PFAM DATASET CLASSIFICATION

Lea Boulos

Detailed Report
20th of May 2023

Table of Contents

Introduction

Two fundamental biological principles can shed light on the complex workings of living organisms. Firstly, DNA serves as the repository or blueprint of genetic information for all living organisms. This remarkable molecule governs the development, growth, functionality, and reproductive processes of all known life forms. It encodes organisms' physical attributes, biochemical reactions, and inherited traits. Secondly, proteins are indispensable building blocks of life, and are ubiquitous in the body, and make up enzymes, hormones, muscles, and bones, etc.

The synthesis of proteins occurs within the cytoplasm, where linear chains of amino acids are assembled according to the instructions encoded in DNA. Among the repertoire of amino acids, there exist 20 common types, while four others exhibit relative rarity. The linear sequence of amino acids, referred to as the primary structure, serves as the foundational basis for protein formation. Proteins possess four primary levels of structure: primary, secondary, tertiary, and quaternary. The secondary structure arises from local patterns, such as alpha helices and beta sheets, forged through hydrogen bonding between amino acid residues. Tertiary structure, on the other hand, encompasses the overall folding and arrangement of secondary structure elements, culminating in the creation of a compact globular shape. Finally, quaternary structure manifests when multiple protein subunits congregate to form intricate functional complexes, further amplifying their biological significance.

The intricate three-dimensional arrangement of amino acids within a protein, known as its structure, is a key determinant of its function. The arrangement of amino acids and the different biochemical interactions between them, under homeostatic conditions, determines the final 3D structure of the protein, **which, in turn, governs the macromolecule's biochemical and biological properties**. The relationship between protein structure and function is intimately connected. Proteins function by interacting with other molecules, such as enzymes catalyzing chemical reactions or receptors binding to signaling molecules. The precise position of amino acid residues within the protein structure enables these interactions and unlocks the protein's functionality. Hence, understanding protein structure is crucial, as it allows us to decipher protein function, predict its behavior, and unravel the underlying mechanisms of biological processes.

Proteins are clustered in families if they share a common evolutionary origin and possess similar structural and functional characteristics. Proteins within the same family typically exhibit sequence similarities and perform related biological activities, often serving common roles or participating in similar biochemical processes. Given the growing number of protein sequences discovered through genomic sequencing projects, experimental characterization of each protein's function becomes a monumental challenge. Here, protein structure classification plays a crucial role. By classifying proteins based on their structural features, we can infer their potential functions and identify similarities between proteins sharing common structural motifs.

This can be done based on **homology**, a fundamental concept in protein structure classification. Proteins sharing a common ancestor tend to have similar structures and functions. Due to evolutionary conservation, homologous proteins possess structural similarities. Thus, by identifying homologous proteins, we can transfer functional annotations from well-studied proteins to uncharacterized ones, providing valuable insights into their potential roles in biological systems. One widely used tool is the Basic Local Alignment Search Tool (**BLAST**). BLAST compares a query protein sequence against a database of known protein sequences to identify homologous proteins and infer functional annotations. BLAST utilizes algorithms that detect sequence similarities, which often correlate with structural similarities.

In recent years, however, the advent of deep learning has revolutionized the field of protein structure classification and function prediction. Alphafold, a Deep Learning algorithm by DeepMind, which uses

transformer-based architecture and attention, was able to solve one of the most central and challenging biological questions by predicting protein structure with astounding accuracy. This further proves the essential role that Artificial intelligence can play in advancing the biological and medical fields. Deep Learning algorithms learn from large datasets and can predict protein functions with remarkable accuracy and offer advantages over traditional methods by capturing complex non-linear relationships and extracting high-level features from protein sequences and structures.

The use of deep learning in protein structure classification has several advantages. Firstly, deep learning models can handle large-scale datasets efficiently, allowing for comprehensive analysis of protein sequence and structural information. Secondly, such models can learn representations of proteins without solely relying on pre-defined features, thus facilitating the discovery of new functional patterns. Lastly, deep learning models can provide a holistic understanding of protein function by integrating various types of biological data, such as genomic, proteomic, and structural data.

Deep learning offers remarkable prospects by building models that learn and extract useful and relevant. These can be shared or unique between protein families. This information can then be leveraged to predict the annotations from unaligned input sequences. (Maxwell L. Bileschi, 2022) This opens new possibilities for advancing our understanding and analysis of complex biological data. Hence, the goal of this project is to build a classifier that, given the amino acid sequence of a protein, is able to predict the protein family class to which it belongs.

