitHub. However, two additional files were created before initiating the processes for each of the models: a vocabulary file and a configuration file.

The vocabulary file was a text file that included a dictionary of the tokens in which the sequences were divided. For the case of the primary structure, the dictionary included the different letters representing the amino acids. On the other hand, the dictionary of the secondary structure model was composed by the three folding patters. Both also included the special tokens.

In the case of the configuration file, the JSON file contained the hyperparameters of each of the models, which are shown in *Table 5.2*.

TABLE 5.2. HYPERPARAMETERS FOR BERT CONFIGURATION

hidden_act	"gelu"
hidden_size	768
intermediate_size	3,072
max_position_embeddings	30
num_attention_heads	12
num_hidden_layers	12
vocab_size	30/8

The *hidden_act* refers to the activation function in the transformations of the FNN in the encoder, which is the Gaussian Error Linear Unit (GELU). The *hidden_size* represents the dimensions of the hidden units of the encoded layers in the FNN while the *intermediate_size* is the number of hidden layers.

The *num_attention_heads* corresponds to the number of encoding parts that form the transformer and the *num_hidden_layers*, the number of hidden layers of these enconding parts.

The only different parameter was the *vocab_size* which corresponds to the number of tokens used in the representation of all the input sequences. It includes 30 components in the primary structure model and 8 in the secondary structure model.

5.3.1. Protein Language Modeling

The protein language models for the protein primary and secondary structures were obtained by pre-training two different BERT models. The protein models were trained on the human proteome datasets with primary and secondary structure information for creating a more generalized language model. Each of the datasets was transformed into a text file with a different sequence in each line and with no labels.

The next step was to run the program *create_pretraining_data.py* on Python. This program creates a TFRecords file, Tensorflow's format that stores the data as binary string sequences. The selected parameters for running the algorithm for both models are shown in *TABLE 5.3.*

TABLE 5.3. PARAMETERS FOR CREATING PRE-TRAINING DATA

max_seq_length	30
max_predictions_per_seq	5
maskel_lm_prob	0.15
dupe_factor	5

The *max_seq_length* parameter was set to 30 as it is the maximum and only length of the input sequences. The *max_predictions_per_seq* is the maximum number of masked predictions for each sequence. The *maskel_lm_prob* was calculated multiplying the two previous parameters. The *dupe_factor* refers to the number of duplication the input data suffers for the training.

The final step for the pre-training of the models was to run the code *run_pretraining.py* with the parameters that appear in *TABLE 5.4.*

TABLE 5.4. PARAMETERS FOR PRE-TRAINING BERT MODEL

train_batch_size	32
max_seq_length	30
max_predictions_per_seq	5
num_train_steps	500,000/14,000,000
learning_rate	1e-4

The *train_batch_size* for the pretraining refers to the number of sequences the algorithm uses every iteration, and it was set to 32 as it was the maximum number the used GPUs could support. The *max_seq_length* and *max_predictions_per_seq* parameters passed to this code must be the same as *create_pretraining_data.py*. The *num_train_steps* differed between the two models. For the primary structure model, the greater number of vocabulary items required more training steps (14,000,000) than the secondary structure model (500,000).

5.3.2. Feature Extraction

After obtaining the pre-trained models for the primary and secondary structure sequences, the embeddings were extracted running the code *extract_features.py* with the parameters shown in *Table 5.5.* The input datasets in this case were the sets containing the epitope and non-epitope sequences in their primary and secondary protein structure.

However, this code was altered in order to get smaller files with just the required information. Therefore, it was set to give as output a text file with the information on token [CLS], which contains the information about the whole sequence and is used for classification purposes.

TABLE 5.5. PARAMETERS FOR FEATURE EXTRACTION

max_seq_length	30
batch_size	32
layers	-1

The *max_seq_length* was again set to 30 as all the sequences were of that length. The selected layer for obtaining the word embeddings of the protein sequences was layer *-1* which is the first hidden layer or word embedding layer. The output generated were 768-long vectors corresponding to the [CLS] token.

The final files contain all the 768-long vectors corresponding to each of the initial protein sequences. Therefore, an initial protein sequence has two associated feature vectors, one for each of the models (primary structure and secondary structure models).

5.4. Neural Network

The neural network was implement with the Keras library of Python. The input features of the developed neural network were the word contextualized embeddings previously obtained from the primary and secondary structure datasets. As it has been stated above, each protein sequence had two corresponding 768-long feature vectors.

There were two proposed network architectures to assess this problem: single-branch and double-branch (see *Fig. 5.5.*). The single-branch model receives as input 1536-long vectors which result from the concatenation of a vector of 768 values. On the other hand, the double-branch architecture receives 768-long vectors in each branch, each corresponding to primary or secondary structure embeddings.

Two first models (model 1 and model 2) were created, each based on one of the proposed architectures, for training a subset of 100,000 sequences. Then, the promising architecture was chosen for the creation of the last model (model 3) for training the whole dataset of 12 million sequences.

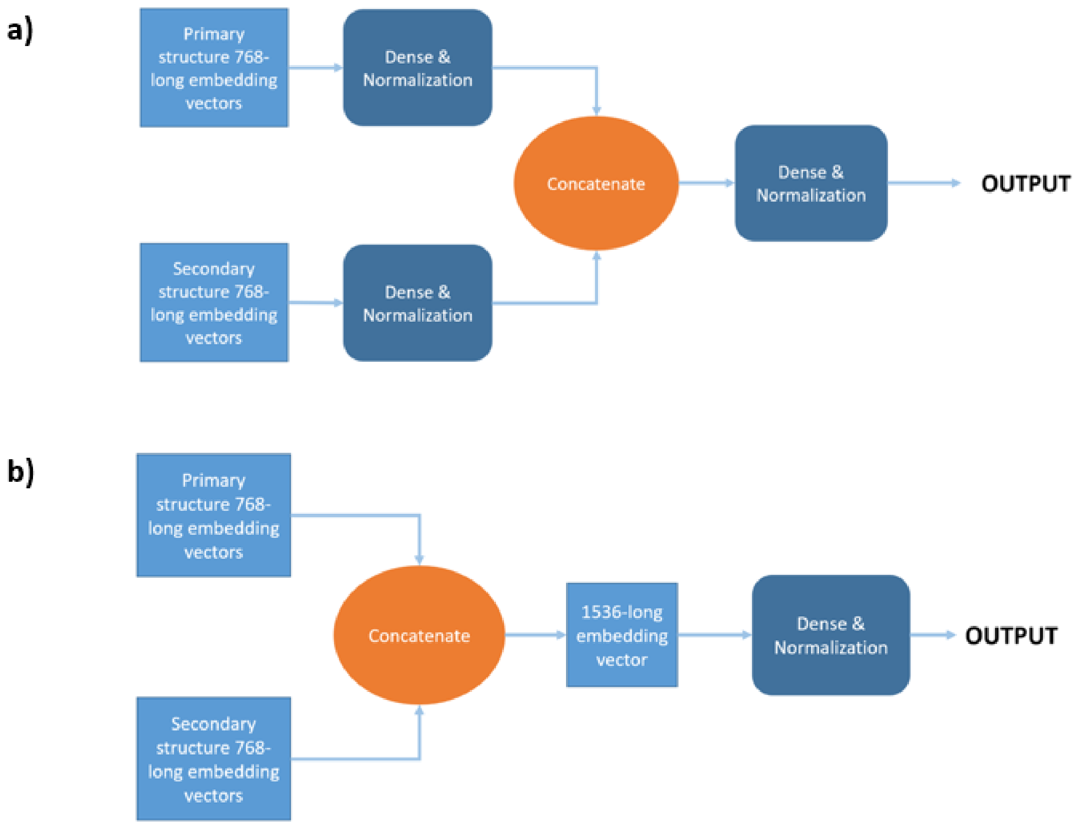

a)
Primary structure 768-long embedding vectors
Dense & Normalization
Concatenate
Dense & Normalization
OUTPUT
Secondary structure 768-long embedding vectors
Dense & Normalization
b)
Primary structure 768-long embedding vectors
Concatenate
1536-long embedding vector
Dense & Normalization
OUTPUT
Secondary structure 768-long embedding vectors

Fig. 5.6. a) Single-branch architecture which receives as input 1536-long vectors which result from the concatenation of two vector of 768 values. b) Double-branch architecture which receives 768-long vectors in each branch.

