UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE POSTGRADO Y EDUCACIÓN CONTINUA

PREDICTION OF THE OPTIMAL GROWTH PH OF ACIDOPHILES BY PROTEIN SEQUENCE ANALYSIS: A DEEP LEARNING APPROACH

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE DATOS,

DIEGO NAHUEL CORTEZ MILÁN

PROFESOR GUÍA:
FELIPE TOBAR HENRÍQUEZ

MIEMBROS DE LA COMISIÓN:
ALEJANDRO MAASS SEPÚLVEDA
IVÁN SIPIRÁN MENDOZA

Este trabajo ha sido parcialmente financiado por:
Proyecto Basal FB0008.

SANTIAGO DE CHILE
2024

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE DATOS
POR: DIEGO NAHUEL CORTEZ MILÁN
FECHA: 2024
PROF. GUÍA: FELIPE TOBAR HENRIQUEZ

PREDICCIÓN DEL PH ÓPTIMO DE ORGANISMOS ACIDÓFILOS MEDIANTE ANÁLISIS DE SUS SECUENCIAS PEPTÍDICAS CON DEEP LEARNING

Las proteínas son moléculas formadas por una cadena de moléculas más pequeñas llamadas aminoácidos. Las propiedades de cada proteína como su función, familia y propiedades fisicoquímicas están codificadas de manera compleja en su secuencia de aminoácidos. Algunas proteínas son capaces de resistir condiciones extremas, como aquellas presentes en el envoltorio de microorganismos llamados acidófilos que viven en condiciones extremadamente ácidas (pH <3). En este trabajo, se desarrollan modelos de *deep learning* para decodificar la resistencia a ácido de las proteínas. Más de 150000 proteínas de envoltorio de organismos que viven a pH 1 a 7 se utilizaron para entrenar múltiples modelos de regresión, desde modelos lineales simples hasta modelos de NLP. Los resultados muestran que existen cambios en los patrones de la secuencia aminoacídica de los proteínas a diferentes pH, los cuales reflejan capacidades de resistencia a condiciones extremadamente ácidas. Los mejores modelos de *machine learning* clásico fueron modelos de tipo *gradient boosting* entrenados en atributos de las proteínas y codificaciones de transformer. El mejor modelo *deep learning* fue una nueva arquitectura que combina LSTM y extracción de atributos mediante CNN y atención. Se diseñó una heurística para predecir el pH óptimo de crecimiento de organismos unicelulares en base a la agregación de las predicciones individuales de cada una de sus proteínas, con un error absoluto medio de 0.61 unidades de pH. Estos resultados representan un importante paso en el desarrollo de herramientas bioinformáticas para la caracterización de proteínas y genomas.

ABSTRACT OF THE REPORT TO QUALIFY TO THE DEGREE OF
MASTER OF DATA SCIENCE
AND TO THE DEGREE OF INSERT ENGINEERING
BY: DIEGO NAHUEL CORTEZ MILÁN
DATE: 2024
PROF. GUÍA: FELIPE TOBAR HENRIQUEZ

PREDICTION OF THE OPTIMAL GROWTH PH OF ACIDOPHILES BY PROTEIN SEQUENCE ANALYSIS: A DEEP LEARNING APPROACH

Proteins are molecules formed by a chain of smaller molecules called amino acids. The properties of the protein are deep coded into its sequence, such as function, family and physichochemical properties. Some proteins are able to resist extreme conditions such as the envelope proteins of microorganisms that live at extremely acid conditions (pH <3) called acidophiles. In this work, machine learning tools were developed to decode the acid resistance of proteins. Over 150.000 envelope proteins from organisms that thrive at pH 1 to 7 were gathered to train multiple regression models from simple linear models to deep learning NLP models. The results show that there are changes in the amino acidic patterns across pH that are probable consequences of the proteins adaptation to resist extremely acidic conditions. The best performing classical machine learning model was a gradient boosting model trained on amino acidic features, transformer encodings and manual features extracted from the proteins sequence. The best performing deep learning model was a novel architecture that mixes a 2 layer LSTM, CNN layers for keys and values extraction, and optimized attention mechanisms to extract multiple features from the LSTM outputs. An heuristic was designed which permitted the prediction of the optimal growth of unicellular organisms from the aggregation of the individual prediction of their exposed proteins, with a mean average error of 0.61 pH units. These results represent an important step in the development of bioinformatic tools for the characterization of proteins and genomes.

Acknowledgements

Quiero agradecer a mi mamá y hermana Gaby por brindar el apoyo necesario en mi casa para poder realizar este magister. A mi polola Nicole que con infinita paciencia me apoyó y me brindó los espacios necesarios para desarrollar esta tesis y el magister. A mi hermano Amaru quien me brindó ayuda en la teoría matemática y así poder nivelarme.

Quiero agradecer también a David Holmes, a quien debo la trayectoria que llevó al desarrollo de esta tesis. También al profesor Felipe Tobar que me guió en el proceso académico y a crecer en el área del *machine learning*. A mis compañeros de magister con los que nos acompañamos en las trasnochadas de los proyectos.

Por último, quiero agradecer a Alexandra Elbakyan, sin quien la ciencia no sería posible en países como Chile.

Table of Content

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Bioinformatics is an area of biology in which living organisms are studied using the informatic analysis of high-volume data that contains information about the biological traits of the organisms. Specifically, this involves the analysis of sequence data such as the genomic DNA sequence and the protein sequence, structure data such as protein structure, and genetic expression arrays [90]. This study revolves around protein sequence analysis.

A genome is the collection of all the DNA sequences that are located inside each cell of an organism. Together, they contain most of the information necessary to identify and virtually recreate all the molecular and, indirectly, non-molecular characteristics of the organism. These sequences are composed of subunits and can be over 10^{11} characters long, storing 2 bits of information in each. The analysis of such high-volume data requires the development of efficient computational algorithms and methods for obtaining, searching, comparing, and analyzing them to mine valuable information that describes genomic traits of the organism. The now-not-so-recent boom in high-throughput sequencing technologies along with the exponential growth of genomic databases allows for ever increasing availability of the genomes of the organisms known to science [80]. This, together with the development of highly efficient bioinformatic tools and publicly available genome repositories, enables the possibility of studying organisms without the need for expensive and time consuming biochemical or biological studies. The usefulness of bioinformatics then relies on how accurate is the prediction of biological traits from the genome of an organism, for which the training of accurate machine learning models is crucial.

This study focuses on proteins, which play a pivotal role in the expression and processes of biological functions [79], and are the primary effectors of the genes that carry the organisms' information across generations. The proteins' sequence is unambiguously encoded in the DNA sequence, meaning there is a direct translation from DNA to protein sequence. The characteristics of the proteins are in turn encoded in the protein sequence. However, these are encoded in a complex way which has not been fully understood. One of these protein traits is the capacity to resist extreme pH conditions, particularly extremely low pH (acid). Some proteins are naturally exposed to extremely low pH and are able to persist in their unaltered conformation where other proteins can't. This is the case of the proteins of acidophilic microorganisms, which are microorganisms that grow optimally at low pH [3].

The main goal of this study is to be able to decode the acid resisting capabilities of proteins encoded in the protein sequence using machine learning, deep learning, and NLP tools.

1.2 Hypothesis

In this study the following hypotheses will be explored and tested:

- It is possible to predict the pH at which the proteins are naturally exposed to from their amino acidic sequences with less than 1 pH units of absolute error.

- The optimal growth pH of unicellular microorganisms can be predicted by aggregating the individual predictions of their proteins with less than 0.5 pH units of absolute error.

1.3 Objectives

1.3.1 General objective

The main objective of this thesis is *to explore and compare classical machine learning algorithms and deep learning architectures to train models that estimate the acid resistance capabilities of proteins, and permit the characterization of their respective source organism in terms of optimal growth pH*.

1.3.2 Specific objectives

The specific objectives of this thesis are the following:

1. To obtain a dataset of proteins exposed to the extracellular pH of bacteria and archaea that have optimal gwoth pH from pH 0 (acidic) to pH 7 (neutral).

2. To extract relevant features of the proteins using existing machine learning models and bioinformatic tools to permit higher accuracy in the obtained models.

3. To explore the protein attributes in relation to their associated pH and test whether there are differences at different pH.

4. To train regression models that predict the optimal pH of the source organism of an exposed protein from its sequence by exploring classical ML models that use features as input and deep learning architectures that use the amino acid sequence, selecting the model with best performance.

5. To design a heuristic to estimate the optimal pH of an organism from the individual prediction

of each of its exposed proteins.

1.4 Contributions

The main contributions of this thesis are the following:

- The determination of which traits correlate or are good predictors of the pH a protein can resist.
- A bioinformatic method to predict the acid resistance capabilities of a protein from its sequence.
- A pipeline of bioinformatic methods that permits the estimation of the optimal growth of an organism from its genomic sequence, thus allowing for characterization of organisms without the need for laboratory cultures.
- The design and testing of novel deep learning architectures for the processing of protein data and potentially human language.
- The availability of multiple Python modules for the design and training of diverse deep learning architectures.
- The exploration and testing of training methods and metrics for the handling of unbalanced data in regression problems.

Chapter 2

Background

2.1 Biological background

Prior to diving into this thesis' core topics, the general biological concepts necessary to understand the context of this work will be shortly overviewed.

2.1.1 Cell biology

The cell is the principal physiological and reproductive unit of all living organisms [91]. The most basic living organisms, the unicellular organisms, are only composed of a single cell, while the most complex organisms like animals and plants are muilticellular organisms, meaning they are composed of multiple cells [92]. Cells can multiply by cloning themselves, producing tissue growth in multicellular organisms and permitting the organisms' reproduction in both multicellular and unicellular organisms.

Physically, cells are aqueous compartments typically in the micrometer scale (10^{-6} m) which are surrounded by the cell membrane (Fig. 2.1A). The cell membrane is a lipidic bilayer composed mainly of phospholipids [91, 92]. In simple terms, cells are microscopic water droplets coated in a thin oil layer. These water droplets are self-sustained by a complex chemical machinery called the cell's metabolism which regulates the membrane's stability and permeability, the cell's reproduction, feeding and motility, the internal environment's homeostasis and even the processing of information [93]. This unlikely machinery operates by multiple components working simultaneously, which in turn feed on the energy produced by the metabolism itself. This defines a delicate stability well – life itself – where if one component fails, the whole system collapses, thus defining the concepts of death and self-sustainable autopoiesis [94].

The organism's gemome can be found inside the cell, either inside the cell nucleous in complex (eukaryotic) cells such as animal cells, or suspended in the cytoplasm (the cell's internal environment) in simple (prokaryotic) cells such as bacteria and archaea [92]. The genome of an organism is one or several DNA molecules which contain the information to recreate all cellular functions and, at least in unicellular organisms, encode all the characteristics of the organism [97]. It is a

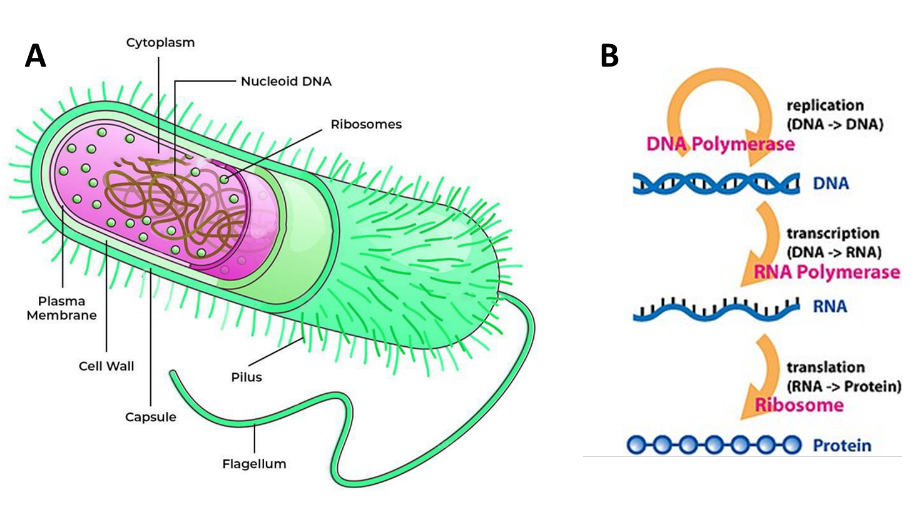

Fig 2.1: A) A schematic representation of a gram-positive bacteria, which have a cell wall. Gram-negative bacteria, on the other hand, have a double membrane and periplasmic space. B) A schematic representtion of the central dogma of molecular biology. Adapted from [120].

long double strand of smaller molecules called nucleobases or nitrogenous bases, of which there are 4 types: adenine (A), cytosine (C), guanine (G), cytosine (C) and thymine (T) [95, 96]. Then, the DNA can be represented as a long string of 4 possible characters with variable length. The genome of an organism can be from about 1 million (10^6) bases long in some bacteria up to about 150 billion ($1.5 \cdot 10^{11}$) bases long in some trees [10, 11].

The genome is subdivided into informational units called genes. Each gene encodes a different biological function or subfunction, which is effectuated in a process called gene expression [96]. In this process, each gene produces different proteins, which play different cellular functions that vary depending on the protein's family and structure (Fig. 2.1B).

2.1.2 Biochemistry of proteins

Proteins are biomolecules involved in most biological functions [79]. Like DNA, proteins are long molecules formed by smaller subunits called amino acids. The amino acids are linked by molecular bonds in a chain-like pattern. There are 20 different types of amino acids, so a protein can be represented by a string of 20 types of characters of variable length. Proteins from unicellular organisms are typically between 50 and 600 amino acids long but can be over 2000 amino acids long [12].

Proteins are encoded in the DNA in an unambiguous manner. Every string of 3 consecutive DNA nucleobases (codon) encodes a single amino acid. As there are 4^3 = 64 codon combinations, this translation from DNA to proteins is redundant. The same codon will always encode the same amino acid [98]. The process in which proteins are created from the information present in genes is called translation [96].

Protein structure

The protein structure is defined in 4 levels: primary, secondary, tertiary and cuaternary structure (Fig. 2.2). The chain of amino acids that forms a protein is its primary structure, as it indirectly defines all the features of the protein. After being created, the proteins fold in a hierarchical way,

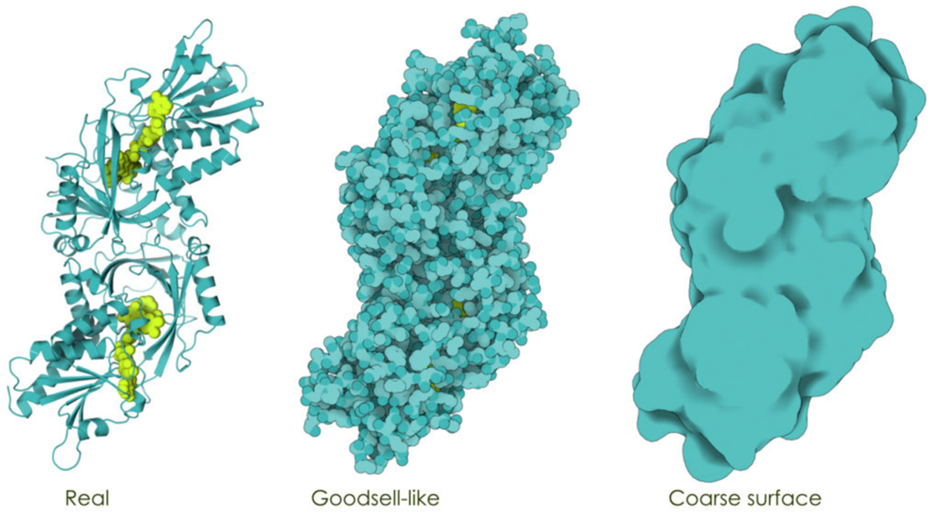

Fig 2.2: Structure of a protein. The 3-dimensional structure of a protein is represented in 3 visualization modes. Left visualization shows the backbone of the amino acidic chain, where the secondary structure is represented as helices for the alpha-helices and arrows for the beta-sheets. Middle visualization shows the volume of each atom, while the rightmost visualization shows the rendered atomic surface of the protein. Taken from [117]

first folding locally in micro-folding patterns that define the secondary structure of the proteins. The tertiary structure is the subsequent macro-folding that gives proteins a 3-dimensional configuration and is correlated with function [99]. The cuaternary structure refers to the aggregation of multiple folded protein molecules into a single functional unit, and will not be further addressed in this thesis [100].

The secondary structure is simple enough to be coded in 3 possible configurations, each of which presents in individual protein segments: alpha helices which are spiral like patterns, beta sheets which are paralelly aligned strands, and loops or coils which are segments with no folding pattern, where the amino acids have more freedom of movement [101, 99]. A single protein can have multiple segments with different secondary structures. The tertiary structure, on the other hand, is far more complex and virtually every type of protein has a different tertiary structure, where some common motifs called domains are observed [102].

The protein undergoes folding due to the physicochemical interactions between the amino acid and the medium or between themselves [103] (Fig. 2.3). Most amino acids are neutral and non-polar (A, V, I, L, M, F, Y, W, P and G) and will “repel” the water molecules of the medium and hence have a propensity to be in the inside of the protein. Other amino acids are neutral and polar (S, T, N, Q and C) and will form hydrogen bonds with water molecules and between themselves and are hence most abundant in the protein surface. Lastly, some amino acids are charged either positively (K, R and H) or negatively (D and E) and will either interact with water, ions or form strong ionic bonds between themselves conferring the protein with greater stability [106, 93].

The structure of a protein is mainly derived from its amino acidic sequence, as the same sequence will normally generate the same 3-dimensional structure [103, 99, 76]. However, the *ab-initio* sequence-to-structure translation is complex and has been unresolved until transformer-based deep learning methods arose [88, 87]. The most common way to bioinformatically determine the struc-

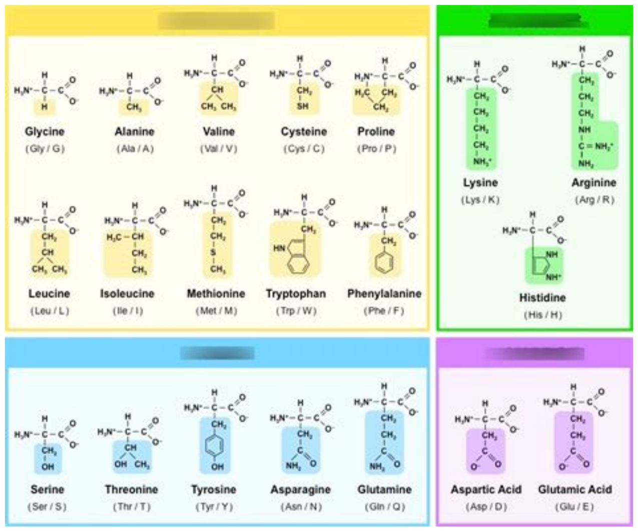

Fig 2.3: Amino acids compositions and classification. Taken from [118]

ture of a protein is to perform homology modelling, which involves adapting the structure of another protein that had its structure determined empirically and has high sequence similarity to the target protein [104]. This is possible because proteins, like DNA, undergo evolution by mutation to acquire novel functions [105]. Proteins that diverged relatively recently from a common ancestor will likely have a very similar 3-dimensional structure and function.

Protein function

Proteins play diverse roles in the cell's complex internal machinery. Proteins can be structural components, permit chemical reactions that define the cell's metabolism, process information, transport molecules across the cell membrane, control gene expression and DNA division, and a plethora of other functions which comprise almost the totality of cellular functions [107]. The biological functions of proteins are, like their structure, complexely coded into their amino acid sequence. Proteins with high similarity will likely have the same function and have similar structures. To this extent, the distribution of protein sequences is far from uniform, as for most proteins there are numerous other slightly or highly similar proteins, even in evolutionarily distant organisms, called homolog proteins [105]. This defines the concept of protein family, which is a cluster of similar (homolog) proteins with a similar or identic function. There are multiple coding systems for protein families, such as PFAM, COGs, PRK, TIGRFAMs, etc, which use different criteria for the definition of protein families [71, 72]. Determining the function of a protein from its sequence is called functional annotation and is usually achieved with sequence similarity methods or with methods derived from NLP such as HMM [1, 4, 71].

Protein properties

In addition to the protein structure and function, proteins have physical properties that define their behavior and stability. Mainly, the molecular weight of a protein is the sum of the mass of all the atoms present in the protein molecule.

Proteins can be charged. The charge of a protein is the sum of the individual charges of its amino acids. The charge of a protein affects how it interacts with other proteins and with DNA, its function, and its localization [13, 14]. However, the charge of a protein changes with pH, as negative amino acids become neutral at acidic pH and positive amino acids become neutral at basic pH.

Then, there is a pH at which a protein is uncharged, which is the isoelectric point of the protein [108].

Proteins can also have different water interacting properties. The proteins can be more hydrophilic (high water affinity) or hydrophobic (low water affinity), depending on their amino acid composition and disposition [109]. The hydrophobicity of a protein affects its location (hydrophobic proteins are more likely found embedded in the cell membrane), its stability, and its protein-protein interaction properties [110, 111, 112].

Proteins can have different stability properties when facing different physicochemical factors. Extreme conditions such as high or low temperature, high or low pH or high salinity normally cause loss of function or even the unfolding or denaturation of proteins [61]. However, some proteins can resist or even prefer extreme conditions [59, 60]. For instance, a protein can be thermostable if it can maintain its folding at high temperatures, or thermophilic if it functions more efficiently at high temperatures [63]. Low temperature usually causes the proteins to function poorly or even denaturate, but cold-resistant proteins can function properly at low temperatures [64]. On the other hand, pH extremes change the ionic proterties of amino acids which affects the charge of the proteins, possibly resulting in denaturation [65, 67]. Proteins can be acidophilic or alkaliphilic if they are able to resist and have improved function at low or high pH, respectively.

2.1.3 Acidophiles

Both multicellular and unicellular organisms need specific conditions for optimal growth. Usually, these conditions range around very common or *normal* levels: around 37°C, neutral pH, 1 atmosphere of pressure, etc. However, some organisms, particularly unicellular organisms, are adapted to extreme conditions to the point they need extreme conditions to grow optimally. Such is the case of extremophiles, which are organisms that withstand different types of extreme conditions [53]. Some extremophiles can resist high temperatures, ($>$ 100 °C), low temperatures, high salinity, high pressure, high pH (alcaline) or low pH (acidic).

This work focuses on the study of acidophiles, which are organisms that live at extremely low pH (< 3)[3, 54]. Acidophiles can be found in both natural and artificial acidic environments, such as marine hydrothermal vents, natural acidic pools like those found in the yellowstone national park (Fig. 2.4), and even acid mine drainages of copper mines. Acidophiles and can even be used in mining operations to recover metals in a process called bioleaching [113, 114]. At these extremely low pH values, the concentration of protons in the medium is so high that the non-resistant organisms will become essentially *cooked*, as their proteins are denaturated. There are several known acid resistence mechanisms characterized in highly studied organisms such as *Escherichia coli* that have also been found in acidophiles, and have been hypothesized to bestow them with resistance to acidic environments [55, 56, 5]. Amongst them, a membrane highly impermeable to protons, transporters that pump protons out of the cell, cytoplasmic buffering of protons and pH damage repair mechanisms have been proposed as acid resistance mechanisms for acidophiles. However, direct evidence of this is lacking. Therefore, it's unknown which genomic traits are determinant for the acidophilic phenotype.