The Dataset

We are provided with the Pfam dataset. It is a large protein database, constituted of over 1 million small and manually curated seed sets of roughly 18,000 protein domain sequences. Each protein constitutes a row, and its features are as follows:

- **sequence:** the linear sequence of amino acids that make up the protein. This would be the **input** feature to the model. The sequence is a string made up of the 20 common amino acids, each represented by a character (frequency > 1,000,000), and 4 amino acids that are quite uncommon: X, U, B, O, Z.
- **family_accession**: this is the family class or label to which the protein sequence belongs, in string format, and constitutes the output of the model. Accession number in form PFxxxxx.y (Pfam), where xxxxx is the family accession, and y is the version number. Some values of y are greater than ten, and so 'y' has two digits.
- **family_id**: a string representing the name of the protein family.
- **sequence_name**: sequence name, in string form "$uniprot_accession_id/$start_index-$end_index".
- **aligned_sequence**: contains a single sequence from the multiple sequence alignment of the different proteins of that family against this protein sequence.

The data has already been split into 3 different sets:

- The training set: the dataset provided to the model to learn from and train on. This set constitutes roughly 80% of the data. It contains 1,086,741 rows, covering all different 17,929 family labels.
- The test set: this is the dataset on which the model's classification performance is evaluated and makes up approx. 10% of the data. It's used to assess the model's ability to generalize and reflect how well it learned the underlying structure of the data.

- The validation set: also called held-out or dev set, is the dataset on which the hyperparameters of the model are tuned. It's also approx. 10% of the data.

A closer exploration of the data suggests that the test and dev sets statistics are quite similar, with the same number of sequences in each and same number and similar label distribution over sequence length. The split has carefully been done to include all possible family IDs in the data represented in the training set. In other words, 100% of the family labels that appear in the held-out and test set appear also in the training set, which helps improve the generalizability of the subsequent model in the later stages of the project.

The most frequent family in all 3 splits is *Methyltransf_25*, with 3,637 sequences in the training set and 454 sequences in the test and dev set, respectively. But given the volume of the data, the sequences belonging to this family are a only about 0.33% of the training set and 0.35% of the test and held-out set.

	Count	Count_unique	Percent
train	1086741	17929	81.16
test	126171	13071	9.42
dev	126171	13071	9.42

Table 1: Count of all sequences, unique set of sequences and the percentage of present sequences in each of the splits.

The representation of the families in the entire dataset is not balanced. If we combine all 3 datasets and plot the count of occurrence of each class, we notice a huge discrepancy between the topmost frequent and the 1000th most frequent, as depicted in Table 2 and Figure 1. This might bias the learning later on and should be considered when assessing and/or building the model. In the event that a model without class weights is considered, the performance shouldn't be assessed solely based on the accuracy but should also consider the model's precision and recall metrics to account for the imbalance in the classes in the data.

	Count of top ith Family
1	4545
10	1512
50	1017
100	884
150	780
200	731
250	675
500	461
1000	284

Table 2: Count of sequences in the ith family

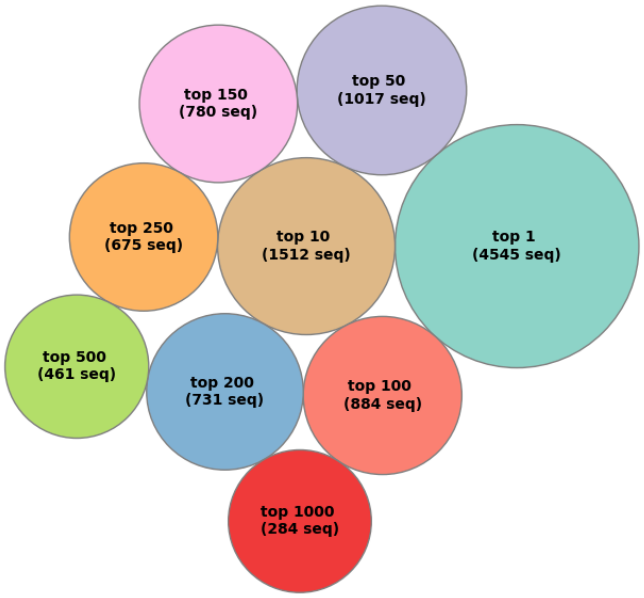


Figure 1: Bubble Chart representing some of the 1000th most frequent protein families in the training set.