Model 1

The first model was based on the single-branch architecture, which receives a 1536-long vector resulting from the previous concatenation of two 768-long vectors, each containing primary or secondary structure information. This network consisted of five dense layers, one batch normalization layer and a dropout layer of 0.2. The activation for the first four layers was ReLU while the last layer had Sigmoid activation.

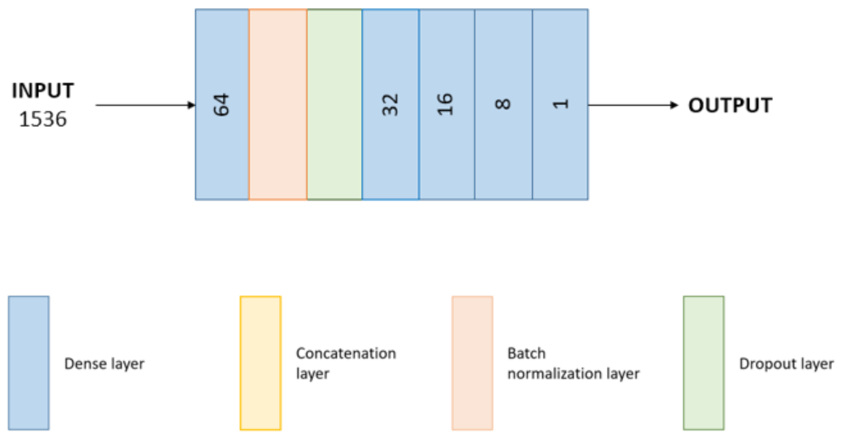


Fig. 5.7. Model 1 design based on single-branch architecture containing 5 dense layers (blue), 1 batch normalization layer (red) and 1 dropout layer (green).

This model was trained with the subset of 100,000 sequences, which were previously divided into train and validation sets as it is depicted in *Fig. 5.7.* In every step of each epoch, the batch was created using a data generator which ensured that the same number of positive and negative samples were taken for training in order to keep the batch balanced. The same procedure was implemented for the validation steps. The *steps per epoch* parameter was calculated, dividing the total number of input samples by the *batch size*. For the case of the *validation steps*, the number of validation samples was divided by the *batch size*.

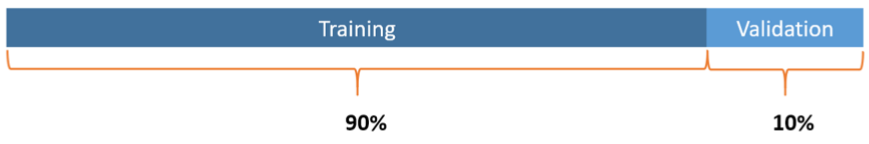


Fig. 5.8. Training and validation split of the dataset. The training included the 90% of the sequences while the validation the rest 10%.

The chosen loss function was the binary crossentropy loss function and the optimization algorithm was Adam.

Model 2

The second proposed model consisted of two branches which are later concatenated. The first branch has as input the feature vectors from the primary structure model, while the second branch takes the feature vectors from the secondary structure model.

Each of the branches contains an input layer, a dense layer, a batch normalization layer and a dropout of 0.2. Afterwards, they are concatenated into a single layer. The final part consists of 5 dense layers with two dropouts of 0.2 and a batch normalization layer. The activation functions for the dense layers were ReLU except for the last one which was Sigmoid.

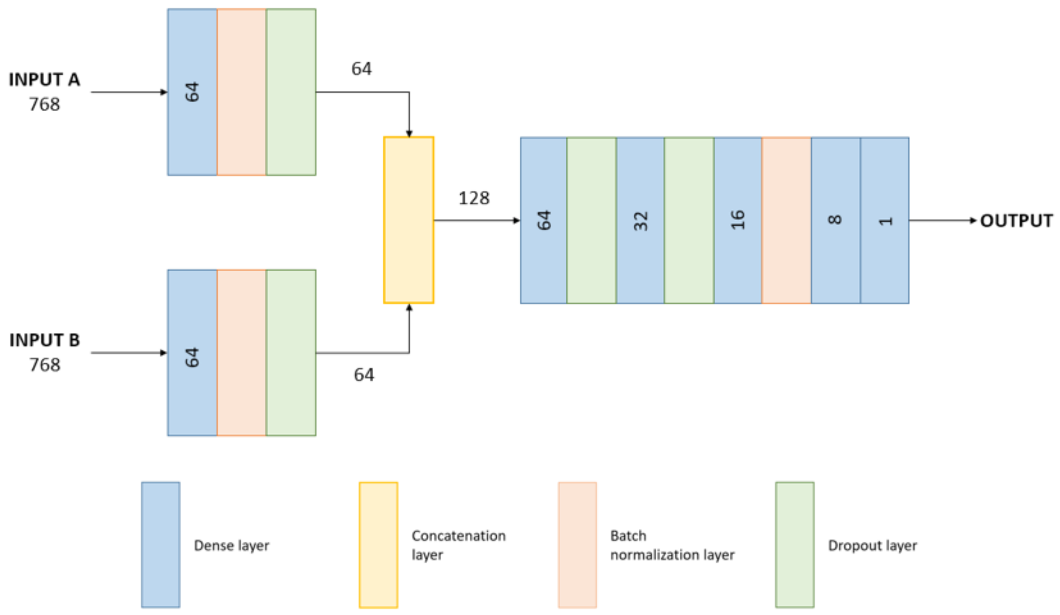


Fig. 5.9. The two-branch architecture of the Neural Network which receives vectors of 768 values with primary or secondary information in each of the branches. Input A refers to the primary structure embedding while input B corresponds to the secondary structure embedding vectors.

This model was also trained on the subset of 100.000 sequences and the training procedure followed was exactly the same as for model 1.

Model 3

This third model was designed after model 2 which showed slightly more promising results than model 1, as it can be observed in the results section. Therefore, the design for model 3 was based on the double-branch architecture.

Each branch contains 3 dense layers, a batch normalization layer and a dropout of 0.2. Then, they are concatenated into a single layer. The rest contains 6 additional dense layers, a batch normalization layer and two dropouts of 0.2. The activation function for all the dense layers was ReLU, except for the last one, which was Sigmoid.

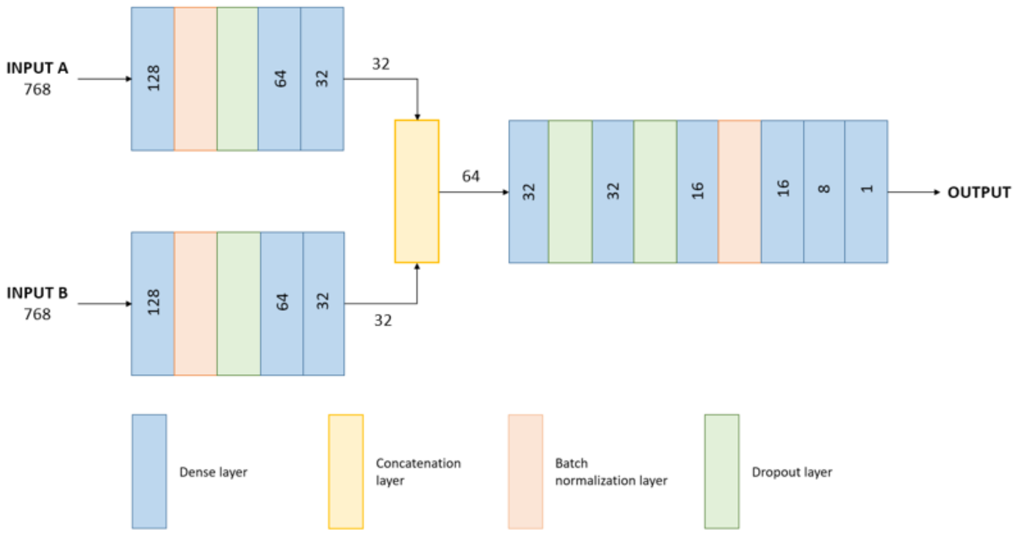


Fig. 5.10. The third model implemented in the whole dataset was based on the double-branch architecture which received vectors of 768 values in each branch. Input A refers to the primary structure embedding while input B corresponds to the secondary structure embedding vectors.