Acidophilic proteins

One of the main reasons why extremely low pH is damaging to the cells is that it affects the proteins of an organism. As it was exposed in the previous sections, extreme conditions such as low pH can

Fig 2.4: *Morning glory* pool from the yellowstone national park, where multiple acidic hot springs are the home of thermoacidophilic microorganisms. Taken from [119]

produce the denaturation of proteins, that is, their unfolding and loss of function [65, 61]. The inside of the cell or internal environment of unicellular organisms (the citoplasm) is usually close to neutral pH (pH 7), even if that organism is an extreme acidophile [57]. On the other hand, the space between membranes called periplasm, which can be found in gram-negative bacteria (most bacteria) has low pH in acidophiles [58]. This implies that, in acidophiles, the proteins located in the periplasm, outer membrane and exported to the outside of the cell are exposed to extremely low pH. On the other hand, proteins embedded in the inner membrane of the cell are only partially exposed to low pH.

As these microorganisms are resistant to low pH, so should, in theory, all their exposed proteins be [67, 66]. The main challenge of acidophilic proteins is that below pH 3.7, the 2 negative amino acids glutamic acid (E) and aspartic acid (D) become protonated, as their proton dissociation constants (pKa) are 3.71 and 4.15, respectively [68]. This causes them to lose their negative charges, which in turn disrupts the protein's capacity of forming strong structure-stabilizing ionic bonds, effectively impairing the stability of the protein.

Several mechanisms have been proposed to confere acidophilic proteins with acid resistance. For instance, the periplasmic proteins of *Acidihiobacillus ferrooxidans* have a marked basic shift, as over 70% of the proteins have high (basic) isoelectric points and therefore are positively charged at neutral pH [6]. A similar remark has been made on other acidophilic proteins and has been proposed as an acid adaptation mechanism [8, 70]. However, a lower (acid) isoelectric point (the opposite adaptation) has also been suggested as an acid resistance adaptation in acidophilic proteins [66, 69]. Therefore, the traits that allow a protein to resist extremely low pH are yet still unknown.

2.2 Machine learning algorithms

In this thesis, multiple machine learning algorithms were used to predict the pH associated with proteins in base of their sequence and several features. As the target variable is a continuous variable (pH), the models described in this section are regressors.

2.2.1 Non-deep learning algorithms

This section refers to the machine learning algorithms used in this thesis that are not deep learning, which will be subsequently referred to as classical machine learning or conventional machine learning [15]. Classical machine learning algorithms are usually suitable for tabular data with thousands of data instances, while deep learning is more suitable for complex unstructured data with millions of data instances.

Ridge regression

Ridge regression is a variation of the ordinary least squares regression (OLS) where an L2 cost is added to the model's parameters to deal with the multicollinearity problem and reduce over-fitting [17, 18]. In ridge regression, the loss function is given by

$$J_p = ||Y - \tilde{X}\theta||_2^2 + \rho||\theta||_2^2 \tag{2.1}$$

Where Y is the target variable, θ are the model parameters and ρ is the regularization hyperparameter that controlls the strength of the L2 regularization. A higher value of ρ imposes a higher quadratic cost on the parameters which can yield more regularized models. The value of the parameters θ that minimizes the loss function J_p can then be directly obtained with the following expression:

$$\theta = \left(\tilde{X}^\intercal\tilde{X} + \rho\mathbb{I}\right)^{-1}\tilde{X}^\intercal Y \tag{2.2}$$

Random forest regression

The random forest regression method is a type of bagging machine learning algorithm composed of multiple decision tree regressors [19]. A single tree is a machine learning method that performs recursive splitting of the data in consecutive nodes to achieve accurate predictions. In each node, the feature that best discriminates the target value is selected, and a threshold (if the feature is numeric) or category (if the feature is categoric) is defined to split the data in 2 sets. For each set, a single prediction of the target variable is performed using the average of the target varable. Each set can then be consecutively splitted for higher accuracy [20]. This process can be perfomed indifenitely until a perfect prediction is achieved in the training data. However, this will usually result in over-fitted trees, as a tree with a high number of nodes will eventually capture the specific details of the training data (data memorization).

Random forests circumvent this issue by training many decision trees [21]. Each individual tree of a random forest is different to a standalone decision tree model, as the random forest trees are usually weak simple predictors that are trained on a subset of the data and for which only a subset of the features is searched for the best split in each node. This results in more regularized models, where each tree is impaired and biased, but the expected tree performs an unbiased prediction. The prediction of the forest is then the avereage of all the trees predictions (Fig. 2.5). This method captures the underlying distribution of the data and target variable better than individual trees (even pruned), and produces better predictions on out-of-sample data (test data).

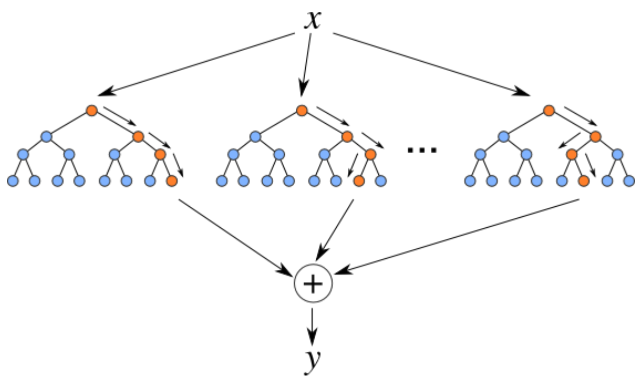

Fig 2.5: Schematics of the random forest regressor algorithm. Taken from [121]

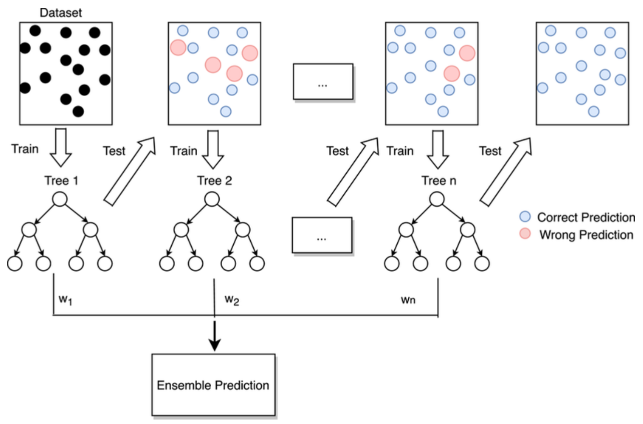

Fig 2.6: Diagram of the boosting machine learning algorithm. Adapted from [116]

Boosting

The bagging models such as random forest are a reliable alternative to reduce overfitting, as many impaired models will collectively capture the general distribution of the data and the rules to classify most data points. For bagging models, this is done by randomly selecting data points in the training of each sub-model. Instead, boosting models [34] accomplish this in a more guided process, where models are specifically designed to be biased towards usually misclassified data points.

Boosting models, just as bagging models, are composed of multiple ”weak” predictors. Usually, these are 1 or 2 node decision trees, for which the prediction is averaged or weighted in the final prediction step. However, this is done by giving additional weight to points with higher error rates, thus forcing the models to be impaired in an iterative way. One of the earliest and most pivotal boosting models, AdaBoost [35], applies the following algorithm: First, a base model is trained in a standard way, where all data points have the same weight. Next, weights are recomputed by giving misclassified points (or otherwise points with a higher error rate for regression models) additional weight. Then, consecutive models are trained using these modiffied sample weights, which lowers the error rates on data points that previously had higher error rates. Then, the weights are again re-computed and the process is repeated a set number of iterations. For the final prediction, the individual predictions are linearly combined by giving additional weight to models with a better overall performance (Fig. 2.6). This process forces models to be impaired and biased towards a subset of points that deviate from the general data distribution, or simply put, points that are *hard to classify*. On the other hand, bagging models expect this to happen from random subsampling.

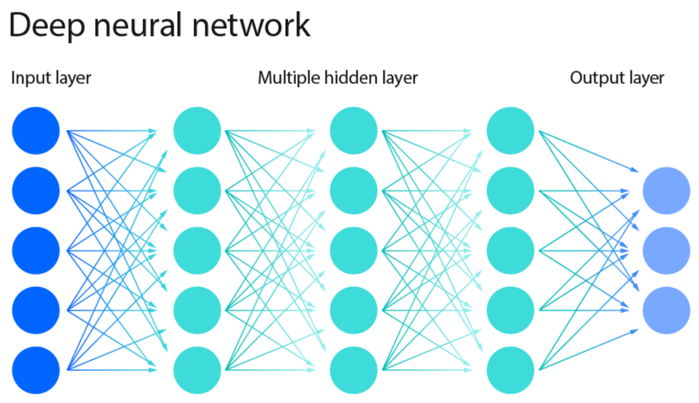

Fig 2.7: Schematics of the architecture of a deep learning model. Adapted from [122]

A subset of boosting models are called the gradient boosting model [34]. For these models, instead of training on a re-weighted dataset, consecutive weak predictors are trained on the residuals of the prediction of the previous iterations, such as when the predictions are added they should converge to the target values. This is done iteratively using a learning rate such that the expected value of the additve model does not immediately converge to the target values, but does after several iterations. A variant of this type of boosting models is LightGBM [16], a boosting algorithm optimized for large datasets that estimates the information gain more efficiently by using a small sample of the data for better performance, producing nearly identical results as greedy boosting algorithms nut with a training process that is over 20 times faster.

2.2.2 Deep learning

Deep learning algorithms are a popular type of machine learning algorithms which are based on neural networks [36]. The principle behind these type of models is that key features for the prediction are calculated automatically as combinations of the different dimensions (or features) of the original data or of other automatically learned features, instead of manually as it is in classical machine learning. The most basic deep learning architecture is the multi-layer perceptron (Fig. 2.7) (MLP). In the MLP, all the original dimensions of the input data are linearly combined with weights present in the model's parameters to obtain a single feature of the first hidden layer, which is called a neuron. This is done multiple times to obtain all the different neurons of the first hidden layer. This is called a fully-connected or linear layer. Fully-connected layers are then consecutively applied to obtain the neurons of the second hidden layer by combining the neurons of the first hidden layer and so on for the next layers. The last layer, the output layer, contains the prediction of the model. In each step, a non-linear function (the activation function) is applied to the layer output, which allows capturing non-linear relations and permits information processing.

Deep learning models are optimal for complex data structures such as images, audio and language. Specific, optimized deep-learning architectures exist for each type of problem and data structures, which involves a more directed information processing rather than comparing all data dimensions between themselves as it is in the multi-layer perceptron.

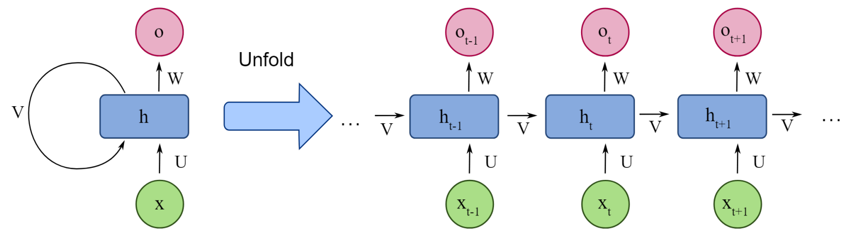

Fig 2.8: Architecture of a typical RNN. x_t, h_t and o_t represent the embedding of the input, the hidden state, and the output at state t. Taken from [123]

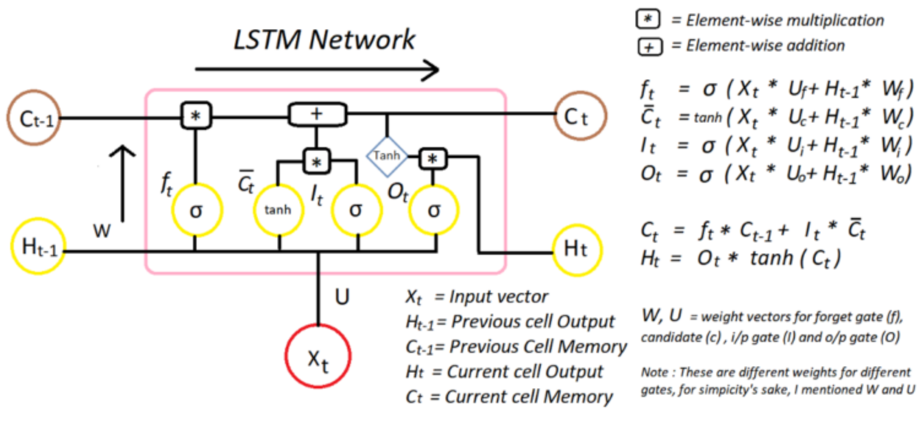
LSTM Network
C_{t-1}
H_{t-1}
W
U
X_t
f_t
$\bar{C}_t$
I_t
O_t
σ
tanh
Tanh
C_t
H_t
∗ = Element-wise multiplication
+ = Element-wise addition
$f_t = \sigma(X_t * U_f + H_{t-1} * W_f)$
$\bar{C}_t = \tanh(X_t * U_c + H_{t-1} * W_c)$
$I_t = \sigma(X_t * U_i + H_{t-1} * W_i)$
$O_t = \sigma(X_t * U_o + H_{t-1} * W_o)$
$C_t = f_t * C_{t-1} + I_t * \bar{C}_t$
$H_t = O_t * \tanh(C_t)$
X_t = Input vector
H_{t-1} = Previous cell Output
C_{t-1} = Previous Cell Memory
H_t = Current cell Output
C_t = Current cell Memory
W, U = weight vectors for forget gate (f), candidate (c), i/p gate (I) and o/p gate (O)
Note : These are different weights for different gates, for simpicity's sake, I mentioned W and U

Fig 2.9: LSTM architecture. Taken from [124]

Recurrent neural networks

Recurrent neural networks (RNNs) are a type of neural network specialized for recursive data structures such as language and time series data [37]. In the RNN, the input x is a series of tokens with varying length. For each token x_i, a different embedding is learned, which is a semantic representation of the information contained in each token. The embeddings are then subsequently passed to the RNN, which contains a hidden state vector h_i. In each step, the embedding vector of the token x_i is concatenated to the hidden state vector h_i, and a fully-connected layer is applied to the concatenated vector, producing the next hidden state h_{i+1} as output by matrix multiplication with the network's parameter matrix (Fig. 2.8). Additionally, a token-wise output o_i can also be produced in each step when necessary. The hidden state h_i of the RNN represents the contextual information of the text up to the i-th token, and in the final hidden state h_n it represents the information of the whole document. The final hidden state can be then concatenated to fully connected layers, similarly to an MLP architecture, to then perform document classification or regression.

The regular RNNs suffer from the vanishing gradients problem [38]. That is, after several recursive iterations, the gradient derived from the first tokens vanishes. This causes the the final state of the hidden layer and therefore the output to mostly depend on the last tokens. The LSTM is a subtype of RNN that has been designed to address the vanishing gradient problems of conventional RNNs [39]. Besides of containing the RNNs' hidden state, the LSTMs contain memory blocks which are connected to the cell's hidden state through the forget gate (Fig. 2.9). This allows for a part of the information to be retained across multiple iterations of the RNN.

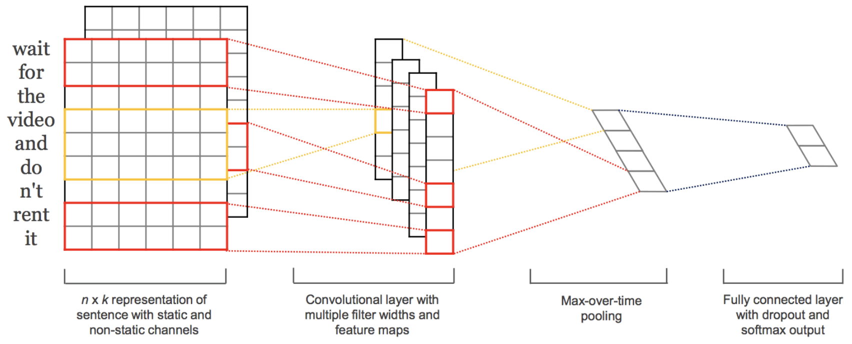
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Fig 2.10: Architecture of 1-dimensional CNNs for text classification. Taken from [125]

Convolutional neural networks

Convolutional neural networks (CNNs) are a sub-type of neural network where the data or features located next to each other are compared using a set-sized kernel that performs the convolution operation [41]. The CNNs were initially designed for image classification, where a 2-dimensional kernel compares the nearby pixels to obtain an activation map of the image which is max-pooled and then passed through consecutive CNNs to obtain a final representation vector of the image (Fig. 2.10). In text document classification and named entity recognition problems [40], this is done with a 1-dimensional convolution, where the embeddings of the nearby tokens are concatenated and processed with linear layers that output a set of channels for each position of the document. This is done by performing matrix multiplication between the layer's weights and a vector of the relative-position-specific concatenated embeddings (Fig. 2.10). This process is repeated by centering the kernel on every token, each time outputing c channels.

More specifically, for an input x composed of n tokens with embedding size k (shape of x is $n \times k$), and a kernel size s, a single convolution operation on the i-th word performs matrix multiplication between the vector of size $k \cdot s$ composed by the concatenation of $[x_i, x_{i+1}, ..., x_{i+s}]$ by the layer weights $W^{(k \cdot s) \times c}$ which outputs the channels C_i^c. When applied to $i = [1, 2, ..., n]$ (if padding is used), the overall output has the shape $n \times c$. Finally, a max-pooling is applied across the *token dimension*, producing an outupt vector of shape c which can be processed with linnear layers to produce a single output per-document. The spirit behind using CNNs for text classification is to be able to automatically learn the relevant n-grams for the task, where the max learnable n-gram is the kernel size s.

Attention

Attention is currently one of the most important concepts in the deep learning field [42]. It is inspired by the biological systems of humans that tend to focus on the distinctive parts when processing large amounts of information.

The general mechanism of attention layers or attention-based models is summarized in Fig. 2.11. The inputs for the attention layer are series of embeddings of shape $n \times k$, similar to the input of an RNN or any text-based deep-learning model. The embedding size k must be pre-defined but the number of embeddings n (or document length) can be of variable length. The attention layer takes two $n \times k$ shaped inputs: The keys and the values. In simple words, the keys determine the weights for the linnear combination of the values. In a single attention operation, a f function takes every

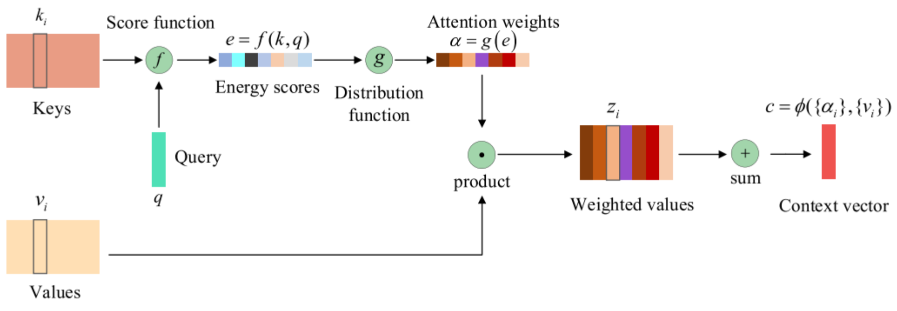

Fig 2.11: General attention mechanism. Addapted from [42]

key and a query vector (usually learnable) to extract the weights to linearly combine the values. One of the most common attention mechanisms used is dot-product attention, where f is the dot product of the key and query. The keys and values are defined separately, but can be the same vectors.

The attention mechanism allows for capturing directed contextual information from the embeddings. In machine translation, attention between the current hidden state and the hidden states of the encoding has been used to extract additional contextual infromation for the translation of each token than can be compressed in the encoder output [43]. Attention mechanisms have also been used for text classification and sentiment analysis as a means to construct a more efficient document representation [44]. Attention mechanisms can even be applied to computer vision, where a spatial atention models learns the part of the image that the network should focus on [45].

Transformers

Transformers are one of the most successful deep learning architectures and the current state-of-the art in generative models for both text and images. They were first proposed on the topic of NLP in 2017 [46]. The transformer was the first model based solely on self-attention mechanisms, where the keys, values and even queries are all derived from the input embeddings using simple linear layers. In a single multi-head attention layer, the keys of all input tokens are compared with the queries of all input tokens to obtain the weights to combine all the values separately for each token by scaled dot-product attention (Fig. 2.12). The positions of every token in the sequence are encoded and concatenated to their vectorial representations (embeddings) with positional encoding. Otherwise this architecture would be position insensitive, as the positional relation of tokens from either CNNs or RNNs is absent. This multi-head attention layer can be performed multiple times to extract different features, and even with multiple layers of depth, making the transformer architecture one of the largest to date.

The transformer architecture has achieved great success in many artificial intelligence fields, such as natural language processing, computer vision, and audio processing [47, 51]. Natural language processing was the starting point of transformers and continues to be one of its main applications. The Bidirectional Encoder Representation from Transformers (BERT) is one of the most populars transformer-based NLP tools, suitable for tasks like document-wise or token-wise classification, and has between 110 million (base) and 340 million (large) parameters [50, 49]. The recent explosion in deep generative transformer-based models like ChatGPT [48] and the use of difussion models for image generation [52] have achieved the level of performing human-like data generation.

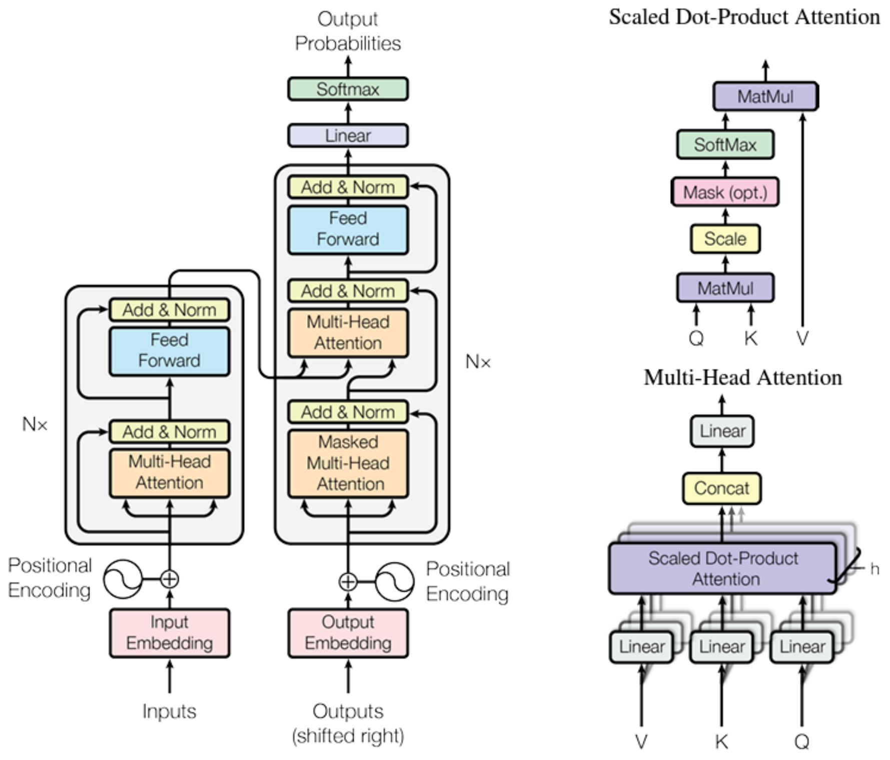

Fig 2.12: Transformer architecture. Adapted from [46]

2.3 Machine learning on proteins

There is currently a large number of softwares and algorithms for protein analysis based only on its amino acid sequence. As previously stated in earlier sections of this thesis, most protein analysis softwares use sequence-similarity-based or rule-based algorithms [1, 72, 2].