Under-sampling

Class imbalance is not uncommon in biology. Many diseases have low prevalence and consequently, cases in a cohort study are less abundant than controls. Similarly, proteins exhibit a wide range of functions, some of which may be more prevalent in nature than others. Certain enzymatic activities or structural functions may be more common compared to rare or specialized functions. This natural imbalance in protein functionality can be translated into imbalanced datasets when collecting protein annotations. Not to mention research bias, which may result in heavier, more extensive research into certain protein functions compared to others, leading to an imbalanced representation of understudied proteins.

Not only is the current dataset imbalanced but contains a very large number of labels. A model that learns to predict the probability of a protein class out of ~18,000 classes would require a massive number of trainable parameters, is susceptible to overfitting and requires considerable computational power to accommodate the depth it needs to learn and extract useful features. Not to mention the time and space complexity needed for the *softmax* function to predict such a probability distribution, which could be unfeasible.

There are many solutions to class imbalance, such as under-sampling, oversampling or augmentation, the latter being more relevant to computer vision tasks. However, we posit that the best approach for now would be to simply decrease the size of the dataset by retaining only the top 100 most frequent class labels that are present in all 3 splits. These labels would have the advantage of rich representation, providing the model with enough examples to learn their features, and would result in a much smaller dataset with 100 labels and 92,983 sequences, or approx. 0.09% the size of the original dataset.

The new training set has varying sequence lengths across classes, ranging from 10 to 500 amino acids per sequence, with a small variation of sequence length per family (Figure 2). The sequence length distribution is right skewed, with most sequences having between 80 and 150 amino acids, and the mean and median values falling between 110 and 130. 95% of the sequences have lengths of 300 or less. Thus, when setting the max length of the sequences for truncating and padding, the length of 300 is a good option.

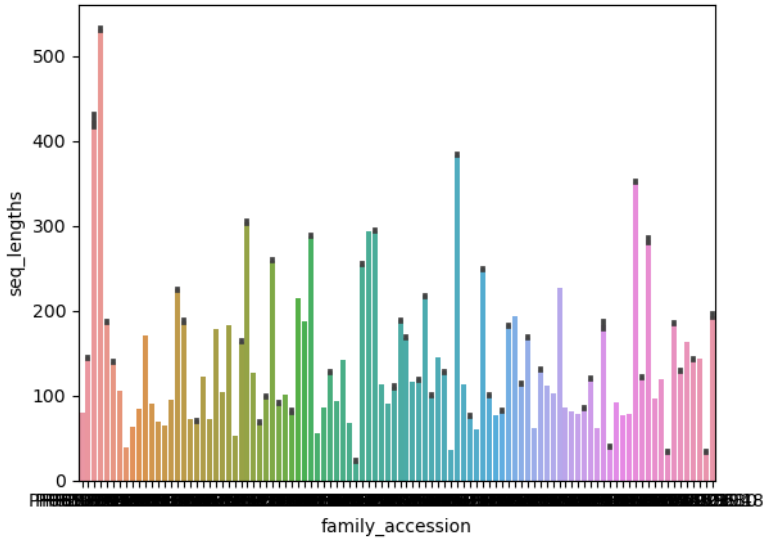
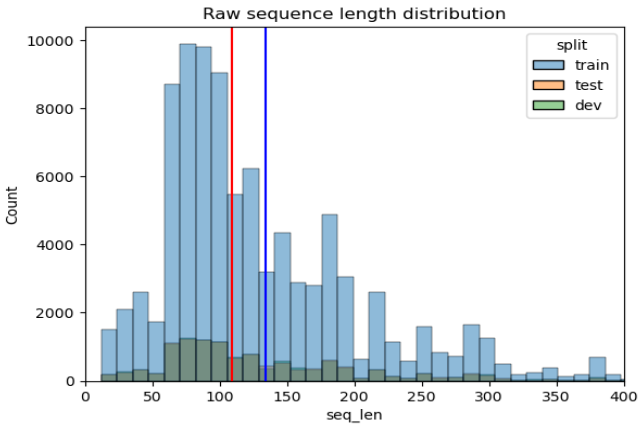


Figure 2: Barplot displaying the average sequence length per family in the data. The black part represents the variation of length within the family.

