A Review of Protein Structure and Function Prediction in Light of the Transformer Archetecture

Benjamin Strauss

December 6, 2022

Abstract

Prediction of protein structure has been revolutionized in the last five years by the incorporation of natural language processing tools, most notably the attention mechanism and the transformer. In this review, we look at a brief history of protein structure prediction in light of language models and provide an overview of how these new natural language processing tools work in the context of proteins. We further look at current research, analyze potential shortcomings, and finally attempt to outline what might be the next steps in the field of protein structure prediction. This is done in a way that is easily understandable by a programmer with a general understanding of machine learning and no expertise in computational biology.

1 Introduction

Amino acids are molecules that are often called the building blocks of life. In biological systems, these molecules are covalently connected together in chains to form proteins, larger molecules that perform functions in an organism. Since many biological processes are initiated by a protein interacting with another molecule, understanding proteins is critical to understanding biology as a whole.

Classically, programmers represent proteins as character strings, with different letters of the alphabet representing each of the 22 proteinogenic amino acids. Each amino acid has the same general backbone structure with a different side chain, also known as an "R" group, to which the letters correspond. This general structure is shown below:

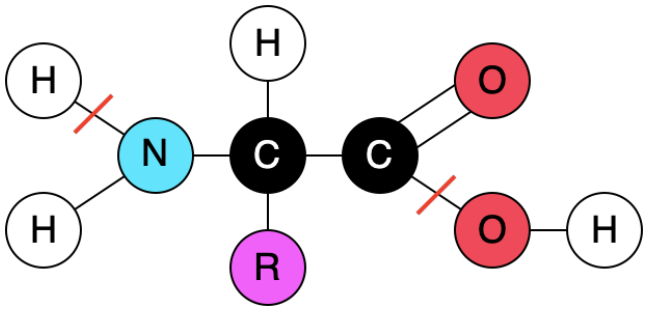

Figure 1: Structure of an amino acid molecule, based on Strauss, 2019, p1. In a protein molecule, these backbones are linked into a chain, where the bonds marked with a red line break, leaving behind joined amino acids and an extra water molecule.

Programmers are typically represented in computational biology character strings (known as a "primary sequence"), with a different letter representing one of the 22 proteinogenic amino acids see Fig 2. This representation is useful at a base level as it captures the physical chain of the molecule, but leaves quite a bit to be desired for more complex analyses. Proteins are nearly never physically straight in nature, but instead fold into complex 3D structures, and these 3D structures determine the function of a protein.

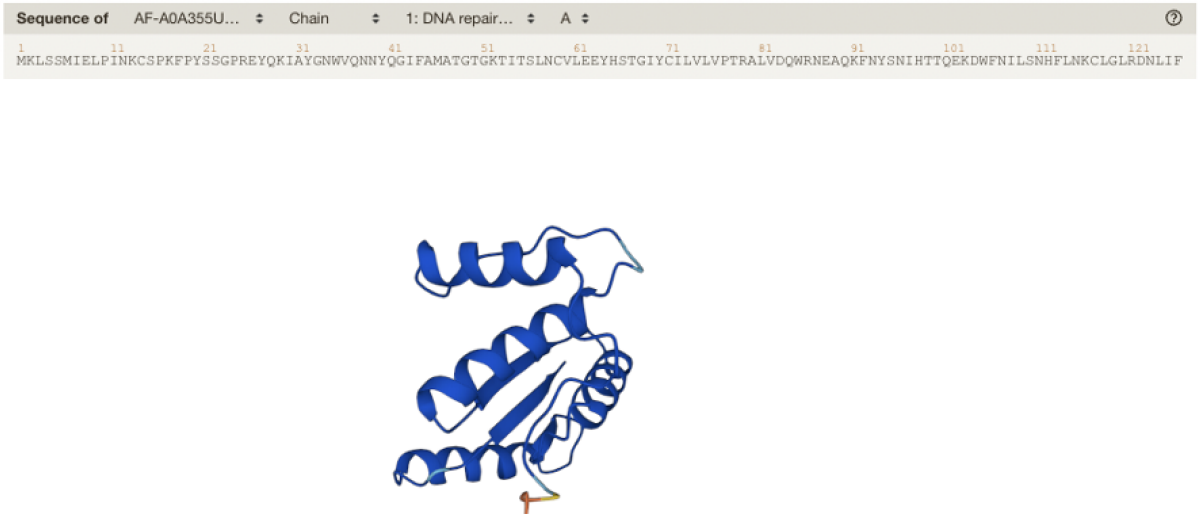

Figure 2: An amino acid text sequence (above) and (visual) protein structure (below).

Some researchers have tried to get around this problem by augmenting the character string with the coordinate locations of the atoms in each amino acid's backbone (non-R-group), and can be seen in PDB format [Berman et al.(2000)]. The trouble is that these atom locations must be obtained experimentally through NMR or x-ray crystallography, and is far more time and effort intensive than obtaining the primary sequence.

Due to the issues with obtaining so many structures experimentally, many researchers have turned to computational approaches to determine a protein's 3D structure from sequence alone. This process came in three phases, first with simple algorithms based on biological constants, then with neural networks (typically of a convolutional or recurrent nature), and finally in the last few years with a type of natural language processing architecture known as a transformer.

Transformers have been wildly successful, achieving far greater accuracy than anything that has been done previously, and for the first time, being able to predict global structure (called tertiary structure) as opposed to the local structure (known as secondary structure). Part of this is that the output of the transformer architecture can carry more information, as each amino acid is encoded as a vector of numbers rather than a single letter. Thus, the resulting encoding of a protein is a matrix of numbers instead of a string of letters, as shown below:

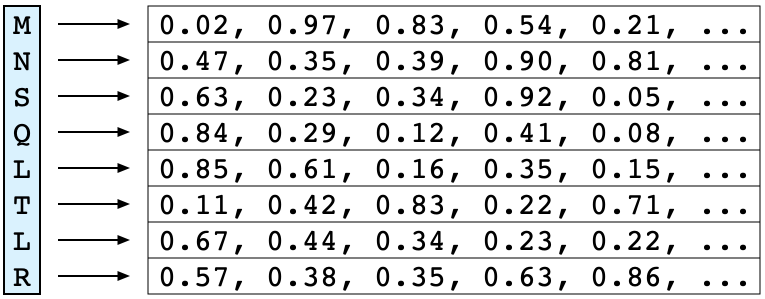

Figure 3: Visual example of vector embeddings of a sequence.

The size of these numerical vectors depends on the encoding method, but every vector produced by a given encoding method will be of the same size. Typical lengths are in the range of 500-1500, with 1280 being commonly used by many neural networks including variants of Facebook's ESM [Rives et al.(2021), p22].

Unlike the character strings, the meaning of a single number in this matrix is nearly indecipherable to human observers. These matrices however are of great use computationally, for such purposes as categorization and similarity problems [Rives et al.(2021), p5], as well as de novo drug discovery [Grechishnikova(2021)].

Other methods tried are those that involve sequence-based descriptors [Strauss(2019)] and counts of kmers (how many times does a sequence of a given length appear in a protein), but these have not met

with nearly the level of success as transformers.

2 History

Predicting the 3-dimensional structure of a protein from its sequence has always been the holy grail of computational biology [Petsko(2000)]. Before neural networks or transformers were invented, many had tried to predict structure algorithmically, with some minor success. Most early attempts worked on predicting the local (aka "secondary") structure, of which there are three major classes: helices, sheets, and other (sometimes known as coils or random coils).

These early secondary structure prediction algorithms used biophysical and biochemical principles and experimentally-determined constants to try to calculate how a chain would fold. This should work in theory, as refolding experiments have shown that information required to specify a protein's folded conformation (its native state) is completely contained in its linear amino acid sequence [Kuhlman and Bradley(2019), p3]. However, in practice these methods underestimate the complexity of protein folding, this can be demonstrated as seen by the early Chou-Fasman algorithm, which is only able to correctly predict protein secondary structure around 50-55% of the time [Kabsch and Sander(1983)].