2.3.1 Classical machine learning on proteins

Several classical machine learning and NLP algorithms have been widely used for protein characterization. Hidden markov models have been applied to both classification of proteins and extraction of features from the sequence of a protein [4, 73, 74]. Machine learning has been used to predict both protein function and structure [76, 77]. SVM has been used to predict protein isoelectric point [78]. Feature extraction coupled with machine learning classification techniques has been used to classify thermophilic proteins and several types of extremophilic proteins [62, 75], not including acidophilic proteins.

Machine learning has also been used to classify the cytoplasmic proteins of acidophiles from their non-acidophilic homologs [81]. However, as only 6 organisms were used, it is possible the models captured specific traits of those organisms' proteins. Additionally, cytoplasmic proteins are not exposed to low pH, so they are not acidophilic proteins.

2.3.2 Deep learning on proteins

In the past decade, there has been a boom in deep learning techniques applied to protein characterization. Several NLP DL architectures have been applied to protein sequence analysis. CNN based models have been used for tasks as simple as isoelectric point estimation [82]. Deep learning models have been used to predict protein interfaces of interaction using graph convolutional networks [83]. Attention-based CNN-LSTM networks have been developed for compound based interactions prediction [84]. LSTM based models and even CNN-LSTM based models have been used to predict the secondary structure of proteins [25, 86]. LSTM based models have also even been used to predict thermophilic traits of the proteins [85].

Proteins have also been the target of transformer-based large models. Most notoriously, the AlphaFold model permits the accurate prediction of a protein's 3-dimensional structure using only its sequence [87]. This problem is called ab-initio structural prediction, which has been historically considered an unsolevd problem before AlphaFold [88]. Different types of complex transformer-based protein encoders resembling the BERT language model have also been developed, such as the ProtBERT model [26, 89]. Even generative transformer-based models have been developed. This is the case of ProtGPT2 [9], a model trained on almost 50 million proteins that the authors claim can sample from unseen regions of the protein space (but that are valid proteins according to several criteria).

2.4 Methodological approach

The ulterior motivation of this thesis is to be able to develop a model or algorithm that predicts the optimal growth pH of an organism using only its genomic sequence as input. This includes using all features that can be obtained with high accuracy from the genome, such as the proteome and subsequent protein annotations.

This could be accomplished with machine learning by using genomic traits such as the presence of certain genes, genome length, GC content (nitrogenous bases frequency), codon usage, genome packaging indicators, or even deep learning language models to classify the genomes. However, genomes are high dimensionality data, as a single genome is usually well over 10^6 nitrogenous bases. Even if a smart feature preprocessing is performed to reduce over-fitting, there are only hundreds of acidophilic species known.

While these issues could be approached with feature engineering, there is still a major problem: there is no guarantee that novel groups of acidophiles to be discovered in the future will share the characteristics of the current acidophiles. Even by performing an efficient model regularization, we can only speculate the features we are using to distinguish acidophiles from non acidophiles are universal acidophilic features.

However, we can posit that all the exposed proteins of an acidophiles should be able to resist low pH. If this genome classification problem is then transformed into a protein classification problem, all the issues here exposed are taken into account. The data dimensionality would drastically decrease, as proteins are usually about only 50-500 amino acids long. Secondly, the number of instances (data points) would increase several times, as the organisms have about 100-300 exposed proteins. Also, as acid resistance is a physical property of proteins, it should be a less complex trait encoded in the protein sequence than other complex traits like structure or function. Then, as proteins are much simpler than DNA, this acid resistance property should be learnable, provided a large enough protein dataset. From an evolutionary perspective, it should be more likely that proteins of novel acidophilic groups have the same acid resistance mechanisms in their exposed proteins than for them to have the same genomic traits (acid resistance mechanism genes).

Independently of the organisms' predictions, obtaining a model for protein classification or regression that predicts its acid resistance capabilities is interesting by itself, as it can be used as a bioinformatic analysis tool.

Consecuently, machine learning models wil be trained to capture as best as possible the acid resistance capacities of proteins, or more specifically, the optimal pH of the protein's source organism, by exploring classical machine learning algorithms and multiple DL architectures. After obtaining the best possible model, organisms' predictions will be performed by exploring the average, the median and the mode of the individual predictions of proteins. By doing this a sort of *bagging* approach is used, where while individual predictions can be erratic and have high variance, the average or median should converge to the actual pH of the organism and produce general predictions, even if the number of organisms used for training is low.

Chapter 3

Methods

3.1 Dataset

3.1.1 Proteomes and bioinformatic features

The genomes and proteomes of acidophiles were obtained from the AciDB database, which contains data and metadata of acidophilic bacteria and archaea [54]. The proteomes of taxonomically related organisms with optimal growths up to around pH 7 were obtained with the methods present in [7]. Also with the methods from this work, protein features such as molecular mass, isoelectric point, transmembrane domains, functional annotation and subcellular localization were obtained using classical bioinformatic tools.

To improve the quality of the data, only the proteins from genomes that passed a CheckM [24] genome completeness of $> 80\%$ and contamination of $< 5\%$ were considered. Additionally, only laboratory grown microorganisms were used. No metagenomic reconstructions were used in this study.

Each organism in this study was associated with a pH, which will be referred to as associated pH. The peer-reviewed laboratory-tested optimal pH of the microorganisms was considered if available. Otherwise, the pH of the medium used to grow the organisms was used.

3.1.2 Protein selection

In this thesis, only proteins theoretically exposed or partially exposed to the environmental pH were considered for the analyses. Proteins were considered exposed to the environmental pH if they followed one of the next conditions:

- PSORTb [22] predicted that the protein was either exported outside the cell, or had a subcellular localization in the periplasmic space or the outer membrane (for gram-negatives) or the cell wall (for gram-positives and archaea).

- PSORTb was not able to predict the protein's subcellular localization (Tag Unknown) and the presence of a signal peptide for protein translocation was predicted with SignalP [23].

In addition, only proteins with 500 or fewer amino acids were selected for training, as the presence of large proteins would drastically reduce the efficiency of the training of deep learning models, even if they are only a few of them.

3.1.3 Other features

The predicted transmembrane segments of each protein were calculated by averaging the prediction of TMHMM and HMMTOP [27, 28]. Additionally, the length of the proteins and the 1-2 gram amino acidic frequencies were obtained manually.

3.2 Deep feature extraction

3.2.1 Secondary structure prediction

Secondary structure prediction was performed with the pytorch based deep learning tool S4PRED [25]. The software was modified to be able to perform batch size predictions by introducing padding. As the original model (a bagging model of 5 LSTMs) did not include a padding index, the pytorch pack_padded_sequences tool was used. The predictions of the modified model were confirmed to be identical to those of the original model using a random sample of 1000 proteins.

3.2.2 Protein autoencoding

The pretrained protein autoencoder transformer type model ProtBERT model [26] was used to encode the protein sequences into vectors that capture the necessary information to reconstruct the amino acidic sequence at high accuracy. The first token of each output of the ProtBert model (containing the whole protein representation) was extracted and saved as numpy vectors (size 1024).

3.3 Data preparation

3.3.1 Datasets split

Each protein and its features were considered a separate datapoint, where the organism's associated pH was used as a target variable.

The proteins were splitted into train and test datasets. For the classical machine learning models, an 85% of the data was placed in the training set, while the remaining 15% was placed in the test set, resulting in a final size of 129975 proteins for the training set and 22937 proteins for the test set. For the deep learning models, 120000 proteins were used for the training set (a 78%), 8000 proteins were used for the validation set (a 5%) and the remaining 24912 proteins (16%) were used for the test set. In all cases, the split was done with sklearn's train_test_split tool using a random state of 1991.

3.3.2 Feature preprocessing

The following numerical features were extracted and preprocessed by standard scaling:

- The amino acidic composition was extracted from each sequence as a vector of unigram and bigrams of amino acids. This means that the frequency of each of the 20 amino acids, along with the frequency of each of the 400 possible amino acid combinations was calculated for each protein, resulting in a total vector of 420 features by protein.

- The secondary structure composition of alpha helix, beta sheets and loops was calculated for each protein, resulting in vector of 3 features.

- The full protBERT encoding vector was used as a feature, resulting in 1024 additional features

- Additional features such as molecular weight, isoelectric point, number of transmembrane segments and protein length were also extracted, resulting in a vector of 4 features.

Additionally, the following categorical features were extracted and encoded:

- The subcellular localization was extracted and encoded with the one hot encoding method. As there were 5 possible localizations in these proteins, this resulted in a vector of 5 features.

- The presence and type of signal peptide for protein export was extracted and encoded with the one hot encoding method. As there were 4 possible localizations in these proteins, this resulted in a vector of 4 features.

- The COG category of the protein was extracted and encoded with the multilabel binarizing method, resulting in an binary vector of 24 features where more than one feature can be non-zero.

The previous transformations and encodings were trained in the training set and then applied to both the training, test and validation (for DL models) sets. All features were concatenated, resulting in a final feature dimension of 1484. For the deep learning models, the amino acidic sequence along with the sequence of secondary structure predictions for each amino acid were encoded to integer sequences and passed to embedding tables.

3.4 Exploratory analyses

3.4.1 Gaussian smoothing

A procedure of smoothing was performed for the expoloratory analyses, the visualization of regression results and even the calculation of regression metrics. Consider the random variable $x \in \mathbb{R}^1$ that follows an unknown distribution and $X = [x_1, ..., x_n]$ a simple random sample of x. A regular kernel density estimate (KDE) of the likelyhood of x at a given value x^* by using the sample X follows the following formula.

$$KDE(x^*, X, \sigma) = \sum_{x_i \in X} \exp\left(-\frac{(x_i - x^*)^2}{2\sigma^2}\right) \tag{3.1}$$

Which is proportional to the estimated probability at x^* and can be converted to probability by normalizing by $\int_{\mathbb{R}} KDE(x)dx$.

Similarly, the gaussian smoothing of a variable produces local averages of such variable at specific values of another variable. Consider the variable $y \in \mathbb{R}$ with $Y = [y_1, .., y_n]$ to be smoothened. The Gaussian smoothing (GS) of Y in relation to X at a given point x^* is the weighted sum of Y using the normalized weights of X obtained from the KDE. The weight W of a single value is given by:

$$W(x^*, x, \sigma) = \exp\left(-\frac{(x - x^*)^2}{2\sigma^2}\right) \tag{3.2}$$

Then, the gaussian smoothing of Y at x^* is:

$$GS(x^*, X, Y, \sigma) = \frac{\sum_{i \in [1,n]} y_i \cdot W(x^*, x_i, \sigma)}{\sum_{i \in [1,n]} W(x^*, x_i, \sigma)} \tag{3.3}$$

For both the kernel density estimate and the gaussian smoothing there is a hyperparameter σ which is the smoothing parameter. A higher σ produces more smooth but undetailed estimation, while a lower σ produces a more sharp but noisy estimation.

3.4.2 Feature exploration

For the protein features, univariate and bivariate analyses were conducted to determine their distribution in an unsupervised manner and as a function of pH. For Amino acid composition, secondary structure composition, molecular weight, isoelectric point and the number of transmembrane segments their distributions were plotted with histograms and density plots and their means were compared across optimal growth pH of proteins' source organisms with gaussian smoothing. The former features were also explored bivariately with correlation matrices.

For the autoencoder vectorial representations, density estimates were performed to explore the distribution of each dimension. For the former features and for the amino acidic frequencies, dimensionality reduction was performed with PCA using sklearn's decomposition.PCA module to explore the general distribution of the features and analyze possible tendencies in relation to pH.

3.5 Machine learning

In this thesis, both classical machine learning models and deep learning models were training to predict the associated pH of the source organism of each protein. Each model was trained in a weighted and unweighted. The aim of the weighted version was to reduce the bias of the model towards overrepresented pH ranges.

3.5.1 Sample weights

For the weighted version of the models, the sample weights were calculated with the following procedure. Consider the vector of target values (pH) $Y = [y_1, ..., y_n]$ and the unnormalized kernel density shown in Eq. 3.1. Then, the initial weight W_0 of $y \in Y$ is given by:

$$W_0(y, Y, \sigma, \alpha) = \frac{1}{KDE(y, Y, \sigma) + \alpha \cdot |Y|} \tag{3.4}$$

Where α is the smoothing parameter which gives all points a base density of α times the number of points. Considering the vector of initial weights $W_0(Y) = [W_0(y_1), ..., W_0(y_n)]$, the final weights W are given by:

$$W(y, Y, \sigma, \alpha, q) = \min\left(\frac{W_0(y, Y, \sigma, \alpha)}{Q(W_0(Y), q)}, 1\right) \tag{3.5}$$

Where Q is the quantile function that returns the value that a q fraction of the data is lower to. This gives a weight of 1 to the $1-q$ fraction of the highest values of Y and a weight in proportion to such quantile to the rest of the data. This dampens the weight of rare pH values that would otherwise have extremely high weights and bias the models.

For all the machine learning models, a $\sigma = 0.2$ (a 3% of the pH range) was used for the KDE of the pH, and a smoothing factor of $\alpha = 0.01$ and a quantile of $q = 0.99$ were used for the weight calculation.

3.5.2 Metrics

Models were trained and evaluated using regression metrics. The main metric used to evaluate the models was the mean absolute weight (MAE). Additionally, models were evaluated using mean squared error (MSE) and R^2. The scikit-learn's metrics submodules mean_absolute_error, mean_squared_error and r2_score were used, respectively.

Weighted regression metrics

A weighted version of these metrics was implemented to evaluate how well the models perform independently of the data imbalance. By doing this, models that outperform other models at all pH values were selected instead of models that perform better at overrepresented pH values. This was done by averaging the mean local errors at each pH with the gaussian smoothing method of Eq. 3.3. For a given metric m of the prediction $Y^* = [y_1^*, ..., y_n^*]$ over the real target values $Y = [y_1, ..., y_n]$, consider the metric vector function $M(Y, Y*)$ so that:

$$M(Y, Y^*)_i = m(y_i, y*_i) \quad ; \quad M(Y, Y^*) = [m(y_i, y_i^*), ..., m(y_n, y_n^*)] \tag{3.6}$$

Then, the weighted version of the metric WM is given by:

$$WM(Y, Y^*, M, \sigma) = \frac{1}{max(Y) - min(Y)} \int_{\min(Y)}^{\max(Y)} GS(\hat{y}, Y, M(Y, Y^*), \sigma) d\hat{y} \tag{3.7}$$

The integral was calculated numerically by using 101 intervals between the minimum and maximum values of the target variable Y. Consider $\hat{Y} = min(Y) + (max(Y) - min(Y)) \cdot [0, 0.01..., 0.99, 1]$. Then the numeric version of the weighted metric $WM*$ is defined by:

$$WM^*(Y, Y^*, M, \sigma) = \frac{1}{101} \sum_{\hat{y} \in \hat{Y}} GS(\hat{y}, Y, M(Y, Y*), \sigma) \tag{3.8}$$

A σ of 0.15 (a 2% of the pH range) was used. This defined the metrics WMAE (weighted mean absolute error) for $m(y, y^*) = |y - y^*|$ and WMSE (weighted mean squared error) for $m(y, y^*) = (y - y^*)^2$. As for the R^2 metric, the weighted version WR^2 was calculated using the r2_score module of sklearn with the sample weights of Eq. 3.5 but using $\sigma = 0.15$, $\alpha = 0$, and $q = 1$.

Regression-to-classification metrics

Classification metrics were also used as an additional approach to manage the unbalanced representation of pH ranges in the dataset. The f1-score macro average is widely used as a metric to evaluate classification models on unbalanced classes [30]. The pH was segmented in 3 ranges: pH 0 – 3.8, pH 3.8 – 6 and pH 6 – 8.

The pH 3.8 threshold was selected because it is about the average between the glutamic acid and aspartic acid proton dissociation constants [68]. Below this pH, all amino acids are protonated and the proteins contain no negative charges, which probably results in entirely different folding mechanisms. On the other hand, the pH 6 threshold was selected arbitrarily by defining a window of ± 1 pH units around neutral pH 7.

3.5.3 Classical machine learning models

The pH of the proteins' source organism was predicted using a linear model, a bagging model, and a boosting model. All models were trained using the machine learning library Scikit-learn [29]. The models were trained on all the 1484 tabular features, including bioinformatic features, 1-2 gram amino acidic frequency, secondary structure frequency and ProtBERT features. For the weighted version, the sample weights were included as a parameter in the training function for a weighted calculation of the loss function.

Ridge regression

For the linear models, the Ridge regression model was used with scikit-learn's Ridge module. Alpha values were explored independently for the weighted and unweighted version of the model. The best alpha hyperparameter was selected considering the best MAE in test data for the unweighted version and best WMAE for the weighted version. Ridge models were then trained on the training data using squared error loss or weighted squared error loss, as it is default by linear regression models.

Random forest

For the bagging models, the random forest model was used with the scikit-learn's RandomForestRegressor module. A total of 200 trees were used in the forests. The hyperparameters max_samples and max_features were shortly explored manually, selecting the best combinations in test data MAE. The models were trained on training data using a squared error loss instead of an absolute error loss, as the absolute error loss was found to be extremely computationally inefficient. All the trees were trained using 8 parallel jobs.

Gradient boosting

For the gradient boosting models, the sklearn's module called HistGradientBoostingRegressor was used, which is an alternate implementation of lightGBM [16] recommended for high volume data. The models were trained using the absolute error as loss function. Besides that, all parameters were set to default.

3.5.4 Deep learning models

All deep learning models were trained using the pytorch framework [31]. Datasets, dataloaders, training functions, and deep learning architecture classes were developed and coded manually for each specific use. For the deep learning models, both the amino acidic sequence and secondary structure sequence are used as input in the form of integer tensors which are then passed to embedding layers. All models were trained using the Adam optimizer [32] without regularization. Unless specified, models were trained using a L1 loss function (absolute error). Models were trained until overfitting or until validation metrics convergence. Dropout probabilities and learning rates were shortly explored in each architecture and selected by best validation metrics. The model with the best validation metrics was saved and used to perform predictions in the test data.

The development of deep learning models in this thesis was an iterative process, on which over 25 architectures were tested. For several architectures, different settings than those reported in this methods section were used, including using squared error loss, different dataset split methods, using fewer or no features, and even classification models. The architectures here reported are those best performing in test data.

Undersampling

For the weighted versions of the models, a different approach was used to that of the classical machine learning models. As the deep learning models are much more computationally intensive than classical machine learning models, using sample weights for the loss' calculation would be highly computationally inefficient, as in each batch most of the data points would have a very small contribution to the loss function. Instead, an undersampling method was used. Instead of performing stratified sampling once, a custom data loader was designed that contains all the data and performs undersampling in each epoch. The data loader selects each data point with probabilities given by the sample weights of Eq. 3.5, where points with rare pH values have a probability of being selected of 1, while points with common pH values have a low probability of being selected. By doing this, the model is eventually trained on all the data, but the most common pH values are used far less frequently. This type of undersampling resulted in epochs of varying size (expected epoch size of about 27500 data points) and a higher training variance.

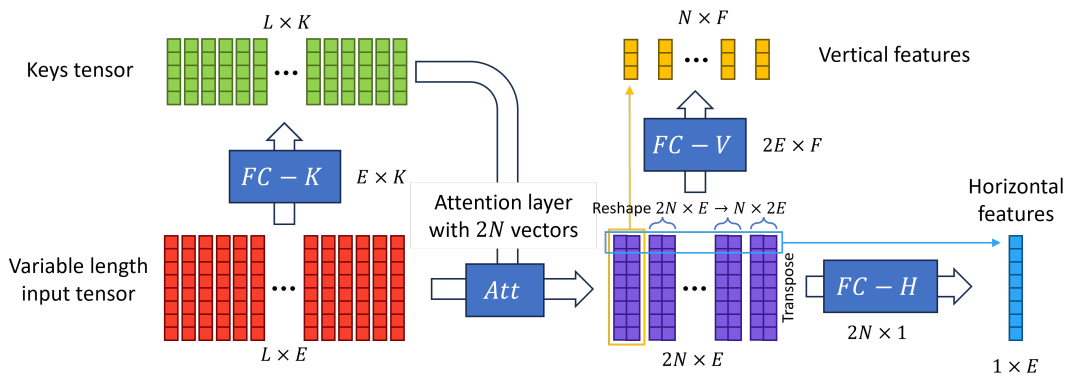

Fig 3.1: Architecture of the proposed attention feature extraction layer. The tensors are displayed in a transposed manner, where the first dimension is shown horizontally and the last dimension (embeddings) is shown vertically, to emulate the language representation of embedding tensors.

Self-attention

For some models, dot product self-attention layers were used for feature extraction. This implementation was adapted from Christos Baziotis' implementation of self-attention in pytorch [33] which was modified to work with multiple attention vectors simultaneously. The self-attention layer takes a tensor of shape $B \times L \times E$, where B is the batch size, L is the sequence length and E is the embedding size, and outputs a tensor of size $B \times N \times E$ by performing dot product attention with N trainable vectors. The batch size B and the sequence length L of the input can be of varying length, but the embedding size E has a fixed length. This condition is met by varying size inputs such as the output of an embedding table or the output of a RNN like a layer. The self-attention layer performs self-attention between all the E sized vectors of a data point and each of the N trainable attention vectors of the layer. In each attention process, the vectors of a data point are linearly combined with weights proportional to the dot product between the vectors and the trainable attention vector. The final output has a fixed size, thus permitting connection to fully connected layers.

Attention features

A novel architecture was developed that permits the use of multiple attention vectors for feature extraction without excessive scaling of the attention layer output (Fig. 3.1). The input tensor of size $L \times E$ first is passed through a fully connected layer to obtain the keys of size $L \times K$. Then, both tensors are passed through an attention layer with $2N$ attention vectors of size K that performs dot product attention by linearly combining the input tensor with weights proportional to the dot products with the key vectors. The output of the attention layer is size $2N \times E$, which scales linearly with the number of attention vectors used.