(a)



(b)

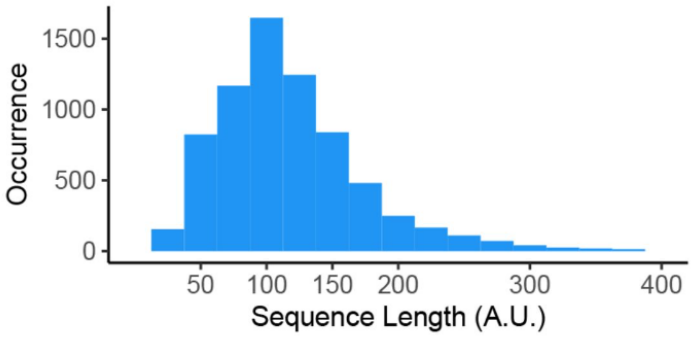


Figure 3: (a) Histogram of the sequence length distribution in the new smaller training set. The blue line represents the mean, and the red line the median. (b) Sequence length distribution in (Kamil Tamiola, 2017).

The amino acids occurrence distribution is not uniform (Figure 3.a). The trend observed in our dataset is almost the same as what is observed in the literature before (Figure 3.b). Other protein databases generate the same distribution (Kamil Tamiola, 2017). In both cases, the most common amino acid is Leucine 'L', Alanine 'A' and Glycine 'G', and the least abundant are Cysteine 'C' and Tryptophan 'W'. It's worth noting that, in the new reduced training set, only 3 out of the 5 rare amino acids are represented. These are 'U' 'X' and 'B' (no sequences with the amino acid 'O' and 'Z')

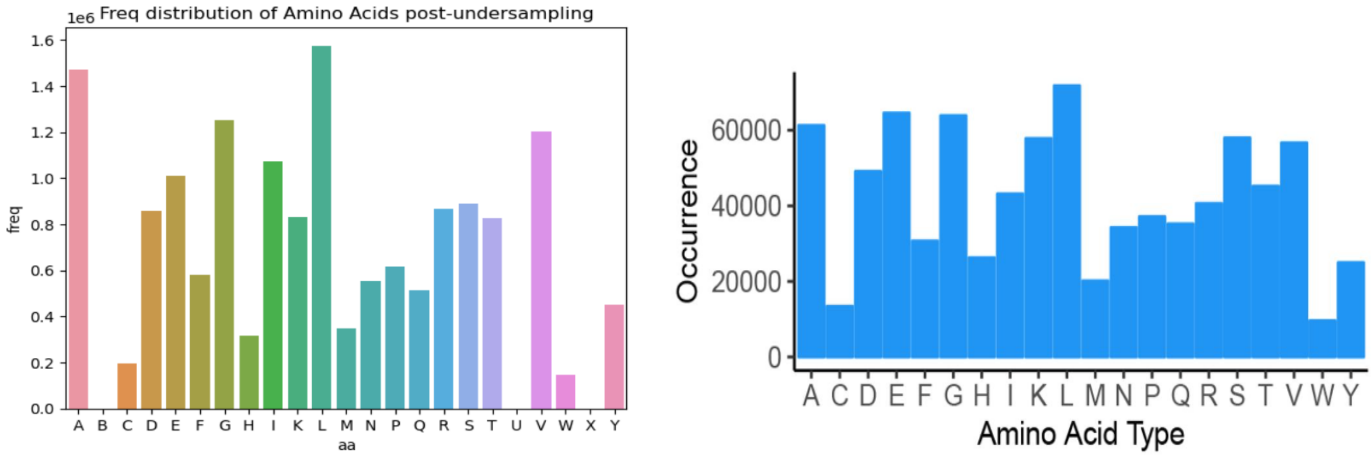


Figure 4: Amino -acid occurrence frequency in (a) the entire dataset (b) in (Kamil Tamiola, 2017)

Preprocessing

Input Padding

Padding and one-hot encoding are common preprocessing techniques used in deep learning for handling variable-length sequences or categorical data, respectively.

Since the sequences at hand vary in length, padding is necessary for efficient processing to make all sequences of uniform length. Padding involves adding extra elements (usually zeros) to the sequences so that they all have the same length. In case the sequence is larger than the input size, it is truncated to the maximum length.

The padding ensures that all sequences have the same length, enabling efficient batch processing in deep learning models. The padded elements are typically ignored during computations.

One-Hot Encoding

One-hot encoding is a technique used to represent categorical variables or labels as binary vectors in deep learning. In this encoding scheme, each category is represented as a binary vector where only one element is "hot" (1), indicating the presence of that category, while all other elements are "cold" (0).

The length of the binary vector is equal to the number of categories. Each category is assigned a unique index, and the corresponding element in the binary vector is set to 1.

One-hot encoding allows the model to interpret categorical data as numerical inputs, enabling mathematical operations and efficient computation. Here, one-hot encoded vectors of the amino acids are fed as input to neural networks, and one-hot encoded family labels are used to represent the 100 target classes at the output layer.

