A Deep Few-Shot Network for Protein Family Classification

Saeedeh Jamali

A Thesis
In the Department of
Mathematics and Statistics

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Science (Mathematics)

at Concordia University
Montreal, Quebec, Canada

March 2024

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared
By: Saeedeh Jamali

Entitled: A Deep Few-Shot Network for Protein Family Classification
and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Mathematics)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____ Chair
Dr. Arusharka Sen
_____ Examiner
Dr. Arusharka Sen
_____ Thesis Supervisor
Dr. Yogendra P. Chaubey
_____ Thesis Supervisor
Dr. Ashkan Ebadi

Approved by _____
Dr. Lea Popovic, Graduate Program Director

_____
Dr. Pascal Sicotte Dean, Faculty of Art, and Science

_____
Date: 2024-03-25

Abstract

A Deep Few-Shot Network for Protein Family Classification

Saeedeh Jamali

Protein sequence analysis is arguably a challenging modern bioinformatics problem covering various applications such as disease research, precision medicine, and therapeutics [1]. Given the emergence of sequencing technologies and the resulting large-scale databases, protein family classification is an open problem in bioinformatics [2]. Recent advances in computer science have opened new gates to researchers in various scientific domains [3]. Bioinformatics, as an intermediary research field, takes advantage of these advancements from conventional machine learning methods to large language models, and biostatistics [4]. Utilized machine learning techniques for protein family classification, are dependent on domain experts to generate features which could be time-consuming and challenging [5] [6]. Deep learning algorithms have shown promising results in proteomics; however, their application is limited to the availability of massive data sets for training. Since the required data comes from experiments, it can be highly complex or incomplete. As an alternative, few-shot models can learn and generalize from a few observations. To address the mentioned limitations, in this research, we designed and implemented a deep few-shot network for protein family classification1 [7] and our result showed outperformance to state-of-the-art baseline models. To the best of our knowledge, this is the first deep network tailored for primary sequence family classification that can highly perform with a very limited number of observations.

---

[1] The short paper of this research work was accepted and presented at the 36th Canadian Conference 2023 on Artificial Intelligence (CanAI - 2023)

Acknowledgments:

I would like to express my profound gratitude to my supervisors, Professor Yogen Chaubey and Professor Ashkan Ebadi, for their exceptional guidance and mentorship throughout my journey in this program. Not only are they outstanding researchers, but they have also shown unwavering support. I am truly grateful and honoured for the remarkable opportunity to have been mentored by them and for the enriching experience of learning from them during this research work and this program.

Additionally, I extend my sincere thanks to Professor Arush Sen and all my professors at Concordia University for their invaluable contributions through insightful discussions, comments, and courses.

Thanks to the Mathematics and Statistics department for providing me with this great opportunity to experience working as a Concordia University employee and having believed in how strong an international student can be to work and study in a new country.

To my first teachers, my parents, Maman, Baba, and older sister, Nastaran, who taught me that "Word," "Education," and "Science" are lights that overcome fear, darkness, and lies.

To my dearest Narges, Reza, Armand, Marmar, and all my friends who supported me to dare to start and finish this challenging and wonderful journey.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

Proteins are represented through four distinct structural forms (Fig 1). The protein primary sequence is the most cost-effective and accessible form of protein. This primary structure is obtained by direct sequencing of the protein, resulting in a sequence of amino acids. Each amino acid in this sequence is represented by a specific character, making it a straightforward and commonly available format for protein analysis in a protein sequence. There are predominantly twenty standard amino acids that frequently occur. Occasionally, this sequence also includes a few rare amino acids. Each of these amino acids, whether common or obscure, is represented by a unique character notation. Advanced high-throughput technologies like next-generation sequencing [4] [8] have significantly accelerated the pace of omics-based research, which includes the study of genes, proteins, and other biomolecules. This acceleration has deepened our understanding of the biological mechanisms associated with specific diseases and has enhanced the speed and efficacy of patient-specific responses to these conditions.

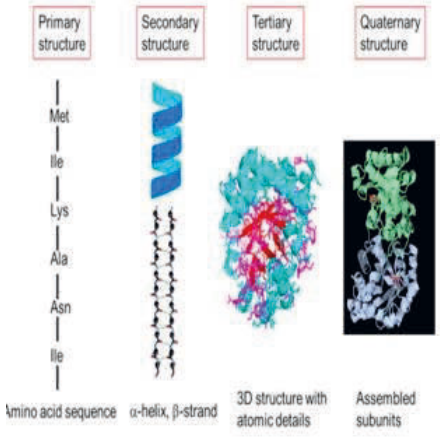

Fig 1. Four protein structures [9]

The novel approach to biomarker research, biological indicators that can predict disease risk, diagnose conditions, or forecast treatment outcomes with high precision, grounded in data-driven methods, results in methods of personalized medicine [10] [11] [4]. Improving the techniques for processing and analyzing underlying data remains a key motivation for bioinformaticians and machine/deep learning specialists. Investigating and exploring *omic* data, without human intervention, is an ongoing challenge. In standard machine learning applications, feature engineering is often employed within the data analytic pipeline to address this issue. Recently, experts in machine learning have started utilizing advanced algorithms from the subfield of deep learning (DL) that do not necessitate the extensive feature engineering typically required in traditional machine learning approaches [4]. This is illustrated in Fig 2, showcasing the growing recent interest in utilizing DL models in *omics-based* research.

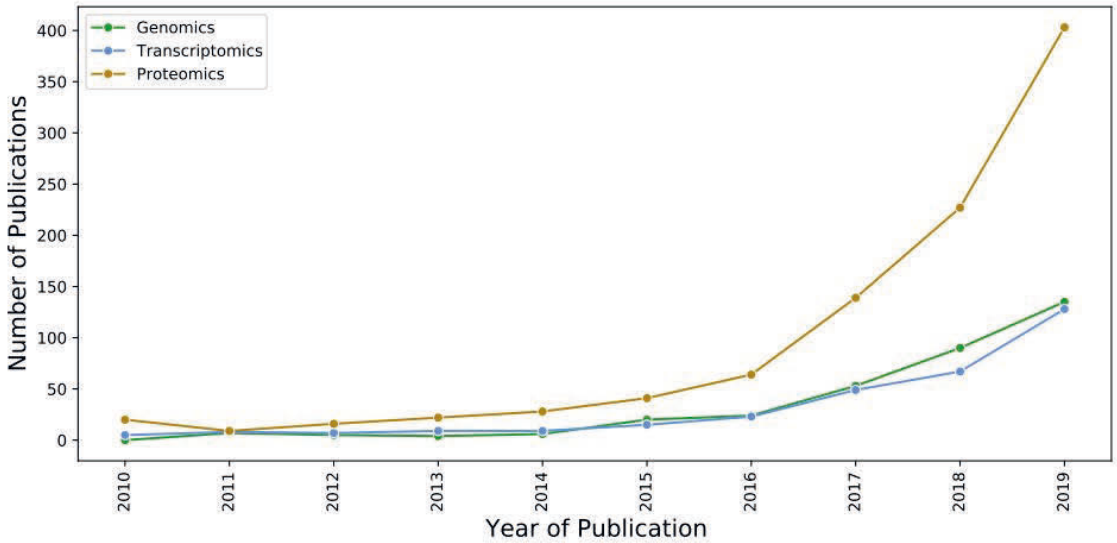

Fig 2 Approximation of the number of published articles based on the search for deep learning models within the *omics* research field [4].

This transition primarily stems from the superior capability of DL in developing predictive models and identifying complex patterns within extensive datasets [12] which requires minimal feature

engineering, offering a unique advantage to analyze omics data at its most fundamental level, bypassing the need for handcrafted features.

A major issue with employing advanced machine learning techniques, like deep learning models, is their dependence on extensive collections of labelled data for effective generalization. Acquiring such large datasets remains both costly and time-consuming. Moreover, the lack of labelled data can result in poor performance of machine learning algorithms. Similar challenges also exist in other domains, including computer vision (CV) and natural language processing (NLP) [4]. Thus, a motivation behind this research is the adaptation of deep learning methods for the analysis of proteomics data, aiming to be effective irrespective of the quantity of available labelled data. Modern deep learning approaches need to be adept at handling both extensive and limited datasets, as they play a crucial role in uncovering new insights and broadening our understanding of diseases. These techniques are instrumental in enhancing diagnostics and formulating customized treatment plans [4].

Before the advent of machine learning, earlier methods focused on programming computers with a specific set of rules for each modelling task. Dealing with the intricate nature of omic data made these early methods both time-consuming and laborious. As the availability of data increased, these traditional approaches soon became impractical for researchers. The need for continual adjustments and modifications to the programs, in response to the escalating complexity of the data, rendered them inefficient. Ultimately, machine learning emerged as a more effective solution to these challenges, surpassing the limitations of rule-based systems. With their capacity to learn from experience, machine learning algorithms could use all available data, offering a more dynamic and adaptable approach to data analysis [4] [11]. Machine learning algorithms have the capability to process vast amounts of data and identify patterns that might be overlooked by

manually crafted rules. Now, machine learning is pivotal in uncovering new patterns and addressing emerging hypotheses in the context of these intricate biological systems. Contemporary machine learning methods, particularly deep learning, have revolutionized the field of computational biology. The adoption of deep learning algorithms has expedited research by enabling the automatic detection and analysis of patterns in large datasets, eliminating the need for manual intervention. Similar to other information processing areas, deep learning is becoming increasingly prominent and effective in the field of bioinformatics. This includes their use in predicting important protein properties directly from protein sequence data. The effectiveness of deep learning solutions lies in their capability to extract intricate, task-specific features from basic input data. Most of the deep learning methods used for protein analysis have been adapted from the field of natural language processing (NLP). The analysis of a protein's primary structure can be viewed as similar to the tasks currently being performed in the field of NLP, which focuses on learning the linguistic structure of sentences [4].

Recent advancements in NLP have highlighted the effectiveness of pre-training. In this method, a model is first trained on a large body of unlabeled text, and then fine-tuned with labelled data for a specific task. Pre-training enables the model to learn the statistical patterns of language, such as the meanings of terms and possible grammatical relationships. Fine-tuning, on the other hand, optimizes the network for a particular function, like discerning the emotional sentiment of a sentence. Training language models on large-scale unlabeled datasets generally demand significant computational resources. Yet, the resulting encodings from such training are versatile enough to be applicable across a broad spectrum of subsequent tasks [13] [14] [15].