Even though most would agree that these algorithms are outclassed in predictive power by neural networks, they still remain useful tools, as they are far less computationally intensive than building and training a neural network. To put this into perspective, many of these algorithms can be easily run on a laptop (often in less than one second for a Java translation of the 1974 Chou-Fasman algorithm) while training neural networks requires massive databases of proteins and immense computational resources, such as those possessed by Facebook and Google.

While an accuracy of 50-55% might not sound impressive, more modern algorithms have drastically improved upon this, including those using first-generation machine learning approaches which fall in the range of 70-75% [Rose et al.(2011), p1] where progress began to slow down [Jones(1999), p195] [Sen et al.(2005), p1]. These algorithms were mostly based on neural networks or hidden Markov Models (HMMs) [Adamczak et al.(2005), p467].

2.1 Other Types of Predictions

While this review will focus mostly on structure prediction, it is important to note that historical protein prediction forts have not been limited to predicting secondary structure. Notable other predictions have been made on residue disorder [Lobanov and Galzitskaya(2011)], where a disordered residue is one not visible on the x-ray of a crystallized protein [Lobanov et al.(2010), p1]. Yet other attempts of prediction were involved in whether residues were solvent accessible [Mucchielli-Giorgi et al.(1999)], meaning whether the amino acid residue touched the water that the protein was floating in, or was entirely internal to the folded protein.

2.2 Templates

There are two distinct methods of predicting protein structure, template-based and template-free modeling. Template-based modeling involves finding a similar protein for which a 3D folded structure is known, and using that as a guide to compute the target protein's structure [Kuhlman and Bradley(2019), p4]. Often similar sequences are found through a database search, and then compared via a multiple sequence alignment [Kuhlman and Bradley(2019), p4]. This is highly effective, as small changes to an amino acid sequence are not likely to have a big impact on overall structure. The problem with this approach lies in the fact that templates may not always be available for a given sequence of interest.

One specific type of template-based model is the evolutionary model, where the template protein in a target species is an already-solved protein in a related species. These models seem to lack experimental accuracy where a close homolog has not been experimentally solved, which limits their predictive range [Jumper et al.(2021)]. This is not surprising, and a testament to the complexity of the problem of protein folding. Such shortcoming of these options have led scientists to keep searching in hopes of finding greener pastures.

Template-free modeling can be conducted in a variety of ways. Perhaps the most well-known of these is fragment assembly, where an algorithm computes the folding of small segments of a protein and stitches them together, trying to find the folds with the lowest energy. Other methods include model variation, where a starting model is modified to lower the total energy, and contact predictions from residue covariation, where a model is built around the conservation of favorable (those resulting

in a low energy state) residue–residue interactions, such as hydrogen bonds or other compatible charges [Kuhlman and Bradley(2019), p5-7].

Kuhlman and Bradley note in their review that template-based and template-free methods are becoming harder and harder to differentiate, as many modern algorithms use a mixture of both. Many of these algorithms employ neural networks, often of the convolutional variety (CNNs) – which have become very popular in computer vision, such as classification or to identify objects in images [Szegedy et al.(2015)] by having each layer recognize patterns based on the information discerned from the previous layer.

Other options for structure prediction include thermodynamic or kinetic models of how proteins fold, but these become computationally challenging for proteins of even a moderate size [Rives et al.(2021)] [Jumper et al.(2021)], due to exponential time complexity. This remains the case even when one assumes that each amino acid has a limited, discrete set of possible backbone states. Yet, there remains hope for this approach if future computing power grows substantially, or alternatively, directional cues may be able to narrow down the options for backbone states [Kuhlman and Bradley(2019), p3].

Even if they have trouble computing most proteins, these tools have been far from useless. Rosetta [Zhou et al.(2018)] has been remarkably successful in free energy estimation of protein folding, even with simplified energy models, and GROMACS [Van Der Spoel et al.(2005)] has seen widespread use for modeling dynamics and fine-grained structure prediction. However, in addition to the aforementioned problems, these models require expert knowledge for setup [Bepler and Berger(2021)] which limits their utility.

2.3 Algorithmic Availability

Perhaps the best thing these algorithms have going for them is accessibility. Unlike complex machine learning models, the vast majority of these algorithms are available for free via batch servers. Many are also open source, including those that use pre-trained neural networks of their own. In a previous effort to attempt to predict protein secondary structure variability, we were able to obtain the source code for Chou-Fasman, PSIPred, JNET, GOR IV, PROF, Sable, and SSpro (via a software bundle called SCRATCH-1D) online, as well as the source code for the DSC algorithm by contacting the authors directly.

While many of these algorithms have extensive instructions, they can still be tricky to set up and run. Some come with a database of protein data, which can reach sizes of 15GB for SSpro v5.2. SSpro v5.2 has now been replaced with a version 6, and the database keeps growing with each successive release. Another problem is that these algorithms sometimes have issues on certain hardware; SSpro v5.2 takes around 20 minutes to run on a single core and does not run on Apple M1 (all things we have discovered first-hand).

2.4 Comparison to Transformers

This brings us to the modern era and work done by neural nets, using Bidirectional Encoding Representation for Transformers or (BERT), covered in-depth in section 3, Models such as Facebook's ESM using BERT have made significant progress for secondary structure prediction, reaching accuracy levels around 72% for eight classes of secondary structure [Rives et al.(2021), p9], the prediction of which is significantly harder than 3-class prediction. (Of the eight secondary structure classes in the DSSP database, the following conversion from 8 to 3 classes is applied: G, I, H to H, B, E to E, and T, S, and "other" to C, where H denotes helix, E denotes -strand, and C denotes coil [Adamczak et al.(2005), p468]).

Prior attempts had reached 65% for conditioning neural fields and 50% for SSpro8 in 2011 [Wang et al.(2010)]. It is worth noting that nowadays, SSpro8 currently claims accuracy of around 84% for proteins with no known homologues [Cheng et al.(2005)], although we could not determine the exact method used in the prediction.

These numbers are unlikely to ever reach 100%, and this is likely due in part to the fact that proteins can sometimes change their secondary structure, based on their surrounding conditions. When a molecule binds to the protein, the protein may undergo structural changes. Since these algorithms only predict a single type of secondary structure for each amino acid, they cannot account for how the protein would behave in every set of circumstances [Chen et al.(2020)].

3 Neural Network Embeddings

Neural network embeddings come as a partial successor to the secondary structure prediction algorithms of the past, and are used because of the additional data they encode, as shown in the table below:

Table 1: Neural network analogized to human cognition.

Secondary Structure Prediction	**Neural Network Embeddings**
Predicts only local structure, usually in the categories of helix, sheet, other (aka coil), and sometimes turns and bends	Predicts numerous metrics including: full 3D structure [Jumper et al.(2021)] protein function [Rives et al.(2021)]
Biological significance of output can be easily understood by humans	Biological significance of output is not human-interpretable
Many algorithms are publicly accessible for research use.	Pre-trained models publicly accessible for research use.
Many use large databases, although a trained programmer can change them to use a different database.	Training a model with transformers requires an enormous number of parameters. Changing databases is a long task. [Bepler and Berger(2021), p659]

This all raises the question of if one could assign different embeddings based on biological characteristics to give better results, or at least comparable results that have obvious biological or biochemical significance. To our knowledge, this has not been achieved, and although we were seemingly able to predict secondary structure variability using this method in previous work [Strauss(2019)], further research, currently unpublished, seems to suggest that this result was likely a coincidence.