Both padding and one-hot encoding are preprocessing steps that transform the input data and/or classes into a suitable format for the model. Padding ensures uniform sequence lengths, while one-hot encoding converts categorical variables into binary vectors, enabling efficient computations, such as the *softmax* calculations for predicting the probability of each label at the output layer.

Methods

Baseline model – Embedding

In machine learning, embedding refers to the process of representing (or encoding) categorical variables, such as words or discrete entities, as continuous vectors in a lower-dimensional space. It is a technique used to transform high-dimensional and discrete data into a continuous representation, and is used in many deep learning models such as the Encoder-Decoder models.

The embedding vectors capture semantic relationships and contextual information between the different elements of a sequence. Positional embedding, as explained above, encodes positional information. The addition of these two embeddings can then be forwarded to a couple of dense/dropout layers. These transformations reduce the dimension gradually until the last classification layer is reached. Such a model constitutes a good baseline, such that more complex and deeper models based on the architectures discussed above can be compared to it, or extrapolated from it.

Embeddings have several advantages. They enable the representation of categorical variables in a continuous and meaningful way, facilitating the use of traditional numerical-based machine learning algorithms. They also help capture latent relationships and similarities in the data, providing richer feature representations for downstream tasks. In NLP, word embeddings have proven to be valuable in various applications, including text classification, sentiment analysis, machine translation, and information retrieval.

Transformers

The Transformer architecture is a deep learning model that has revolutionized natural language processing (NLP) tasks, particularly in machine translation and language understanding. It was introduced in the paper "Attention is All You Need" (Ashish Vaswani, 2017). The Transformer model relies on self-attention mechanisms to capture relationships between words in a sentence, while also enabling efficient parallel computation and avoiding vanishing gradient problems. Parallel computations and superior performance are the two main advantages of Transformers over other architectures such as traditional Recurrent Neural Networks (RNNs),

At its core, the Transformer architecture consists of an encoder-decoder structure with multiple layers. Each layer in the encoder and decoder consists of two main sub-modules: the self-attention mechanism and the feed-forward neural network.

Feed-forward neural networks are traditional fully connected neural networks. The self-attention mechanism, as the name implies, allows the model to focus on different parts of the input sequence during processing. It computes attention weights for each word in the sequence based on its relationships with all other words in the sequence. The attention weights reflect the importance of each word in the context of the entire sentence. This mechanism helps the model capture long-range dependencies and contextual information efficiently, information that is lost in convolutions but is crucial in many sequence-based tasks such as language processing and modeling.

The encoder and decoder layers in the Transformer architecture are connected via a series of attention mechanisms. The encoder processes the input sequence and generates contextualized representations for each word. The decoder takes these representations and generates an output sequence, word by word, by attending to relevant parts of the encoded input sequence.

But since the location of a word (in this case, the amino acid) is an essential feature in sequence-based tasks that is lost in convolution, positional embeddings are added to the input embeddings, as described in the previous section. These encodings provide the model with information about the position of each word in the sequence.

Results

Metrics

Accuracy

Accuracy is a metric that measures the overall correctness of the model's predictions. It calculates the proportion of correctly predicted instances (both positive and negative) among all instances. Accuracy provides a general view of the model's performance but may be misleading when the classes are imbalanced.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Where:

- TP = True Positives (the number of correctly predicted positive instances)
- TN = True Negatives (the number of correctly predicted negative instances)
- FP = False Positives (the number of incorrectly predicted positive instances)
- FN = False Negatives (the number of incorrectly predicted negative instances)

Recall

Recall measures the proportion of correctly predicted positive instances among all actual positive instances. It reflects the ability of the model to correctly capture positive cases, indicating how well it avoids missing positive instances.

Recall = TP / (TP + FN)

Precision

Precision measures the proportion of correctly predicted positive instances among all instances predicted as positive. It focuses on the accuracy of positive predictions, indicating how precise the model is in identifying positive cases.

Precision = TP / (TP + FP)

Baseline model – Embedding.

As discussed in the previous section, we started with a basic embedding model. The model summary is displayed below:

```
Model: "pos_embedding_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 position_embedding_layer (P  multiple                 10368
 ositionEmbeddingLayer)

 dropout (Dropout)           multiple                  0

 flatten (Flatten)           multiple                  0

 dense (Dense)               multiple                  1228928

 dropout_1 (Dropout)         multiple                  0

 dense_1 (Dense)             multiple                  12900

=================================================================
Total params: 1,252,196
Trainable params: 1,252,196
Non-trainable params: 0
_________________________________________________________________
```

Despite its simplicity, the performance of this model is remarkably impressive from a technical standpoint. After only 7 epochs of training, the model achieved an outstanding validation accuracy of 99.6%. While accuracy alone can sometimes be deceptive as a metric, it is crucial to evaluate the model based on various criteria to gain a comprehensive understanding of its performance. Consequently, both the recall and precision metrics serve as valuable indicators, further confirming the model's remarkable fitness with validation scores of 0.99 each.