To reduce the output dimension, 2 types of feature extraction procedures are performed. First, the attention output is reshaped so the outputs of every 2 self-attention operations are paired, resulting in a tensor of size $N \times 2E$ which is passed through a linear layer that outputs a tensor of size $N \times F$, where F is the feature dimension which can be as small as desired. We will call these features *vertical features*. By doing this, the information is compressed so every vertical feature captures the information extracted by 2 different attention vectors. Secondly, one dimension of every attention output is combined in a linear layer with output 1. As this is done to one dimension of each attention output, the final dimension of this process is a single vector of the same size as the attention output vectors, which are the same size as the input embedding dimension E. We

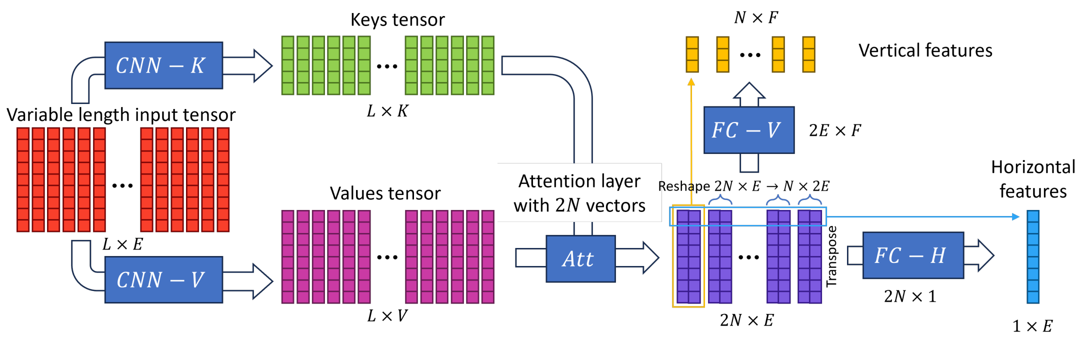

Fig 3.2: Architecture of the proposed CNN attention feature extraction layer. Analog to Fig. 3.1

will call these *horizontal features*. Finally, the vertical and horizontal features are flattened and concatenated, resulting in an output of size $N \cdot F + E$.

The dimensions displayed in this section are for an unbatched example. It is also valid for a batched input if the batch size dimension B is just be added as the first dimension of all tensors. The custom python *AttentionFeatures* class takes the hyperparameters E for the input embedding dimension, K for the key dimension, N for the number of horizontal attention features and F for the horizontal feature dimension.

CNN attention

Finally, a variation of the attention features architecture was used (Fig. 3.2). This architecture is identical to that of the attention features architecture (Fig. 3.1), except that both the key vectors and value vectors are extracted with convolutional neural networks. By doing this, contextual information is used to both determine how the information of the vectors is combined and what information is extracted from each vector. Then, in addition to the hyperparameters of the *AttentionFeatures* class, the *CNNAttention* class takes the hyperparameters V for the value dimension and ks for the CNN kernel size. Then, as the vectors that are combined through dot product attention are sized $L \times V$, the attention output is $2N \times V$, the horizontal featues size is $1 \times V$ and the overall output of the CNN attention is $N \cdot F + V$.

3.6 Organism predictions

The pH associated with each organism was predicted by aggregating the individual predictions for each of its exposed proteins. The model family with best test metrics (MAE) was selected to perform protein pH predictions. Predictions were then made in test + validation data to first select between the weighted and unweighted version of the model by best MAE of the average predictions by genome.

After selection of the weighted or unweighted version, different aggregation heuristics were tested in train data to obtain the best metrics. The mean of the prediction, median prediction, and mode prediction with density using different σ were be tested for the best organism predictions on train data. The selected hyperparameters were then used to perform predictions on test + validation data to obtain the final model performance.

Chapter 4

Results

4.1 Dataset Description

In this study, the proteins of organisms that grow optimally at a wide pH range from extremely acidic to circumneutral (around pH 7) were studied. The main aim of this study is to predict the pH that a protein is exposed to and can resist based solely on its amino acid (the building blocks of proteins) sequence. The organisms here used are acidophilic and neutrophilic Bacteria and Archaea, which are single cell organisms that contain actively functioning proteins, some of which are found near or in their envelope [91]. The inner cavity of these organisms or cytoplasm is usually neutral (pH close to 7), while their envelope is exposed to the outer pH [58]. Hence, proteins located in the envelope of the cell are theoretically exposed to the pH of the medium that the organisms thrive optimally in.

In total, the genomes of 696 organisms were obtained, of which 180 were from Archaea and 516 were from Bacteria (Fig. 4.1A). All the genomes had experimentally validated information of their optimal growth pH, and at least one exposed protein. The optimal pH of the organisms ranged from pH 0.7 to pH 8, and there was a higher density of genomes with optimal growths around pH 7 (Fig. 4.1C). For each genome, from 9 to up to 1200 exposed proteins were extracted, 314 in average for Bacteria and 31 in average for Archaea (Fig. 4.1B). Only proteins with 500 amino acids or fewer were considered as a means to simplify deep learning models which analyze the full amino acidic sequence. A total of **152912** proteins were obtained.

4.2 Exploratory Data Analyses

After the extraction and obtention of the protein dataset, the source orgenism pH, henceforth referred to as *pH*, was obtained and assigned as the target variable to each protein. The proteins' features and pH were explored in terms of distribution and correlation. The proteins were in average 266 amino acids long, and had a nearly uniform length distribution between around 50 and 500 amino acids, with a highest concentration around 155 - 180 amino acids long (Fig. 4.2).

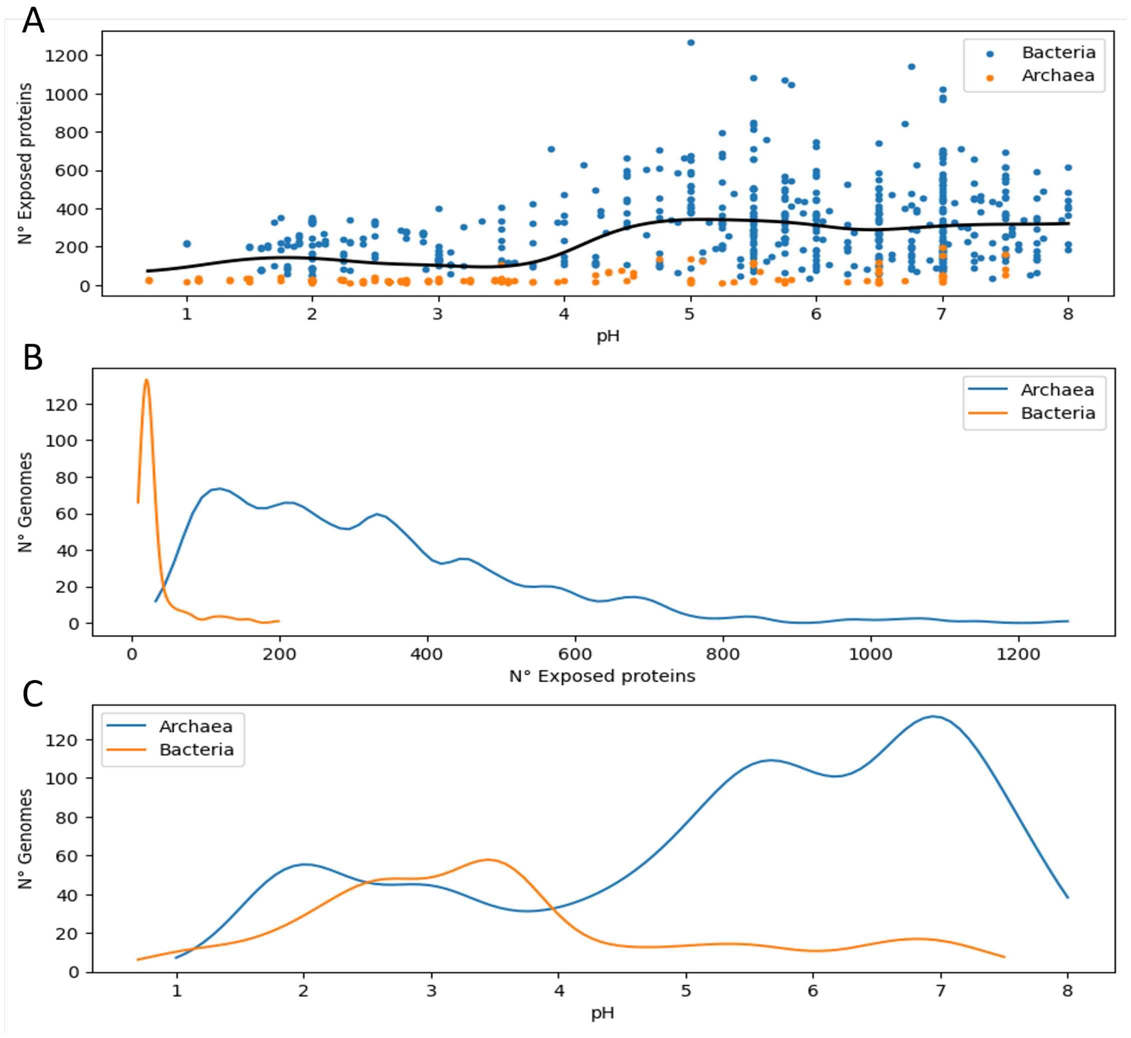

Fig 4.1: A) Bacterial and archaeal genomes obtained for this study ranging from pH 0.7 up to pH 8. The average number of exposed proteins extracted from each genome is plotted alongside their optimal growth pH. Locally averaged number of exposed proteins are shown with a black line (gaussian smoothing) B) Distribution of the number of exposed proteins extracted from each genome for Bacteria and Archaea as a density plot. C) Distribution of optimal growth pH for Bacteria and Archaea as a density plot.

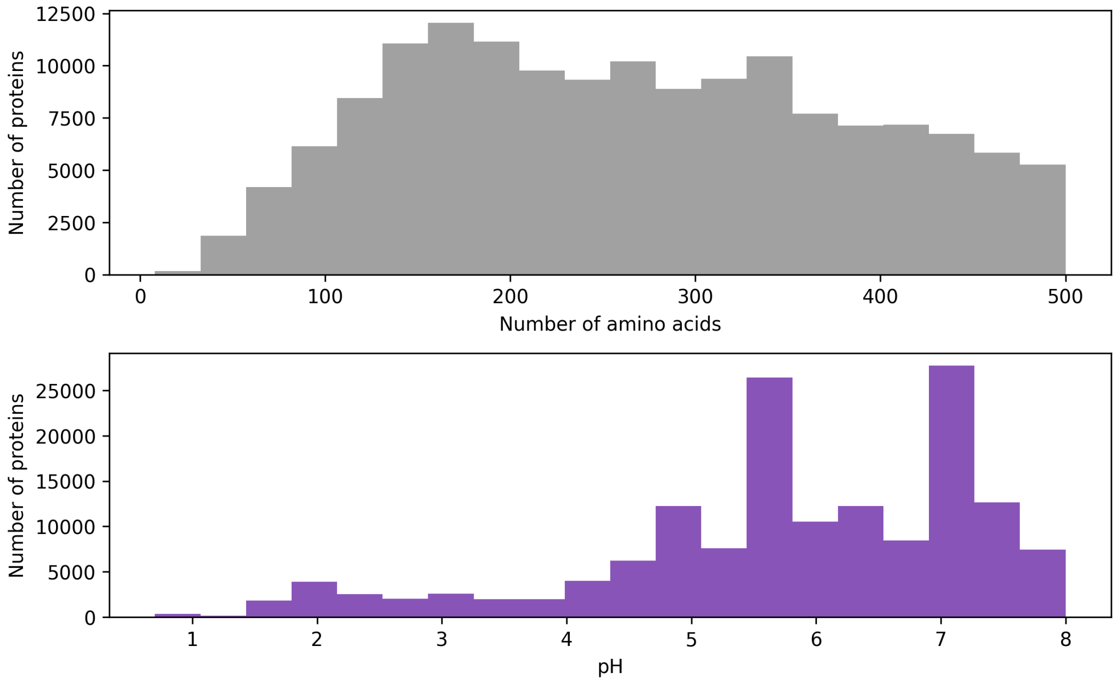

Fig 4.2: Distribution of the protein length (above) and pH (below) of the proteins used in this study.

As for the pH distribution, it was far from uniform. The proteins were highly overrepresented at around pH 5.5 and pH 7, and only an 11.4% of the proteins had pH < 4. This uneven distribution raised the need for normalization techniques to handle unbalanced datasets for regression problems. With this aim, sample weight calculation methods were developed, which assign weights inversely proportional to the pH density. Also, metrics to measure the performance of models which take the inbalance in consideration were developed (see methods).

In addition to protein length and pH, protein sequence patterns and other features obtained with several bioinformatic tools were extracted. Changes in these features across pH were explored to the extent of evaluating their predictive capacity.

4.2.1 Amino acid composition

The main determinant of the characteristics and features of a protein is its amino acidic sequence. Technically, all the features used in this study stem directly or indirectly from the amino acidic sequence of the proteins, and their traits are complexely encoded in it.

To explore the full amino acidic sequence of proteins and changes in them across pH is not straightforward. A workaround to this issue is to explore changes in the simple amino acidic frequencies. Fig. 4.3 shows the distribution of the amino acidic frequencies, where the most common amino acids are Alanine (A) and glycine (G) while the least common are tryptophan (W) and cysteine (C). The PCA analysis shows a slight tendency between the vertical (PC2) axis and pH. Indeed, there is a slight negative correlation between the second principal component of the amino acidic frequency and pH of 0.14 (p-value ~ 0), which is an indicator that the amino acidic frequencies are a good predictor of pH and should change with pH.

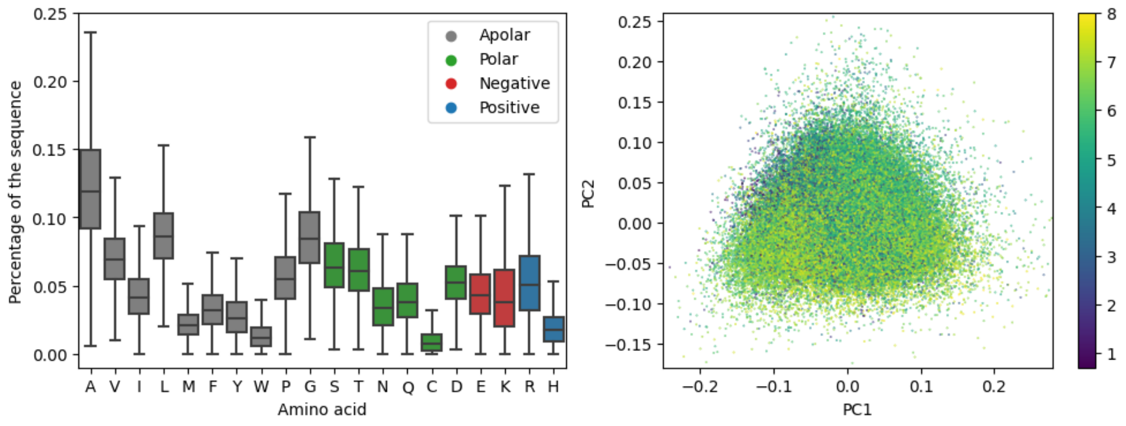

Fig 4.3: Left plot shows amino acid frequency distribution with boxplots for each amino acid in terms of percentage of the protein. Right plot shows a dimensionality reduction analysis where the dimension of the frequency of all 20 amino acids was reduced with PCA. The first 2 principal components are shown with a scatterplot. Points are colored by pH.

An additional analysis was made by comparing the amino acidic frequencies across pH. This is, to determine if some amino acids are more or less frequent at different pH values. In Fig. 4.4, the 10 amino acids with highest variation across pH are shown. Negatively charged acidic amino acids exhibit a lower frequency at lower pH (red lines, amino acids D and E). The same is observed for the positively charged basic amino acid (blue lines) arginine (R), but not for histidine (H). On the other hand, polar amino acids exhibit an increase in frequency at lower pH, indicating an exchange of salt bonds by hydrogen bonds as a stabilizing mechanism at lower pH [106]. These results indicate that pH has some level of predictability, even only using simple sequence traits such as amino acid frequency, which is the equivalent of bag of words.

4.2.2 Secondary structure

As secondary structure mostly depends on local amino acid neighborhood, efficient machine learning methods for secondary structure prediction have been developed [25] and were used in this study as features. A secondary structure prediction is done on each amino acid, and therefore the output can be summarized as a chain of characters as long as the protein (with only 3 possible different characters). E.g., ”HHHBBBBHHHCCCC...”. Similarly to the amino acid sequences, it is simpler to explore the frequency of each type of secondary structure than the complete secondary structure sequence. In Fig. 4.5, the secondary structure distribution shows that the most common secondary structure is the loop, with a peak abundance at around 55% of the protein's secondary structure. Both the alpha helix and beta sheet secondary structure have a peak frequency at 0% of the protein, indicating that purely alpha helix or beta sheet proteins are common. Besides, secondary peaks at around 15% for the beta sheet and around 35% for the alpha helix were found. The joint distribution indicates that there are peaks of pure alpha helix structures with around 65% alpha helix, pure beta sheet structures with around 40% beta sheets, and mixed structures with about 40% alpha helix and 14% beta sheets.

In Fig. 4.6, local averages of secondary structure composition (as frequency) are shown across pH. Besides from a slight increase in the ratio of beta sheets at pH ~ 1, no significant differences are observed. However, as the number of organisms and therefore proteins found at these low pHs is low, this tendency is likely not significant and probably reflects the individual charasteristics of the few organisms and proteins that are found at those pH values.

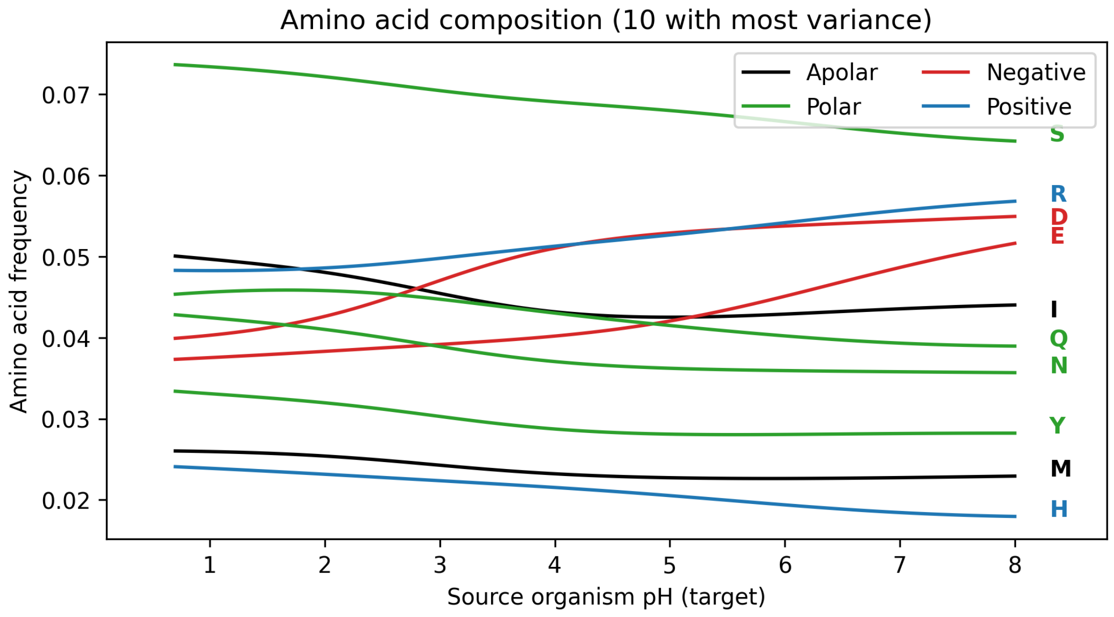

Fig 4.4: Amino acid composition across pH. The frequency of each amino acid was calculated as a fraction of the total number of amino acids. The amino acid frequency was then smoothened against pH with gaussian smoothing and plotted. Only the top 10 amino acids with highest variance relative to the mean frequency of the amino acid were plotted

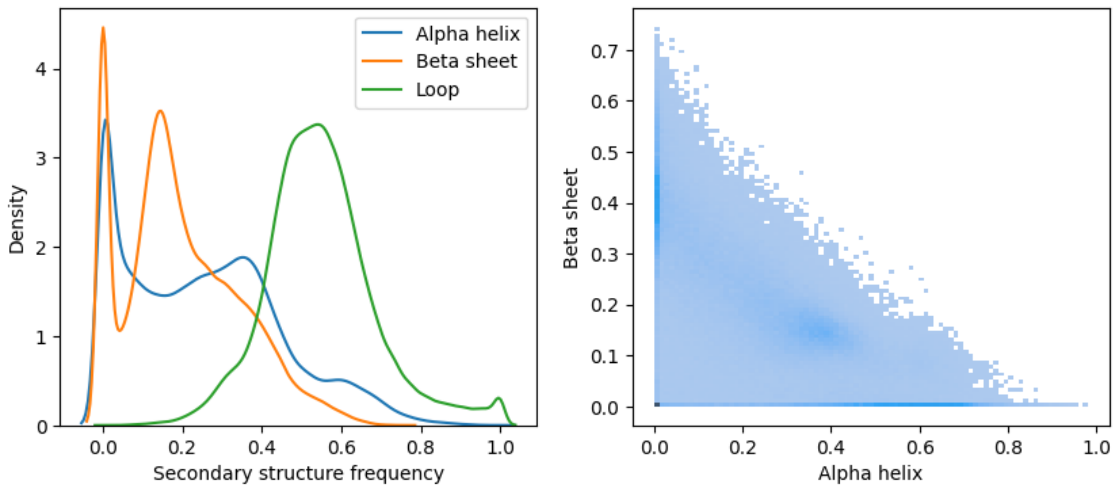

Fig 4.5: Left plot shows a KDE plot of the distribution of the frequency of alpha helixes, beta sheets and loops in terms of percentage of the protein. Right plot shows the joint distribution of alpha helix and beta sheet frequency as a 2D histogram. As all the frequencies add up to 1, this plot indirectly includes the loop frequency.