3.1 Relationship to Natural Language Processing (NLP)

The idea to assign vector embeddings to amino acid residues in a protein is one that biologists took from natural language processing, through something known as the transformer model. The high-level concept is fairly simple, to think of a protein as a sentence where each amino acid is a word in that sentence. The meaning of the sentence then corresponds to the function of a protein. The analogy mostly breaks down as follows:

Table 2: Natural language processing (NLP) vs computational proteomics.

Natural Language Processing	**Computational Proteomics**
An English word	A single amino acid residue
A sentence (sequence of space delimited words)	A protein (sequence or string of amino acids)
The meaning of a sentence	The function of a protein
The vector embedding of a word from a neural network	The vector embedding of an amino acid residue from a neural network

Like English, proteins are a highly context-sensitive language. For instance, amino acids that occur in similar contexts tend to have similar properties, just as words in a similar context have a similar meaning [Bepler and Berger(2021), p657].

While 22-ish (occasionally, amino acids are modified post-translation) words may seem insufficient to describe a language as complex as protein function, like English, amino acid words have more than one meaning. For example, the English word “play” can mean (1) to have fun with or (2) a performed drama. The words are similar syntactically and phonetically, but their semantics differ. Likewise, given a random amino acid, say a cysteine, once cannot tell its functional implications. Without context, there is no way to determine if the position of the amino acid is an active site, an allosteric site, or even if it is stabilizing the protein with a sulfur-sulfur bond to another cysteine. This is analogous to how a human cannot tell which version of “play” is being referred to when seeing the word out of context.

This language-to-protein similarity has led many researchers to apply the transformer model to protein structure prediction, and it rapidly became state-of-the-art, as evidenced by its use in the wildly successful

AlphaFold, GTP-3 [Floridi and Chiriatti(2020)], etc. However, researchers are also quick to point out that there are many key differences between human languages and nature's language of protein biology [Brandes et al.(2022)].

3.2 Understanding Transformers

To fully appreciate transformers, one needs to understand their predecessor, the recurrent neural network, or RNN [Vaswani et al.(2017)]. When an RNN processes input, such as a word in a sentence, it stores some of the data in an internal variable, known as a hidden state. When the RNN processes the next word, the RNN uses data stored in the hidden state to influence how it processes that next word. It then updates the hidden state, and repeats.

This technique gives the RNN a lot of power, as it can take past data into consideration when processing current data, providing a bit of context for the data that is currently being processed. However, this comes with significant drawbacks. First, because the RNN needs to learn from past processes to process the current data, information must be processed sequentially, and thus, full parallelization is precluded [Vaswani et al.(2017), p2]. This makes RNNs slow and clunky when processing a large amount of data (i.e. a protein database).

The second problem with RNNs is a bit more nuanced: because the hidden state is continuously being updated, the impact of earlier processes on the hidden state gets diluted as the network progresses. This is not necessarily a problem in every situation, but it can be when dealing with folding molecules like proteins, where distant residues can interact. Third, and perhaps most damningly for proteins, the RNN can only go in a single direction, typically known as "left-to-right" [Brandes et al.(2022)], whereas proteins in nature are not as thus constrained.

An improvement to the classic RNN is the Long Short-Term Memory network. The LSTM deals with one of the problems with the RNN, that there is no longer a hidden state being diluted. And thus, the standard RNN can only look back about 10 timesteps without running into problems with exploding or vanishing gradients [Staudemeyer and Morris(2019), p2]. LSTMs are a way of circumventing this issue, as they have gated memory that allows them to store values for much longer periods of time, potentially over 1,000+ time steps. This also gives the network the ability to filter out irrelevant input, helping to address the RNN's dilution issue [Hochreiter and Schmidhuber(1997)].

While the LSTM does somewhat fix the dilution issue of the standard RNN, it does not address the problem of requiring the input to be processed sequentially [Hochreiter and Schmidhuber(1997), 22]. LSTMs also run into trouble if too much data must be remembered [Staudemeyer and Morris(2019), p28], as their storage is limited.

However, these problems are addressable, and this is where the attention mechanism comes in, on which transformers are based. Instead of sequential processing with a hidden state, the alternative is to process the entire input sequence at once. Simultaneous processing of the input means no long sequential operations, thus enabling parallelization [Vaswani et al.(2017)], mitigating dilution by intermediate operations. Finally, this means no future processes that the hidden state could not yet have been influenced by, but should theoretically reflect.

The attention mechanism's main contribution to protein structure research is not turning words or amino acids into numerical vectors, as earlier tools such as Word2Vec [Mikolov et al.(2013), p1] did this as well, but in adding the context. Word2Vec's initial contribution was putting words in vector space, i.e., producing a sparse word encoding, something that would be of far less use for amino acids than English words, (because there are far more words in English than amino acids in proteins).

3.3 How Attention Works

The attention model gets its name because it, in at least a manner of speaking, mimics how human attention works when reading a sentence. When a human reads a sentence (at least for most of us) our brains do not assign equal importance to every word in the sentence. We might not be conscious of it, but our brain parses out things like prepositions, and comes away with the key points. Then the brain groups related words together to give us our comprehension of subjects, actions, et cetera. The attention model does essentially the same thing, except computationally with numerical weights instead of neural connections. The result in memory is known as an attention matrix, which describes how words (or amino acids) are connected to each other. To show how this works, we will use the example sentence "The spotted dog sat." An attention matrix for this sentence might look like Fig. 4:

Table 3: Sample attention matrix for a simple sentence.

Word:		The	spotted	dog	sat
Relatedness to word:	The	0.4	0.2	0.15	0.15
Relatedness to word:	spotted	0.15	0.3	0.25	0.15
Relatedness to word:	dog	0.3	0.25	0.35	0.25
Relatedness to word:	sat	0.15	0.2	0.25	0.45

The above matrix was not generated by an algorithm but by our guesswork for use as an example only. For each word in the top row, the weights below it are how related it is to each other word in the sentence. In the algorithmically generated matrix, each column would actually be multiple normalized vectors instead of a single vector.

For each word in the sentence we can see that each word is maximally related to itself. Words like "spotted" and "dog" are highly interrelated as one describes the other, and marked with a high value, while words like "spotted" and "sat" are only slightly interrelated. This same technique can be used with proteins, perhaps even more quantifiably, as a matrix can be made of how connected a pair of amino acids are in terms of the overall structure.

In practice, generating attention matrices is done via a neural network. The process is described in figure 4.

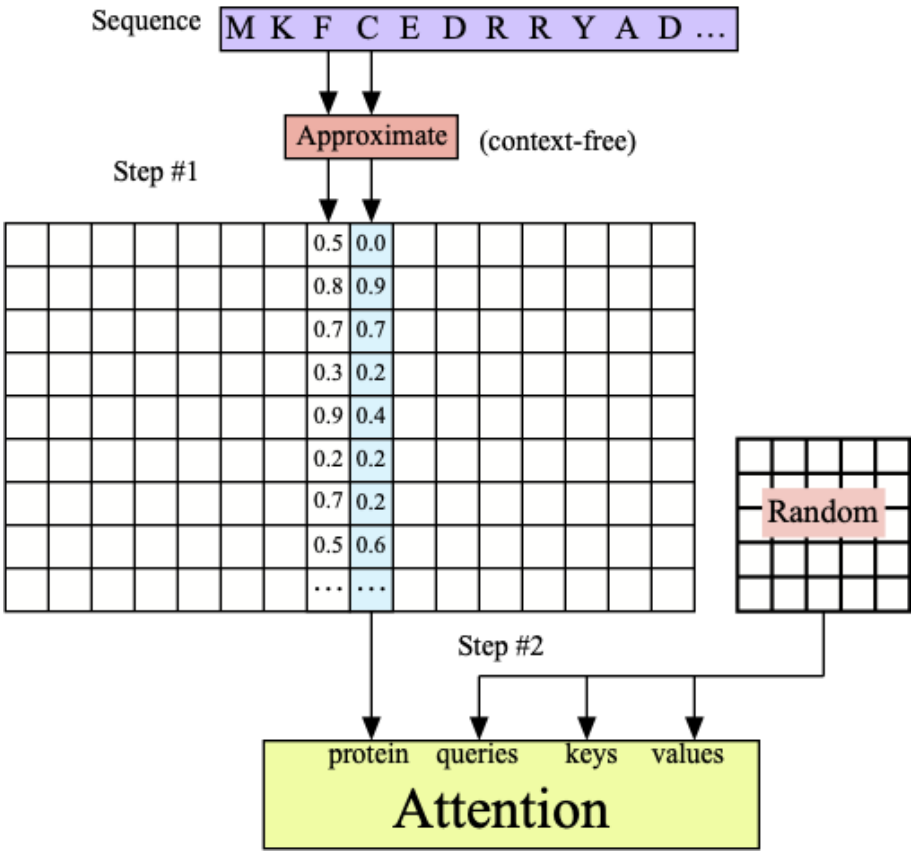

Figure 4: Start of the attention mechanism.