Pre-training has now become a standard in NLP, leading to the emergence of networks such as

ELMo [16], BERT [17] [4]. These models have achieved state-of-the-art results in language modelling. Typically, they start by using a sub-word algorithm to re-encode the original text. This process enables the system to deconstruct infrequently used words in the vocabulary into more common sub-words. For instance, the word 'cars' could be divided into the word piece tokens 'car' and 's'. By fragmenting the rare words in a corpus, the modelling process is simplified. The network can then utilize these sub word representations to depict words, rather than relying on the original character sequences of the words.

In recent years, the application of pre-trained techniques from the field of NLP has significantly influenced protein bioinformatics. These techniques are trained on extensive databases of protein sequences to extract meaningful characteristics [13] [18] [19].

A notable example built upon the principles of the BERT model is the work by Elnaggar *et al.* [20], who utilized over 2.1 billion sequences of proteins to train, a set of transformer models originally developed for NLP. These models enable the conversion of protein sequences into vector formats, this transformer can then be effectively utilized for a range of applications, such as classifying protein families [21] [19]. The use of these pre-trained transformer models offers numerous benefits. They eliminate the need for creating manual, error-prone features to represent protein sequences, leading to a more streamlined and efficient approach to protein sequence analysis and related developments [19].

Given the limitations in the field of omics research regarding the application of deep learning models, a highly accurate protein family classification task is an open problem. Understanding the unknown properties and functions of proteins, based on measurable features, is crucial for advancing disease research, precision medicine, and therapeutics. For instance, in the task of

protein sequence classification, we often encounter families with insufficient examples. As previously discussed, most deep/machine learning models, such as common convolutional neural networks (CNN), are applicable for classification tasks only when a large-scale dataset is available [4] [12]. Metric-based meta-learning models, such as Siamese networks, offer an innovative approach to generalizing from limited examples. These models use a distinctive structure to assess the similarity between inputs [22], minimizing the need for extensive retraining. Siamese networks are noted for their scalability, allowing for the inclusion of additional categories [22] [23].

1.1.1 Contribution

To address the mentioned limitations, in this research, we designed and implemented a novel deep few-shot learning network for protein family classification. While Siamese networks have been explored in various contexts, including object tracking [24] and COVID-19 detection from X-ray images [22], Protein-Protein Interaction [19], to the best of our knowledge, this is the first work that employs transfer learning and presents a few-shot deep Siamese network model for protein family classification task. The goal of our deep few-shot learning architecture is to classify unseen primary protein sequences that can be highly performed with a very limited number of observations.

The effectiveness of our architecture will be tested with three different baseline architectures including Support Vector Machines, Convolutional Neural Networks, Gated Recurrent Unit.

1.2 Literature Review and the Outline of Thesis

In this chapter, we review the relevant literature, organized into three main sections. First, we examine the conventional models used for protein family classification. This is followed by a

discussion on the advantages and limitations of machine learning methods, particularly in the context of protein family classification. In the subsequent section, we review what deep learning methods offer as alternatives to conventional machine learning approaches for this field of science.

We conclude the chapter with a literature review of works and techniques, such as the few-shot learning strategies that provide the necessary background for our novel architecture proposal.

Protein plays crucial roles in various functions within an organism, ranging from growth to cellular maintenance. Differentiating between known and unknown proteins and assigning them to their appropriate protein families can yield deeper insights into their specific functionalities and behaviours. Protein family classification models consider the inherent nature of proteins, which are one-dimensional sequences composed of 20 unique amino acids, referred to as the primary protein sequence. When a new protein is discovered, researchers strive to categorize its primary sequence into a particular family. This classification assesses the probability that the novel protein shares similarities, in terms of properties, functions, and general behaviour, with previously identified protein families [2]. Such understanding is vital for medical and biology research areas like disease identification, comparative genomics, and the development of medications and drug designs [4] [25] [26]. Traditionally, methods like X-Ray crystallography and nuclear magnetic resonance have been employed by experts in laboratories to discern protein structures and functions. However, with the advent of large-scale genome projects and technological advancements, there has been an exponential increase in the number of novel protein sequences. As a result, conducting biological experiments to characterize these protein sequences has become costly, time-intensive, and laborious [27] [23]. Thanks to advancements in computer science and digital technologies, both traditional and modern machine learning techniques have significantly transformed computational

biology and data-driven methods in bioinformatics [4]. In this section, we will discuss this progress and its implications.

The remainder of this thesis is organized as follows: Chapter 1 continues with a review of relevant research work. Chapter 2 discusses data methodologies and the components of our proposed architecture. In Chapter 3, we discuss the results and conclusions, along with directions for future research.

1.3 Conventional Models for Protein family classification

1.3.1 K-mer

The K-mer approach offers a technique for embedding bioinformatics sequences, but it's shallow to a neural network that can't be trained beyond a single convolutional layer. This approach was established by Karlin and Burge in 1995 [28]. K-mer methods work based on the frequency of words and are one alignment-free method applied as the most used model in sequence comparison and many other bioinformatics problems [5]. The method initiates by generating a comprehensive dictionary of all potential sub-sequences that are k units long (known as a K-mer), and then each k-length sub-sequence is moved across against the given sequence. Concurrently, a quantitative vector is constructed wherein each element precisely represents the occurrences of its corresponding K-mer within the sequence.

Although it forms the foundation for numerous bioinformatics methods, such as ESPRIT by Sun *et al.* (2009) and SLAD by Zheng *et al.* (2018) for sequence binning, as well as the RDP classifier by Wang *et al.* (2007) and Kraken by Wood and Salzberg (2014) for sequence annotation generating features from words of varying lengths typically demands more complex data frameworks, making it substantially more computationally intensive [18]. Moreover, a crucial

aspect to understand about the K-mer approach is that its filters, which correspond to the adjustable weights within the network, are pre-defined and manual rather than learned which can be problematic [5].

1.4 Machine Learning

The advent of Big Data has been instrumental in the evolution of Machine Learning (ML), a subfield of Artificial Intelligence (AI). Machine learning focuses on creating and refining mathematical algorithms that are capable of enhancing their performance autonomously through experience and task modelling. This subfield has significantly transformed the methodologies employed by biologists in modelling and analysis.

Machine learning is a subfield of computer science that employs a series of statistical and mathematical rules and assumptions to enable machines (such as computers), to learn from data and make decisions or predictions based on that data [4]. A machine learning model can essentially be viewed as a function that approximates mathematical relationships. It is designed to learn the connection between input data (x) and output data (y), especially in cases where the precise relationship is not initially clear. For instance, in the field of bioinformatics, a machine-learning model might analyze numerous instances of peptide sequences (inputs) along with their corresponding retention times (outputs). By doing so, the model is trained to predict the retention times for other peptides that have not yet been measured. The key advantage of machine learning lies in its ability to leverage existing data to learn a relationship between various features of the data. Once a model is developed, it can predict future outputs based solely on input data, thereby reducing the need and expense of additional measurements [29]. The choice of machine learning algorithm largely depends on the nature of the input data (i.e., features) and the type of output (i.e.,

discrete, or continuous labels). Transforming the input data through data processing feature engineering, which involves applying a series of mathematical functions, is a crucial step in this process. The machine learning model efficiency is related to the quality of its input data, as this data is essential for optimizing the model's parameters. Additionally, the method by which the input data is processed and transformed via feature engineering can significantly influence the overall performance of the model [4].

1.4.1 General Workflow of a Machine Learning Model

Now we discuss the general workflow when developing a machine learning model. After a machine learning model is developed, it enters a process called training. This involves using a set of mathematical functions to optimize the model's parameters. Throughout this iterative process, the parameters are repeatedly estimated and evaluated on how accurately the model's predictions align with the expected output. Then a predefined loss function (Equation 1) measures the errors of these predictions, and adjustments are made to the parameters to minimize this loss function.

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, f_\theta(x_i)) \quad (1)$$

In Equation 1, $\mathcal{L}(\theta)$ represents the loss function dependent on the parameters θ, N is the number of examples, ℓ is a loss measure (e.g., squared error for regression or cross-entropy for classification), y_i are the true values, and $f_\theta(x_i)$ are the model's predictions.

This process continues until no further reduction in loss is possible, signifying that the model's performance has peaked. The model's predictions are then validated using a separate holdout dataset, confirming the model's effective training. This typically involves training the model on a

designated dataset (the training dataset) and then assessing its performance on entirely separate datasets (known as test and validation datasets).

When a machine learning model shows high accuracy on the training dataset but underperforms on the validation or testing sets, this is indicative of overfitting the training data [4]. This situation may occur when a model's parameters are overly trained, losing their generalizability for the overall task, or when the model is excessively complex (with too many trainable parameters) [4].

1.4.2 Support Vector Machines (SVM)

Support Vector Machines (SVM) is a common machine learning model that can be utilized for analyzing bioinformatics data. SVM utilizes one or more hyperplanes in a multi-dimensional space for predictions. The design of each hyperplane aims to maximize its distance from any individual data point during the training process. SVM relies on a kernel function that elevates the original data to a higher-dimensional space. Then, the model learns the hyperplane's shape by adjusting weights. The resulting hyperplane, when translated back to the original data space, provides a nonlinear decision boundary [4].

Lee *et al.* [13] used SVM for the protein family classification task, resulting in an 87.9% F1 performance metric evaluation. In another work by Yuvaraj *et al.* [30], the SVM model was utilized for tumour classification using gene expression data. Additionally, SVMs have been prominently utilized in various domains of omic research, particularly in identifying biomarkers. Moreover, these models have played a significant role in predicting protein thermostability and localization [4]. Shen *et al.* [6] developed a notable SVM-based method for protein sequence analysis. In their study, they categorized the 20 amino acids into seven groups based on the dipoles and volumes of their side chains. Shen *et al.*'s approach, aimed at predicting human protein-protein interactions

(PPIs) achieved a high prediction accuracy of 83.9 percent. Building on this, Guo *et al.* [31] introduced a technique that combines to extract interaction data from discontinuous amino acid segments in the sequences. Their method further enhanced the prediction accuracy, reaching 86.55 percent, when predicting PPIs in the Saccharomyces cerevisiae species.

With SVMs, users need to select a particular optimization or regularization parameter, which influences the geometry of the hyperplane. Besides, discovering the most optimal kernel function can be a time-consuming process [4] [32].