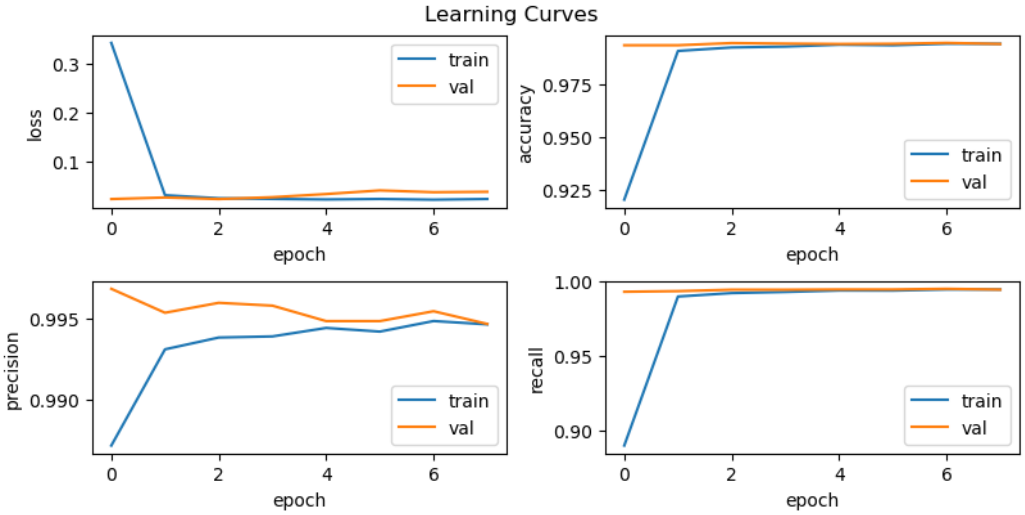


Figure 5: Learning curves of the baseline embedding model. Loss, Accuracy, precision, recall in train and validation are plotted.

Figure 5 illustrates the model's performance, depicting its efficacy even as early as the first epoch, followed by a subsequent plateauing of its learning curve by the conclusion of the second epoch. The evaluation of the model on the test set further supports this conclusion, with test accuracy reaching an impressive 99.4%. Moreover, the validation precision and recall scores of 0.995 and 0.993, respectively, highlight its ability to generalize effectively over previously unseen data. These results collectively demonstrate not only the model's excellent fit but also its capacity to extrapolate its learned knowledge to new and unseen instances.

Transformer Model

Transformers are bulky models. When testing the compatibility of this architecture to the problem, we opted for simple model with only one transformer block and a feedforward layer of size 32.

```
Model: "protein_transformer_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 position_embedding_layer_3   multiple                 10368
 (PositionEmbeddingLayer)

 transformer_block (Transfor  multiple                 10656
 merBlock)

 global_average_pooling1d (G  multiple                 0
 lobalAveragePooling1D)

 dropout_4 (Dropout)         multiple                  0

 dense_4 (Dense)             multiple                  3300

=================================================================
Total params: 24,324
Trainable params: 24,324
Non-trainable params: 0
_________________________________________________________________
```

When designing this experiment, we held high expectations for the Transformer model, given its reputation for excelling in diverse natural language processing tasks. However, surprisingly, the Transformer model exhibited the poorest performance among the three models explored. It demanded a considerable amount of time, surpassing the 30-minute mark, to train adequately and achieved a moderate validation accuracy of 96%. Similarly, the recall and precision metrics of the model also failed to demonstrate impressive results, attaining values of 0.95 and 0.96, respectively. Given it's training time, we did not further explore potential fine-tuning, as the baseline performance in terms of speed and efficiency is more promising.

These outcomes imply that the mere incorporation of increased complexity does not invariably yield optimal outcomes when tackling intricate problems. The poor performance could stem from over-parametrization, where the number of trainable parameters in the model potentially exceed the needs and characteristics of the provided dataset. This revelation highlights the significance of striking a delicate balance between model complexity and dataset constraints to attain optimal performance in classification tasks.