The attention layers then call each other recursively as can is shown by figure 5.

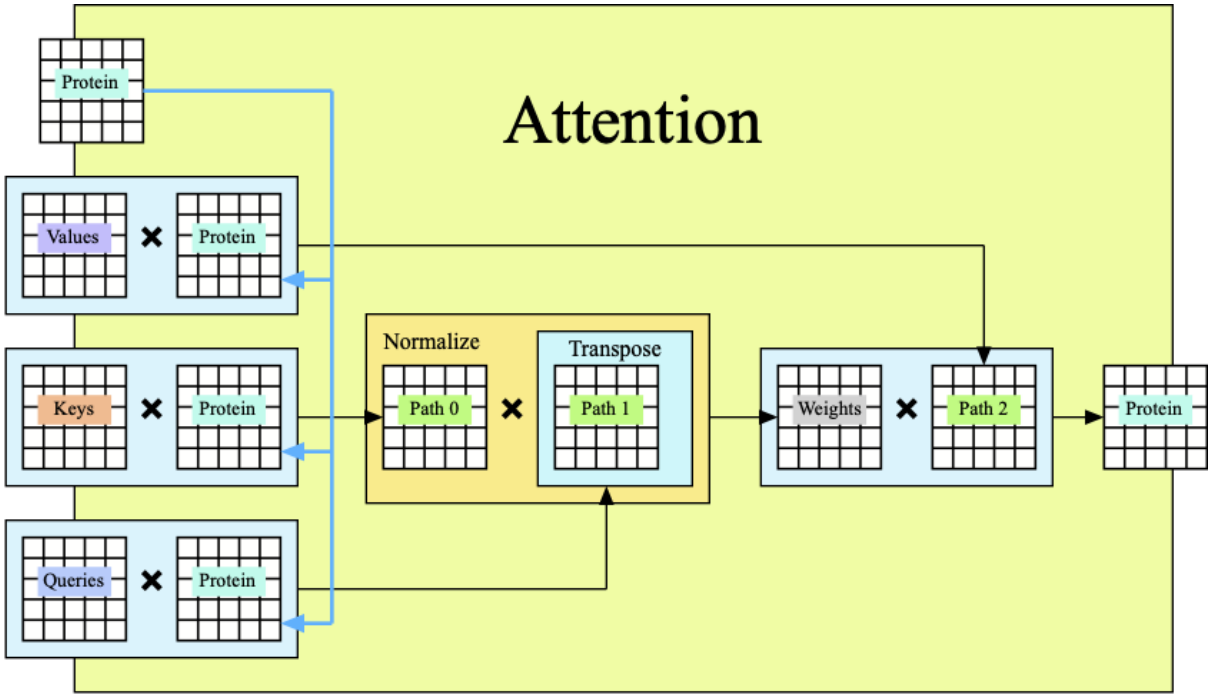

Figure 5: A typical attention layer.

Once an attention layer returns, the keys matrix, queries matrix, and values matrix will be updated (via gradient descent) with the results of that layer. This process of updating the aforementioned matrices is what allows the program to learn.

Over time, this context-free encoding will be trained by the effects of nearby vectors into something context-sensitive, with the weights of the keys, queries, and values representing the relatedness of each word to every other word.

This approach is also parallelizable; and parallelization of this approach results in what is known as multi-headed attention. In multi-headed attention, the "learning loop" would be done in parallel instead of iteratively in a loop. This parallelization is shown via the diagram from [Vaswani et al.(2017), p4]:

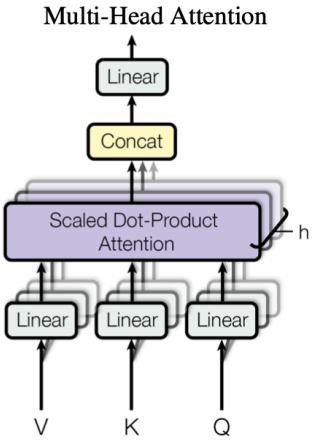

Figure 6: Multi-Head Attention, V, K, and Q stand for "values," "keys," and "queries" respectively. Their results are finally concatenated into a single matrix, which is further sent through more attention layers.

3.4 Inner Workings of Transformers

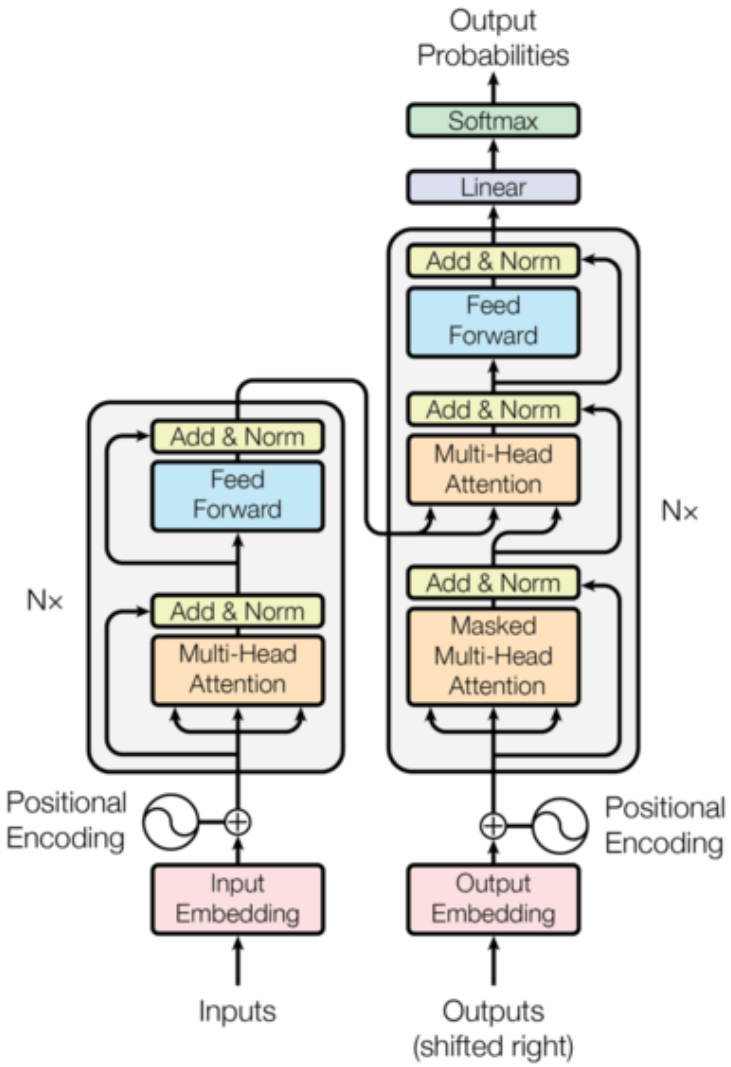

Figure 7: How data flows through a transformer [Vaswani et al.(2017), p3]

The attention mechanism, first a multi-headed variant followed by a single variant, comprises the bulk of what the input module of the transformer does, repeated several times (as denoted by the "Nx" in the above diagram). For our purposes, we will focus on the input module, because we are mostly focused on classification, so that is what is most important to us. It is worth noting that the code in the previous section depicts a recursive attention mechanism, rather than the complete transformer. Notably, the attention mechanism does not encode position, just interrelatedness. This is why both input and output modules have separate positional encoding mechanisms to account for this shortfall.