1.5 Deep Learning Models for Protein Family Classification

Deep learning techniques are now available for virtually every stage of proteomics research, which enhances capabilities in feature selection, identifying peptides, and protein structure inference [29]. In the past few years, there has been a rapid increase in research focused on applying deep learning to the field of omics. Despite deep learning offering innovative ways to analyze omics data, the methods employed in computational biology generally lag behind the more advanced techniques found in fields like computer vision and natural language processing [4].

1.5.1 General Workflow of a Neural Network Model

Essentially, a basic artificial neural network is composed of three layers: an input layer, one hidden layer, and a final output layer (Fig 2). Expanding upon this structure, deep neural networks (DNNs) feature multiple hidden layers positioned between the input and output layers. Equation (2) is a vector of input variables for a neural network (NN), its included p elements and the network will build a nonlinear function $f(X)$ (Equation 3), that is prediction of the response Y for this vector. In neural network terminology, the features X_1 to X_p represent the units in the input layer. Fig 3 shows a basic feed-forward or forward-pass neural network designed for modelling a quantitative

response. The arrows signify that every input from the input layer feeds into each of the units in the hidden layer.

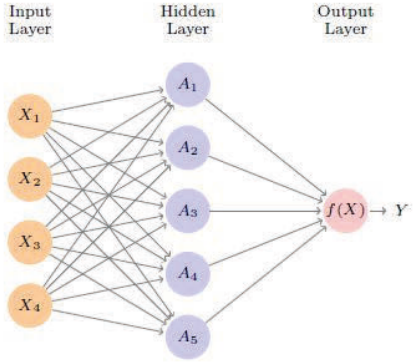

Fig 3: A neural network designed with one hidden layer functions by calculating activations $A_k = h_k(X)$. These activations are nonlinear transformations of the inputs $X_1, X_2, \ldots, X_p$ here $p = 4$, created through linear combinations. Notably, these A_k activations are internal to the network and not directly observable. These $h_k(.)$ functions to define these transformations that are not predetermined but rather learned during the network's training process. The final output of the network is generated by a linear model in the output layer, which takes these activations A_k as its inputs to form the overall output function $f(X)$ [33].

$$X = (X_1, X_2, \ldots, X_p) \tag{2}$$

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k \tag{3}$$

$$A_k = h_k(X) = g(z) = g\left(w_{k0} + \sum_{j=1}^{p} \beta_k w_{kj} X_j\right) \tag{4}$$

The function $f(X)$ is constructed through a two-step process. Initially, the K activations A_k , Equation (3), where k ranges from 1 to K, are determined in the hidden layer. These activations are the results of applying functions to the input features $X_1, X_2, \ldots, X_p$ utilizing a predetermined

nonlinear activation function $g(z)$. Each A_k (Equation 4) can be viewed as a unique transformation $h_k(X)$ of the inputs. To complete this process, all the parameters β_0, ..., β_K and w_{k0}, ..., w_{Kp} must be estimated from the data. This estimation is typically done in the early stages of training of the neural network.

In earlier neural network models, the sigmoid activation function was commonly used (Equation 5). This is the same as that used in logistic regression, transforming a linear function into probabilities ranging between zero and one.

$$g(z) = \frac{1}{1+e^{-z}} \qquad (5)$$

However, in modern neural networks, the ReLU (Rectified Linear Unit) is also used as an activation function (Equation 6).

$$g(z) = \max(0, z) \qquad (6)$$

Similar to their more basic counterparts, DNNs are trained through a process of continuous adjustment of the model's internal parameters to minimize the overall error in predictions (Equation 7) [33].

$$R(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - f_\theta(x_i)\right)^2 \qquad (7)$$

The principal technique used for this purpose is back-propagation, initially put forward by Werbos *et al.* [34] The gradient of the function $R(\theta)$, when assessed at a current value $\theta = \theta_m$, is represented by the vector composed of its partial derivatives at that specific point (Equation 8).

$$\nabla R(\theta_m) = \frac{\partial R(\theta)}{\partial \theta}\Big|_{\theta=\theta_m} \qquad (8)$$

Since the function, $R(\theta)$ is defined as a sum, (Equation 9), where it is the sum of $R_i(\theta)$ over n observations, its gradient is also the sum over these n observations. Therefore, we will focus on examining one of these terms (Equation 10).

$$R(\theta) = \sum_{i=1}^{n} R_i(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - f_\theta(x_i)\right)^2 \quad (9)$$

$$R_i(\theta) = \frac{1}{2}\left(y_i - \beta_0 - \sum_{k=1}^{K}\beta_k g\left(w_{k0} + \sum_{j=1}^{P} w_{kj}\, x_{ij}\right)\right)^2 \quad (10)$$

$$z_{ik} = w_{k0} + \sum_{j=1}^{P} w_{kj}\, x_{ij} \quad (11)$$

$$\frac{\partial R_i(\theta)}{\partial \beta_k} = \frac{\partial R_i(\theta)}{\partial f(x_i)} \cdot \frac{\partial f(x_i)}{\partial \beta_k} = -\left(y_i - f(x_i)\right) \cdot g(z_{ik}) \quad (12)$$

$$\frac{\partial R_i(\theta)}{\partial u_{kj}} = \frac{\partial R_i(\theta)}{\partial f(x_i)} \cdot \frac{\partial f(x_i)}{\partial g(z_{ik})} \cdot \frac{\partial g(z_{ik})}{\partial z_{ik}} \cdot \frac{\partial z_{ik}}{\partial u_{kj}} = -\left(y_i - f(x_i)\right) \cdot \beta_k \cdot g'(z_{ik}) \cdot x_{ij} \quad (13)$$

We use Equation 11 to simplify the expressions. First, we take the derivative with respect to (Equation 12) and with respect to (Equation 13). Both formulas of Equation 12 and Equation 13 incorporate the residual term $y_i - f_\theta(x_i)$. Equation 12 demonstrates how a portion of this residual is allocated among the hidden units, which is proportional to the function $g'(\boldsymbol{z_k})$. Following this, Equation 13 illustrates a comparable allocation process where the residual's portion is distributed to the input j mediated by hidden unit k. This differentiation process effectively apportions parts of the residual to each parameter, a mechanism facilitated by the chain rule. This method is commonly referred to as backpropagation, a fundamental concept in the domain of neural networks. This technique was pivotal in the development of neural networks containing several layers.

It operates by relaying an error signal backward through the network's layers. During this backpropagation, the model's parameters are systematically adjusted to minimize the overall error throughout each layer of the network [31]. During the forward pass, each layer's output is calculated, and the activation signals are carried onward across the network with every training iteration. To measure the error between what the network predicts and what is expected (the labels), a loss function is employed. During training, the error signals are backpropagated through the network via the chain rule, which computes the gradients to all the weights in each layer [32]. This process is completed several times during training until the error between the network's prediction and the expected output reaches an acceptable minimum level. An optimization algorithm based on a form of stochastic gradient descent (SGD) [33] is typically used to update the weight parameters in the network. Often, such adjustments involve a type of mini-batch gradient descent where an optimizer based on SGD gradually fine-tunes the model's parameters. This fine-tuning is accomplished by stochastic approximation. In recent times, a range of sophisticated learning optimization algorithms has emerged, advancing the training efficacy of neural networks, notable examples include Adagrad [34] and Adam [35].

When creating a deep learning model, it's crucial to employ regularization techniques to avoid overfitting the training data. One common regularization strategy is to apply weight decay during the training process [36]. This method imposes a penalty on the loss function if the weights in the network are too large. Additionally, dropout [37] is a widely used regularization technique, which involves randomly omitting a number of hidden units in each applicable layer throughout the training, to enhance the model's ability to generalize to new data (Fig 4).

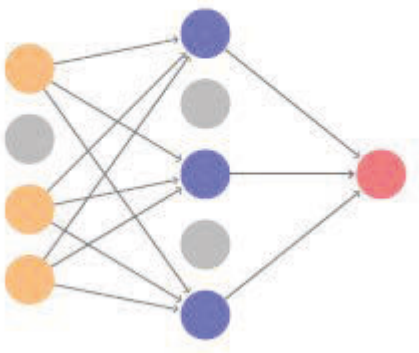

Fig 4: A network utilizing dropout technique, both the input and hidden layers have certain nodes marked in grey. These grey nodes are randomly chosen and are excluded during a specific training iteration [33].

The idea is to randomly remove a percentage (φ) of the neurons in a layer during the training phase. The remaining active neurons compensate for those that are inactive by adjusting their weights, which are scaled up by a factor of $1/(1-\varphi)$. This process ensures that neurons do not become overly dependent on specific patterns, promoting a form of regularization. Practically, dropout is executed by setting the activations of the deactivated neurons to zero, while the overall structure of the network remains unchanged. This technique helps in preventing overfitting and enhances the generalization capability of the model [33].

Batch normalization [38] stands out as another significant regularization technique. It standardizes the input features for each activation in a mini-batch by adjusting and scaling based on the batch's mean and variance.

The upcoming sections will delve into how deep learning is applied to the fields of proteomics, expanding an in-depth exploration of the most common models for protein family classification.

1.5.2 Convolutional Neural Network (CNN)

Deep learning models have been applied for complex problems such as Natural Language Processing (NLP) with state-of-the-art performance [35] [36]. This progress in solving NLP problems makes it possible to apply neural network architecture to bioinformatics problems [6] [37] [36] [38] [39]. Among all deep learning models, one-dimensional Convolutional Neural Network, (1D)-CNN is one of the most common models, which reached remarkable results for sequence classification problems [6]. CNN, which was first introduced by LeCun and team [40], stands as one of the most common deep learning architectures. This versatile framework has been effectively employed in various research fields, notably achieving impressive results in computer vision and language-related tasks. A standard CNN consists of successive layers of convolution and pooling that work to refine the input data. Each layer is crafted to improve the input by extracting and transforming essential features from a prespecified segment (called window) of the data. Each convolutional layer analyzes the complete input, whether it is in the form of one-hot or embedding encodings, by focusing on smaller segments (such as windows) and applying specific filters, including kernel weight and bias [4]. The outcome from this layer is a weighted sum of these filtered features, yielding a condensed representation of each window. Often, a convolutional layer is paired with a subsequent pooling layer that further summarizes the input. The purpose of the pooling layer is to preserve the most important features identified by the convolutional layer while discarding any extraneous ones, thereby helping to avoid overfitting. The two prevalent types of pooling include max-pooling, which picks out the feature with maximum values within a defined window of the input, and mean-pooling, which computes the mean value of the features within the same section [4]. Using a fully connected CNN model in which every input is connected to every

output by a learnable weight incorporating max pooling. Lee *et al.* achieved an F1 score of 89.7% in protein family classification with the UniProt dataset [13].