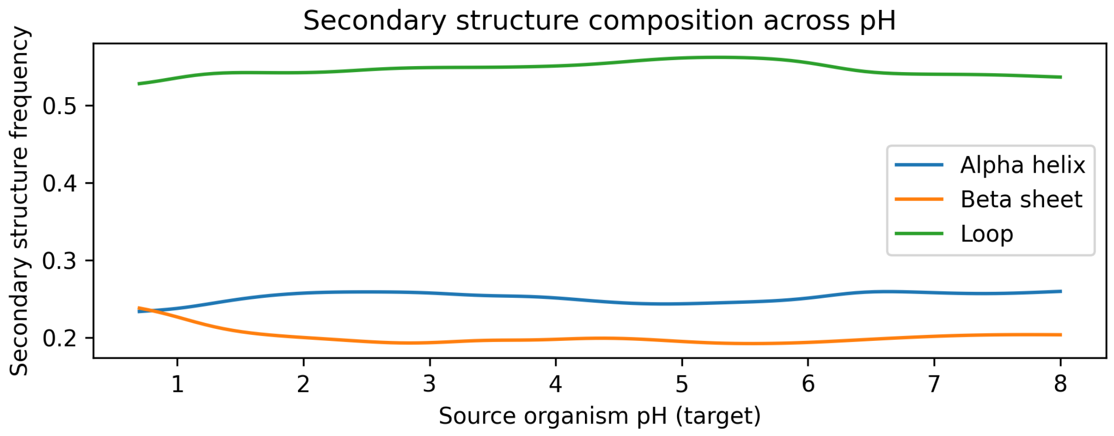

Fig 4.6: Predicted secondary structure of proteins across pH. The average frequency of each type of secondary structure (alpha helix, beta sheet or loop/coil) was smoothened as in Fig. 4.4.

While the secondary structure frequency tendencies were not robust enough to support them as a strong pH predictor, the secondary structure sequence could still be used alongside amino acidic sequence in more complex NLP models. The secondary structure frequencies could still be useful alongside other features, so they will be included in the training of classical machine learning models.

4.2.3 Transformer autoencoder output

The pretrained protein autoencoder transformer type model protBERT [26] was used to encode the protein sequences into vectors that capture the necessary information to reconstruct the amino acidic sequence at high accuracy. The outputs of the autoencoder were vectors of size 1024. All dimensions of the vectors had normal-like distributions but with different means and variances (Fig. 4.7). The average means of the encoding features was around 0.002 ± 0.18, while the average standard deviations where around 0.034 ± 0.023.

Supervised explorations were performed to evaluate if the dimensions of the encoder had any correlation with the target variable pH. Pearson's correlation index was calculated between each encoder dimension and pH and the distribution of the correlations was plotted (Fig. 4.7, lower right plot). Correlations of up to 0.11 and down to -0.11 were found. While no feature had significant correlation with pH individually, the fact that a significant number of them had a slight correlation with pH indicates that collectively they could be a valid predictor. Similarly, principal component analysis (Fig. 4.8) shows slight variations in the local pH averages for components 1, 2, 3 and 5.

It is not expected for these encoder dimensions to be individually correlated with pH. The principle behind the autoencoder is to capture the complex information behind the amino acidic sequence of a protein in a much lower dimension than all the information a protein can have. This is possible because the real distribution of the protein sequences is not random: proteins evolve to adapt and attain new functions, thus producing divergence over time. This causes proteins to retain features of their ancestors, causing proteins to have a similar sequence than those that have similar function or structure. Hence, the sequence of a protein could be encoded as slight variations (or noise) of the evolutionary backbone, which has limited possibilites. By this focus, the encoding of protBERT would represent the general family, function or structure of a protein, not necessarily the pH.

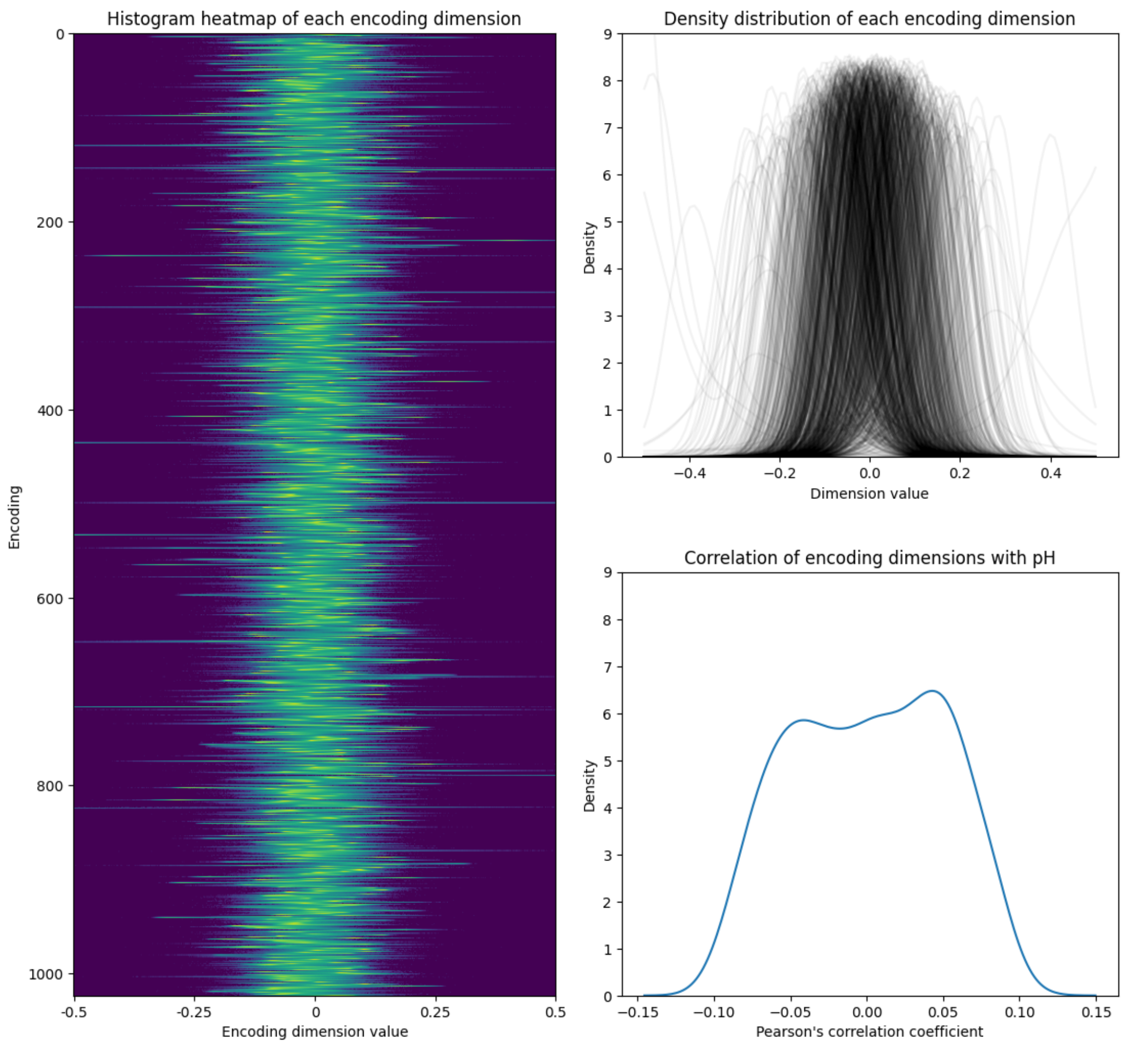

Fig 4.7: Distribution of each individual dimension of the protBERT encoding. Left plot is an histogram heatmap where each row represents a different dimension of the encoding vector (1024 dimension). For each, its distribution from -0.5 to 0.5 is plotted as a heatmap. Upper right plot is the same analysis but as distribution density, where each semitransparent line represents the density of a different dimension. Lower right plot is the distribution of the correlation coefficients of each dimension with pH.

As acidophilic proteins can have the same functions than non-acidophilic proteins, the encoding of these proteins is probably similar to that of their non-acidophilic counterparts. However, as the encoding of the protein captures the family, it could still be useful as a feature, as some of the adaptations could be family-specific.

4.2.4 Other features

Next, the molecular weight, isoelectric point and number of transmembrane segments were explored. The distributions of the features are shown in Fig. 4.9, and the mean as a function of pH is shown in Fig. 4.10.

The molecular weight of a protein is the total mass in Daltons of the molecule, which correlates strongly with the length of the protein. This is reflected on an almost identical distribution shape to that of protein length (Fig. 4.2 and Fig. 4.9). When compared with pH, a slight decrease in molecular weight is observed at pH < 3, indicating that the exposed proteins of extreme acidophiles

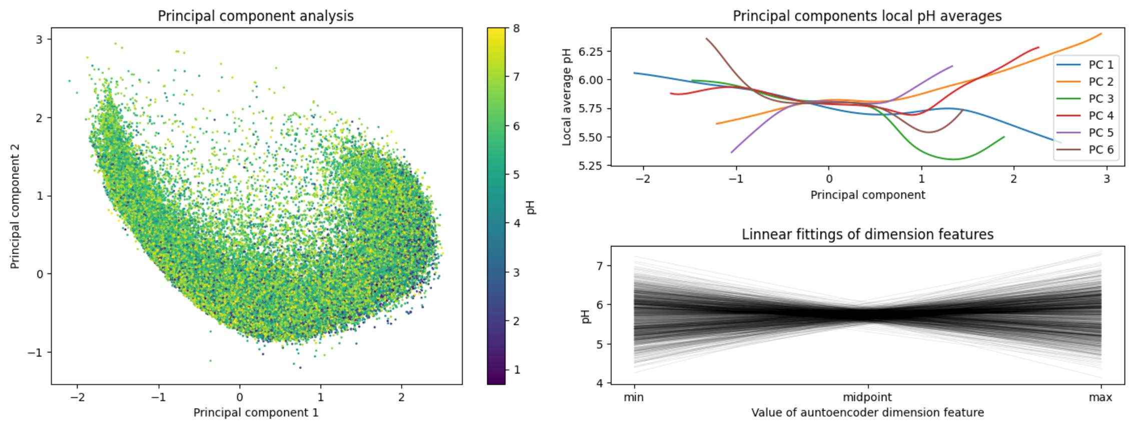

Fig 4.8: Unsupervised and supervised analysis of encoding feature dimensions. Left plot is a scatterplot of the principal component analysis (PCA) of the feature dimension. The first 2 principal components of the feature dimensions are plotted, where each point represents a protein. Points are colored according to the protein's pH. Upper right plot is a smoothed local average pH of the proteins versus the 6 first principal components. Lower right plot shows the predictions of linnear regressions fitted on each encoder dimension on the lowest value, the highest value and the midpoint of each encoder dimension.

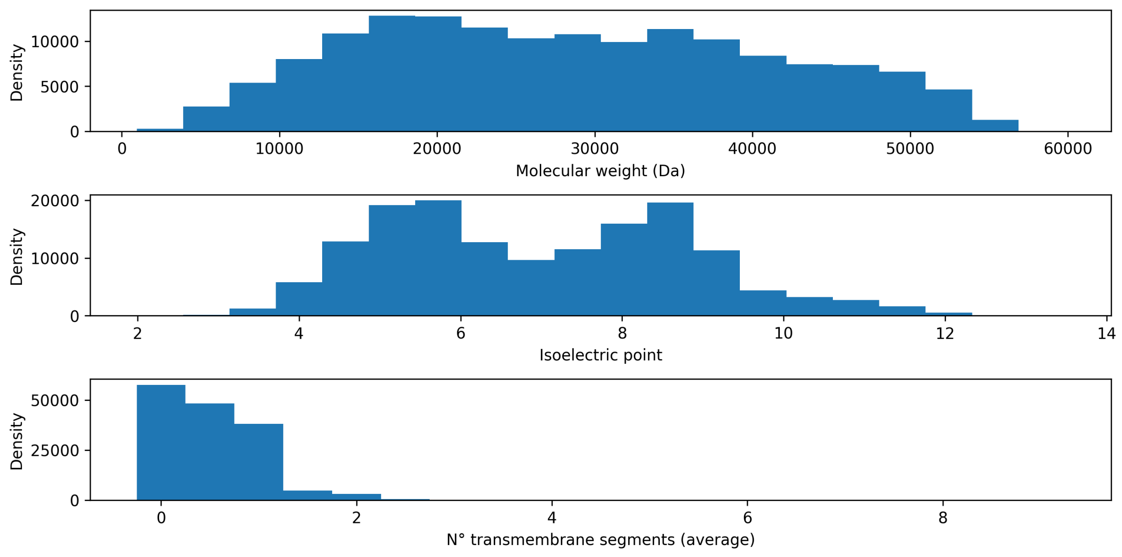

Fig 4.9: Distribution of other features is shown as histograms. Top plot: molecular weight (in Daltons). Mid plot: isoelectric point. Bottom plot: average number of transmembrane segments.

are smaller. However, this decrease represents only about a 5% decrease of the protein mass and is negligible when compared to the molecular weight range.

The isoelectric point is one of the most important physicochemical features of a protein. It represents the pH at which a protein has no charge. The charge of a protein is determined by the presence of charged amino acids, which are acidic or basic molecules that either release or accept protons to become negative or positively charged, respectively. This confers them ionic properties that permit the formation of salt bonds that stabilize the protein [106]. Because these are weak acids or bases, a different pH may reverse the proton coupling thus removing their charge. As a consequence, there is always a pH at which the number of negative charges is equal to the number of positive charges, producing a net charge of zero. This pH is the isoelectric point.

The distribution of isoelectric point follows a typical 2 peak distribution usually observed in this variable, where proteins are either classified as basic or acid proteins if their isoelectric point is > 7 or < 7, respectively [108]. Contraintuitively, proteins with low pH had higher average isoelectric points than neutral proteins, meaning that the proteins that resist low pH need a high pH to be neutral. Therefore, at low pH these proteins are positively charged, which could constitute a defence mechanism or a protein stabilization mechanism [8, 70]. The change in mean isoelectric point shifts from around 7 at pH 7 to about 8 at pH 1, which is relatively significant when compared to the isoelectric point distribution. This implies that basic and acid proteins are equilibrated at neutral pH but at low pH basic proteins are the majority of the exposed proteins.

Finally, the average number of trans membrane segments was analyzed. These are sections of the protein that are embeded in the cellular membrane (the envelope), which is a lipid layer. Proteins with transmembrane segments are not freely exposed to the external medium, but are anchored and slightly protected from it inside the cell envelope. In this study, the proteins were observed to have 2 or fewer transmembrane segments in average. Exposed proteins with over 2 transmembrane segments do exist, but are rare. Proteins with pH < 4 were found to have a higher number of average transmembrane segments. This could mean these proteins are more protected from the low pH, or that these organisms have a higher proportion of membrane embedded proteins in relation to exposed proteins in the periplasmic space (space between cell envelopes in bacteria with double membrane).

4.2.5 Bivariate analysis

Lastly, all features except for the bert encodings were explored in a bivariate analysis. A correlation matrix between these features and pH is shown in Fig. 4.11. Protein length and molecular weight have an almost perfect correlation, which is to be expected as the molecular weight of a protein deppends mostly on its length. A strong negative correlation is also observed between the secondary structure frequences. This is expected as the frequencies add up to 1. Next, there are some somewhat strong correlations between the isoelectric point and basic (positive) or acid (negative) amino acids, which are determinant of the isoelectric point. Other notable correlation is between the loops or coils (ss_C) and proline (aa_P). Proline is a rigid amino acid which usually disrupts the secondary structure of proteins.

However, there aren't many strong correlations between pH and other features. The highest correlations observed are with aspartic acid (aa_D) and glutamic acid (aa_E), and a very slight negative correlation with isoelectric poing and histidine (aa_H).

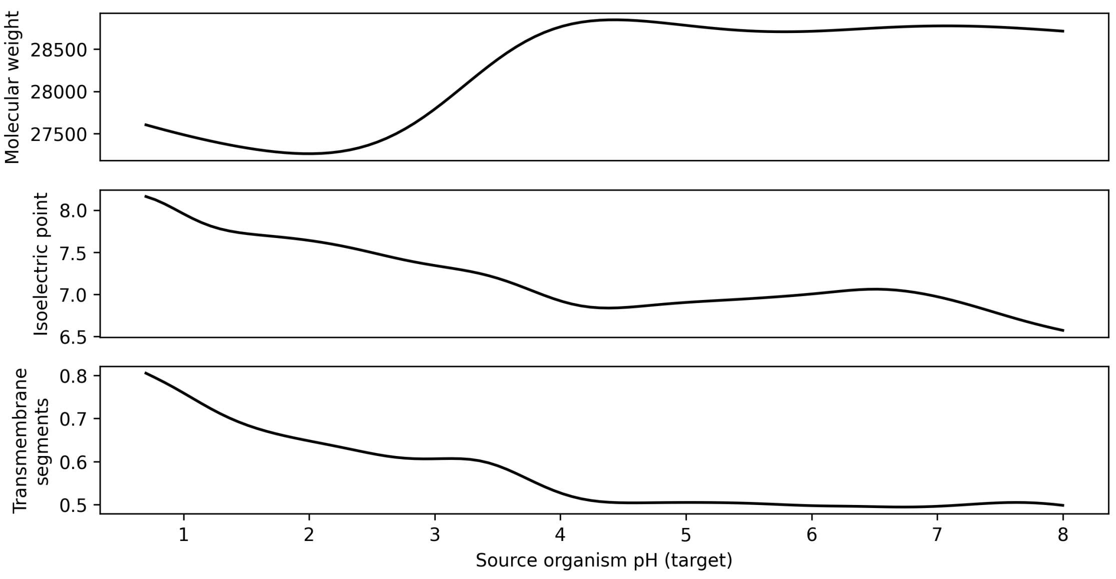

Fig 4.10: Average of other features was plotted as a function of pH. The features were averaged locally against pH with gaussian smoothing. Top plot: molecular weight (in Daltons) average across pH. Mid plot: isoelectric point across pH. Bottom plot: average number of transmembrane segments across pH.

4.3 Classical machine learning models

The term classical machine learning is used to refer all machine learning algorithms that do not involve deep learning and do not use the full amino acidic sequence as input. This includes from simple linnear models to complex bagging and boosting models. Some of the models described in this section do use the output of complex transformer type models as a complementary input along with other features. As no fine-tuning was done to obtain this transformer output, it will only be considered as an additional feature.

4.3.1 Ridge regression

For the ridge regression models, different regulariation factors (alpha) were tested and selected with best metrics in test data. For the unweighted version, alpha values from 10 to 10000 were tested, while for the weighted versions, alpha values from 10^5 up to 10^8 were tested (Fig. 4.12). For the unweighted version, an alpha factor of 1000 was found to be optimal, while for the weighted version, an alpha factor of $3 \cdot 10^6$ was found to be optimal. This was determined with the metric MAE for the unweighted version and WMAE for the unweighted version.

The full metrics are present in the table 4.1. The weighted version of the model had inferior performance according to all metrics except for the WMAE, which reflects more accurately how good the predictions is at all pHs.

4.3.2 Random forest

The random forest models had a high computational cost, due to the large dimension of the data and the high number of proteins to be classified. Features were then shortly explored, aiming for a high number of trees as long as the computational cost permits it, and testing several max features and max samples values.

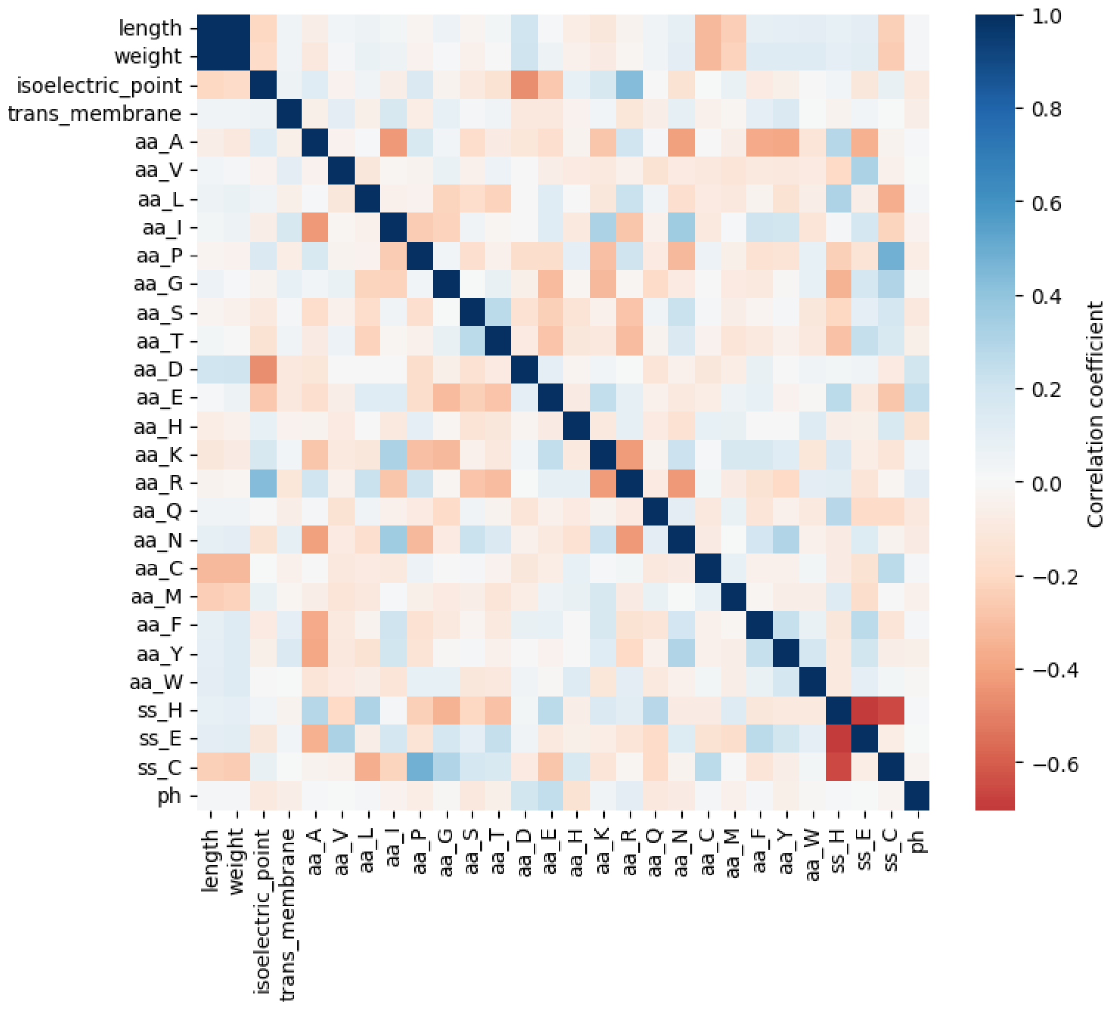

Fig 4.11: Correlation matrix heatmap between all protein features, amino acidic frequency, secondary structure frequency and pH. The *aa* preffix indicates amino acid frequency, while the *ss* prefix indicates secondary structure frequency.