3.5 Machine Learning with Transformers

To provide a simple analogy, machine learning with transformers can be easily compared to a child learning. Since the machine is learning the "language" of proteins, we will make our analogy between the machine and a young child learning to read. There are multiple methods for the machine to learn, which we will cover briefly. These methods each come with various pros and cons which are mostly outside the scope of this work.

The first of these methods is supervised learning. The neural network will interpret a sentence, then ask its checksum (analogous to a teacher) if it correctly interpreted the passage. Each epoch of the neural network then, can be seen as a child reading a text from start to finish and asking the teacher if it means a certain thing. If the computer's interpretation is wrong, it tries again. We attempt to illustrate this metaphor with the following table.

Table 4: Neural network analogized to human cognition.

Human Cognition	**Neural Network**
Neural connections in the human brain to build association	Numerical weights assigned to words to show what is most closely related to what
A parent, teacher, etc, checking a child's reading comprehension	The neural network checking the comparisons it made
A child reading a book from start to finish	An "Epoch" – all of the data running through the network at least once
A child reading a sentence	The network encoding a sentence
A child writing book reports on what he/she/(insert pronoun) read	The network decoding the encoding of the sentence back into a sentence

This above method is employed by Vaswani et al, used in language translation. The encoder turns the words of language A into embeddings, which are then decoded into language B. Training a neural network in this way requires a large amount of data [Brandes et al.(2022)], akin to a child reading many books and writing many reports, which are then graded and given to the child as direct feedback.

Not all transformer systems employ both an encoder and a decoder; only one of the two may be needed, depending on the task at hand. Encoder-only systems are typically used for classification or sequence labeling problems. Such is useful for quantifying inherently qualitative data, such as words in a sentence, or the type of an amino acid residue in a protein. By contrast, decoder-only systems are used for sequence generation, such as language modeling [Lin et al.(2022b)]. Using this in the context of proteins would likely be in the form of generating a protein sequence with specific given characteristics [Grechishnikova(2021)]. Our research mostly involves using encoders, so that will be the primary focus of this section.

Returning to the metaphor of a child learning to read, one cannot help but notice that after a certain point, children begin to pick up new words on their own. Often people can determine what a word means just by the context in which it occurs, without consulting others or a dictionary. There is a machine learning equivalent to this, which is called "self-supervised learning" [Devlin et al.(2018)] because the machine is checking itself, instead of using a hard-coded answer label [Bepler and Berger(2021), p657]. Using this technique Bepler & Berger describe three different models. This first, is called "left-to-right," although a more accurate term might be "start-to-finish" as not all languages are read left to right. A typical manifestation would be a machine parsing a prompt then answering a question. The left to right aspect comes into play because after the prompt, there is nothing for the machine to read (in the other direction) to give it a clue to answer the question.

The second method is what is known as bidirectional, and would be more like a person trying to find the definition of a word they have never seen before in a novel. (For example, imagine a boomer coming across a new word like "selfie.") Here, the bidirectionality means that the person or machine has access to text on both sides of the new word, and can read the response, and in doing so determine things like changes in tone, but can do so going backwards and forwards.

Finally, there is the masked language model (MLM), which would be similar to a reading comprehension exam, where a student would read a story and then need to fill in a blank. The machine is essentially doing the same thing, except that the machine has to fill in a blank further on in the story, where a word in the text has been blanked out (i.e. masked). This requires processing an entire sentence at once in parallel. This is displayed by Bepler and Berger in the following graphic:

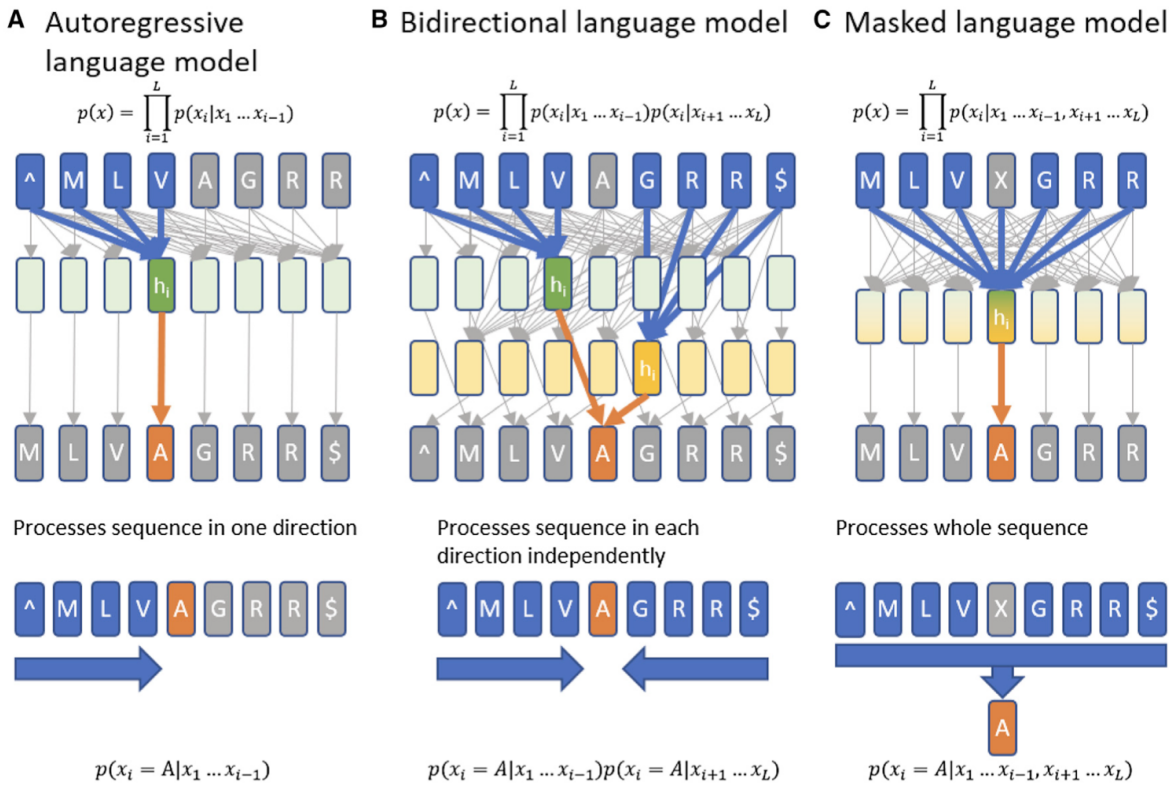

Figure 8: Comparison of language models

It may be worth noting that these are not the only possible ways to process sequential information, however, it would be a hard sell to say that the alternatives, at least the ones we can think of, make as much sense. Employed by many secondary structure prediction algorithms of the past, there is the moving window, however this is vastly inferior to processing the entire sequence at once, for obvious reasons. One could also preprocess the data by instead of moving towards the value to predict, first moving away, or even doing multiple passes. How much value this would add to the mono- and bi-directional approaches though is anyone's guess, and we think it would be likely inferior to the masked version. Finally, one could check the data in small chunks of k characters for a given k, slowly grouping those chunks together recursively. The trouble with this method is that the initial chunking may be arbitrary, and so it could create biases not in touch with reality.