A CNN-based deep learning framework is versatile enough to address diverse applications. These models excel in detecting patterns that are invariant to spatial transformations directly from raw data, e.g., images or text, thus reducing the need for pre-processing and feature engineering. However, their effectiveness is limited when it comes to capturing long-range dependencies within sequences. Success with CNNs often hinges on access to extensive datasets that enable the model to train effectively and generalize findings. The following section will delve into two other prevalent deep learning approaches used for protein family classification [4].

1.5.3 Bidirectional Long Short-Term Memory

Within the field of deep learning, Recurrent Neural Networks (RNNs) and their improved versions, Long Short-Term Memory (LSTM) Hochreiter and Schmid Huber, 1997 [41] and Bidirectional LSTM networks, are models that are well-suited to sequence data [4] [29] [42]. In this section, we will discuss these models and review the results of related work in protein family classification. RNNs have gained prominence for their ability to process sequential data. RNNs, originally developed by Williams and others [43], have been utilized across multiple domains like natural language processing and computer vision. Unlike CNNs, RNNs consist of a memory component in their structure, enabling the past pattern in the sequence. Like other Deep Neural Networks (DNNs), an RNN includes multiple hidden layers. These layers serve as the network's memory, holding onto data about past sequences, which is continually updated and applied with each new step in the sequence. Each layer's function is to process and modify the incoming data concerning the preceding sequence element. Throughout its training phase, the output at each layer is

influenced by the immediate input and previous state on the network [4]. RNN[s] have garnered impressive results in tasks like language modelling, machine translation, and speech recognition due to their ability to precisely assess sequence log-likelihoods and their differentiability concerning these log-likelihoods, but initially, training RNNs posed challenges because of the vanishing gradient issue [44]. However, the development of LSTM networks and gated recurrent unit (GRU) architectures has overcome these obstacles, paving the way for the success of RNNs in various applications. Nevertheless, advances like long short-term memory, LSTM, and gated recurrent unit, GRU [4] designs have resolved these issues [4] [45]. These adaptations of the RNN architecture have made them more resistant to issues like the exploding gradient when training. In an LSTM network, the hidden layers consist of memory blocks that house one or more LSTM units [46]. The use of a bidirectional structure as an advance of LSTM enables the network to process and capture information in both forward and reverse directions within protein sequences. Hanson et al applied a bidirectional LSTM recurrent neural network to solve the problem of protein disorder prediction [46]. In another work, Zolg et al [47]. made significant progress in predicting the intensity of peptide fragments by training a bidirectional LSTM on fragmentation spectra from proteome Tools. However, this approach was confined to peptides that were no longer than 20 amino acids [47].

1.5.4 Gated Recurrent Unit Network

Gated Recurrent Unit (GRU) networks enhance the traditional recurrent neural network, RNN [48] architecture by addressing their shortcomings in maintaining information across varied time spans and mitigating the vanishing gradient problem that arises with long sequences. GRUs incorporate two specific mechanisms known as the update gate and the reset gate. With each new input, the

model calculates fresh memory content. The reset gate's role in this process is to regulate the extent to which the previous hidden state contributes to this calculation before applying the nonlinear activation function. Subsequently, the update gate determines the degree to which the new hidden state should combine the new memory content with the previous hidden state. This dual-gating system enables the GRU to effectively manage both short-term and long-term dependencies within the data [13]. In a study by Guiyang *et al.*, a BiGRU model was implemented to predict phosphorylation sites during SARS-CoV-2 infection. The model regarded amino acids in protein sequences as analogous to words in natural language, aiming to extract the characteristics present within the sequences of proteins. The BiGRU-based model achieved accuracies of 83.9% and 83.37% for identifying phosphorylated S/T and Y sites, respectively. Lee *et al.* [13] applied a GRU model for a protein family classification problem with F1 as a metric performance equal to 94.8%.

1.6 Few Shot Learning strategies

One common issue in medical research is the imbalance of data and the overall insufficiency of it. This becomes particularly problematic in classification problems where, although there may be ample examples for one class, there is a shortage of appropriately labelled examples for other classes. In deep learning, access to a vast amount of data is essential for enhancing model performance. However, acquiring such extensive datasets is often not feasible due to practical limitations. For instance, the process of labelling data can be both costly and time-consuming [22] [49]. Recently, Few Shot Learning (FSL), a type of meta-learning, has been introduced [50]. This approach seeks to learn from just a few examples. Few-shot learning strategies are predominantly utilized in the field of computer vision. The model that is used with a few-shot learning strategy can discern patterns in the available data. As implied by its name, few-shot learning involves using

a small set of observations for each class, contrary to the standard practice of deep learning datasets that is accurate based on inputting large datasets into a learning model [13], [32]. A prime example of FSL can be seen in character generation tasks [51], where computer programs are tasked with parsing and generating new handwritten characters from a limited set of examples. To accomplish this, the method involves decomposing the characters into smaller, transferable elements, which the second step is aggregating to form new characters [51]. Another classic FSL application arises in situations where obtaining examples with supervised information is challenging or impossible due to privacy, safety, or ethical concerns. Drug discovery is an illustrative case: it aims to explore the properties of new molecules for potential drug identification. However, due to issues like potential toxicity, low efficacy, or poor solubility, these new molecules often lack extensive real-world biological data on clinical candidates. Therefore, learning efficiently from small examples becomes crucial in such research areas [52].

Current FSL challenges are predominantly focused on supervised learning. Specifically, few-shot classification involves training classifiers with only a limited number of labelled examples per class. This approach has applications in various areas, such as image classification [53], sentiment classification for short text [53], and object recognition [24]. Typically, the method employs the N-way-K-shot classification strategy [54] where the training dataset comprises KN examples, distributed across N classes with K examples each. Transfer learning approaches are frequently utilized in FSL [55] [56], allowing the transfer of previously acquired knowledge from a source task to a few-shot task. In this process, transfer learning [57] insights from a source domain or task, for which training data is large and rich, to a target domain or task that faces a shortage of training data [57]. In FSL, when each class is represented by only one example with supervised information, FSL is termed one-shot learning. Conversely, if the classes lack any examples with supervised

information for the target, FSL is called zero-shot learning (ZSL). Since ZSL's target class doesn't have examples with supervised information, it relies on examples containing data from different modalities, such as attributes to facilitate the transfer of some level of supervised knowledge, enabling the learning process [57].

Few-shot learning represents a relatively new area of study within NLP, which has seen a growing interest recently. Unlike the field of computer vision, where standardized benchmarks for few-shot learning are well-established, the NLP domain still lacks such universally accepted benchmarks and various studies in this area often showcase their performance results on large-scale datasets [50].

1.7 Transfer learning

One common application of transfer learning [50] involves initially training a model on a specific task within one domain, and then adapting it to another task or domain. This approach leverages the extensive data available from the initial task or domain to create models capable of generating well in a target task or domain, which might have limited data availability. For example, a classifier might be developed from a model that has already learned to extract data representation such as features or from a model trained in a different but related domain [50]. The effectiveness of pre-training (transfer learning) is evident in computer vision [4], where fine-tuning, pre-trained deep neural networks for specific purposes has become standard practice within this field [4]. One example is once, the network is trained to map images into a feature space and then compares or matches these images using metrics like Euclidean or cosine distance. Siamese networks are a prime example of this approach. The initial application of deep metric learning in computer vision involved a Siamese network [58], tailored for signature verification. This network was designed to

learn a feature set that assessed the similarity between two signature inputs. Due to the significant similarities between techniques in NLP and computational biology, methods from one field are often applicable to the other [4]. BERT, Bidirectional Encoder Representations from Transformers [17], is known as a standard transformer developed for natural language processing. The architecture of BERT centers around a multi-layered bidirectional Transformer encoder. Its novel approach involves masked language modelling over its pre-training, where certain words or tokens in the input are randomly obscured. The model then learns to predict these hidden tokens, a process fundamental to its training. The entirety of BERT's structure depends on a self-attention mechanism, also known as intra-attention, Fig 5 [59]. This mechanism (Equation 15) connects different parts of a single sequence to form a comprehensive representation [17].

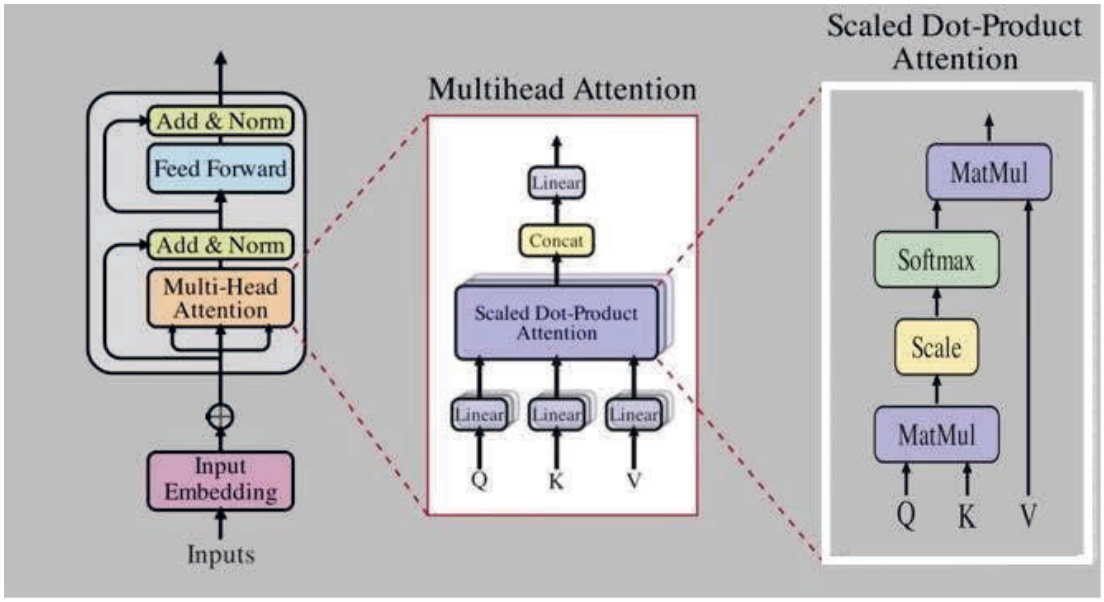

Fig 5. Stack of Encoder, Multiheaded Attention Mechanism and Scaled Dot Produced Attention

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (15)$$

In transformer models, q, k, and v represent queries, keys, and values, respectively. These are vectors that are derived from the input tokens, words from a sentence. The tokens are first converted into vectors x_1, x_2, ..., x_T, where T represents the total number of tokens in the input sequence. Each input vector x_i is then transformed into three different vectors: a query vector q_i a key vector k_i, and a value vector v_i. This transformation is performed using three different weight matrices that are learned during the training process of the transformer.