Model 3 was trained using the whole dataset containing 12 million epitope/non-epitope sequences and their corresponding primary and secondary structure features. The training process in this case was more complex than the previous one.

Before initiating the training, it was important to divide the large datasets of features into smaller files containing 100,000 sequences, given that the RAM could not operate with such large files. As a result, 120 files from each large feature datasets were obtained. They were named carefully to ensure that each primary structure features file corresponded to the correct secondary structure features file.

After starting the training, the training and validation set was updated randomly taking two associated files (primary structure features and secondary structure features) every two epochs. This set of 100,000 sequences was divided into training and validation sets as shown in Fig. 5.7..

For the loss function and optimization algorithm, binary crossentropy and Adam methods were chosen.

Model 4

One last model was designed, again with the double-branch architecture, to solve overfitting issues increasing the number of dropout layers. It was basically an improvement of model 3 but adding 4 extra layers of dropout and changing the dropout to 0.3 (see *Fig. 5.11.*)

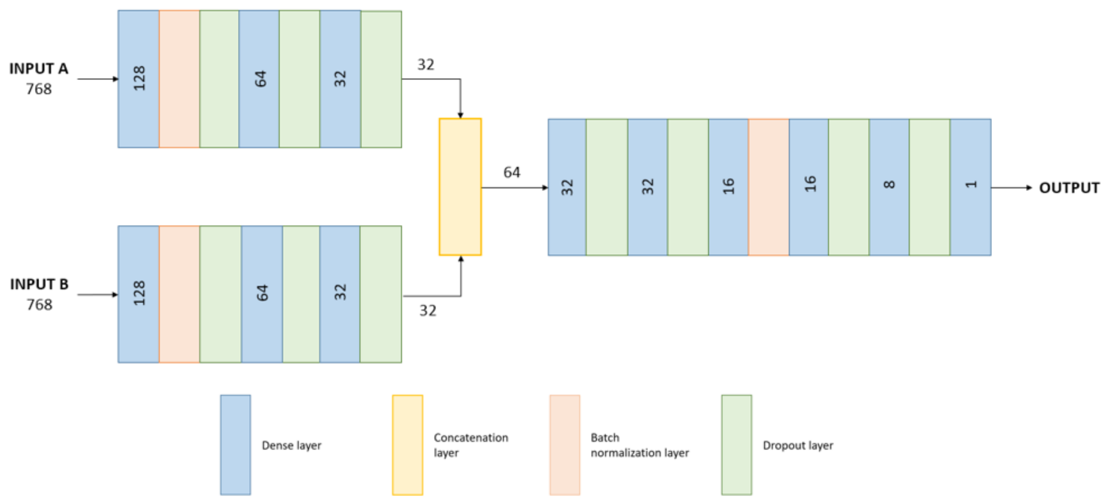


Fig. 5.11. The forth and final model implemented in the whole dataset was based on the double-branch architecture which received vectors of 768 values in each branch. Input A refers to the primary structure embedding while input B corresponds to the secondary structure embedding vectors.

This model was again trained in the whole dataset containing 12 million sequences following the training process described above for model 3.

Testing and result processing

The trained model 4 was tested on a protein sequence with 2 known epitope regions. The chosen protein was Eukaryotic translation initiation factor 4 gamma 1, composed by 1560 amino acids, which contains two known epitopes (see *Table*). This protein was chosen as it contained only 2 known epitopes and thus, the epitope regions did not overlap which could have made the correct assessment of the prediction accuracy more difficult.

This protein comprised 1531 subsequences after the sliding window of 30 amino acids was applied. At the same time, each of the epitopes created a set of positive subsequences which depended on the epitope length (L):

$$\text{size} = 30 - L + 1 \tag{5.2}$$

TABLE 5.6. EPITOPE SEQUENCES IN EUKARYOTIC TRANSLATION INITIATION FACTOR 4 GAMMA 1

EPITOPE	**LENGTH (aa)**	**LENGTH (subsequences)**
YYPAQGVQQF	10	21
AARPATSTL	9	22

After conducting the prediction, the results were compared to the actual labels. As it was explained in the data extraction part, the subsequence was considered positive or epitope-containing if a complete epitope was contained in the selected region or at least half of the subsequence was part of one or more epitopes. The accuracy of the model on the sequence was obtained by calculating the amount of correctly predicted labels.

Then, the results were analyzed and processed, implementing a noise reduction system. This noise reduction system considered the minimum size of a set of positive subsequences that could be predicted, given that the maximum length of an epitope (26) comprises 5 subsequences.

Finally, the epitope predictor approach was completed with all the steps to follow when using a novel protein sequence.

6. RESULTS AND DISCUSSION

6.1. Secondary Structure Prediction

The secondary structure prediction model (ProteinUnet) was executed in Google Colab for several weeks in order to obtain the secondary structure information from the human proteome and the epitope-containing sequences.

The obtained results consisted of a 3-state classification of each amino acid into alpha-helix (H), beta-strand (E) or coil (C) as shown in *Fig. 6.1.*

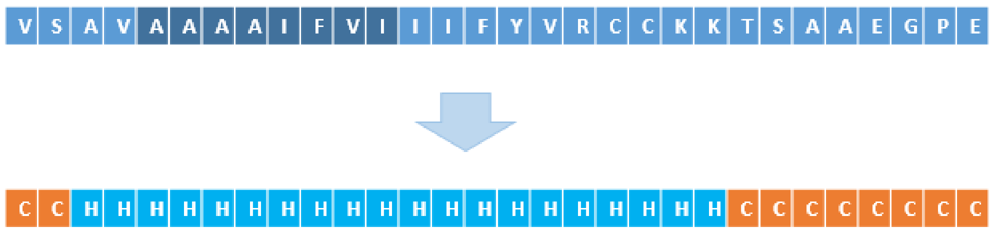

V S A V A A A A I F V I I I F Y V R C C K K T S A A E G P E
C C H H H H H H H H H H H H H H H H H H H H C C C C C C C C

Fig. 6.1. Secondary structure prediction example of the resulting 3-state classification of a sequence of 30 amino acids containing an epitope (dark blue). H (light blue) represents alpha-helix and C (orange), coil state.

An approximate accuracy was calculated from the obtained results checking 10 epitope-containing proteins. The successes, length and resulting accuracy are depicted in *Table 6.2.*

The average resulted in a 0.7089 accuracy value, which corresponds to the 0.7358 accuracy shown by the ProteinUnet model in other datasets. In addition, the computational time of calculating the 3-state classification was lowered.

TABLE 6.1. PROTEINUNET ACCURACY RESULTS

PROTEIN	**SUCCESSES**	**LENGTH**	**ACCURACY**
MHC class I related protein A	117	196	0.5969
Vimentin	193	203	0.9507
Interleukin-3 receptor subunit alpha	94	170	0.5529
Beta-2-microglobulin precursor	41	73	0.5616
Laminin-B1	120	137	0.8759
Histone H4	50	74	0.6757
Prostatic acid phosphatase precursor	170	233	0.7296
Ribosomal protein P2	32	53	0.6038
Whirlin	101	139	0.7266

6.2. Language Models

The pre-training of the two BERT models for the development of the language models was run for several weeks in four different GPUs (NVIDIA GeForceGTX 1070), two for each pre-training.