3.6 Results of Transformers

It seems that even without significant consideration of the differences, natural language processing tools are extremely powerful – at least when applied to massive amounts of data. One of the most successful types of transformer BERT, which was used by both Jumper et al., and Rives et al., with the latter trying four different versions of transformers, and BERT being the most successful, but only by a small margin. Facebook's ESM [Rives et al.(2021)] trained on 86 billion amino acids from a quarter billion sequences from across the tree of life. The model was trained on the Uniref50 protein database, which is a clustering of UniParc at 50% sequence identity. While the Facebook ESM team acknowledges the differences between protein and natural language, specifically the differences in cases in which each arises, the results are surprisingly useful given this drawback. The information encoded in the embeddings from the NLP models contain significant information about the secondary and tertiary structure of a sequence. This was then verified by linear projections [Rives et al.(2021), p1].

The ESM team also goes further to make the claim that given a table of amino acid vector embeddings, the protein can be represented by a single vector embedding with each feature in that new embedding being the average of all features across the amino acids in the protein, effectively allowing a 2D matrix to be compressed to a 1D vector by a simple average. [Rives et al.(2021), p5] Given the amount of information irretrievably compressed in this manner, we are somewhat skeptical of the value of this approach.

The BERT model was also employed about a year later by Jumper et al. of Google's AlphaFold, which reached near experimental accuracy most of the time, in a blind test of structures that were not yet publicly disclosed, with results being within 5 angstroms about 95% of the time.

Despite its success, BERT is not without critics. The team behind ProteinBERT points out in their 2021 paper that most sequence-based language models have been pre-trained on natural languages, and may not be optimal for proteins. The team trained a model using a version of the natural language model BERT specifically for proteins, and found noticeable improvements, despite only training on 106 million proteins [Brandes et al.(2022)].

3.7 Limits of Current Models

While breakthroughs have occurred with current models, these models are far from flawless. AlphaFold, and along with it, CASP require large sets of protein sequences that can be aligned with high confidence, and are unable to learn patterns across groups of unrelated proteins, making their effectiveness highly reliant on the availability of specific data to the problem in question [Bepler and Berger(2021)]. Even with models such as Facebook's ESM, the accuracy of the predicted contacts varies as a function of the number of evolutionarily related sequences in the training set [Lin et al.(2022a), p4].

Transformers also in and of themselves have a few shortcomings, such as inefficiency at processing long sequences (due to processing an entire sequence at once) and difficulty training on small data sets [Lin et al.(2022b)]. However, transformers have been so successful that rather than abandoning the concept of transformers, researchers have modified the initial concept of transformers (known as x-formers) specifically to address these shortcomings. Lin et al. lists three major classes of these modified transformers, those devoted to dealing with both of the shortcomings listed above, along with those which adapt the transformers to specific tasks (such as ProteinBERT for proteins). The details of the techniques used to achieve these aims tend to be fairly technical, and are generally irrelevant to the end result, (except obviously when creating new transformer architectures from scratch), suffice to say that if a specific transformer architecture does not fit your purposes, many others do exist. Inefficiency can be mitigated via divide-and-conquer methods (commonly seen in recursive algorithms) or sparse methods (if a lot of the data is very similar). Artificially introducing biases, regularization, or pre-training on unlabeled data can help with small data sets [Lin et al.(2022b)]. (Regularization is a technique for modifying a loss function, and is explained in detail at https://www.geeksforgeeks.org/regularization-in-machine-learning/.)

3.7.1 The Difference between Proteins and Natural Language

While proteins may seem like a language, it is not hard to see where the similarities end. First, a human language such as English has around a million words [**?**], and while there are a large number of amino acids in nature, only 22 are known to be translated into proteins. Even if we include post-translational modifications, that number does not increase by more than an order of magnitude or two.

Second is the design of the language. Human languages are designed to make sense to human intelligence, whereas biology is not, which has been noted by various researchers. Excluding the concept of intelligent design, which for purposes of this work we will consider having been shown to not be a thing, biological systems are not designed to make sense to anything. (One could thus argue that the language of proteins is closer in fact to Pokémon™-speak, something that was not designed for humans to understand). This fact is not so much an issue for our machines, as given enough data to process, patterns tend to emerge.

This comparison with language may indeed go further, although it is speculation at this point, post-translational modifications may have some effect on proteins as well. To our knowledge, post-translational modifications were not included in the training of neural networks like AlphaFold, ProteinBERT, and ESM. Given the success of these models, which take nothing more than a standard amino acid sequence, it is likely that post-translational modifications play a minimal role; perhaps even analogous to the accent over the "e" in words like "Pokémon" or "résumé": just about everyone knows what the word is even if you exclude the accent.

It is however possible that these neural networks are missing something. Some post-translational modifications could play a role analogous to the tones in English, which can occasionally modify the meaning of a word, usually only slightly, except in instances such as sarcasm (which may or may not have an equivalent in the world of proteins). It is unlikely that these modifications play the same role they do in Pokémon™-speak, or even in Thai, where different tones can give a (spoken) word very

different meanings. Still, we stress that the importance of post-translational modifications is currently our speculation, and has no experimental data backing it up.

What is much more likely in our opinion is that like language, proteins have patterns, and our machines are really, really good at picking out patterns, when fed with enough data. This is evidenced by the sheer amount of predictive power machine learning (and neural nets) have had in other fields, from other branches of science to predicting consumer behavior.

Other researchers, including [Brandes et al.(2022)] have identified further differences between proteins and spoken language. Aside from mentioning the previous point (although they phrase it differently, saying that proteins lack multi-letter building blocks like words in sentences), they mention that proteins are much more variable in length than sentences. This is an interesting point, because while proteins do have significant length variability, sentences have a great deal of variability too, from small exclamatory comments (e.g. "The dog sat!") to long-winded run-ons (e.g. "It was a dark and stormy night. . . ").

Despite this, the vast majority of sentences fall somewhere in the middle of that range, perhaps because we as the human species have a fairly uniform attention span when it comes to sentences, and while certainly not universal, fairly uniform standards when it comes to readability. Nature is not similarly constrained, as there are no rules when it comes to what can be a protein, save for evolutionary success and failure. And while there does not seem to be a hard limit for protein length, it is worth noting however, that longer proteins are more difficult for nature to fold.

And Brandes et al., have a final point on how proteins are not like sentences, in that distantly placed amino acids can interact with each other far more than distantly placed words can. In general, when sentences are long, they tend to be broken up into small sections, usually by commas, because human brains are more easily able to process things that way. For this reason, it is very rare for a word at the end of a sentence to describe a term at the beginning of the sentence (and if it did happen, we would likely consider that sentence badly written). This becomes an issue when comparing amino acids to words and proteins to sentences because nature has no such guidelines on organization. A protein will stick around if it is evolutionarily beneficial, rather than if it is pretty.

3.8 BERT for Proteins

The difference between natural language and proteins begs the question of how one actually makes a BERT model for proteins. The team behind ProteinBERT created a language of 26 words which they call tokens. 20 were used: one for each of the standard amino acids, presumably their standard single character representations, as well as 'U' for selenocysteine, 'X' for unknown amino acid, and OTHER for other amino acids that did not have their own tokens. They also added 3 additional tokens (START, END and PAD). For each sequence, START and END tokens were added before the first amino acid and after the last amino acid in a chain, respectively. The PAD token was added to pad sequences shorter than the sequence length chosen for the minibatch, thus assuring that all sequences' input was of the same number of tokens [Brandes et al.(2022), p4].

This approach takes out one of the variables that a language model would normally have to contend with when learning on proteins: that being the length, and secretly (for the machine) encodes it by the use of tokens. To train ProteinBERT, this encoding process was done to the roughly 106 million proteins from UniprotKB / UniRef90. Next, some tokens were corrupted (replaced with something random) with 5% probability per token of being corrupted. ProteinBERT was then tasked with detecting and correcting corrupted tokens, which meant restoring them to their original value.