Self-attention has been useful in enhancing performance across various tasks, such as reading comprehension, abstractive summarization, and generating sentence representations independent of the task. Through its attention mechanism, BERT effectively concentrates on diverse segments of the input sequence during predictions, thereby it learns the context and interrelations within the sequence. This capability enables BERT to deeply understand both context and relationships in text sequences. The embeddings for each token in the sequence are derived from the output of BERT's final layer [60].

1.7.1 ProtBert

Recently, transfer learning techniques from NLP have significantly influenced protein bioinformatics [19] [20] [61]. These approaches, trained on extensive protein sequence databases, effectively learn informative features of these sequences of proteins. For example, Elnaggar *et al.* [20] utilized about 2.1 billion protein sequences to pre-train ProtTrans, a collection of transformer models derived from NLP. These methods enable the transforming of protein sequences into vector forms, which can be effectively used in a variety of applications, such as protein family classification. The use of existing pre-trained transformer models offers several advantages. It eliminates the need for manually creating complex features to represent protein sequences, leading

to more streamlined development of new neural network models. This approach also tends to enhance the predictive accuracy of these models [19].

1.8 Conclusion

In this chapter, we have discussed the benefits and challenges of implementing deep learning in protein domain research. We highlighted the potential of machine learning to advance computational biology while acknowledging the limitations of basic machine learning models in this field. Further on, we discussed how deep learning surpasses the limitations of traditional methods. Despite these advancements, there remain significant challenges in applying deep learning effectively in this area. We examined previous instances where deep learning was applied to specific challenges in this field, noting their promising nature. However, we concluded by identifying critical issues that need resolution for computational biologists to utilize. A primary concern is a need for large, labelled datasets for training of deep learning models. Hence, the amount of accessible data is a major challenge for deep learning methods in protein research, especially for protein classification tasks. Deep learning models need more data compared to traditional machine learning models to achieve optimal task-specific performance. Shortage of sufficient data for the training step of a deep learning model often results in overfitting to the training set, which in turn compromises its ability to generalize effectively. While next-generation sequencing methods are producing vast volumes of unlabeled omic data, the scarcity of adequately labelled data remains an issue. Additionally, if the available labels are not evenly distributed among different classes, there's a risk that the model will be overfit to the more prevalent class, further complicating the training process. A related issue arises when there is an imbalance in the labels;

the model may tend to overfit the class that is more heavily represented. This skew in label distribution can lead to a bias in the model's performance, favouring the majority class [4]. Quality of data is another challenge. Proteomics data, primarily derived from experimental sources, can be complex, noisy, heterogeneous and often incomplete, with only a portion of observations having valid labels. Besides these challenges, deep learning models require many parameters that need significant time and financial resources for optimization. The varied lengths of biological sequences often necessitate extensive padding, further slowing down the training process. Additionally, testing various model architectures to find the optimal one is both time-intensive and expensive, as each model's hyperparameters need to be precisely adjusted for the specific task [5].

Continuously analyzing and examining all omic data, without human involvement, remains a persistent challenge. In traditional machine learning approaches, feature engineering is typically a common step in data processing to address this issue. However, in recent times, machine learning professionals have started using advanced algorithms from the deep learning (DL) subfield. These algorithms reduce the necessity for extensive feature engineering but still require the specialized domain knowledge of experts [5]. Consequently, classifying a new protein sequence still relies heavily on feature engineering, utilizing prior knowledge and the expertise of professionals for accurate annotation [13]. The most successful applications of DL in omic research to date have been through supervised learning. Before starting the explanation of our promising model, which aims to address the challenges existing in the application of deep learning for proteomics data, and specifically for protein family classification, it is important to mention that this chapter also reviews two strategies for deep learning models: few-shot learning and transformer learning. We discussed how these approaches mitigate the need for large-scale datasets and explored the use of

transfer learning for non-manual feature selection. In the next chapter, we will delve into how these techniques have been utilized to achieve high accuracy in our findings. These results will assist omic data researchers in their investigations to produce groundbreaking results, not just in protein family classification tasks but in broader research problems as well.

Chapter 2: Deep Few-Shot Network for Protein Family Classification

2.1 Deep Few-Shot Network for Protein Family Classification

This chapter outlines the methodology employed in the experiments. We discussed the DL algorithms promising results in proteomics; however, their application is limited to the availability of massive data sets for training. Since the required data comes from experiments, it can be highly complex or incomplete [62]. As an alternative to address the mentioned limitations, techniques adopted from computer vision such as meta-learning can be generalized from a few observations. To rank the similarity between inputs, these networks employ a unique ranking structure, not requiring extensive training. In this research, we implemented a few-shot deep Siamese neural network for protein family classification. Our advanced few-shot Siamese neural network integrates twin pre-trained transformer ProtBERT [20] models. Siamese networks, a type of metric-based meta-learning model, excel in adapting to new tasks with a few examples as training data aka shot. This network consists of two identical sub-networks. These twin sub-networks with identical configurations, parameters, and weights and include an embedding extractor. The input of the network is a pair of samples, and each sub-network processes its input. Then their outputs are merged to evaluate the similarity between the inputs [63]. In our network, each pair of protein sequences is processed through ProtBERT to generate sequence embeddings. ProtBERT, offered by Elnaggar *et al.* (2021), has been pre-trained on an extensive dataset of 2.1 billion protein sequences [19]. It's built upon the principles of the BERT model. The embeddings produced by ProtBERT are passed through fully connected layers in our network, leading to the final classification results.

2.2 Data

We use the UniProtKB/Swiss-Prot dataset [64]. First, we retrieved reviewed protein sequences resulting from human protein in 20,426 (as of October 2023) with various lengths. Despite other sequence classification networks, we do not filter out the records according to the length of sequences, making sure the model is insensitive to the length of the sequence.

2.2.1 Data Preparation

Our dataset consists of labelled protein primary sequences from the UniProt database [64]. A total of 20,426 protein sequences, distributed across 5098 different families. We selected sequences with family names that resulted in 14,431 sequences being selected for training, validation, and testing. The data was then allocated into training, validation, and test subsets, with respective distributions of 70%, 10%, and 20%. Throughout this division, care was taken to maintain a consistent ratio of different classes in each subset (Table 1).

Total # of sequences.	20426
Total # of sequences with Family Name	14431
Total # of Families	5098
Total # of families with at least > 25 examples (Shot)	33

Table 1: Data distribution

We defined the problem of protein family classification as an N-way-K-shot classification [49] problem, where 'K' represents the count of training examples of protein sequences for each family, and 'N' denotes the total categories in our problem protein families or classes involved. The objective of N-shot learning is to accurately classify new data using these limited training samples. Modern deep learning methods need to efficiently utilize both large and small datasets. These techniques are pivotal in uncovering new understanding and expanding our comprehension of diseases, enhancing diagnostic accuracy, and crafting tailored treatment strategies [4].

2.3 Methodology

2.3.1 Architecture of Deep Few-Shot Network for Protein Family Classification

We utilized a deep few-shot Siamese neural network architecture applying a transformer to learn, latent features of protein family classification based on the primary sequences of protein pairs (Fig 6). These components will be explained in detail in the following sub-sections.

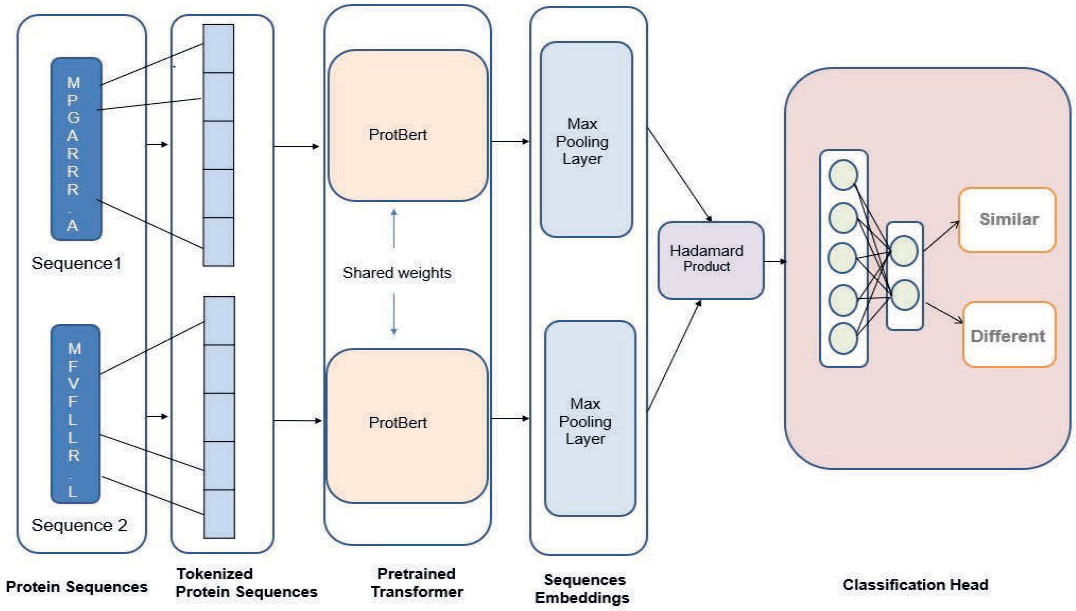

Fig 6: The high-level architecture design of Deep Few-Shot Network for Protein Family Classification

2.3.1.1 ProtBERT

In model architecture, we employed ProtBERT as the embedding extractor. ProtBERT [20], trained on roughly 2 billion protein sequences, employing a masked language modelling approach [17]. In the context of our research, ProtBERT interprets protein sequences as sentences, with amino acids functioning as the 'words' or fundamental elements. Specifically, we used the BFD variant of ProtBERT [19], which includes 30 layers, 16 attention heads, and 1,024 hidden layers. The model was trained over approximately 23.5 days using the Lamb optimizer [65] across 128 compute nodes, each equipped with 1,024 tensor processing units. Throughout its training, ProtBERT has learned to identify the biophysical properties of proteins as features, drawing on the billions of protein sequences it was exposed to during its training phase [19].