The two pre-trained BERT models on primary and secondary structure information showed similar accuracies. However, the model corresponding to the secondary structure information achieved a higher accuracy in less steps than the primary structure model did. Both results are illustrated in *Table 6.1.* and *Table 6.2.*

TABLE 6.2. PRIMARY STRUCTURE PRE-TRAINED MODEL RESULTS

global_step	14,000,000
masked_lm_accuracy	0.79
masked_lm_loss	0.3218

TABLE 6.3. SECONDARY STRUCTURE PRE-TRAINED MODEL RESULTS

global_step	500,000
masked_lm_accuracy	0.89
masked_lm_loss	0.2606

The *global_step* referring to the number of samples that have passed through the model differed between the two BERT models. This number was higher for the primary structure model, which contained 30 vocabulary items. On the contrary, the secondary structure vocabulary was formed by 8 tokens, which resulted in a lower number of steps for reaching the desired accuracy. It can be concluded that the greater vocabulary size, the higher number of steps are needed to reach meaningful results.

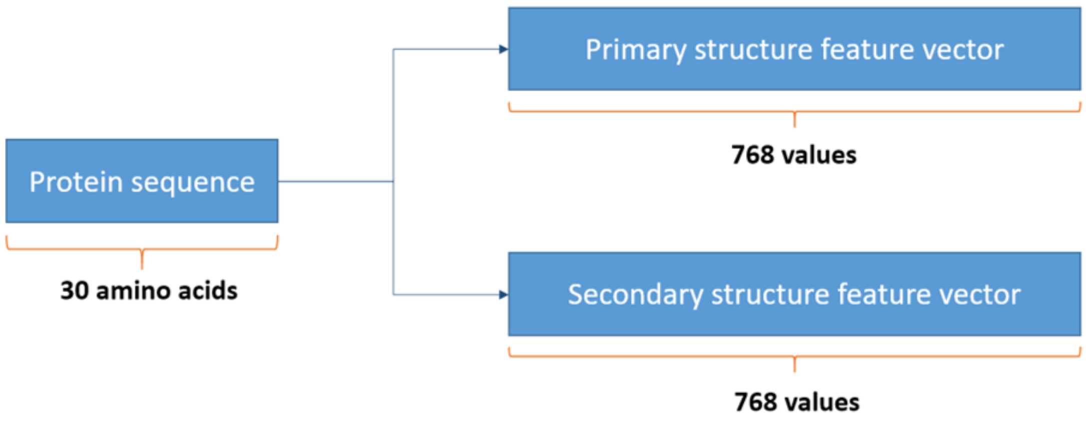
The accuracy for the secondary structure model was slightly higher than the one from the primary structure model. At the same time, the secondary structure model loss is lower than the loss from the primary structure model. For both cases, the accuracy increased with higher number of steps and the loss decreased over the steps, which agrees with the results of a common machine learning model.

6.3. Feature extraction

The next step was to extract the word embeddings of the epitope and non-epitope datasets using the created language models on primary and secondary structure information. This process required two weeks and four GPUs (NVIDIA GeForceGTX 1070), two for each of the models.

The resulting files contained 12 million vectors of 768 values corresponding to the [CLS] special token of classification. These values ranged from -1 to 1.

Each initial protein sequences had, as a result, two associated 768-long feature vectors, each from one model (see *Fig 6.2.*). This corresponds to having 1536 features which were fed into a neural network.



Protein sequence
30 amino acids
Primary structure feature vector
768 values
Secondary structure feature vector
768 values

Fig. 6.2. Each protein sequence of 30 amino acids has two associated vectors (primary and secondary structure features) resulting from the feature extraction process. Both have 768 values from -1 to 1.

6.4. Neural Network

The results of the training using the different designed models are shown in this section. Firstly, the two generated models were trained using a subset of 100,000 sequences, which permitted a rapid execution for tuning and evaluating the results. After choosing the best architecture, the other two models were trained on the large dataset.

Model 1 and Model 2

The two first models trained with the subset of 100,000 sequences were ran in one GPU (NVIDIA GeForce GTX 950M). The results obtained from both models were slightly different.

The first model was the single-branch architecture design which received as input a set of concatenated 1536-long vectors from the primary and secondary structure features. The results from the training and evaluation can be observed in *Fig 6.3*.

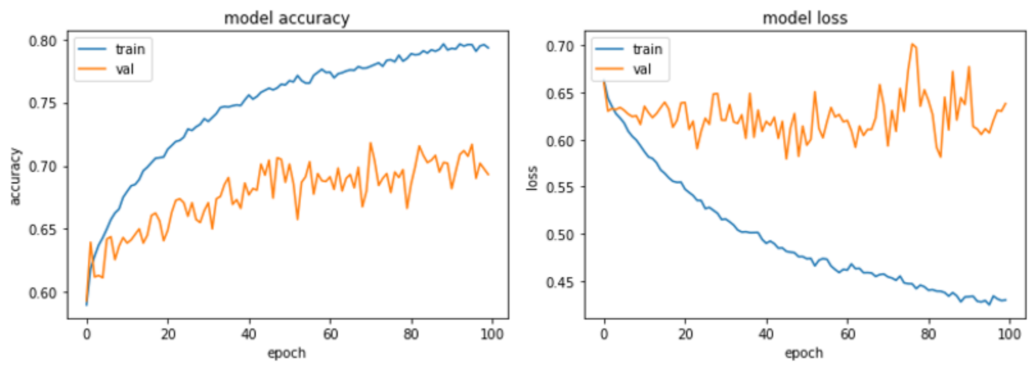


Fig. 6.3. Model 1 results. The validation accuracy (left) converged at 0.7 while the model loss (right) was maintained around 0.65.

The training reached an accuracy level of 0.8 while the evaluation converged at 0.7. Although the evaluation is normally lower than the training, this difference could have been associated with overfitting of the model, which corresponds to an extremely close fit to a limited data samples. In addition, the loss of the validation set does not have a decreasing tendency, which also corresponds to overfitting.

The second model was the double-branch architecture design. Each of the branches received as input 768-long vectors from primary and secondary structure embeddings. These vectors were later concatenated.

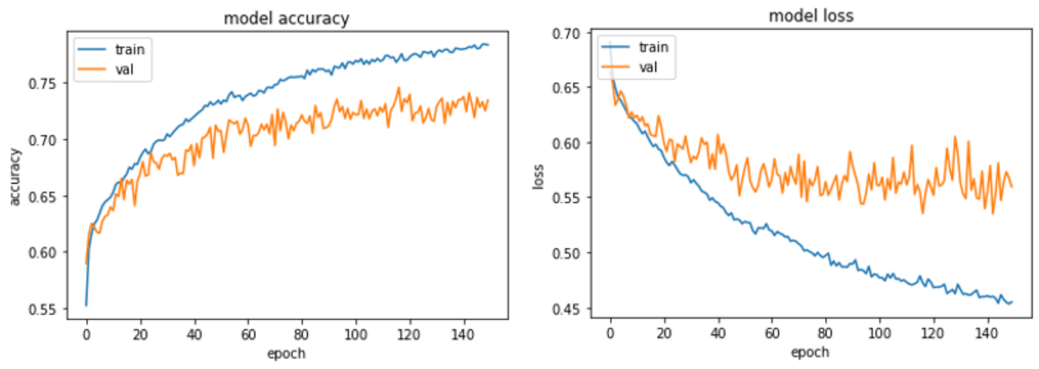


Fig. 6.4. Model 2 results. The validation accuracy (left) converged at 0.73 while the model loss (right) was maintained around 0.57.