The choice of tokens however is a bit puzzling, as the ProteinBERT team chose to include selenocysteine (U), but not for pyrrolysine (O); when both are non-standard proteinogenic amino acids. One reason for doing this would be if there were not enough instances of pyrrolysine to determine any significant pattern, and the same could be said for various modified residues. Pyrrolysine only occurs in methanogenic bacteria and archaea [Rother and Krzycki(2010), p1], while selenocysteine occurs in all three major domains of life [Johansson et al.(2005), p1], so this would make sense if a large percentage of UniRef90's sequences came from Eukaryotic organisms. As for the lack of accounting for modified residues, this could be because they are also too rare, or they behave identically to their unmodified counterparts.

3.9 Results from ProteinBERT

ProteinBERT was pre-trained for about 6.4 epochs [Brandes et al.(2022), p8], and the team noted that the pretraining did have a noticeable positive effect. ProteinBERT was able to detect patterns better

in longer sequences, having similar results for lengths of 512 and 1024, but significantly less accurate results for lengths of 128. Interestingly, for lengths of 1024, there seems to be more variability in the loss than there is for lengths of 512.

One possible explanation for this is that while protein lengths are highly varied, most tend to have lengths around 200-300 residues. It is possible then that there is less loss for shorter proteins, as these would have more padding, and corruption should be much easier to detect in padding. Why? Because if a non-PAD token is surrounded by PAD tokens, it is very likely corrupted (as the only alternative would be having the other PAD tokens all corrupted). With lengths of 1024, there is more padding for shorter proteins, and because padding errors are far easier to detect, one would expect far less loss. While the 512-token results look the strongest, one has to wonder if these results are optimal, or if there is another token length that would give even clearer results, with a smaller error margin.

4 Comparison of Protein Function

Given these new matrix embeddings, scientists are faced with a problem: how can we compare protein functions? Protein functions are typically described by Gene Ontology (GO) notation in terms of molecular function, cellular components, and biological processes [Ashburner et al.(2000)], and ¡0.1% of proteins have GO annotations assigned experimentally [Cai et al.(2020), p1]. Biological roles are then described by sets of terms from the hierarchically organized GO domains [Fa et al.(2018), p2].

Fa et al took the approach of trying to predict these GO annotations with a 258-feature neural network on the Uniprot-GOA database, and tested their results on human proteins that had no experimentally verified annotations at that time, but received some in the following 24 months. It is worth noting that these 258-features were not the result of transformers or NLP tools, but sequence motifs including, but not limited to secondary structure, intrinsically disordered regions, transmembrane segments and more. Since GO domains can overlap, Fa et al. used a deep neural network for each different GO domain that they were looking for (effectively turning the assignment into a decision problem). Results improved considerably at a decision threshold of 0.5, [Fa et al.(2018), p13].

While running the neural network, the team came across a serious problem: a very large negative to positive example ratio. This can cause a neural network (or even a regression) to minimize error by predicting everything as negative. Two approaches were used to fix this, the first was to make false negatives have more of a penalty than false positives. The second was to oversample the minority class to ensure that positive and negative examples were balanced, then penalize the minority class with a weight in the prediction phase, and this is the strategy that the team decided to go with, after testing both approaches on the smallest annotation and finding it the better result.

Fa et al did not use transformers, but another group [Cai et al.(2020)] did. Their experiment included encoding amino acid trigrams (3 consecutive amino acids) into vectors with NLP tools and then letting a convolutional neural network find patterns matching the patterns of known GO notations. This experiment was limited in that it only used proteins less than or equal to 1,500 residues in length, and that shorter proteins were padded. This still runs into the arbitrary constant problem as padded amino acids result in vectors of zeros [Cai et al.(2020), p4]. While this may not be as robust as the ProteinBERT approach, it is still a logical choice. The precision-recall curve mostly exceeded that of BLAST and other metrics [Cai et al.(2020), p9].

4.1 Function without GO notation

GO notation seems to be the gold standard when determining protein function, but it does not necessarily have to be. Instead, in theory function could be done via comparing proteins directly, which is briefly mentioned in [Rives et al.(2021)]. Since a protein is essentially a vector of amino acids, and each amino acid is represented by a vector embedding (as per section 1), the protein itself is represented by a matrix of size $m \times n$, where m is the size of the vector embedding and n is the length of the protein sequence. This could, potentially, give us a more quantifiable comparison.

A hypothetical example of a protein-turned-matrix can be visualized by the following table:

$$Feature_m = \frac{\sum_{i=1}^{n} i_m}{n} \tag{1}$$

1: Equation for the m-th feature (m = feature number, n = number of amino acid residues)

Table 5: Example of a matrix embedding for a protein. Used only to explain the general concept, these numbers were generated and assigned randomly.

Amino Acid	M	A	E	M	C	F	R	Q	W	D
Feature 1	0.490	0.776	0.813	0.867	0.647	0.310	0.505	0.164	0.183	0.868
Feature 2	0.991	0.203	0.186	0.834	0.025	0.857	0.152	0.798	0.069	0.190
Feature 3	0.765	0.601	0.781	0.747	0.010	0.401	0.188	0.159	0.564	0.568
...	...	...	...	...	...	...	...	...	...	...
Feature m-1	0.255	0.616	0.885	0.219	0.137	0.268	0.323	0.786	0.590	0.355
Feature m	0.424	0.185	0.362	0.281	0.045	0.608	0.179	0.181	0.242	0.283

Typically, the numbers of these vectors come from the last hidden layer of a neural network, i.e. the layer right before the output layer. Thus, obtaining these vector embeddings is an encoder-only task, and no decoder is necessary for this application. However, this matrix is just a starting point; while in theory, we could compare proteins to each other by comparing the tables, this is a bit more complicated than it first appears. One major problem is that different proteins have different lengths, and therefore different values for n. (Given protein #1 with length p1 and protein #2 with length p2, it is likely $p1 \neq p2$).

There are two ways to get around this issue: the first is to somehow standardize p1 and p2. This is done by constructing a single vector for the protein chain, by averaging across each feature [Rives et al.(2021)]. For an embedding of size m, and a protein chain of length n, the value for m is computed as follows:

This leaves an $m \times 1$ vector, which Rives et al say is somewhat indicative of protein function, although this method leaves a bit to be desired: while an average might give one an overall sense of a protein, it does not indicate what parts of the protein do what. A protein with radically different values may give the same average as one that has similar values throughout, and that data cannot be re-extracted from the per-feature average. And since averaging values is lossy in this way, it inevitably loses much of the power that assigning embeddings has in the first place.

This does not necessarily mean one should dismiss the averages outright, as more information can be added to this vector while still keeping the size constant. One could augment these averages with values such as medians, minimums, maximums, standard deviations, etc, which would each grow the vector by a constant size, but keep the size of the vector constant.

A second way of comparing two proteins is to compare them by amino acids. While this does not lose information in the same way averaging does, it does beg the question of which amino acid should be compared to which. In regards to an alignment though, while the standard Needleman-Wunsch is a baseline, it ultimately falls short as it relies only on amino acid type, which does not necessarily mean functionally similar amino acids will end up aligned together, although it may be able to be used as a rough estimate. Building a matrix based on cosine distances is an attractive idea, however, there is nothing intuitive to use as a gap penalty – any constant chosen will ultimately be arbitrary. Yet the idea of aligning via embeddings is not new, and is explored by Bepler and Berger [Bepler and Berger(2021)], where they attempted to compare structure with this method by employing the SCOP database, and then training a neural network to predict the levels of the SCOP hierarchy shared [Bepler and Berger(2019)]. Still, the SCOP levels are ultimately arbitrary, so this method (although it works in practice) does not have the mathematical backbone we were looking for.