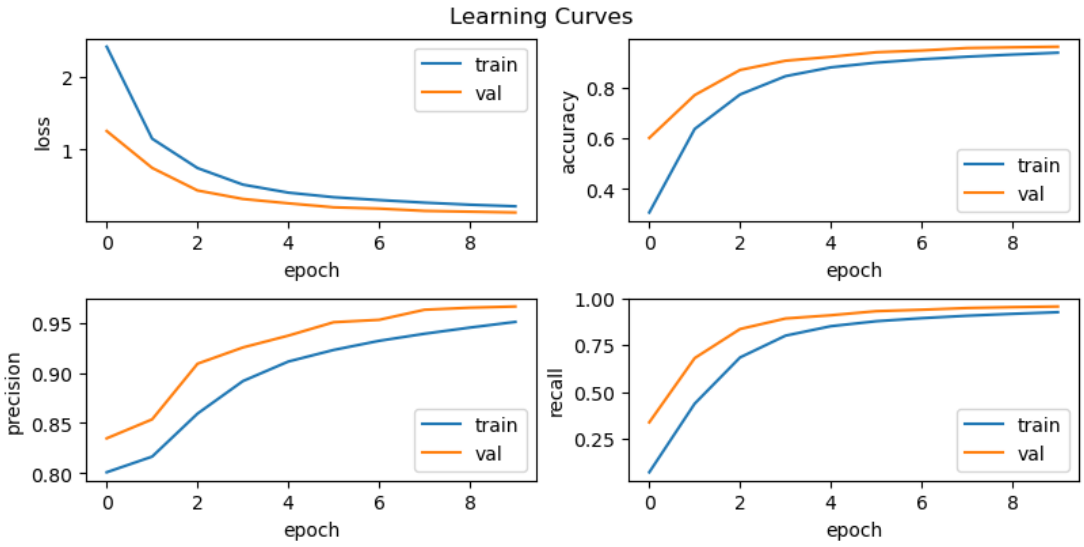


Figure 6 Learning curves of the baseline embedding model. Loss, Accuracy, precision, recall in train and validation are plotted.

Discussion

Occam's Razor stipulates that simpler explanations or hypotheses are generally preferable to more complex ones. Hence, the embedding model, which also resulted in the best performance, is the best fit for the classification task at hand.

However, the evaluation and validation of deep learning models applied to biological data for addressing biological problems necessitate meticulous examination. Despite their reputation as black boxes, it is important to delve into the decision-making processes of the model. While conventional machine learning metrics and scores offer initial insights into their efficacy, it is necessary to conduct a comprehensive problem-relevant assessment to establish their credibility.

One promising approach in this context involves visualizing the embeddings generated by the model on the test set. This step is essential to verify the embeddings' ability to accurately represent the underlying data in a lower-dimensional space. We leveraged the dimensionality reduction technique known as t-distributed Stochastic Neighbor Embedding (tSNE) to assess the extent to which the model has successfully captured the distinct label-specific features. We would expect that sequences belonging to the same family exhibit a propensity to cluster together, signifying the model's ability in learning family-specific characteristics.

Interestingly, the tSNE plots, depicted in Figure 7 and 8, validate this expectation, as they demonstrate a compelling tendency for most families to form distinct clusters. This clear segregation among the majority of families further demonstrates the model's ability to extract and encapsulate the relevant features that distinguish one family from another. Consequently, these findings significantly contribute to reinforcing the model's validity and further validate its capacity to effectively address this problem.

Another interesting way to further evaluate the model, which is outside the scope of this project, is to extract the learned embeddings and compute a similarity matrix of learned amino acid representations, as demonstrated in (Maxwell L. Bileschi, 2022). The resulting matrix should resemble the one of the BLOSUMX (BLOcks SUbstitution Matrix) matrices. These matrices quantify the likelihood of amino acid substitutions based on the observed frequencies of substitutions in a database of aligned protein sequences. One would expect the learned frequencies of substitution to be captured by the model as well.

In order to address the challenge posed by the substantial number of labels associated with the ~18,000 protein families, the construction of a classifier capable of accommodating such diversity necessitates the utilization of a larger and more complex architecture, trained on the entirety of the original dataset. Furthermore, the modification of the final classification output layer becomes indispensable, since the *softmax* function is unable to adapt to such a large number of labels. One potential approach involves the development of a comprehensive classifier based on broader protein classes, wherein the protein families are grouped together based on their orthology or biological functions, effectively reducing the number of distinct classes. Subsequently, each cluster would be assigned its own classifier responsible for discriminating between the various families predicted to belong to that particular cluster. It is evident that the adoption of such a strategy requires more computational power and demands meticulous research and thorough examination prior to its implementation.