In this case, the accuracy was 0.78 in the training process and the evaluation converged at 0.73. In this case, the evaluation process is closer to the accuracy values. Moreover, the loss is lower than for model 1 showing a better decreasing tendency.

It could be concluded that the double-branch model resulted in a better performance that the first model. Although the training reached slightly higher accuracy, the evaluation process showed a better overall behavior. Therefore, the double-branch architecture was chosen to create the third and final model for the training of the whole dataset.

Model 3 and Model 4

The final two models were trained on the full dataset of 12 million of sequences in one GPU (NVIDIA GeForceGTX 1070). As this training had the particularity of changing the training and validation file every two epochs, the training was left to perform 300 epochs so that the model could be trained in all the 120 files.

For model 3, the accuracy showed a consistent behavior increasing the values for higher number of epochs and being slightly lower for the case of validation. On the other hand, the model loss for the training decreased while the validation loss did not. The results obtained can be observed in *Fig. 6.5.*

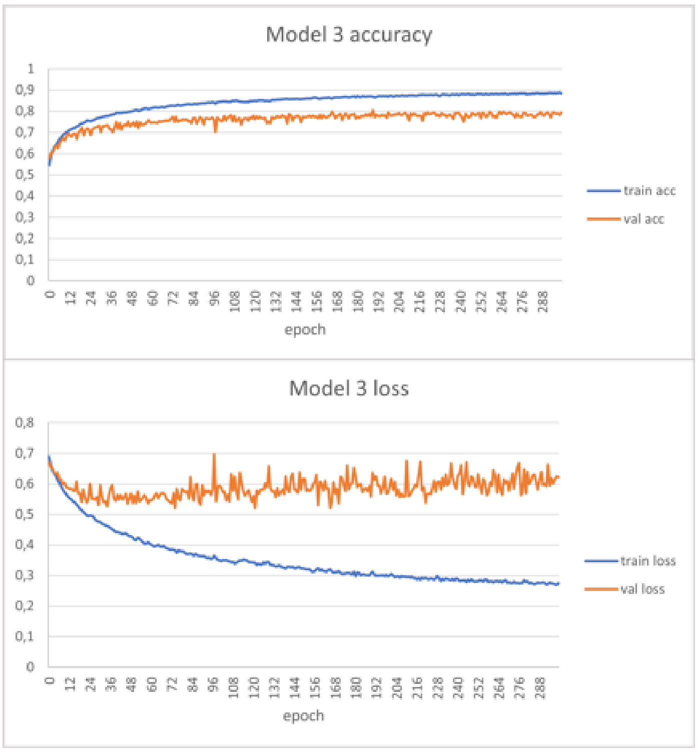


Fig. 6.5. Model 3 results. The accuracy (top) showed an increasing accuracy while the loss (bottom) slightly increased.

The training accuracy reached a value of 0.9. The validation process converged at 0.8. In terms of the loss, it stayed between values of 0.5 and 0.6. This can be explained with overfitting. The model learns specific patterns in the training data and, thus, fails to generalize when validating with new data.

With the improvements implemented in model 4 which consisted of adding dropout layers, the overfitting was slightly lowered which can be observed in the validation loss (see *Fig 6.6.*). However, the accuracy was lowered to a maximum of 0.7 in the validation process.

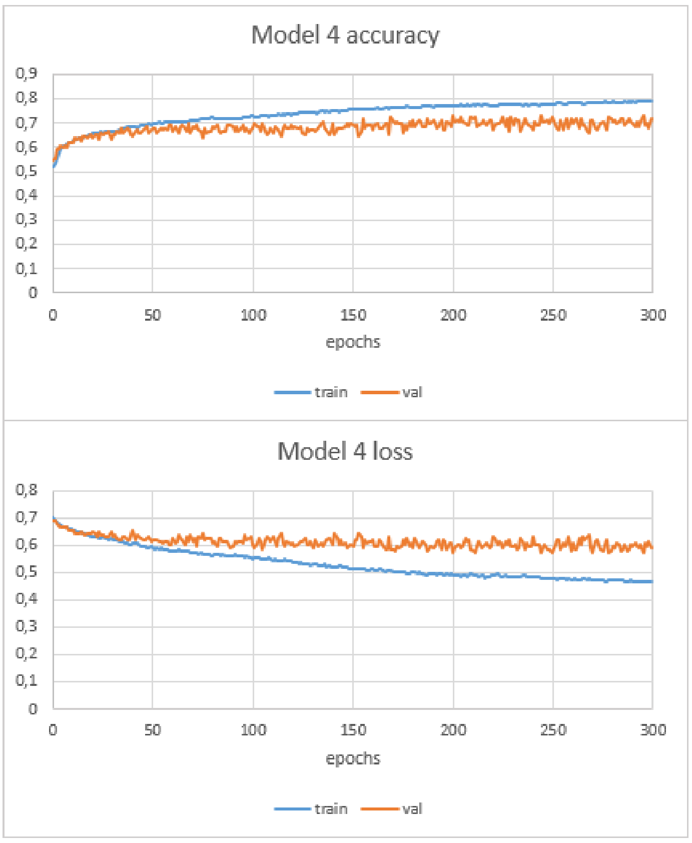


Fig. 6.6. Model 4 results. The accuracy (top) showed an increasing accuracy while the loss (bottom) was maintained.

The overall behavior of model 4 was improved in terms of overfitting. However, given that the loss was still higher than expected, the model was tested in a whole protein in order to analyze the results.

Testing and result processing

The last model was tested on a protein (Eukaryotic translation initiation factor 4 gamma 1) with two known epitope regions. The 1560 amino acids that form the protein sequence comprised 1531 subsequences.

The overall results of the 1531 subsequences of the protein can be observed in the confusion matrix (see *Fig. 6.7*). Focusing on the known epitope regions, the accuracy for epitope **YYPAQGVQQF**(21 subsequences) was 71% (see *Table 6.4.*) and for epitope **AARPATSTL** (22 subsequences) was 82% (see *Table 6.5*). These two tables are a visual representation of the set of positive subsequences each epitope creates with the predicted output from the model.

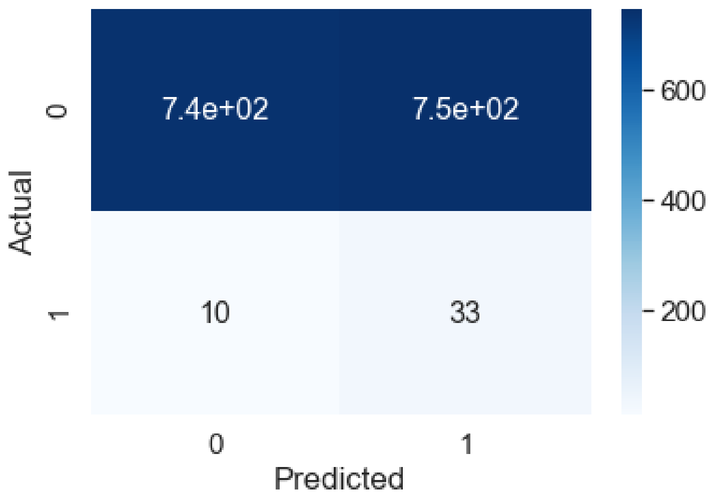


Fig. 6.7. Confusion chart for model 4 tested in the protein Eukaryotic translation initiation factor 4 gamma 1. The upper left section refers to the True Negatives and the lower right section represents the True Positive, both correspond to correct predictions. The upper right part represents the non-epitope sequences predicted as positive (False Positives) and the lower left values are the epitope sequences predicted as negative (False Negatives).

The resulting accuracy was of 53%. The results show a great number of false positives, meaning that those subsequences do not contain any epitopes but were predicted as such. This could be related to having noise in the protein sequence. As the minimum size of a set of subsequences comprising an epitope is 5, consecutive subsequences of size lower than 5 predicted as epitopes could be removed as they could be assumed as noise. After removing these sections, the accuracy increased to a 74% in the whole protein.