2.3.1.2 Siamese neural network

A deep Siamese neural network is utilized to analyze protein pairs. Our Siamese network contains two identical pre-trained ProtBERT models as embedding extractors, both sharing the same weights. The input consists of pairs of proteins, and the model learns the similarity between them over the training phase. Initially, we built an N-way K-shot few shot data sets and built the sequences of each protein pair. These sequences were then processed through our Siamese network model, as depicted in Figure 6. Within this framework, we utilized the twin pre-trained ProtBERT model to generate embeddings for each protein sequence. There are various methods to interpret the relationship between these sequence embeddings. While some researchers prefer to concatenate these embeddings, others focus on element-wise multiplication, commonly known as the Hadamard product. In our methodology, we utilized an integration layer that employs the

Hadamard product for combining the sequence embeddings. This technique has often been identified as particularly effective in representing the symmetric characteristics of the primary sequence of proteins [66] [19]. We utilized this deep Siamese model since protein data contains some classes or protein families with limited examples. Besides, the number of classes is very large. Previous works have shown the Siamese model's outperformance compared to other deep learning models, particularly for data where the number of classes is not known at the time of training, when the number of classes is very large, and when the number of examples per class is very small [67].

2.3.1.4 Classification Head

We integrated a classification head above the integration layer, consisting of fully connected layers. This classification head is structured in a bottleneck form, incorporating a mix of dropout and linear layers, and culminates in an output layer that employs a logistic function, sigmoid function (Equation 14).

$$S(x) = \frac{1}{1+e^{-x}} \qquad (14)$$

The sigmoid function stands as a frequently utilized activation function in deep/machine learning. It is mathematically expressed as a function where the input x is a linear blend of weights and feature values from the preceding layer. As x becomes smaller, the sigmoid function's output nears 0, whereas, with larger x values, it approaches 1. This function serves to map continuous real numbers into a bounded interval of (0,1). Such transformation ensures that the inputs to the subsequent layer are confined within a predictable range, thereby contributing to greater stability

in the network's weights. This setup enables the ranking of protein pairs, categorizing them as either the same family class, *genuine* or a different family class *imposite*. The bottleneck design is significant because it gradually reduces the number of neurons in each layer, enabling the network to concentrate on relative information while discarding what is redundant or irrelevant [19].

Loss Function

The goal of our designed Siamese-based Protein family classification network is to learn a model that can detect whether the paired proteins that are the input of the Siamese model are similar or not and detect the family classification of unknown proteins. The classifier is trained to distinguish between *genuine* instances; where protein sequences are paired from similar family classes, and *imposite* instances, where protein sequences are paired with different ones. Therefore, this task is often considered a binary classification problem [19].

Binary cross entropy is a loss function used for binary classification tasks involving two classes. In the proposed model, the predicted labels are 0 and 1, like logistic regression. Therefore, using the cross-entropy loss function is optimal, as it measures the disparity between the predicted and actual labels. The formula for this function, where 'y' represents the label and '$p(y)$' denotes the predicted probability, is as

follows:

$$\text{Cross Entropy Cost Function} = -\frac{1}{N}\sum_{i=1}^{N} y_i \, log(p(y_i)) + (1 - y_i) \, log(1 - p(y_i)) \quad (15)$$

In binary cross-entropy, the task is to distinguish between two classes. In the provided equation, 'p' refers to the predicted probability of the occurrence of a certain class, while 'y' indicates the actual output or label. This loss function effectively measures how well the model's probability predictions align with the actual labels.

2.4 Training and Validation

For our N-way-K-shot classification [54] problem, where 'K' represents the count of training examples, and protein sequences for each family, and 'N' denotes the total categories in our problem protein families or classes involved, we trained the model using pairs of protein sequences derived from training examples. In these pairs, sequences in one half were from the same class, while in the other half, the paired sequences were from different protein family classes. The goal of Protein family classification is to learn a model capable of identifying the similarity between pairs of proteins and determining the family classification of proteins whose classification is unknown. This task is commonly viewed as a binary classification problem [19] that utilized a classifier to detect between *genuine* instances; where protein sequences pair with similar family classes, and *imposite* instances, where protein sequences pair with different ones.

We chose protein families with at least 25 examples to accommodate various N-way, K-shot learning scenarios ($n \in \{5,10, 20,25\}$). For instance, in a 5-shot learning setup, we selected five samples from each family and then created pairs as described. By initially selecting families with more than 25 examples and we have 33 families or classes. There our experiment was a 33-way, K-shot learning scenarios ($n \in \{5,10, 20,25\}$).

We ensured the presence of the same families across all 4 datasets, facilitating a consistent comparison of our architecture's performance. After splitting our dataset into a training set, we created 4 different data setups for n, with n ranging from {5,10,20,25}. For each setup, we randomly selected n sequences from each family within the training set. Within each new setup, these sequences were used to form pairs as previously described. Half of these pairs were *'Genuine*

pairs', consisting of two sequences from the same family class. The other half were *'Imposite pairs'*, combining sequences from different family classes. The same approach was used for the validation data set.

2.5 Model Performance Metrics (Evaluation)

For the performance evaluation of classifiers in a binary classification task, where outcomes are labelled either *genuine* or positive (p) or *imposite* negative (n), a range of standard performance metrics are employed.

The outcomes in binary classification are categorized into four types:

- TP: True Positive, when a positive class is correctly predicted,
- FP: False Positive, when a negative class is incorrectly predicted as positive,
- TP: True Negative, when a negative class is correctly predicted,
- FN: False Negative when a positive class is incorrectly predicted as negative.

For instance, if a prediction outcome is positive (p) and the true value is also positive (p), it is classified as TP, but if the true value is negative (n), it falls under FP. Similarly, a prediction and true value both being negative (n) is classified as TN, while a positive true value (p) predicted as negative (n) is an FN. Various metrics derived from these four outcomes, such as accuracy, precision, recall, and F1 score, provide a comprehensive evaluation of machine learning classifiers.

2.5.1 Recall

Recall is an important model evaluation performance metric in machine/deep learning that represents how well a model is at correctly identifying positive cases. Recall (Eq 16) is defined as

the ratio of True Positive (TP) predictions to the total number of actual positive examples [68]. In essence, a higher Recall value is desirable as it implies fewer positive instances being incorrectly predicted as negative, which is vital in applications, where failing to identify positive cases could lead to serious repercussions such as in cancer detection models Powers
(2020).

$$\text{Recall} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}} \qquad (16)$$

2.5.2 Precision

Precision is defined as the number of true positive predictions divided by the total number of positive predictions (which includes both true positives and false positives) (17). In simpler terms, it measures the accuracy of the positive predictions made by the classifier [68] In the context of predicting protein sequence family classes, a high precision value indicates that the classifier is effective at correctly identifying members of a given protein family, with fewer instances of incorrectly predicting a protein sequence as belonging to that family when it does not. This is particularly important in biological and medical research, where accurate classification of protein sequences can have significant implications for understanding biological processes and developing treatments.

$$\text{Precision} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Positive (FP)}} \qquad (17)$$

2.5.3 F1 Score

The F1 score, often referred to as the F-score or F-measure, is determined by computing the weighted harmonic mean of precision and recall. This is exemplified in Equation (18), where the F1 score is represented as the harmonic mean of these two metrics. The F1 score can vary from 0 to 1, indicating its range of possible values.

$$\text{F1 score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{18}$$

2.5.4 Accuracy

Accuracy is one metric for evaluating classification models. For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{\text{True Positive (TP)} + \text{True Negative(TN)}}{\text{True Positive (TP)} + \text{True Negative(TN)} + \text{False Positive (FP)} + \text{False Negative(FN)}} \tag{19}$$

2.6 Testing Strategy

We tested our architecture with an N-way one-shot learning testing strategy [22] [50]. In this task, within our separated testing data test, we created sets of protein sequence pairs, where each set consisted of one protein sequence paired with N different sequences. Within each set, there was one *genuine* and one *imposite* pair. To ensure a comprehensive and reliable evaluation, this process was repeated for all family classes. Moreover, to enhance the reliability of the results, the testing was done on 10 one-shot tasks, providing a robust and thorough assessment of the model's ability to recognize and differentiate between various protein families.

Chapter 3: Results and Discussion

3.1 Results

Our results, as shown in Table (2) for training setup and Table (3) for validation setup, indicate that our proposed Deep Few-Shot Network for Protein Family Classification accurately detects protein family classifications. Based on evaluation performance metrics, our experimental results demonstrate a precision of 97.3% and an F1 score of 96.2% for only 25 shot (examples) for each class. Our findings show superior performance compared to baseline models for protein family classification, which typically utilize the entire dataset rather than a few shot, a small sample size for training models, as is the nature of this biological dataset (Fig 7). As seen in Table (4) test setup of our Deep few shot network can detect family class with a high F1 score 94.2%.

Training setup Shot Size	F1	Precision	Recall	Training Loss
5 Shot	40.5%	40.3%	55.0%	0.873
10 Shot	64.7%	63.8%	64.5%	0.336
20 Shot	82.8%	82.1%	83.6%	0.085
25 Shot	96.2%	97.3%	95.2%	0.071

Table 2: Training performance evaluation for n-shot (n-example) setup

Validation setup Shot Size	F1	Precision	Recall	Training Loss
5 Shot	38.5%	38.2%	39.8%	1.08
10 Shot	63.2%	60.9%	61.6%	0.434
20 Shot	79.6%	79.2%	80.1%	0.181
25 Shot	93.9%	94.2%	93.6%	0.079

Table 3: Validation performance evaluation for n-shot (n-example) setup

Test setup Shot Size	F1	Precision	Recall	Cross Entropy
5 Shot	36.1%	42.1%	31.1%	0.876
10 Shot	59.6%	60.1%	59.2%	0.395
20 Shot	80.9%	79.5%	82.7%	0.198
25 Shot	**94.2%**	95%	93.5%	0.11

Table 4: Testing setup performance evaluation for n-shot (n-example)

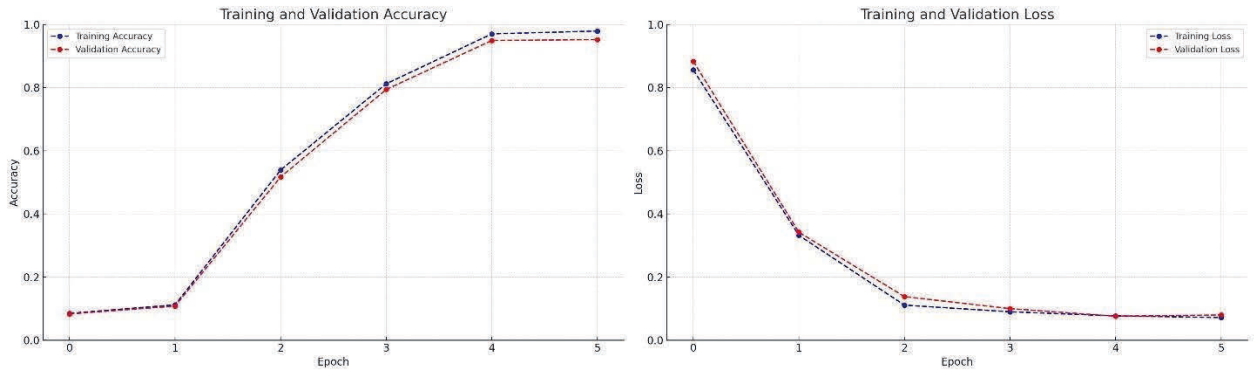

Fig 7: Training and validation accuracy for 33-way, 25-shot learning setting.