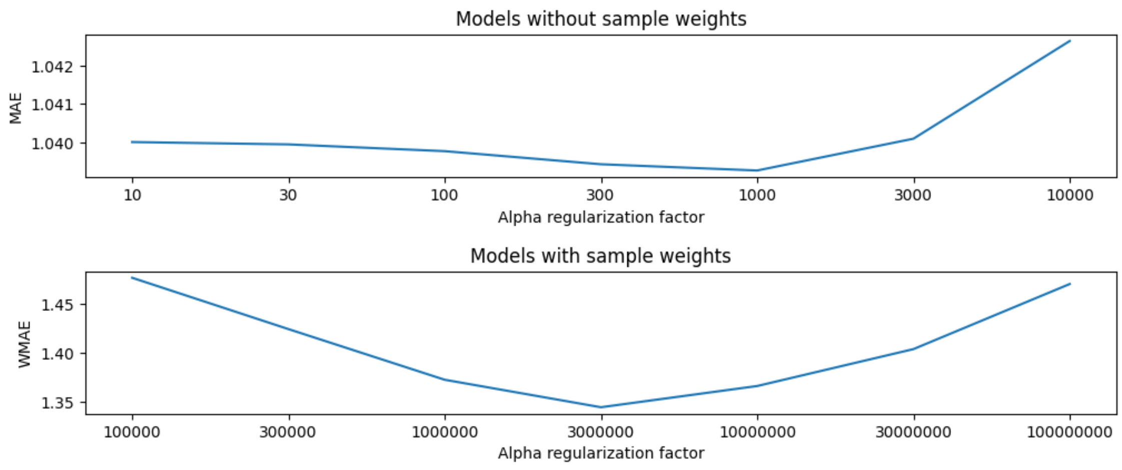

Fig 4.12: Metrics of the ridge regression models in test data using different regularization factors. Upper plot shows the mean absolute error between the test pH and the prediction with ridge regression models trained without sample weights. Lower plot shows the weighted mean absolute error between the test pH and the prediction with ridge regression models trained with sample weights.

Metric	Unweighted model	weighted model
Mean absolute error (MAE)	1.039	1.326
Weighted mean absolute error (WMAE)	1.609	1.480
R^2 score	0.22	-0.11
Classification f1-score macro	0.42	0.31

Table 4.1: Metrics of Ridge regression models in the weighted and unweighted versions.

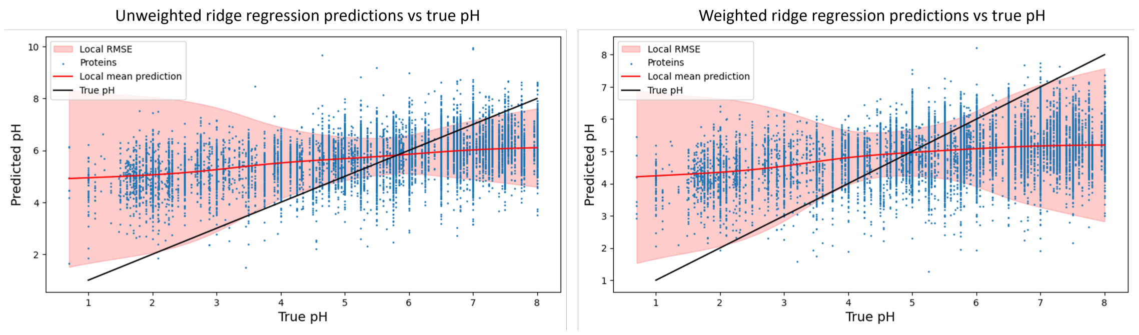

Fig 4.13: Ridge regression models result. In each plot, the predicted pH is plotted against the real pH of each protein. Each point represents a single protein. The local average was calculated for each pH with gaussian smoothing (red line) with a standard deviation of 5% the range of the target variable. Local RMSE was also calculated by weighting the total RMSE with gaussian smoothing, using a 2% of the pH range as standard deviation. The red shadow shows the area between the local average $\pm$ the local RMSE. A black line shows the perfect prediction (true pH = predicted pH). Left plot shows the unweighted model while the right plot shows the weighted model

The best models trained had 200 trees, max features of 0.4 and max samples of 0.1. The metrics and predictions are shown in Table 4.2 and Fig. 4.14, respectively. Surprisingly, there was no significant difference between the weighted and unweighted version. In fact, the weighted version had slightly worst weighted metrics, as well as absolute metrics.

4.3.3 Gradient boosting

For the gradient boosting models, the Histogram-based gradient boosting regressor was used. All hyperaparameters were set to default, except for the use of the absolute error as loss function.

These models had the best results so far and were computationally more efficient than random forest. This, in terms of unweighted MAE for the unweighted model and weighted MAE for the weighted model. The metrics and predictions are shown in Table 4.3 and Fig. 4.15, respectively. Surprisingly, the best f1-score so far of 0.51 was obtained in the weighted version of this model. This highlights the importance of using weighted models to train on unbalanced date and validates the novel metric WMAE as a regression analog to the f1-score.

Metric	Unweighted model	weighted model
Mean absolute error (MAE)	1.015	1.021
Weighted mean absolute error (WMAE)	1.674	1.691
R^2 score	0.26	0.25
Classification f1-score macro	0.43	0.42

Table 4.2: Metrics of random forest models in the weighted and unweighted versions.

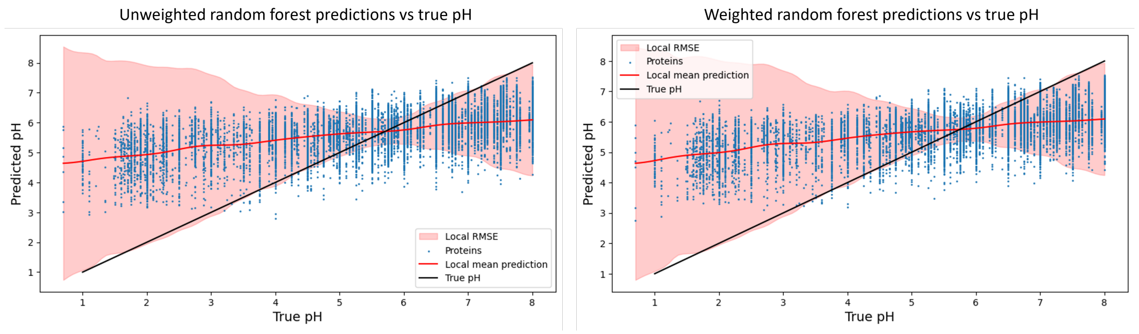

Fig 4.14: Random forest models result. Analog to Fig. 4.13 but for random forest. Left plot shows the unweighted model while the right plot shows the weighted model

Metric	Unweighted model	weighted model
Mean absolute error (MAE)	0.975	1.152
Weighted mean absolute error (WMAE)	1.75	1.270
R^2 score	0.24	0.06
Classification f1-score macro	0.46	0.51

Table 4.3: Metrics of gradient boosting models in the weighted and unweighted versions.

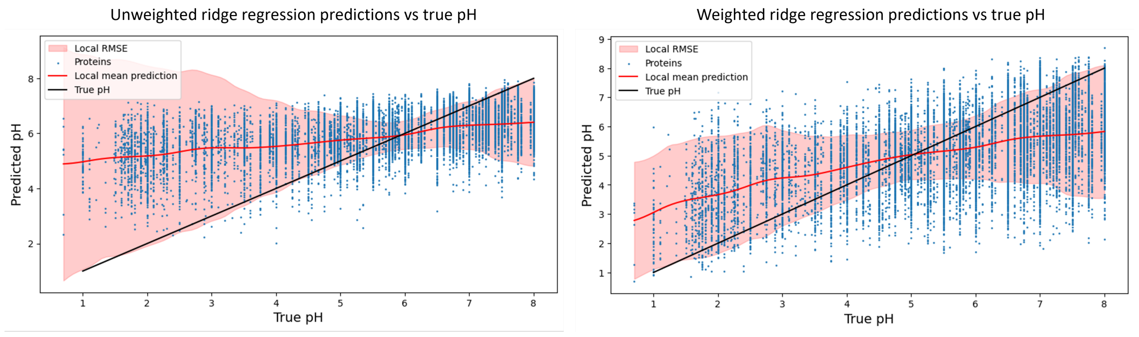

Fig 4.15: Gradient boosting models result. Analog to Fig. 4.13 but for gradient boosting. Left plot shows the unweighted model while the right plot shows the weighted model

4.4 Deep learning models

As it is specified in the methods section, the development of deep learning models was an iterative process, where the first models were trained and evaluated on different metrics and settings. Also, the first models were simpler and did not include all the features used for the classical machine learning models. Part of the focus of this thesis is to explore different deep learning architectures to find which is more suitable for this type of problem.

4.4.1 Model history

Models with a total number of 26 different architectures were trained for this thesis. The complete list is present in Table 4.4. The architectures were explored manually by seeking to maximize efficiency and prediction capacity while minimizing overfitting. The models were all trained locally in a 4GB GPU, which limited the models capacity.

The models can be subdivided into 5 generations. From first generation to fifth generation, the models cycle through bigger and smaller architectures while gradually increasing performance. The models metrics are shown in Table 4.5. Eventually, some architectures were found to have superior metrics and were then used in an undersampling approach which increased the performance in weighted metrics (highlighted in bold in 4.5). Some of them were re-trained using updated settings. The general development history of all models will be overviewed in this section. However, the best performing and more complex architectures (generations 3-5) will be described in detail in the next sections.

Models 1-7 are first generation models, which used a classifier approach where the pH was segmented into 3 intervals: pH 0-3.8, pH 3.8-6 and pH 6-8. These are the same intervals used for the f1 score calculation in regression models. The aim was to follow the hypothesis that there is a significant change at around pH 3.8 because of the protonation of acid (negative) amino acids. All these models were trained with cross-entropy loss. Models 1-2 had simple LSTM architectures, while models 3-7 used the amino acid frequency (a 20 sized vector) as an additional input, which is appended to the LSTM output before the fully connected layers. Besides that, models differed only on the hidden and embedding sizes and number of layers.

Models 8-12 are second generation models. These models have an identic architecture to that of models 3-7, but are regressors. That is, they have a 1 dimensional output instead of a 3 dimensional output and are optimized with MSE using the pH as target. Similarly to first generation models, they only vary in model size. For these models, f1 score was calculated by performing a post-prediction discretization, as described in the methods section.

In the next sections, third, fourth and fifth generation models are described. All of them had different attention mechanisms which boosted the extraction of information from the amino acidic sequence.

4.4.2 LSTM with attention models

Models 13-18 are third generation models. These models are LSTM-based regressors with an attention layer that extracts relevant protein vectorial representations from the LSTM outputs by performing dot product attention. The embedings of the input are passed through a Bi-directional

Model	Architecture	Layers	Hidden	Params	D.O.	A.a.	S.s.
1	LSTM clasif.	2	64	150 k	0.6	32	
2	LSTM clasif.	3	128	991 k	0.7	64	
3	LSTM clasif. + a.a freq	2	64	151 k	0.9	32	
4	LSTM clasif. + a.a freq	2	256	2.63 M	0.9	256	
5	LSTM clasif. + a.a freq	2	128	596 k	0.85	64	
6	LSTM clasif. + a.a freq	2	64	168 k	0	64	
7	LSTM clasif. + a.a freq	2	256	2.14 M	0.93	16	
8	LSTM reg. + a.a freq	2	64	142 k	0.8	16	
9	LSTM reg. + a.a freq	2	112	668 k	0.5	16	
10	LSTM reg. + a.a freq	2	256	3.29 M	0.6	16	
11	LSTM reg. + a.a freq	2	64	324 k	0.55	16	
12	LSTM reg. + a.a freq	2	128	1.14 M	0.95	16	
13	**LSTM reg. + 1/1 Att + a.a freq**	2	64	219 k	0.4	16	
14	LSTM reg. + 3/3 Att + a.a freq	2	64	767 k	0.7	16	
15	LSTM reg. + 0/0/8 Att + a.a freq	3	128	5.28 M	0.6	64	
16	LSTM-CNN reg. + 0/3 Att + feats.	2	128	1.31 M	0.6	64	16
18	LSTM reg. + 0/2 Att +bert	3	128	3.61 M	0.6	104	24
19	Cross LSTM-CNN + 2*16 AttFe	2	64	219 k	0.6	12	4
20	CNN reg. + 2*32 AttFe	2	128	409 k	0.1	24	8
20b	**CNN reg. + 3*32 AttFe +bert**	3	128	1.16 M	0	24	8
21	LSTM reg. + [CNN 2*AttFe] *2	2	64	373 k	0.6	24	8
21b	LSTM reg. + [CNN 2*AttFe] *2 +bert	2	64	847 k	0.6	24	8
22	CNN reg. + 2*16 AttFe + Emb.	3	128	2.91 M	0	1024	24
23	CNN reg. + 2*16 CNNAttFe	2	64	562 k	0	48	16
23b	**CNN reg. + 2*16 CNNAttFe +bert**	3	128	1.66 M	0	48	16
24	**LSTM reg. + 2*16 CNNAttFe +bert**	2	64	472 k	0.35	20	4

Table 4.4: Model architecture history. The architecture of all model architectures trained in this thesis is shown. Layers and Hidden refers to the number of layers and hidden size of sequence analysis (either LSTM or CNN) layers, not the FC layers. Params is the total number of model parameters. D.O= dropout probability of the sequence analysis layers. A.a= dimension of the amino acid embeddings. S.s= dimension of the secondary structure embeddings. The architecture shortly describes the type of architecture of the model. All LSTM are bidirectional LSTMs. clasif.= classifier. reg.= regressor. a.a. freq= amino acid frequency. feats.= features (frequencies and other features). bert= all features including bert features, 1-2 gram frequencies and other features. Att= attention. n/n Att describes the number of attention layers applied to each sequence analysis (LSTM or CNN) layer. AttFe= attention features. n*F describes F attention features were extracted from each of the n layers. CNNAttFe= CNN attention features.

Model	Epochs	MAE	MSE	R^2	WMAE	WMSE	WR2	F1
1	30							0.378
2	30							0.443
3	20							0.429
4	20							0.317
5	40							0.417
6	50							0.313
7	50							0.506
8	60		1.58					0.54
9	105		1.55					0.58
10	95		1.57					0.59
11	215		1.63					
12	20		1.78					
13	80	**0.905**	1.59	0.31	1.42	3.58	0.17	0.58
14	120		1.469					
15	40	0.91	1.494	0.34				0.6
16	50	0.904	1.53		1.365			0.61
18	35				1.407			
19	105	0.904	1.59		1.463			0.59
20	890	0.942	1.69	0.255	1.662	4.64	-0.01	0.539
20b	340	**0.872**	1.439	0.366	1.441	3.548	0.232	0.586
21	60	0.904	1.61	0.291	1.361	3.431	0.2	0.6
21b	250	0.88	1.534	0.324	1.274	3.078	0.281	0.616
22	16	0.944	1.7	0.253	1.663	4.644	-0.01	0.539
23	50	0.941	1.691	0.255	1.664	4.648	-0.02	0.538
23b	370	**0.869**	1.425	0.373	1.355	3.242	0.294	0.609
24	95	**0.855**	1.444	0.364	1.223	2.907	0.318	0.631

Table 4.5: Metrics and training epochs of the models present in Table 4.4. The metrics represent the performance of the best model selected with validation metrics, which was MSE for models 1-15 and MAE for models 16-24, except for model 13 which was re-trained with MAE. Whenever possible, the models were tested in test data. Otherwise, validation data was used.

2 or 3 layer LST, and its outputs are fed to self-attention layers. The attention layers extracted features from each Bi-LSTM layer for models 13 and 14, and from only the last layer in models 15, 16 and 18. The advantage of doing the latter is that it allowed for some level of parallelization between the forward pass of the multiple LSTM layers, thus enabling a more efficient training.

Up to 8 attention layers were used, each extracting a vector of the same size as the LSTM hidden size. This posed a problem, as the output dimension of the attention layers scaled linearly with the number of attention vectors used. The outputs of the attention layers were then concatenated to the features associated with each protein. For models 13-15, only the amino acid frequency was used. For model 16, other additional features like secondary structure frequency were included, and for model 18, the bert autoencoder output was also included as feature. After concatenation, the tensors are passed through a 2 layer MLP (usually with a higher dropout probability) that preocesses the information and outputs a 1 dimensional value. Model 17 was a different version of model 16 with the same architecture but different dropout, so it was not mentioned.

Additionally, from model 16 onwards the secondary structure sequence is used along the amino acidic sequence. Both sequences are transformed to integer tensors, passed through embedding tables and concatenated, hence giving the input embeddings more possible combinations (60 possible embeddings rather than 20). Also, from model 16 onwards all models use MAE for training.

From these models, model 13 was simple and the best performing model than the ones that came before, hence it was selected for re-trained to include all the metrics and settings used in the latter models. Upon re-training, the model effectively had a better performance than several models that

Model	D.O.	Epochs	Tuning	MAE	MSE	R2	WMAE	WMSE	WR2	F1
13	0.4	80	Yes	0.988	1.886	0.176	1.136	2.46	0.409	0.605
20b	0	500	No	1.012	1.778	0.217	1.146	2.25	0.503	0.6
23b	**0.05**	370	Yes	0.976	1.659	0.269	1.138	2.218	0.508	0.613
24	**0.4**	750	No	0.938	1.715	0.244	1.049	2.251	0.464	0.633

Table 4.6: Results of models that were further trained with undersampling. D.O. indicates the dropout probability used, where bold indicates the dropout is different to that of the model trained regularly. Tuning indicates if the model was *fine-tuned* using the best model trained regularly as a starting point.

came after, with a MAE of 0.905 (Fig. 4.16, top row).

Model 13 had a simple architecture, where one attention vector is trained by self-attention layer for each of the 2 LSTM layer outputs. Each attention layer performs dot product attention, extracting a 128 dimensional vector (bidirectional LSTMs have double the hidden size) (Fig. 4.17). Both attention outputs are concatenated with the amino acid frequencies, totallying 276 dimensions which are passed through 2 FC layers (with dropout probability 0.3) until an output of dimension 1.

This model was trained with both a regular approach and with undersampling. Undersampling deals with the unbalance in the distribution of the target value by giving additional weight to less common pH values (in this case, low pH values). The weighted metrics improved significantly, where the WMAE hits record value so far (and the second best of all models) of **1.136** (Table. 4.6).

4.4.3 Attention features models

Models 19-22b are fourth generation models. These exhibit more complex and highly creative and diverse architectures. They differ from the previous models in that the attention layers are attention feature layers (see methods section). These attention layers are able to perform dot product attention with multiple attention vectors, but are able to compress the information in multiple ways so not to have extremely high number of dimensions before the fully connected layers.

Model 19 is a cross LSTM-CNN, meaning that both an LSTM and a CNN process the input layer, their outputs are concatenated and used as input for consecutive LSTM-CNNs. From each layer, an ”attention features” layer extracts information. This model was very unefficient to train due to lack of parallelization, and no better than other much simpler models. On the other hand, model 20 is a purely CNN model, in which the output of the first CNN is used by consecutive CNNs. It was very efficient to train, not prone to overfitting but had low predictive capacity. Model 21 was highly complex. In it, a 2 layer bidirectional LSTM is applied to the input, and then the same architecture of model 20 is applied to both the LSTM output and the input. Finally, model 22 had a similar architecture to that of model 20 but using the pretrained ProtBERT amino acid embeddings. The pre-trained embeddings' size was 1024 dimension, which caused performance issues so the development of this type of model was stopped early.

The architectures so far do not use any features as input, only the amino acidic sequence and secondary structure sequence. These were included in latter versions (**b** models) to test if good results were obtained in simpler architectures, which would optimize the computational costs of the final model. Models 20b and 21b are have similar architectures to model 20 and 21, respectively, but use all features, including the bert autoencoder output. The models' performance improved significantly. Remarkably, the worst performing model of the fourth generation, model 20, had the

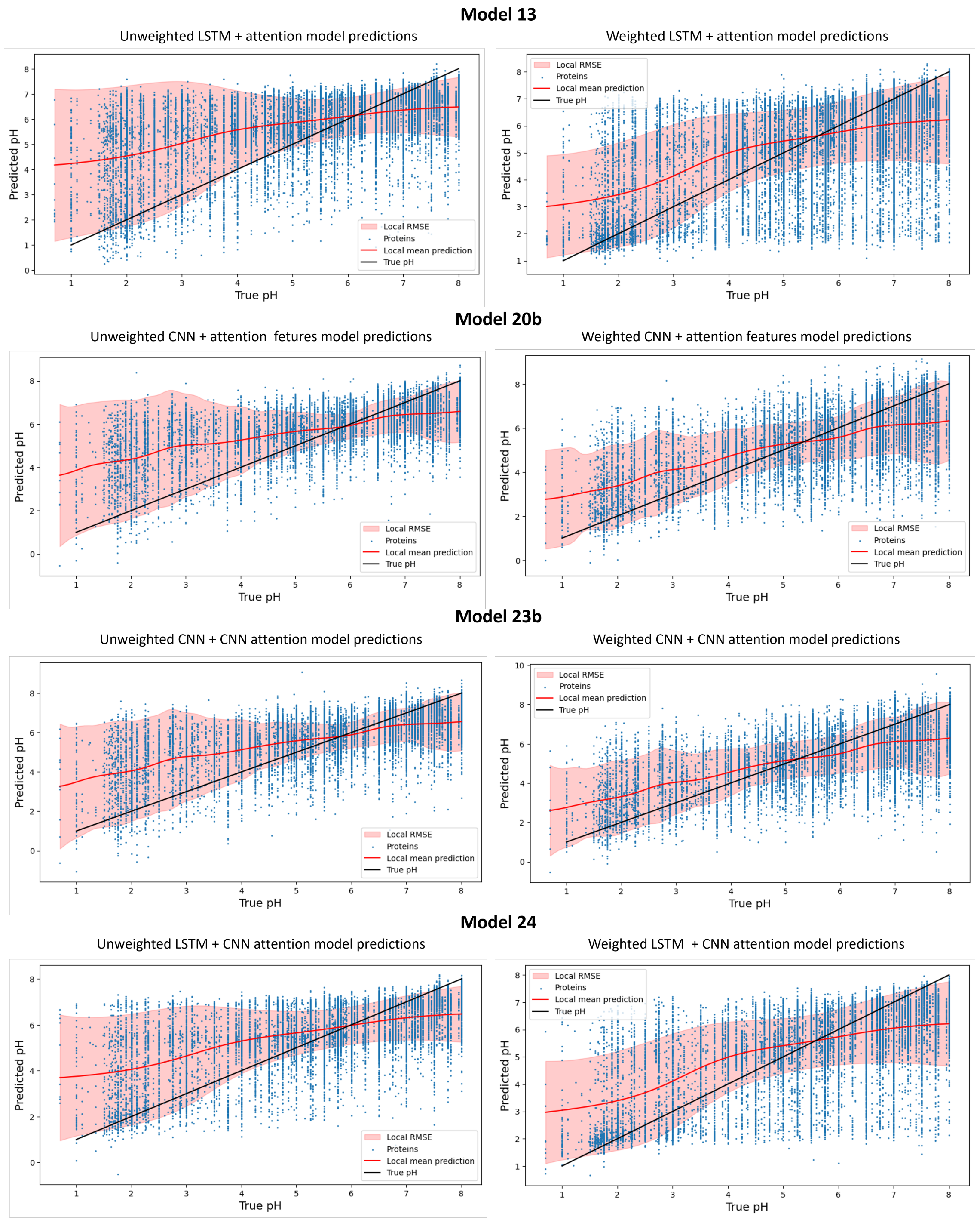

Fig 4.16: Best performing deep learning models in their weighted and unweighted versions. Analog to Fig. 4.13 but for deep learning models.