TABLE 6.4. RESULTS FOR EPITOPE YYPAQGVQQF

SUBSEQUENCE	**CORRECT**	**PREDICTED**
GAPGFYPGASPTEFGTYAGA**YYPAQGVQQF**	1	0
APGFYPGASPTEFGTYAGA**YYPAQGVQQF**P	1	1
PGFYPGASPTEFGTYAGA**YYPAQGVQQF**PT	1	0
GFYPGASPTEFGTYAGA**YYPAQGVQQF**PTG	1	1
FYPGASPTEFGTYAGA**YYPAQGVQQF**PTGV	1	1
YPGASPTEFGTYAGA**YYPAQGVQQF**PTGVA	1	1
PGASPTEFGTYAGA**YYPAQGVQQF**PTGVAP	1	1
GASPTEFGTYAGA**YYPAQGVQQF**PTGVAPT	1	1
ASPTEFGTYAGA**YYPAQGVQQF**PTGVAPTP	1	1
SPTEFGTYAGA**YYPAQGVQQF**PTGVAPTPV	1	1
PTEFGTYAGA**YYPAQGVQQF**PTGVAPTPVL	1	1
TEFGTYAGA**YYPAQGVQQF**PTGVAPTPVLM	1	0
EFGTYAGA**YYPAQGVQQF**PTGVAPTPVLMN	1	0
FGTYAGA**YYPAQGVQQF**PTGVAPTPVLMNQ	1	0
GTYAGA**YYPAQGVQQF**PTGVAPTPVLMNQP	1	0
TYAGA**YYPAQGVQQF**PTGVAPTPVLMNQPP	1	1
YAGA**YYPAQGVQQF**PTGVAPTPVLMNQPPQ	1	1
AGA**YYPAQGVQQF**PTGVAPTPVLMNQPPQI	1	1
GA**YYPAQGVQQF**PTGVAPTPVLMNQPPQIA	1	1
A**YYPAQGVQQF**PTGVAPTPVLMNQPPQIAP	1	1
YYPAQGVQQFPTGVAPTPVLMNQPPQIAPK	1	1

TABLE 6.5. RESULTS FOR EPITOPE AARPATSTL

SUBSEQUENCE	**CORRECT**	**PREDICTED**
LSWGKGSSGGSGAKPSDAASE**AARPATSTL**	1	0
WGKGSSGGSGAKPSDAASE**AARPATSTL**NR	1	1
GKGSSGGSGAKPSDAASE**AARPATSTL**NRF	1	1
KGSSGGSGAKPSDAASE**AARPATSTL**NRFS	1	1
GSSGGSGAKPSDAASE**AARPATSTL**NRFSA	1	1
SSGGSGAKPSDAASE**AARPATSTL**NRFSAL	1	1
SGGSGAKPSDAASE**AARPATSTL**NRFSALQ	1	1
GGSGAKPSDAASE**AARPATSTL**NRFSALQQ	1	1
GSGAKPSDAASE**AARPATSTL**NRFSALQQA	1	0
SGAKPSDAASE**AARPATSTL**NRFSALQQAV	1	0
GAKPSDAASE**AARPATSTL**NRFSALQQAVP	1	1
AKPSDAASE**AARPATSTL**NRFSALQQAVPT	1	1
KPSDAASE**AARPATSTL**NRFSALQQAVPTE	1	1
PSDAASE**AARPATSTL**NRFSALQQAVPTES	1	1
SDAASE**AARPATSTL**NRFSALQQAVPTEST	1	1
DAASE**AARPATSTL**NRFSALQQAVPTESTD	1	1
AASE**AARPATSTL**NRFSALQQAVPTESTDN	1	1
ASE**AARPATSTL**NRFSALQQAVPTESTDNR	1	1
SE**AARPATSTL**NRFSALQQAVPTESTDNRR	1	1
E**AARPATSTL**NRFSALQQAVPTESTDNRRV	1	1
AARPATSTLNRFSALQQAVPTESTDNRRVV	1	0

6.5. Epitope Predictor

The resulting epitope predictor approach for a novel protein sequence involves different steps which are summarized in these 7 points:

1. Obtaining the novel protein sequence as a sequence of one-letter amino acids, which represent the primary structure.

2. Calculating the prediction of the secondary structure using ProteinUnet algorithm resulting in a sequence of the same length composed by the three possible foldings.
3. Obtaining the 30 amino acid long subsequences of the primary and secondary structure sequences passing the sliding window.
4. Extracting the features with the corresponding primary and secondary language models resulting in vectors of 768 values.
5. Making the predictions with the trained model using two vectors for each sequence corresponding to primary and secondary structure information.
6. Processing the results with the noise reduction system by removing the short consecutive positive subsequences of size lower than 5.
7. Analyzing the resulting sections that represent epitope candidates.

Improvements could be implemented in the different steps of the epitope predictor for an increased accuracy in the final results.

7. CONCLUSION AND FUTURE WORK

Data mining and machine learning techniques for extracting novel information from biological data has become a potential area of research for the healthcare industry. The purpose of this work was to apply these methods for obtaining a bioinformatic tool for the prediction of epitopes.

Secondary structure prediction models based on machine learning algorithms have shown great results. In this work, the ProteinUnet model, a method based on U-Net algorithm, was utilized, which yielded useful and fast results for obtaining this information from the protein sequences.

The application of natural language processing for understanding protein primary and secondary structure has shown promising outcomes. The architecture used in this thesis for modeling protein language was BERT, which exhibited great accuracy results for both primary and secondary structure. The feature-based approach permitted the obtention of the word-embedding information in the form of vectors. The approach followed consisted on taking the classification vector of each sequence. Future research could be done by extracting a matrix composed by all the vectors of every token in each sequence.

The use of machine learning algorithms in data mining is essential to properly analyze the large amount of data. The designed models have shown positive results regarding the classification of epitope and non-epitopes. However, refinement in the architecture could be done to increase the accuracy and lower the overfitting. In addition, further information on the functionality of epitopes could be introduced to improve the model.

The combination of efforts from NLP and machine learning algorithms demonstrate potential outcomes on developing an epitope prediction tool. This would ultimately have an impact in the development of vaccines, accelerating and improving the process and results.

8. SOCIO-ECONOMIC IMPACT

Biotechnological advancements have caused a flood of biological data, mostly molecular sequences. In addition, the healthcare industry has begun to employ big data technology, resulting in large amounts of clinical information. A point has been reached in which the ability to generate biomedical data has far outpaced the capacity to analyse it [54]. This has resulted in an increased demand for data mining techniques to extract relevant information.

In molecular biology, as well as the pharmaceutical and clinical sectors, data mining allows biologists and medical researchers to produce insightful observations and significant discoveries. Findings on genes and proteins and their structures and functions have remarkable implications on unknown disease patterns and drug and vaccine development [55].

The introduction of bioinformatics tools like epitope predictors in the vaccine development pipeline signify the achievement of significant advantages over traditional approaches. Using computational methods in the epitope-based vaccine pipeline reduces considerably the cost and time of production [6]. In addition, the emergence of epitope-based vaccines have the potential to be the next generation of cancer immunotherapy [56].

Cancer is one of the most deadly diseases for which there is no definitive therapy. In 2008, cancer lead to 7.6 million deaths, with 13.1 million expected by 2030 [57]. Among all the types of cancer treatments, immunotherapy has proven to be the most important one due to the specificity in the eradication of cancerous cells.

On the other hand, the development of a new vaccine costs between $200 and $500 million US dollars [58]. Although it is estimated that there are over 400 vaccination projects worldwide [59], the majority of these initiatives will not result in a licensed vaccine due to economic obstacles. Besides, the time required for the vaccine development is around 15 years or more [60].