We compared our deep Few-Shot Network for Protein Family Classification with 3 baseline models. 1) CNN model, 2) SVM 3) BiGRU (Gated Recurrent Unit). We were primarily focused on analyzing the changes in performance of baseline models with different numbers of shot sizes, and whether the baseline models can perform well at those shot sizes. Our highly accurate offered model ability for detecting family classes only having a few examples can be seen in Fig 6. Besides, in the testing setup, shown in Table 4 our model resulted in 94.2% for the F1 metric with only 25 shot in the testing setup while Lee *et al.*[13] GRU model showed 94.8% for test F1 metric using the entire UniProt dataset.

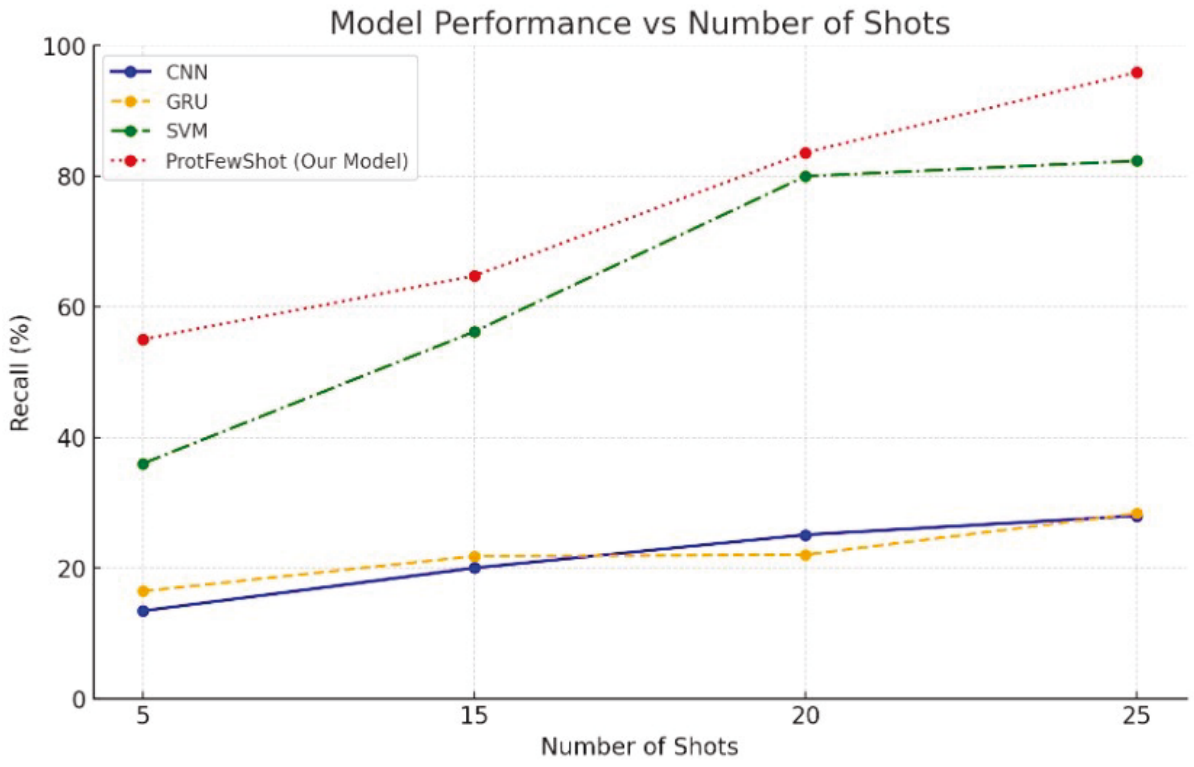

Fig 8: Performance of the Deep Few Shot model (ProtFewShot) and three baseline models, i.e., CNN, GRU, SVM for n-shot learning setting $(n \in \{5,10, 20, 25\})$.

3.2 Conclusion and Future Work

Although deep learning algorithms, which are mostly adopted from NLP, have shown breathtaking performance in Proteomics data analysis, they largely depend on large-scale labelled datasets, expert knowledge, and intervention for feature engineering. However, due to the nature of biological data, access to such large datasets is not possible because of the complexity and noisiness of the data. Specifically, in our research field data, there are family classes of proteins that have few examples available. Considering these limitations, we presented a deep few-shot learning architecture capable of detecting protein family classes with high accuracy, even when

only a few examples of each family are available. In our model, we used a pre-trained model, ProtBert, that was a novel transformer for protein sequences and a fully connected model for our classifier section and reached the high-accuracy performance. Although this pre-trained model increased the accuracy of the model they extracted a large number of parameters that can be officialized by solutions like distillation or sparse parameters that lead selection of a more optimized number of parameters.

For future work, we are working on the explainability of our deep learning architecture. It provides insights into how the classifier makes its predictions by highlighting the features that the model relies on. This explanatory framework aids in making the model more interpretable for end users.

References

[1] P. M. Sonsare and C. Gunavathi, “Investigation of machine learning techniques on proteomics: A comprehensive survey,” Prog. Biophys. Mol. Biol., vol. 149, pp. 54–69, Dec. 2019, doi: 10.1016/j.pbiomolbio.2019.09.004.

[2] A. Wang, “Deep Learning Methods for Protein Family Classification on PDB Sequencing Data.” arXiv, Jul. 14, 2022. Accessed: Nov. 21, 2023. Available: http://arxiv.org/abs/2207.06678

[3] A. Ghaemmaghami, A. Schiffauerova, and A. Ebadi, “Which Keyword Extraction Method Performs Better for Emerging Technology Detection?,” in 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey: IEEE, Oct. 2022, pp. 613–618. doi: 10.1109/ISMSIT56059.2022.9932656.

[4] Lennox,M. “Deep learning of proteomics data,” 2021. Student thesis: Doctoral Thesis › Doctor of Philosophy, https://pure.qub.ac.uk/en/studentTheses/deep-learning-of-proteomics-data

[5] W. Zheng, L. Yang, R. J. Genco, J. Wactawski-Wende, M. Buck, and Y. Sun, “SENSE: Siamese neural network for sequence embedding and alignment-free comparison,” Bioinformatics, vol. 35, no. 11, pp. 1820–1828, Jun. 2019, doi: 10.1093/bioinformatics/bty887.

[6] D. Zhang and M. R. Kabuka, “Protein Family Classification from Scratch: A CNN Based Deep Learning Approach,” IEEE/ACM Trans. Comput. Biol. Bioinform., vol. 18, no. 5, pp. 1996–2007, 2021, doi: 10.1109/TCBB.2020.2966633.

[7] S. Jamali, “An Explainable Deep Few-shot Network for Protein Family Classification,” Proc. Can. Conf. Artif. Intell., Jun. 2023, doi: 10.21428/594757db.033df5af.

[8] J. Shen *et al.*, “Predictingprotein–protein interactions based only on sequences information,” Proc. Natl. Acad. Sci., vol. 104, no. 11, pp. 4337–4341, Mar. 2007, doi: 10.1073/pnas.0607879104. [9] “https://www.sciencedirect.com/topics/medicine-and-dentistry/protein-structure.”

[10] C. E. Cook, M. T. Bergman, R. D. Finn, G. Cochrane, E. Birney, and R. Apweiler, “The European Bioinformatics Institute in 2016: Data growth and integration,” Nucleic Acids Res., vol. 44, no. D1, pp. D20–D26, Jan. 2016, doi: 10.1093/nar/gkv1352.

[11] M. Kim and I. Tagkopoulos, “Data integration and predictive modelling methods for multi-omics datasets,” Mol. Omics, vol. 14, no. 1, pp. 8–25, 2018, doi: 10.1039/C7MO00051K.

[12] D. Grapov, J. Fahrmann, K. Wanichthanarak, and S. Khoomrung, “Rise of Deep Learning for Genomic, Proteomic, and Metabolomic Data Integration in Precision Medicine,” Omics J. Integr. Biol., vol. 22, no. 10, pp. 630–636, Oct. 2018, doi: 10.1089/omi.2018.0097.

[13] J. Lee *et al.*, “BioBERT: a pre-trained biomedical language representation model for biomedical text mining,” 2019, doi: 10.48550/ARXIV.1901.08746.

[14] “‘Semeval-2019 task 3: Emocontext contextual emotion detection in text,’ in Proceedings of the 13th International Workshop on Semantic Evaluation, pp. 39–48”.

[15] V. Tshitoyan *et al.*, “Unsupervised word embeddings capture latent knowledge from materials science literature,” Nature, vol. 571, no. 7763, pp. 95–98, Jul. 2019, doi: 10.1038/s41586-019-1335-8.