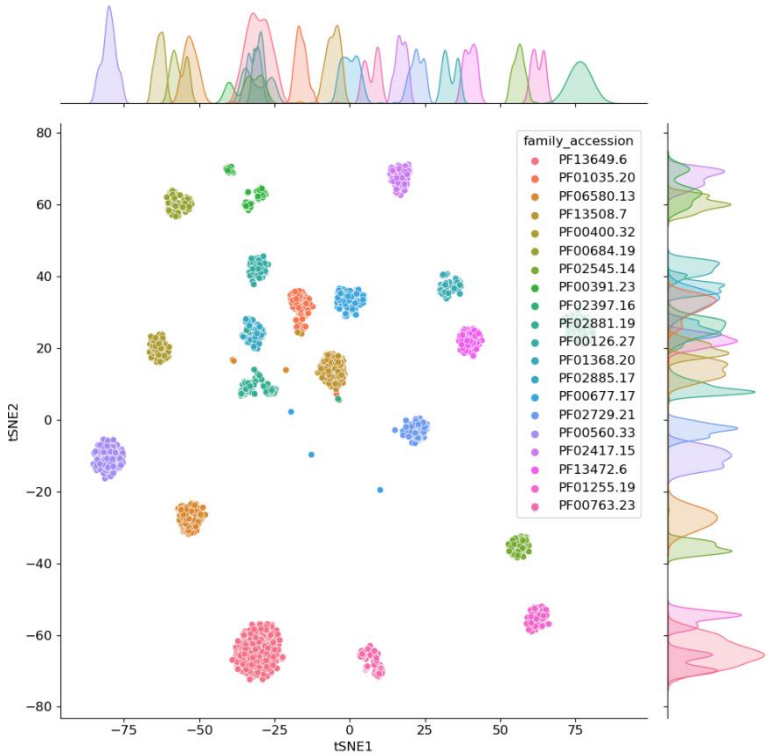


Figure 7: tSNE of the embeddings of the 20 topmost frequent protein families

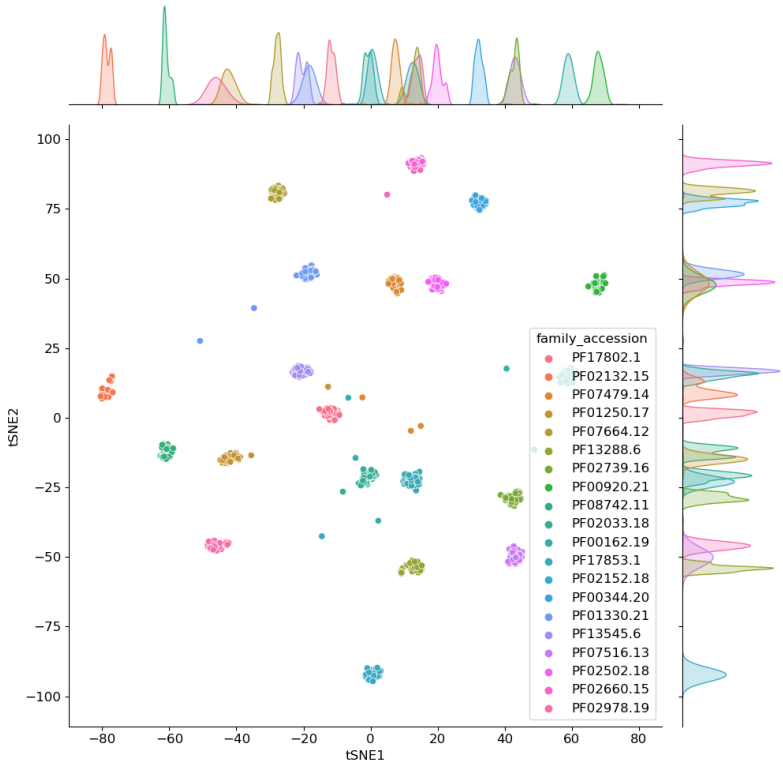


Figure 8: tSNE of the embeddings of the 80th to 100th topmost frequent protein families

References

Ashish Vaswani, N. S. (2017). Attention is all you need. *bioRxiv*.

Kamil Tamiola, M. M. (2017). Structural propensity database of proteins. *bioRxiv*.

Maxwell L. Bileschi, D. B. (2022). Using Deep Learning to annotate the protein universe. *nature biotechnology*, 932.