It is worth mentioning the importance of the fast development of vaccines in the Covid-19 world pandemic. Several manufacturers have developed vaccines in less than a year, which is a remarkable achievement given that new vaccines generally take more than a decade or longer to be produced [61]. In fact, great investments on COVID-19 vaccines have aided in this accelerated process. Currently, there are more than a dozen vaccines that have received regulatory authorization. In addition to the approved ones, there are many more candidates in clinical trials [62].

9. REGULATORY FRAMEWORK

The methodologies followed in this thesis are not subject to any regulation or intellectual property protection, and they do not violate any legal code of ethics. However, the techniques that have been used in this work were implemented in programs that have their own regulations.

The chosen language of programming for writing the scripts for this project was Python (versions 3.6 and 3.8). All python releases are open source. The Python scripts were executed in Spyder and Google Colaboratory. Both of them are 100% free and open sources.

The ProteinUnet model is licensed by Creative Commons Attribution-NonCommercial 4.0 International Public License. This permits distribution and modification of the code, crediting the author for the original creation.

The files for implementing the BERT architecture are licensed under the Apache License 2.0. This is a permissive license which permits private use of the scripts with the possibility of making modifications and distributing it without the source code.

10. BUDGET

The budget required to develop this thesis is represented in these three tables. The two first tables include the human resources and materials needed with their corresponding costs. The final table summarizes the total costs including the human resources, materials and indirect costs, which account for 15% of the previous costs.

TABLE 10.1. HUMAN RESOURCES COSTS

Category	**Cost (€/hour)**	**Time investment (hours)**	**Total Cost (€)**
Student	20	400	7,600
Tutor 1	55	25	1,375
Tutor 2	55	25	1,375
TOTAL			**10,350**

TABLE 10.2. MATERIALS COSTS

Category	**Description**	**Cost (€)**	**Time (months)**	**Amortization (€)**
Erazer P6679 MD60474	Intel(r) Core (TM) i7-7500U CPU 2.70GHz (4 CPUs), NVIDIA GeForce GTX 950M	1.199	5	500
T-Series SP Intel Xeon	Intel(r) Xeon(r) CPU e5-2630 v4 2.20ghz - 40 cores, Nvidia GeForce GTX 1070 379x4 (x4)	4,500	5	1,850
TOTAL				**2,350**

TABLE 10.3. SUMMARY OF TOTAL COSTS

Category	**Cost (€)**
Human resources	10,350
Materials	2,350
Indirect (15% of materials and human resources)	1,905
TOTAL	**14,650**

BIBLIOGRAPHY

[1] K. Vougas *et al.*, “Machine learning and data mining frameworks for predicting drug response in cancer: An overview and a novel in silico screening process based on association rule mining,” *Pharmacology & Therapeutics*, vol. 203, p. 107 395, Nov. 2019. DOI: 10.1016/j.pharmthera.2019.107395. [Online]. Available: https://doi.org/10.1016/j.pharmthera.2019.107395.

[2] J. T. L. Wang, M. J. Zaki, H. T. T. Toivonen, and D. Shasha, “Introduction to data mining in bioinformatics,” in *Data Mining in Bioinformatics*, X. Wu, L. Jain, J. T. Wang, M. J. Zaki, H. T. Toivonen, and D. Shasha, Eds. London: Springer London, 2005, pp. 3–8. DOI: 10.1007/1-84628-059-1_1.

[3] S. E. E. Profile, “Preface to Biological Data Mining and Its Applications in Healthcare,” no. December, pp. xxii–xxiii, 2014. DOI: 10.1109/icdmw.2013.6.

[4] M. Mahmud, M. S. Kaiser, T. M. McGinnity, and A. Hussain, “Deep learning in mining biological data,” *Cognitive Computation*, vol. 13, no. 1, pp. 1–33, Jan. 2021. DOI: 10.1007/s12559-020-09773-x.

[5] R. E. Soria-Guerra, R. Nieto-Gomez, D. O. Govea-Alonso, and S. Rosales-Mendoza, “An overview of bioinformatics tools for epitope prediction: Implications on vaccine development,” *Journal of Biomedical Informatics*, vol. 53, pp. 405–414, 2015. DOI: https://doi.org/10.1016/j.jbi.2014.11.003.

[6] S. Parvizpour, M. M. Pourseif, J. Razmara, M. A. Rafi, and Y. Omidi, “Epitope-based vaccine design: A comprehensive overview of bioinformatics approaches,” *Drug Discovery Today*, vol. 25, no. 6, pp. 1034–1042, 2020. DOI: https://doi.org/10.1016/j.drudis.2020.03.006.

[7] *Molecular Biology of the Cell*, ser. 4th Edition. New York: Garland Science, 2002.

[8] *Biochemistry*, ser. 5th Edition. New York: W H Freeman, 2002.

[9] C. Branden and J. Tooze, *Introduction to Protein Structure*. CRC Press, 2012.

[10] J. Abbass, “Secondary structure-based template selection for fragment-assembly protein structure prediction,” Ph.D. dissertation, Jul. 2018.

[11] D. Whitford, *Proteins: Structure and Function*. Wiley, 2013.

[12] G. Qi, Q. Wang, J. Xu, and F. Deng, "Solid-state NMR studies of internuclear correlations for characterizing catalytic materials," *Chemical Society Reviews*, 2021. DOI: 10.1039/d0cs01130d.

[13] J. Parkin and B. Cohen, "An overview of the immune system," *The Lancet*, vol. 357, no. 9270, pp. 1777–1789, 2001.

[14] D. D. Chaplin, "1. overview of the human immune response," *Journal of Allergy and Clinical Immunology*, vol. 117, no. 2, Supplement 2, S430–S435, 2006, Mini-Primer on Allergic and Immunologic Diseases.

[15] M. Wilk-Blaszczak, *Cell Physiology*. MAVS OPEN PRESS.

[16] "7 - antigen presentation," in *Immunology Guidebook*, J. M. Cruse, R. E. Lewis, and H. Wang, Eds., San Diego: Academic Press, 2004, pp. 267–276.

[17] *Immunobiology: The Immune System in Health and Disease*, ser. 5th Edition. New York: Garland Science, 2001.

[18] S. Bonini, "Immunology IV: Clinical applications in health and disease by joseph a. bellanti," *World Allergy Organization Journal*, vol. 5, no. 8, p. 94, 2012. DOI: 10.1097/wox.0b013e3182641db0.

[19] G. Tortora and B. Derrickson, *Principles of Anatomy and Physiology*. Wiley, 2018.

[20] X. Yang and X. Yu, "An introduction to epitope prediction methods and software," *Reviews in Medical Virology*, vol. 19, no. 2, pp. 77–96, 2009. DOI: 10.1002/rmv.602.

[21] B. Peters, W. Tong, J. Sidney, A. Sette, and Z. Weng, "Examining the independent binding assumption for binding of peptide epitopes to MHC-i molecules," *Bioinformatics*, vol. 19, no. 14, pp. 1765–1772, Sep. 2003. DOI: 10.1093/bioinformatics/btg247.

[22] (). "Immunotherapy | national cancer institute," [Online]. Available: https://www.cancer.gov/publications/dictionaries/cancer-terms/def/immunotherapy.

[23] (). “"immunotherapy | memorial sloan kettering cancer center",” [Online]. Available: https://www.mskcc.org/cancer-care/diagnosis-treatment/cancer-treatments/immunotherapy. (accessed: 09.05.2021).

[24] “Epitope-based vaccines: An update on epitope identification, vaccine design and delivery,” *Current Opinion in Immunology*, vol. 15, no. 4, pp. 461–470, 2003. DOI: https://doi.org/10.1016/S0952-7915(03)00083-9.

[25] F. Afrin, H. Hemeg, and H. Ozbak, *Vaccines*. IntechOpen, 2017.