[16] M. E. Peters *et al.*, “Deep contextualized word representations,” 2018, doi: 10.48550/ARXIV.1802.05365.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018, doi: 10.48550/ARXIV.1810.04805.
[18] C.-A. Leimeister and B. Morgenstern, "kmacs: the k -mismatch average common substring approach to alignment-free sequence comparison," Bioinformatics, vol. 30, no. 14, pp. 2000–2008, Jul. 2014, doi: 10.1093/bioinformatics/btu331.
[19] S. Madan, V. Demina, M. Stapf, O. Ernst, and H. Fröhlich, "Accurate prediction of virus-host protein-protein interactions via a Siamese neural network using deep protein sequence embeddings," Patterns N. Y. N, vol. 3, no. 9, p. 100551, Sep. 2022, doi: 10.1016/j.patter.2022.100551.
[20] A. Elnaggar *et al.*, "ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning," IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 10, pp. 7112–7127, Oct. 2022, doi: 10.1109/TPAMI.2021.3095381.
[21] M. Heinzinger *et al.*, "Modeling aspects of the language of life through transfer-learning protein sequences," BMC Bioinformatics, vol. 20, no. 1, p. 723, Dec. 2019, doi: 10.1186/s12859-019-3220-8
[22] A. Ebadi, Azimi, H, Xi, P, Tremblay, S, and Wong, A, "COVID-Net FewSE: An Open-Source Deep Siamese Convolutional Network Model for Few-Shot Detection of COVID-19 Infection from X-Ray Images,", Accessed on openjournals. waterloo: doi:10.15353/jcvis.v7i1.4891
[23] B. R. Szymczyna, R. E. Taurog, M. J. Young, J. C. Snyder, J. E. Johnson, and J. R. Williamson, "Synergy of NMR, computation, and X-ray crystallography for structural biology," Struct. Lond. Engl. 1993, vol. 17, no. 4, pp. 499–507, Apr. 2009, doi: 10.1016/j.str.2009.03.001.
[24] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking," 2016, doi: 10.48550/ARXIV.1606.09549.
[25] H. Öztürk, A. Özgür, and E. Ozkirimli, "DeepDTA: deep drug-target binding affinity prediction," Bioinforma. Oxf. Engl., vol. 34, no. 17, pp. i821–i829, Sep. 2018, doi: 10.1093/bioinformatics/bty593.
[26] Cancer Genome Atlas Research Network *et al.*, "The Cancer Genome Atlas Pan-Cancer analysis project," Nat. Genet., vol. 45, no. 10, pp. 1113–1120, Oct. 2013, doi: 10.1038/ng.2764.
[27] P. D. Sandaruwan and C. T. Wannige, "An improved deep learning model for hierarchical classification of protein families," PLOS ONE, vol. 16, no. 10, p. e0258625, Oct. 2021, doi: 10.1371/journal.pone.0258625.
[28] S. Karlin, "Dinucleotide relative abundance extremes: a genomic signature," Trends Genet. TIG, vol. 11, no. 7, pp. 283–290, Jul. 1995, doi: 10.1016/s0168-9525(00)89076-9.
[29] J. G. Meyer, "Deep learning neural network tools for proteomics," Cell Rep. Methods, vol. 1, no. 2, p. 100003, Jun. 2021, doi: 10.1016/j.crmeth.2021.100003.
[30] N. Yuvaraj and P. Vivekanandan, "An efficient SVM-based tumour classification with symmetry Nonnegative Matrix Factorization using gene expression data," in 2013 International Conference on Information Communication and Embedded Systems (ICICES), Chennai: IEEE, Feb. 2013, pp. 761–768. doi: 10.1109/ICICES.2013.6508193.
[31] Y. Guo, L. Yu, Z. Wen, and M. Li, "Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences," Nucleic Acids Res., vol. 36, no. 9, pp. 3025–3030, May 2008, doi: 10.1093/nar/gkn159.
[32] C. Leslie, E. Eskin, and W. S. Noble, "'The spectrum kernel: A string kernel for SVM protein classification,' in Biocomputing 2002, pp. 564–575, World Scientific, 2001.".
[33] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*, 2nd ed. Springer International Publishing, 2021.

[34] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” Proc. IEEE, vol. 78, no. 10, pp. 1550–1560, Oct. 1990, doi: 10.1109/5.58337.
[35] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” 2013, doi: 10.48550/ARXIV.1310.4546.
[36] B. Alipanahi, A. Delong, M. Weirauch, and B. Frey, “Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning,” Nat. Biotechnol., vol. 33, Jul. 2015, doi: 10.1038/nbt.3300.
[37] S. Seo, M. Oh, Y. Park, and S. Kim, “DeepFam: deep learning based alignment-free method for protein family modelling and prediction,” Bioinformatics, vol. 34, no. 13, pp. i254–i262, Jul. 2018, doi: 10.1093/bioinformatics/bty275.
[38] B. Szalkai and V. Grolmusz, “Near perfect protein multi-label classification with deep neural networks,” Methods San Diego Calif, vol. 132, pp. 50–56, Jan. 2018, doi: 10.1016/j.ymeth.2017.06.034.
[39] D. Quang and X. Xie, “DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences,” Nucleic Acids Res., vol. 44, no. 11, pp. e107–e107, Jun. 2016, doi: 10.1093/nar/gkw226.
[40] Y. LeCun *et al.*, “Backpropagation Applied to Handwritten Zip Code Recognition,” Neural Comput., vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.
[41] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
[42] T. Bepler and B. Berger, “Learning protein sequence embeddings using information from structure.” arXiv, Oct. 16, 2019. Accessed: Nov. 23, 2023. Available: http://arxiv.org/abs/1902.08661
[43] R. J. Williams and D. Zipser, “A Learning Algorithm for Continually Running Fully Recurrent Neural Networks,” Neural Comput., vol. 1, no. 2, pp. 270–280, Jun. 1989, doi: 10.1162/neco.1989.1.2.270.
[44] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” IEEE Trans. Neural Netw., vol. 5, no. 2, pp. 157–166, Mar. 1994, doi: 10.1109/72.279181.
[45] B. Krause, L. Lu, I. Murray, and S. Renals, “Multiplicative LSTM for sequence modelling.” arXiv, Oct. 12, 2017. Accessed: Nov. 23, 2023. Available: http://arxiv.org/abs/1609.07959
[46] J. Hanson, Y. Yang, K. Paliwal, and Y. Zhou, “Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks,” Bioinformatics, vol. 33, no. 5, pp. 685–692, Mar. 2017, doi: 10.1093/bioinformatics/btw678.
[47] X.-X. Zhou *et al.*, “pDeep: Predicting MS/MS Spectra of Peptides with Deep Learning,” Anal. Chem., vol. 89, no. 23, pp. 12690–12697, Dec. 2017, doi: 10.1021/acs.analchem.7b02566.
[48] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Gated Feedback Recurrent Neural Networks.” arXiv, Jun. 17, 2015. Accessed: Nov. 24, 2023. [Online]. Available: http://arxiv.org/abs/1502.02367 [49] V. Vasnt Alur, “Detection of Melanoma Cancer Using Few-shot Learning,” The University of Toledo, 2021.
[50] R. R. Chowdhury and D. R. Bathula, “Influential Prototypical Networks for Few Shot Learning: A Dermatological Case Study.” arXiv, Dec. 24, 2021. Accessed: Nov. 25, 2023. [Online]. Available: http://arxiv.org/abs/2111.00698
[51] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, “Low Data Drug Discovery with One-shot Learning.” arXiv, Nov. 10, 2016. Accessed: Nov. 25, 2023. [Online]. Available: http://arxiv.org/abs/1611.03199
[52] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a Few Examples: A Survey on Fewshot Learning,” ACM Comput. Surv., vol. 53, no. 3, pp. 1–34, May 2021, doi: 10.1145/3386252.

[53] M. Yu *et al.*, “Diverse Few-Shot Text Classification with Multiple Metrics.” arXiv, May 19, 2018. Accessed: Nov. 25, 2023. [Online]. Available: http://arxiv.org/abs/1805.07513
[54] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.
[55] S. Azadi, M. Fisher, V. Kim, Z. Wang, E. Shechtman, and T. Darrell, “Multi-Content GAN for FewShot Font Style Transfer.” arXiv, Dec. 01, 2017. Accessed: Nov. 25, 2023. [Online]. Available: http://arxiv.org/abs/1712.00516
[56] Z. Luo, Y. Zou, J. Hoffman, and L. Fei-Fei, “Label Efficient Learning of Transferable Representations across Domains and Tasks.” arXiv, Nov. 30, 2017. Accessed: Nov. 25, 2023. [Online]. Available: http://arxiv.org/abs/1712.00123
[57] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.
[58] J. Bromley et al., “Signature Verification Using a ‘Siamese’ Time Delay Neural Network,” Int. J. Pattern Recognit. Artif. Intell., vol. 07, no. 04, pp. 669–688, Aug. 1993, doi: 10.1142/S0218001493000339.
[59] “Self-Attention Using Scaled Dot-Product Approachhttps://youtu.be/1IKrHh2X0F0?si=LI5qeP5QOtBFI70z.” YouTube video
[60] A. Vaswani et al., “Attention Is All You Need.” arXiv, Aug. 01, 2023. Accessed: Jan. 27, 2024. [Online]. Available: http://arxiv.org/abs/1706.03762
[61] S. Min, S. Park, S. Kim, H.-S. Choi, B. Lee, and S. Yoon, “Pre-Training of Deep Bidirectional Protein Sequence Representations with Structural Information,” 2019, doi: 10.48550/ARXIV.1912.05625.
[49] P. D. Sandaruwan and C. T. Wannige, “An improved deep learning model for hierarchical classification of protein families,” PLOS ONE, vol. 16, no. 10, p. e0258625, Oct. 2021, doi:
[50] 10.1371/journal.pone.0258625.
[51] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese Neural Networks for One-shot Image Recognition,” 2015. https://www.semanticscholar.org/paper/Siamese-Neural-Networks-for-One-Shot-Image-Koch/f216444d4f2959b4520c61d20003fa30a199670a
[52] “UniProt, ‘ Proteins UniProt Knowledgebase’ From : http://www.uniprot.org, 2023.”
[53] You,Y and et L., “Large Batch Optimization for Deep Learning: Training BERT in 76 minutes”. https://arxiv.org/pdf/2111.00856.pdf
[54] M. Chen et al., “Multifacetedprotein–protein interaction prediction based on Siamese residual RCNN,” Bioinformatics, vol. 35, no. 14, pp. i305–i314, Jul. 2019, doi: 10.1093/bioinformatics/btz328.
[55] N. Roberts, P. S. Purushothama, V. T. Vasudevan, and S. Ravichandran, C. Zhang, W. H. Gerwick, and G. W. Garrison, “Using deep siamese neural networks to speed up natural products research.,” 2018.
[56] K. M. Ting, “Confusion Matrix,” in Encyclopedia of Machine Learning, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2011, pp. 209–209. doi: 10.1007/978-0-387-30164-8_157.