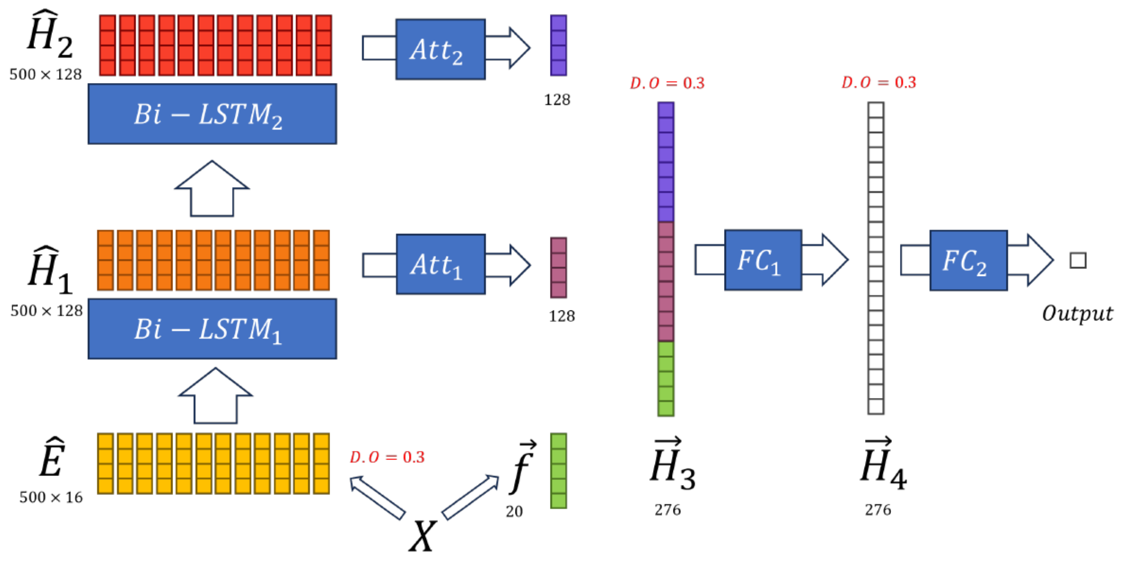

Fig 4.17: Best LSTM with attention model architecture.

best metrics in the version with features (model 20b, Fig. 4.5 and Fig. 4.16 second row), and hence was selected for undersampling.

The complete architecture of model 20b can be found in Fig. 4.18. First, amino acidic sequences and secondary structure sequences are converted to 24 dimensional and 8 dimensional embeddings, respectively, and concatenated to produce 32 dimensional embeddings. As stated above, this model did not contain any recurrent neural networks. Instead, the embeddings are passed through 3 consecutive one dimensional CNN layers of hidden size (N° of channels) 128 and kernel size 11. From each output and from the input embeddings, attention features extract 32 vectors of size 4 (vertical features) and 1 vector of size 128 and 32 for the hidden layers and input layers, respectively (horizontal features). Parallely, the features are also preprocessed with a fully connected layer that reduces their dimension to 256. The frequencies are not processed this way. Finally, all features are concatenated into a 1184 sized vector which is passed to 2 consecutive fully connected layers, with an intermediate hidden size of 384.

With resampling, this model reached the best weighted R^2 score so far (Fig. 4.16 second row, Table 4.6), but did not perform better than model 13 trained with undersampling in terms of WMAE or MAE.

4.4.4 CNN attention

Finally, models 23 - 24 are the fifth and final generation of models developed for this thesis. In these models, CNN attention is used to extract information from the hidden layers (see methods). The CNN attention mechanism is similar to the attention features mechanism, but both keys and values are extracted with convolutional networks. By doing this, complex relationships between the sequences can be captured.

The architecture of model 23 is very simmilar to model 20, as both have consecutive CNN layers from which features are extracted with attention. They differ on the use of CNN attention layers and other minor dimension tweaks. It showed similar performance than model 20 in all metrics.

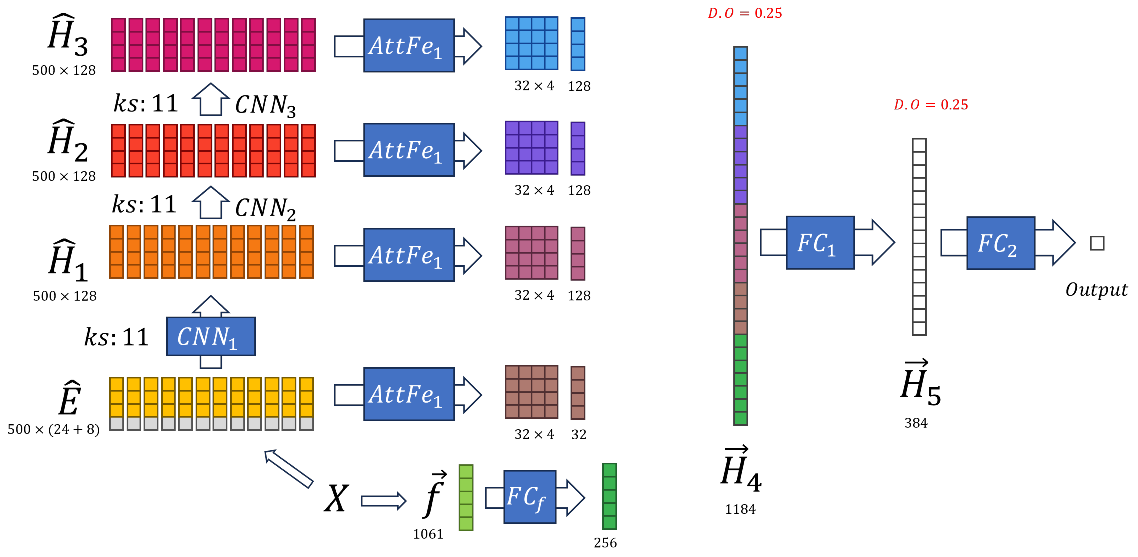

Fig 4.18: Best attention features architecture: model 20b. A consecutive CNN with attention features is shown. Different colored blocks in the embedding tensor indicate that part of the embeddings come from secondary structure data.

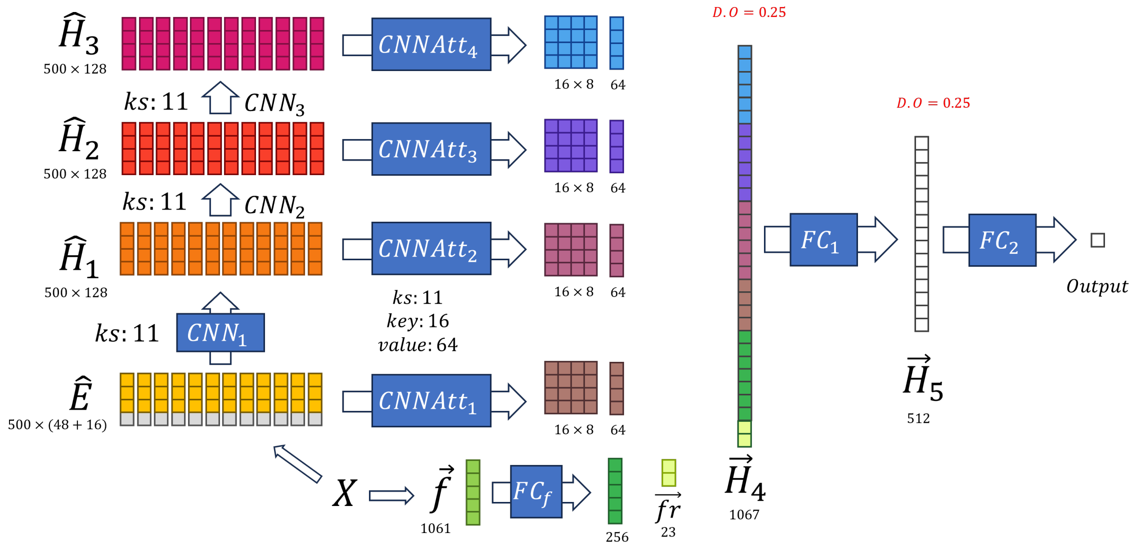

Fig 4.19: Best CNN attention architectures: model 23b. A consecutive CNN with CNN attention is shown. Different colored blocks in the embedding tensor indicate that part of the embeddings come from secondary structure data.

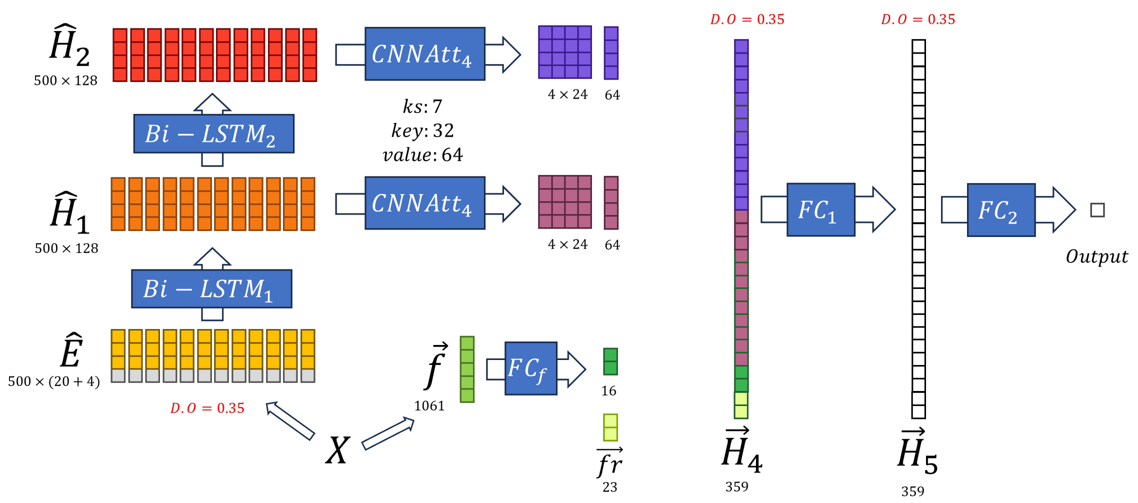

Fig 4.20: Best CNN attention architectures: model 24. A 2 layer Bi-LSTM with CNN attention is shown. Different colored blocks in the embedding tensor indicate that part of the embeddings come from secondary structure data.

The version with features (model 23b) also had a similar architecture to model 20b (Fig. 4.19). Specifically, amino acids and secondary structure are converted to embedding tensors of size 48 and 16, respectively. 3 Consecutive CNN layers with kernel size 11 and 128 channels extract different levels of hidden layers. From each, CNN attention layers with kernel size 11 extract 16 vectors of size 8 (vertical features) and 1 vector of size 64 (horizontal features). Parallely, features are preprocessed similarly than model 20b, but with an intermediate hidden size of 512 instead of 256.

As for model 24, it was constructed directly in a version with features, as at this point it was clear that the features increased the predictive capacity of the models. As seen in Fig. 4.20, this model's architecture is a 2 layer Bi-LSTM, where CNN attention layers are used to extract information from their outputs. In this sense, this architecture is a mix between model 13, where attention is extracted after every LSTM layer, and model 23b, where CNN attention is used. The CNN attention layers are applied only to the LSTM outputs and not the raw data. In this architecture, amino acids are converted to embeddings of size 20, and secondary structure to embeddings of size 4. These are concatenated, producing embeddings of size 24. A dropout of 0.35 is applied to the embeddings, which produces the regularization of the LSTM layers. The embeddings are passed through 2 consecutive Bi-LSTM layers. From their outputs, CNN attention layers with kernel size 7 are applied. Each CNN attention layer extracts 4 vertical features of size 24 and 1 horizontal feature of size 64. Parallely, the features are heavily compressed to 16 dimensions. When all features, including the frequencies, are concatenated, a vector of size 356 is obtained, which is passed through a fully connected layer of equal output and finally to the model's output.

Both models had outstanding metrics (Fig. 4.16 third and lower rows), where the model 23b obtained the best MSE of all models, while model 24 obtained the best MAE. Then, undersampling was applied to both models to obtain more representative predictions. The resulting models have the best weighted metrics of all the models (Fig. 4.6), with model 24 having the best WMAE and f1 of all models and model 23b having the best WMSE and WR^2.

Metric	Unweighted	Weighted
MAE	0.826	0.724
WMAE	1.015	0.778
MSE	1.258	0.863
WMSE	1.809	1.016
R2	0.652	0.761
WR2	0.566	0.762
F1	0.736	0.760

Table 4.7: Metrics of organism averages in the weighted and unweighted versions of model 24.

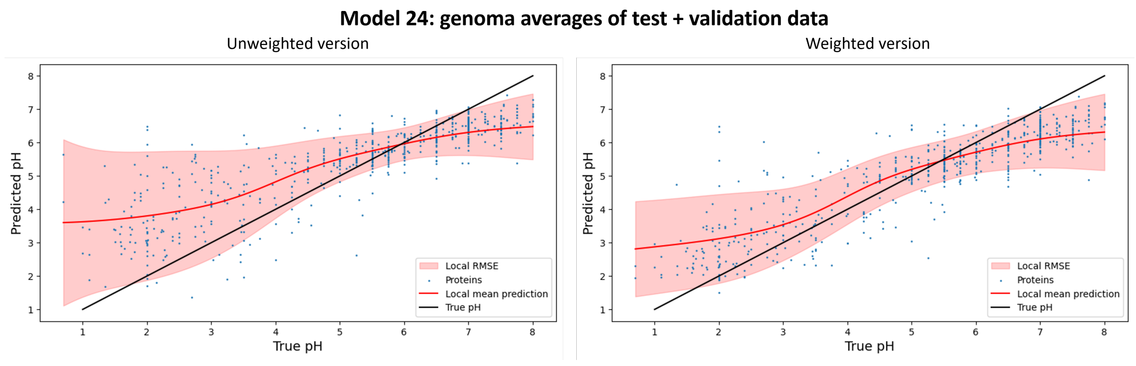

Fig 4.21: Genome average predictions for both the unweighted (left) and weighted (right) versions of model 24.

4.5 Organisms predictions

4.5.1 Model selection

Of all the architectures tested in this thesis, the best performing architecture in terms of MAE, WMAE and f1-score in both the weighted and unweighted versions was that of model 24. Between both version, a first selection was made by best metrics in genome average predictions in test + validation datasets (Fig. 4.21). The weighted version of the model had better performance on all metrics tested (Table 4.7).

4.5.2 Heuristic definition

Then, after selection of the best model and specific version of the model, several heuristics were tested for the best method to calculate the organism prediction. This was done in training data, and then applied to test + validation data, as a means to reduce the overfitting.

For the mode method, different σ values were explored. The optimal σ was selected with best MAE (Fig. 4.22). While using the median of the proteins predictions for the genome predictions had the best MAE in training data in comparison to the mode and mean, the mode had best performance according all other metrics, so it was given priority (Table 4.8). The heuristic was then applied to test and validation data to confirm the metric's performance. Indeed, the mode had a significantly better performance than using the mean according to all metrics (Table 4.7). The genome predictions using the mode of the proteins predictions with a $\sigma = 0.6$ had a mean average prediction error of about 0.609 pH units, a R^2 error of about 0.79, and an equivalent f1 score of 0.825.

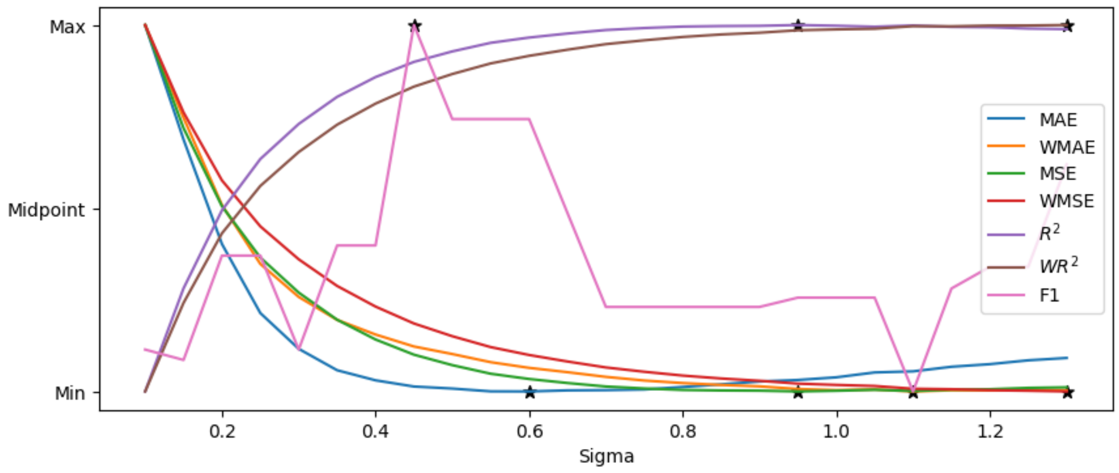

Fig 4.22: Selection of best σ hyperparameter for density calculation and mode prediction. For each metric, the min-max scaled metric is plotted and the best metric is shown with a black star.

Metric	Train - Mean	Train - Median	Train - Mode	Test + Val - Mode
MAE	0.379	0.367	0.369	0.609
WMAE	0.358	0.357	0.356	0.648
MSE	0.235	0.232	0.228	0.749
WMSE	0.214	0.221	0.215	0.903
R2	0.936	0.937	0.938	0.793
WR2	0.945	0.945	0.945	0.789
F1	0.91	0.908	0.914	0.825

Table 4.8: Organisms pH prediction metrics using mean, median and mode in train and test data.

Chapter 5

Discussion

The resulting model for the prediction of source organism's optimal growth pH from the proteins' amino acidic sequence was the result of an exhaustive search for the best machine learning model, where 6 variations of classical machine learning models and over 25 architectures of deep learning models were tested. The resulting model used a combination of features extracted with bioinformatic softwares, features extracted with deep learning tools, and the amino acidic sequence of the protein. The model's architecture was a combination of the architectures of the best performing models of each subfamily, containing LSTM, CNN and novel attention mechanisms for feature extraction.

While the genome predictions are satisfactory at least in terms of the weighted and unweighted R^2 metric, this thesis' objective of a MAE < 0.5 was not accomplished and there is still much room for improvement for both the genome and the individual protein predictions. In the idividual protein predictions in test data, the MAE is barely below 1 pH unit in all the models of this thesis. Considering the full range of the target variable is only 7 pH units and how the target variable is overrepresented around pH 5.5 and 7, these are poor results. If the mean of the target variable is used as a constant prediction, a MAE of 1.18 would be obtained, making this model not much better than a naive model.

The suboptimal performance and metrics of the best performing model of this thesis can be interpreted in at least 2 ways:

- The models lacked predictive capacity: A better model could have achieved much better results.
- The dataset had separability issues: The dataset and preprocessing methods are not clean enough.
- The problem has separability issues: It is just not possible to predict the pH resistance capacity of proteins.

The first hypothesis is plausible. While advanced deep learning architectures with high predictive capacity were used, these architectures were not the state-of-the-art for deep learning on proteins. Transformer based architectures were left out from this thesis' work for computational performance reasons. Transformer-based encoder models like ProtBERT were still used for feature extraction, and using these features as input of deep learning models is the technical equivalent of doing transfer learning with freezed weights. Perhaps a fine-tuning approach might have been the missing ingredient for a model that effectively predicts the pH of the proteins. However, this model is excesively large, as it took over 12 hours to just perform inference over the protein dataset (1 epoch), while in this thesis' trained models 1 training epoch took up to 15 minutes. Hence, this possibility was put aside for this thesis in favor of simpler and computationally efficient models. This possibility can be further developed in future work.

On the other hand, the second hypothesis has interesting possibilities. First, in this model it is assumed that all exposed proteins have the same optimal pH as the source organism, and that might not be true. Exposed proteins do not need to function optimally at the organism's optimal growth pH, they only need to resist it. By that extent, there could be proteins that share identical physichochemical features and have the same optimal pH, but can be present in organisms that grow optimally at different pHs as long as the protein can resist both pHs. For such case, the example proteins would not be separable. Moreover, exposed proteins are not necessarily expressed at the same time. Organisms that grow over a wide range of pH [115] could have exposed proteins that are preferentially expressed at different pH values. Such potential variation is not being addressed in this work, as only the optimal growth is being used for each organism. Additionally, the obtention of the optimal growth pH of an organism also has unavoidable errors. The ideal scenario would be a dataset of manually curated optimal pH of individual proteins, but such database was not found.

Another interesting possibility regarding this second hypothesis is related to the prediction of exposed proteins. In this work, the classification of proteins as ”exposed” was achieved by combining the output of 2 different subcellular localization prediction softwares: PSORTb and SignalP. Psortb [22] predicts subcellular localization by similarity-based algorithms, while SignalP [23] predicts the presence of a translocation signal peptide in the proteins. One or both methods could be prone to classification errors, producing a contamination of the dataset with cytoplasmic proteins which are not exposed to the pH that the organism is exposed to. Even so, some of the subcellular localizations considered in this work could have different forms of exposure to the external pH, or even no exposure.

To explore this last hypothesis, the main metrics for this thesis MAE and WMAE were calculated for each prediction of subcellular localization (Table 5.1) and signal peptide presence (Table 5.2). Effectively, proteins that were predicted to be exported (11.3 % of the proteins) had worse metrics than the rest of proteins. It is unknown why this was obtained. But there was a specially marked difference in the metrics by different signal peptide types. Lipoproteins exhibited the best metrics, while proteins without a signal peptide (11.9%) exhibited by far the worst metrics of all the categories here analyzed. This could mean that part of these proteins are actually cytoplasmic proteins which contaminate the dataset.

These possibilities of contamination, and optimal pH diversity by genome were taken into account in the training of models. Using mean squared error as a loss function gives a high penalty to outliers, which causes the predictions to adjust so all points are not-that-far of the target value. MAE,

Subcellular localization	N° of proteins in the test dataset (%)	MAE	WMAE
Unknown	22863 (69.5%)	0.923	1.04
Periplasm	4236 (12.5%)	0.937	1.006
Exported	3709(11.3%)	1.055	1.172
Outer membrane	1740 (5.3%)	0.924	0.956
Cell wall	364 (1.1%)	0.86	1.021

Table 5.1: Metrics obtained with the weighted version of model 24 by different subcellular localizations

Signal peptide type	N° of proteins in the test dataset (%)	MAE	WMAE
SP	18516 (56.3%)	0.92	1.004
LIPO	7270 (22.1%)	0.884	0.981
None	3574(10.9%)	1.152	1.239
TAT	3552 (10.8%)	0.933	1.178

Table 5.2: Metrics obtained with the weighted version of model 24 by different types of signal peptides

on the other hand, allows for outliers to have high errors as long as a high number of predictions is very close to the target value. This is probably why aggregating the protein predictions with mode had better results than using median and than using mean. With mode, the model only expects a high concentration of predictions around the actual value, even if there are numerous outliers with huge errors.