[26] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, “Deep learning applications and challenges in big data analytics,” *Journal of Big Data*, vol. 2, no. 1, Feb. 2015. DOI: 10.1186/s40537-014-0007-7.

[27] L. Deng, “A tutorial survey of architectures, algorithms, and applications for deep learning,” *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014. DOI: 10.1017/atsip.2013.9.

[28] J. Zou, Y. Han, and S.-S. So, *The Principles of Quantum Mechanics*, ser. vol 458. Humana Press, 2008.

[29] S. Sharma, S. Sharma, and A. Anidhya, “Understanding Activation Functions in Neural Networks,” *International Journal of Engineering Applied Sciences and Technology*, vol. 4, no. 12, pp. 310–316, 2017.

[30] (). “Neural network models,” [Online]. Available: https://www.datacamp.com/community/tutorials/neural-network-models-r. (accessed: 11.06.2021).

[31] D. Krishnamurthy. (). “The components of a neural network,” [Online]. Available: https://towardsdatascience.com/the-components-of-a-neural-network-af6244493b5b.

[32] F. Bre, J. M. Gimenez, and V. D. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using artificial neural networks,” *Energy and Buildings*, vol. 158, pp. 1429–1441, Jan. 2018. DOI: 10.1016/j.enbuild.2017.11.045.

[33] C. C. Aggarwal, *Neural Networks and Deep Learning*. 2018. DOI: 10.1007/978-3-319-94463-0.

[34] A. Zhang, N. Ballas, and J. Pineau, “A dissection of overfitting and generalization in continuous reinforcement learning,” *arXiv*, 2018. arXiv: 1806.07937.

[35] S. Gupta, R. Gupta, M. Ojha, and K. P. Singh, “A comparative analysis of various regularization techniques to solve overfitting problem in artificial neural network,” in *Data Science and Analytics*, Springer Singapore, 2018, pp. 363–371. DOI: 10.1007/978-981-10-8527-7_30. [Online]. Available: https://doi.org/10.1007/978-981-10-8527-7_30.

[36] B. Rost, C. Sander, and R. Schneider, “Redefining the goals of protein secondary structure prediction,” *Journal of Molecular Biology*, vol. 235, no. 1, pp. 13–26, Jan. 1994. DOI: 10.1016/s0022-2836(05)80007-5. [Online]. Available: https://doi.org/10.1016/s0022-2836(05)80007-5.

[37] Y. Yang *et al.*, “Sixty-five years of the long march in protein secondary structure prediction: The final stretch?” *Briefings in Bioinformatics*, bbw129, Dec. 2016. DOI: 10.1093/bib/bbw129. [Online]. Available: https://doi.org/10.1093/bib/bbw129.

[38] R. Heffernan, K. Paliwal, J. Lyons, J. Singh, Y. Yang, and Y. Zhou, “Single-sequence-based prediction of protein secondary structures and solvent accessibility by deep whole-sequence learning,” *Journal of Computational Chemistry*, vol. 39, no. 26, pp. 2210–2216, Oct. 2018. DOI: 10.1002/jcc.25534.

[39] (). “Multiple sequence alignment,” [Online]. Available: https://bioinf.comav.upv.es/courses/biotech3/theory/multiple.html. (accessed: 11.06.2021).

[40] U. Hansmann, “Protein folding in silico: An overview,” *Computing in Science & Engineering*, vol. 5, no. 1, pp. 64–69, Jan. 2003. DOI: 10.1109/mcise.2003.1166554.

[41] K. Kotowski, T. Smolarczyk, I. Roterman-Konieczna, and K. Stapor, “Protein-Unet—An efficient alternative to SPIDER3-single for sequence-based prediction of protein secondary structures,” *Journal of Computational Chemistry*, vol. 42, no. 1, pp. 50–59, 2021. DOI: 10.1002/jcc.26432.

[42] G. G. Chowdhury, “Natural language processing,” *Annual Review of Information Science and Technology*, vol. 37, no. 1, pp. 51–89, 2003.

[43] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association

for Computational Linguistics, Sep. 2015, pp. 632–642. DOI: 10.18653/v1/D15-1075. [Online]. Available: https://www.aclweb.org/anthology/D15-1075.

[44] J. Lee *et al.*, “BioBERT: a pre-trained biomedical language representation model for biomedical text mining,” *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Sep. 2019. DOI: 10.1093/bioinformatics/btz682.

[45] J. Vig, A. Madani, L. R. Varshney, C. Xiong, R. Socher, and N. F. Rajani, “Bertology meets biology: Interpreting attention in protein language models,” *bioRxiv*, 2020. DOI: 10.1101/2020.06.26.174417.

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].

[47] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[48] T. Wood. (). “"deepai-softmax function",” [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer.

[49] (). “What is a feed forward neural network?” [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network. (accessed: 11.06.2021).

[50] F. A. Furfari(tony), “The Transformer,” *IEEE Industry Applications Magazine*, vol. 8, no. 1, pp. 8–15, 2002. DOI: 10.1109/2943.974352.

[51] A. Rogers, O. Kovaleva, and A. Rumshisky, “A Primer in BERTology: What we know about how BERT works,” *arXiv*, 2020. DOI: 10.1162/tacl_a_00349. arXiv: 2002.12327.

[52] L. Breuza *et al.*, “The UniProtKB guide to the human proteome,” *Database*, vol. 2016, bav120, 2016. DOI: 10.1093/database/bav120. [Online]. Available: https://doi.org/10.1093/database/bav120.

[53] “UniProt: The universal protein knowledgebase in 2021,” *Nucleic Acids Research*, vol. 49, no. D1, pp. D480–D489, 2021. DOI: 10.1093/nar/gkaa1100.

[54] F. Wang, X.-L. Li, J. T. L. Wang, and S.-K. Ng, “Guest editorial: Special section on biological data mining and its applications in healthcare,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 3, pp. 501–502, 2017. DOI: 10.1109/TCBB.2016.2612558.

[55] F. Kabli, “Complex biological data mining and knowledge discovery,” in *Biotechnology*, IGI Global, 2019, pp. 305–321. DOI: 10.4018/978-1-5225-8903-7.ch011.

[56] N. Nezafat, Y. Ghasemi, G. Javadi, M. J. Khoshnoud, and E. Omidinia, “A novel multi-epitope peptide vaccine against cancer: An in silico approach,” *Journal of Theoretical Biology*, vol. 349, pp. 121–134, May 2014. DOI: 10.1016/j.jtbi.2014.01.018.

[57] W. H. Organization, *World Health Statistics 2010*, ser. Nonserial Publication Series. World Health Organization, 2010.

[58] S. Plotkin, J. M. Robinson, G. Cunningham, R. Iqbal, and S. Larsen, “The complexity and cost of vaccine manufacturing – an overview,” *Vaccine*, vol. 35, no. 33, pp. 4064–4071, Jul. 2017. DOI: 10.1016/j.vaccine.2017.06.003. [Online]. Available: https://doi.org/10.1016/j.vaccine.2017.06.003.

[59] F. André, “How the research-based industry approaches vaccine development and establishes priorities,” *Developments in biologicals*, vol. 110, pp. 25–29, 2002.

[60] S. McElligott, “Addressing supply side barriers to introduction of new vaccines to the developing world,” *American Journal of Law & Medicine*, vol. 35, no. 2-3, pp. 415–441, Jun. 2009. DOI: 10.1177/009885880903500210.

[61] O. J. Wouters *et al.*, “Challenges in ensuring global access to COVID-19 vaccines: Production, affordability, allocation, and deployment,” *The Lancet*, vol. 397, no. 10278, pp. 1023–1034, Mar. 2021. DOI: 10.1016/s0140-6736(21)00306-8.

[62] (). “"covid-19 vaccine tracker",” [Online]. Available: https://www.raps.org/news-and-articles/news-articles/2020/3/covid-19-vaccine-tracker. (accessed: 11.06.2021).