Weighted models had a much better performance than unweighted models in the organisms predictions according to both weighted and unweighted metrics. This is probably because acidophiles have much fewer proteins than the rest of the organisms (Fig. 4.1). Then, when aggregating by genome, the number of data points proportionally increases at low pH, and the dataset slightly balances itself, so models that have better predictive capacity at low pH have a better performance.

As it was mentioned, this work didn't achieve one of the objectives of this thesis of a MAE < 0.5 in the genome predictions. This could be improved by selecting only a few benchmark proteins for which the prediction is highly accurate and that are present in most or all organisms. This would simultaneously tackle the aforementioned issue of an *unclean* dataset. An alternative could be aggregating the proteins with different weights, using an additional simple model to learn the weights from the protein features.

Chapter 6

Conclusions and future work

The model here developed correspont to the first approach to predicting the acid resistance capacities of proteins. In turn, the organism prediction heuristics corresponds to the first protein-based approach to predict the acid resistance of an organism based solely on its genome and without relying on homology based methods. This could involve the development of a future tool that permits the detection of acidophiles from genomes of organisms that have never been grown in the laboratory (such as metagenomic reconstructions). The advantage of this method over homology-based methods is that it could detect acidophiles even if they are not simmilar to known acidophiles.

Future developments for the improvement of the current model include the use of simple transformer architectures and the cleaning of potential cytoplasmic proteins or other difficult-to-classify proteins which impair the learning capacity of the models, by eliminating proteins predicted to be exported and proteins without a signal peptide. Additionally, a classification approach could be implemented, where genomes of organisms with optimal pH growths of 4 - 6 are removed and the model only learns to distinguish from acidophiles and not acidophiles. While this would reduce the granularity of the prediction, it could potentially increase the predictor's performance. Additionally, fine-tuning of pre-trained transformer models could be performed as a next step for improved predictions. The organisms' prediction heuristic from the aggregation of individual proteins' predictions could also be improved by training a simple model that predicts if the protein would have high or low error rate. which would clean the dataset and also reduce inference costs.

Bibliography

[1] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. Journal of molecular biology, 215(3), 403-410.

[2] Artimo, P., Jonnalagedda, M., Arnold, K., Baratin, D., Csardi, G., De Castro, E., ... & Stockinger, H. (2012). ExPASy: SIB bioinformatics resource portal. Nucleic acids research, 40(W1), W597-W603.

[3] Baker-Austin, C., & Dopson, M. (2007). Life in acid: pH homeostasis in acidophiles. Trends in microbiology, 15(4), 165-171.

[4] Bateman, A., & Haft, D. H. (2002). HMM-based databases in InterPro. Briefings in bioinformatics, 3(3), 236-245.

[5] Chen, X. (2021). Thriving at Low pH: Adaptation Mechanisms of Acidophiles. Acidophiles - Fundamentals and Applications.

[6] Chi, A., Valenzuela, L., Beard, S., Mackey, A. J., Shabanowitz, J., Hunt, D. F., & Jerez, C. A. (2007). Periplasmic proteins of the extremophile Acidithiobacillus ferrooxidans: a high throughput proteomics analysis. Molecular & Cellular Proteomics, 6(12), 2239-2251.

[7] Cortez, D., Neira, G., González, C., Vergara, E., & Holmes, D. S. (2022). A large-scale genome-based survey of acidophilic bacteria suggests that genome streamlining is an adaption for life at low pH. Frontiers in Microbiology, 13, 803241.

[8] Duarte, F., Sepulveda, R., Araya, R., Flores, S., Perez-Acle, T., Gonzales, W., ... & Holmes, D. S. (2011). Mechanisms of protein stabilization at very low pH. In Proc. 19th International Biohydrometallurgy Symposium, Changsha, China (pp. 349-353).

[9] Ferruz, N., Schmidt, S., & Höcker, B. (2022). ProtGPT2 is a deep unsupervised language model for protein design. Nature communications, 13(1), 4348.

[10] Giovannoni, S. J., Cameron Thrash, J., & Temperton, B. (2014). Implications of streamlining

theory for microbial ecology. The ISME journal, 8(8), 1553-1565.

[11] Hidalgo, O., Pellicer, J., Christenhusz, M., Schneider, H., Leitch, A. R., & Leitch, I. J. (2017). Is there an upper limit to genome size?. Trends in Plant Science, 22(7), 567-573.

[12] Zhang, J. (2000). Protein-length distributions for the three domains of life. Trends in Genetics, 16(3), 107-109.

[13] Sheinerman, F. B., Norel, R., & Honig, B. (2000). Electrostatic aspects of protein–protein interactions. Current opinion in structural biology, 10(2), 153-159.

[14] Goldenberg, N. M., & Steinberg, B. E. (2010). Surface charge: a key determinant of protein localization and function. Cancer research, 70(4), 1277-1280.

[15] Chauhan, N. K., & Singh, K. (2018, September). A review on conventional machine learning vs deep learning. In 2018 International conference on computing, power and communication technologies (GUCON) (pp. 347-352). IEEE.

[16] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30.

[17] Dorugade, A. V. (2014). New ridge parameters for ridge regression. Journal of the Association of Arab Universities for Basic and Applied Sciences, 15, 94-99.

[18] Gupta, A., Sharma, A., & Goel, A. (2017). Review of regression analysis models. Int. J. Eng. Res. Technol, 6(08), 58-61.

[19] Bühlmann, P., & Yu, B. (2002). Analyzing bagging. The annals of Statistics, 30(4), 927-961.

[20] Segal, M. R. Machine Learning Benchmarks and Random Forest Regression. Info:.

[21] Loh, W. Y. (2011). Classification and regression trees. Wiley interdisciplinary reviews: data mining and knowledge discovery, 1(1), 14-23.

[22] Yu, N. Y., Wagner, J. R., Laird, M. R., Melli, G., Rey, S., Lo, R., ... & Brinkman, F. S. (2010). PSORTb 3.0: improved protein subcellular localization prediction with refined localization subcategories and predictive capabilities for all prokaryotes. Bioinform., 26(13), 1608-1615. doi: 10.1093/bioinformatics/btq249

[23] Almagro, J. J., Tsirigos, K. D., Sønderby, C. K., Petersen, T. N., Winther, O., Brunak, S., ... & Nielsen, H. (2019). SignalP 5.0 improves signal peptide predictions using deep neural networks. Nat. Biotechnol., 37(4), 420-423. doi: 10.1038/s41587-019-0036-z

[24] Parks, D. H., Imelfort, M., Skennerton, C. T., Hugenholtz, P., & Tyson, G. W. (2015). CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. Genome research, 25(7), 1043-1055.

[25] Moffat, L., & Jones, D. T. (2021). Increasing the accuracy of single sequence prediction methods using a deep semi-supervised learning framework. Bioinformatics, 37(21), 3744-3751.

[26] Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., ... & Rost, B. (2021). Prottrans: Toward understanding the language of life through self-supervised learning. IEEE transactions on pattern analysis and machine intelligence, 44(10), 7112-7127.

[27] Sonnhammer, E. L., Von Heijne, G., & Krogh, A. (1998, June). A hidden Markov model for predicting transmembrane helices in protein sequences. In Ismb (Vol. 6, pp. 175-182).

[28] Tusnady, G. E., & Simon, I. (2001). The HMMTOP transmembrane topology prediction server. Bioinformatics, 17(9), 849-850.

[29] Kramer, O., & Kramer, O. (2016). Scikit-learn. Machine learning for evolution strategies, 45-53.

[30] Pereira, J., & Saraiva, F. (2020, July). A comparative analysis of unbalanced data handling techniques for machine learning algorithms to electricity theft detection. In 2020 IEEE congress on evolutionary computation (CEC) (pp. 1-8). IEEE.

[31] Stevens, E., Antiga, L., & Viehmann, T. (2020). Deep learning with PyTorch. Manning Publications.

[32] Zhang, Z. (2018, June). Improved adam optimizer for deep neural networks. In 2018 IEEE/ACM 26th international symposium on quality of service (IWQoS) (pp. 1-2). Ieee.

[33] Baziotis, C. (2017, August 2). Self-attention (on words) and masking. PyTorch Forums. https://discuss.pytorch.org/t/self-attention-on-words-and-masking/5671/4

[34] Mayr, A., Binder, H., Gefeller, O., & Schmid, M. (2014). The evolution of boosting algorithms. Methods of information in medicine, 53(06), 419-427.

[35] Solomatine, D. P., & Shrestha, D. L. (2004, July). AdaBoost. RT: a boosting algorithm for regression problems. In 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541) (Vol. 2, pp. 1163-1168). IEEE.

[36] Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F., & Campbell, J. P. (2020). Introduction to machine learning, neural networks, and deep learning. Translational vision science & technology, 9(2), 14-14.

[37] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent advances in recurrent neural networks. arXiv preprint arXiv:1801.01078.

[38] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02), 107-116.

[39] Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. arXiv preprint arXiv:1402.1128.

[40] Jacovi, A., Shalom, O. S., & Goldberg, Y. (2018). Understanding convolutional neural networks for text classification. arXiv preprint arXiv:1809.08037.

[41] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems.

[42] Niu, Z., Zhong, G., & Yu, H. (2021). A review on the attention mechanism of deep learning. Neurocomputing, 452, 48-62.

[43] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[44] Letarte, G., Paradis, F., Giguère, P., & Laviolette, F. (2018, November). Importance of self-attention for sentiment analysis. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP (pp. 267-275).

[45] Mnih, V., Heess, N., & Graves, A. (2014). Recurrent models of visual attention. Advances in neural information processing systems, 27.

[46] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[47] Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. AI Open.

[48] Johnson, D., Goodman, R., Patrinely, J., Stone, C., Zimmerman, E., Donald, R., ... & Wheless, L. (2023). Assessing the accuracy and reliability of AI-generated medical responses: an evaluation of the Chat-GPT model. Research square.

[49] Acheampong, F. A., Nunoo-Mensah, H., & Chen, W. (2021). Transformer models for text-based emotion detection: a review of BERT-based approaches. Artificial Intelligence Review, 1-41.

[50] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[51] Liu, Y., Zhang, Y., Wang, Y., Hou, F., Yuan, J., Tian, J., ... & He, Z. (2023). A survey of visual transformers. IEEE Transactions on Neural Networks and Learning Systems.

[52] Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., & Salimans, T. (2022). Cascaded diffusion models for high fidelity image generation. The Journal of Machine Learning Research, 23(1), 2249-2281.

[53] Rampelotto, P. H. (2013). Extremophiles and extreme environments. Life, 3(3), 482-485.

[54] Neira, G., Cortez, D., Jil, J., & Holmes, D. S. (2020). AciDB 1.0: a database of acidophilic organisms, their genomic information and associated metadata. Bioinformatics.

[55] Foster, J. W. (2004). Escherichia coli acid resistance: tales of an amateur acidophile. Nature Reviews Microbiology, 2(11), 898-907.

[56] Vergara, E., Neira, G., González, C., Cortez, D., Dopson, M., & Holmes, D. S. (2020). Evolution of predicted acid resistance mechanisms in the extremely acidophilic Leptospirillum genus. Genes, 11(4), 389.

[57] Slonczewski, Joan L., et al. ”Cytoplasmic pH measurement and homeostasis in bacteria and archaea.” Advances in microbial physiology 55 (2009): 1-317.

[58] Ingledew, W. John. ”Thiobacillus ferrooxidans the bioenergetics of an acidophilic chemolithotroph.” Biochimica et Biophysica Acta (BBA)-Reviews on Bioenergetics 683.2 (1982): 89-117.

[59] Greaves, R. B., & Warwicker, J. (2009). Stability and solubility of proteins from extremophiles. Biochemical and biophysical research communications, 380(3), 581-585.

[60] Brininger, C., Spradlin, S., Cobani, L., & Evilia, C. (2018, December). The more adaptive to change, the more likely you are to survive: Protein adaptation in extremophiles. In Seminars in cell & developmental biology (Vol. 84, pp. 158-169). Academic Press.

[61] Tanford, C. (1968). Protein denaturation. Advances in protein chemistry, 23, 121-282.

[62] Wang, X. F., Gao, P., Liu, Y. F., Li, H. F., & Lu, F. (2020). Predicting thermophilic proteins by machine learning. Current Bioinformatics, 15(5), 493-502.

[63] Sawle, L., & Ghosh, K. (2011). How do thermophilic proteins and proteomes withstand high temperature?. Biophysical journal, 101(1), 217-227.

[64] Siddiqui, K. S., & Cavicchioli, R. (2006). Cold-adapted enzymes. Annu. Rev. Biochem., 75, 403-433.

[65] Yang, A. S., & Honig, B. (1993). On the pH dependence of protein stability. Journal of molecular biology, 231(2), 459-474.

[66] Reed, Christopher J., et al. ”Protein adaptations in archaeal extremophiles.” Archaea 2013 (2013).

[67] Talley, K., & Alexov, E. (2010). On the pH-optimum of activity and stability of proteins. Proteins: Structure, Function, and Bioinformatics, 78(12), 2699-2706.

[68] Lundblad, Roger L., and Fiona Macdonald, eds. Handbook of biochemistry and molecular biology. Crc Press, 2018.

[69] Michaux, C., Pouyez, J., Mayard, A., Vandurm, P., Housen, I., & Wouters, J. (2010). Structural insights into the acidophilic pH adaptation of a novel endo-1, 4-β-xylanase from Scytalidium acidophilum. Biochimie, 92(10), 1407-1415.

[70] Chen, X. (2021). Thriving at Low pH: Adaptation Mechanisms of Acidophiles. In Acidophiles-Fundamentals and Applications. IntechOpen.

[71] Sivashankari, S., & Shanmughavel, P. (2006). Functional annotation of hypothetical proteins–A review. Bioinformation, 1(8), 335.

[72] Higdon, R., Louie, B., & Kolker, E. (2010, June). Modeling sequence and function similarity between proteins for protein functional annotation. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (pp. 499-502).

[73] Söding, Johannes. ”Protein homology detection by HMM–HMM comparison.” Bioinformatics 21.7 (2005): 951-960.

[74] Mirceva, Georgina, and Dancho Davchev. ”HMM based approach for classifying protein structures.” International Journal of Bio-Science and Bio-Technology (2012).

[75] Nath, A., & Karthikeyan, S. (2020). Understanding the protein sequence and structural adaptation in extremophilic organisms through machine learning techniques. In Physiological and Biotechnological Aspects of Extremophiles (pp. 307-314). Academic Press.

[76] AlQuraishi, Mohammed. ”Machine learning in protein structure prediction.” Current opinion in chemical biology 65 (2021): 1-8.

[77] Bonetta, Rosalin, and Gianluca Valentino. ”Machine learning techniques for protein function

prediction.” Proteins: Structure, Function, and Bioinformatics 88.3 (2020): 397-413.

[78] Audain, Enrique, et al. ”Accurate estimation of isoelectric point of protein and peptide based on amino acid sequences.” Bioinformatics 32.6 (2016): 821-827.

[79] Kessel, A., & Ben-Tal, N. (2018). Introduction to proteins: structure, function, and motion. Crc Press.

[80] Nasko, D. J., Koren, S., Phillippy, A. M., & Treangen, T. J. (2018). RefSeq database growth influences the accuracy of k-mer-based lowest common ancestor species identification. Genome biology, 19(1), 1-10.

[81] Fang, Yaping, C. Russell Middaugh, and Jianwen Fang. ”In silico classification of proteins from acidic and neutral cytoplasms.” (2012): e45585.

[82] Kozlowski, L. P. (2021). IPC 2.0: prediction of isoelectric point and p K a dissociation constants. Nucleic acids research, 49(W1), W285-W292.

[83] Fout, A., Byrd, J., Shariat, B., & Ben-Hur, A. (2017). Protein interface prediction using graph convolutional networks. Advances in neural information processing systems, 30.

[84] Shi, Z. (2022). Graph neural networks and attention-based CNN-LSTM for protein classification. arXiv preprint arXiv:2204.09486.

[85] Susanty, M., Hertadi, R., Purwarianti, A., & Rajab, T. L. E. (2022). Low Complexity Classification of Thermophilic Protein using One Hot Encoding as Protein Representation. International Journal of Advanced Computer Science and Applications, 13(12).

[86] Cheng, **yong, Yihui Liu, and Yuming Ma. ”Protein secondary structure prediction based on integration of CNN and LSTM model.” Journal of Visual Communication and Image Representation 71 (2020): 102844.

[87] Jumper, John, et al. ”Highly accurate protein structure prediction with AlphaFold.” Nature 596.7873 (2021): 583-589.

[88] Lee, Jooyoung, Peter L. Freddolino, and Yang Zhang. ”Ab initio protein structure prediction.” From protein structure to function with bioinformatics (2017): 3-35.

[89] Vig, J., Madani, A., Varshney, L. R., Xiong, C., Socher, R., & Rajani, N. F. (2020). BERTology meets biology: interpreting attention in protein language models. arXiv preprint arXiv:2006.15222.

[90] Gauthier, J., Vincent, A. T., Charette, S. J., & Derome, N. (2019). A brief history of bioinfor-

matics. Briefings in bioinformatics, 20(6), 1981-1996.

[91] Alberts, B., Bray, D., Hopkin, K., Johnson, A. D., Lewis, J., Raff, M., ... & Walter, P. (2015). Essential cell biology. Garland Science.

[92] Schlegel, H. G., & Zaborosch, C. (1993). General microbiology. Cambridge university press.

[93] Champe, P. C., Harvey, R. A., & Ferrier, D. R. (2005). Biochemistry. Lippincott Williams & Wilkins.

[94] Maturana, H. R., & Varela, F. J. (1991). Autopoiesis and cognition: The realization of the living (Vol. 42). Springer Science & Business Media.

[95] Travers, A., & Muskhelishvili, G. (2015). DNA structure and function. The FEBS journal, 282(12), 2279-2295.

[96] Cox, M. M., Doudna, J. A., & O'Donnell, M. (2012). Molecular biology: principles and practice (p. 809). New York, NY. USA:: WH Freeman and Company.

[97] Brooker, R. J. (1999). Genetics: analysis & principles. Reading, MA, USA:: Addison-Wesley.

[98] Wang, L., & Schultz, P. G. (2005). Expanding the genetic code. Angewandte Chemie International Edition, 44(1), 34-66.

[99] Lehrman, S. R. (2017). Protein structure. Fundamentals of protein biotechnology, 9-38.

[100] Janin, J., Bahadur, R. P., & Chakrabarti, P. (2008). Protein–protein interaction and quaternary structure. Quarterly reviews of biophysics, 41(2), 133-180.

[101] Pirovano, W., & Heringa, J. (2010). Protein secondary structure prediction. Data Mining Techniques for the Life Sciences, 327-348.

[102] Ponting, C. P., & Russell, R. R. (2002). The natural history of protein domains. Annual review of biophysics and biomolecular structure, 31(1), 45-71.

[103] Ghélis, C. (2012). Protein folding. Academic Press.

[104] Krieger, E., Nabuurs, S. B., & Vriend, G. (2003). Homology modeling. Structural bioinformatics, 44, 509-523.

[105] Pál, C., Papp, B., & Lercher, M. J. (2006). An integrated view of protein evolution. Nature reviews genetics, 7(5), 337-348.

[106] Vogt, G., Woell, S., & Argos, P. (1997). Protein thermal stability, hydrogen bonds, and ion pairs. Journal of molecular biology, 269(4), 631-643.

[107] Eisenberg, D., Marcotte, E. M., Xenarios, I., & Yeates, T. O. (2000). Protein function in the post-genomic era. Nature, 405(6788), 823-826.

[108] Xia, X. (2007). Protein isoelectric point. Bioinformatics and the Cell: Modern Computational Approaches in Genomics, Proteomics and Transcriptomics, 207-219.

[109] Salgado, J. C., Rapaport, I., & Asenjo, J. A. (2005). Is it possible to predict the average surface hydrophobicity of a protein using only its amino acid composition?. Journal of Chromatography A, 1075(1-2), 133-143.

[110] Pace, C. N., Fu, H., Fryar, K. L., Landua, J., Trevino, S. R., Shirley, B. A., ... & Grimsley, G. R. (2011). Contribution of hydrophobic interactions to protein stability. Journal of molecular biology, 408(3), 514-528.

[111] Chanphai, P., Bekale, L., & Tajmir-Riahi, H. A. (2015). Effect of hydrophobicity on protein–protein interactions. European Polymer Journal, 67, 224-231.

[112] Rees, D. C., DeAntonio, L., & Eisenberg, D. (1989). Hydrophobic organization of membrane proteins. Science, 245(4917), 510-513.

[113] Goebel, B. M., Norris, P. R., & Burton, N. P. (2000). Acidophiles in biomining. In Applied microbial systematics (pp. 293-314). Dordrecht: Springer Netherlands.

[114] Valdés, J., Pedroso, I., Quatrini, R., Dodson, R. J., Tettelin, H., Blake, R., ... & Holmes, D. S. (2008). Acidithiobacillus ferrooxidans metabolism: from genome sequence to industrial applications. BMC genomics, 9, 1-24.

[115] Dhakar, K., & Pandey, A. (2016). Wide pH range tolerance in extremophiles: towards understanding an important phenomenon for future biotechnology. Applied microbiology and biotechnology, 100, 2499-2510.

[116] Zhang, T., Lin, W., Vogelmann, A. M., Zhang, M., Xie, S., Qin, Y., & Golaz, J. C. (2021). Improving convection trigger functions in deep convective parameterization schemes using machine learning. Journal of Advances in Modeling Earth Systems, 13(5), e2020MS002365.

[117] Soldera, B. (2023, January 5). Mind the graph and 3D protein imaging. Mind the Graph Blog. https://mindthegraph.com/blog/mind-the-graph-and-3d-protein-imaging-2/

[118] Cornell, B. (n.d.). Amino Acids. BioNinja. https://ib.bioninja.com.au/standard-level/topic-2-molecular-biology/24-proteins/amino-acids.html

[119] Szydlowski, M. (2017, May 24). Extremophiles. Columbia Daily Tribune. https://www.columbiatribune.com/story/lifestyle/family/2017/05/24 /extremophiles/20804947007/

[120] GeeksforGeeks. (2023, August 8). Bacteria - definition, structure, diagram, Classification. GeeksforGeeks. https://www.geeksforgeeks.org/bacteria/

[121] Chaya. (2022, April 14). Random Forest regression. Medium. https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

[122] What are neural networks?. IBM. (n.d.). https://www.ibm.com/topics/neural-networks

[123] Recurrent neural network. AILEPHANT. (2018, July 11). https://ailephant.com/glossary/recurrent-neural-network/

[124] K., P. (2021, April 20). Week 7 - creating initial LSTM model. BASIS Independent Silicon Valley. https://siliconvalley.basisindependent.com/2021/04/09/week-7-creating-initial-lstm-model/

[125] Brownlee, J. (2020, September 2). How to develop a multichannel CNN model for text classification. MachineLearningMastery.com. https://machinelearningmastery.com/develop-n-gram-multichannel-convolutional-neural-network-sentiment-analysis