This begs the question of whether a standard alignment will work at all, as two proteins with similar structure and function may have parts swapped, but swapped in just the right way as so they will both solve the same problem. This is plausible, as proteins twist and turn and fold back on themselves in so many different ways that just because a given site is in a different place in the raw amino acid sequence does not mean it will be in a corresponding place in the finished product.

5 Future Work

The success of the transformer model has put most other machine learning models on the long road out. Despite this, transformers require a lot of CPU/GPU power, and thus are limited to those with the necessary computational resources. Efforts are currently undergoing in improving efficiency [Paszke et al.(2019)], and will hopefully make such models more widely available.

5.1 Modeling Function

But what could be the next steps for the models themselves? One such improvement is from a 3D model to a 4D model, incorporating the protein's behavior over time as it interacted with another protein or ligand – effectively a 3D prediction of function. However, unlike GO notation where function is only a label, here function would be a process of residues physically moving around with respect to time.

This is likely a much harder task as there are not a lot of, if any, protein crystal structures crystallized in the process of folding. Such a model however could be extremely powerful, as it could bypass much wet lab work of drug development. However, it is also possible that this is infeasible or, at the very least, impractical. One of the strengths of transformers is that they do not account for direction, however direction is a key aspect of time. This could be factored in by RNNs, but this would forgo the possibility of parallelization.

This brings us to protein structure variability, the prediction of which could easily be seen as an intermediate step in this process. To do this, the attention mechanism would not only take in the protein's sequence, but also additional data about the surroundings, such as temperature, pH, and presence or absence of a ligand, and then generate a 3D structure.

Other work that could be done is predicting the behavior of individual residues: such as active and allosteric sites. Ideally, somehow one could annotate these regions with GO or something similar and then weave the annotations into the generated embeddings. Some work is already being done in this area in terms of classic algorithms [Tian et al.(2021)], including that of convolutional neural networks [Tian et al.(2021)], but we could not find any Transformer or BERT models working on these sorts of problems. This may in practice be fairly difficult at this time as many proteins have unknown allosteric sites, so training data may not be accurate enough for a machine to make decent predictions.

5.2 AlphaFold Improvements

Moving away from structural and functional models, there are two issues that could be addressed in the short term, perhaps to enormous benefit. The first is Google's AlphaFold's reliance on a cluster of near-homologous proteins – this makes AlphaFold unable to process proteins without homologues, thus making it of limited use on such proteins. In theory, one could attempt to address this by making mutation to the proteins, as biochemists sometimes do to test functional areas (if the protein retains the same function after mutation, then it could be considered a homologue), but this requires access to a wet lab. If a computer could predict these areas, it could allow AlphaFold to run on single protein chains.

5.3 Tweaking ProteinBERT

ProteinBERT was powerful, but there are several ways that it could be taken to the next level. One of these ways would be a ProteinBERT-like model that took secondary structure into consideration, not just the primary structure. Assuming this just meant including the 3-class secondary structure, this would nearly triple the amount of tokens from 26 to 71. Running this model anytime soon, however, would run into trouble.

The first of these problems is a lack of training data. Secondary structure is determined by x-ray crystallography, and this can be more of an art than a science. Sometimes amino acids will not show up in the x-ray, and so no secondary structure can be recorded. ProteinBERT was trained on roughly 106 million proteins from UniprotKB/UniRef90, where the RCSB-PDB (the mainstream database containing proteins that have a known secondary structure) only contains a mere 195,093 sequences, at least when we last checked on 9/7/2022. To make matters worse, among these sequences, it is common for not all residues to have known secondary structure for the aforementioned reasons. In previous work [Strauss(2019)], we dealt with this problem by assigning a 4th secondary class to these residues, and work by Lobanov and Galzitskaya [Lobanov and Galzitskaya(2011)] seem to suggest that there is a pattern for these residues. Still, until more proteins are crystallized, this may have to wait.

This is not the only way one could tweak ProteinBERT: one could expand the "OTHER" tag into all of the different amino acids known, such as pyrrolysine and selenomethionine, although this risks there not being enough of any specified other residue for a pattern to be able to be picked up, as modified residues are fairly uncommon. This has to be weighed against the chance of finding a false pattern amongst all residues considered OTHER, that would not exist among those residues individually.

References

[Adamczak et al.(2005)] Rafał Adamczak, Aleksey Porollo, and Jarosław Meller. 2005. Combining Prediction of Secondary Structure and Solvent Accessibility in Proteins. *PROTEINS: Structure, Function, and Bioinformatics* 59, 3 (March 2005), 467–475.

[Ashburner et al.(2000)] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. 2000. Gene Ontology: tool for the unification of biology. *Nature genetics* 25, 1 (2000), 25–29.

[Bepler and Berger(2019)] Tristan Bepler and Bonnie Berger. 2019. Learning protein sequence embeddings using information from structure. *arXiv preprint arXiv:1902.08661* (2019).

[Bepler and Berger(2021)] Tristan Bepler and Bonnie Berger. 2021. Learning the protein language: Evolution, structure, and function. *Cell systems* 12, 6 (2021), 654–669.

[Berman et al.(2000)] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. 2000. The protein data bank. *Nucleic acids research* 28, 1 (January 2000), 235–242.

[Brandes et al.(2022)] Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. 2022. ProteinBERT: A universal deep-learning model of protein sequence and function. *Bioinformatics* 38, 8 (May 2022), 2102–2110.

[Cai et al.(2020)] Yideng Cai, Jiacheng Wang, and Lei Deng. 2020. SDN2GO: an integrated deep learning model for protein function prediction. *Frontiers in bioengineering and biotechnology* 8 (April 2020), 391.

[Chen et al.(2020)] Nanhao Chen, Madhurima Das, Andy LiWang, and Lee-Ping Wang. 2020. Sequence-based prediction of metamorphic behavior in proteins. *Biophysical journal* 1197, 7 (2020), 1380–1390.

[Cheng et al.(2005)] Jianlin Cheng, Arlo Z Randall, Michael J Sweredoski, and Pierre Baldi. 2005. SCRATCH: a protein structure and structural feature prediction server. *Nucleic acids research* 33 (2005), W72–W76.

[Devlin et al.(2018)] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (May 2018).

[Fa et al.(2018)] Rui Fa, Domenico Cozzetto, Cen Wan, and David T Jones. 2018. Predicting human protein function with multi-task deep neural networks. *PloS one* 13, 6 (June 2018), e0198216.

[Floridi and Chiriatti(2020)] Luciano Floridi and Massimo Chiriatti. 2020. GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines* 30, 4 (2020), 681–694. https://doi.org/10.1007/s11023-020-09548-1

[Grechishnikova(2021)] Daria Grechishnikova. 2021. Transformer neural network for protein-specific de novo drug generation as a machine translation problem. *Scientific Reports* 11, 1 (2021), 321. https://doi.org/10.1038/s41598-020-79682-4

[Hochreiter and Schmidhuber(1997)] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[Johansson et al.(2005)] Linda Johansson, Guro Gafvelin, and Elias SJ Arnér. 2005. Selenocysteine in proteins—properties and biotechnological use. *Biochimica et Biophysica Acta (BBA)-General Subjects* 1726, 1 (June 2005), 1–13.

[Jones(1999)] David T Jones. 1999. Protein secondary structure prediction based on position-specific scoring matrices11Edited by G. Von Heijne. *Journal of Molecular Biology* 292, 2 (1999), 195–202. https://doi.org/10.1006/jmbi.1999.3091

[Jumper et al.(2021)] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* 596, 7873 (2021), 583–589. https://doi.org/10.1038/s41586-021-03819-2

[Kabsch and Sander(1983)] Wolfgang Kabsch and Christian Sander. 1983. How good are predictions of protein secondary structure? *FEBS letters* 155, 2 (May 1983), 179–182.

[Kuhlman and Bradley(2019)] Brian Kuhlman and Philip Bradley. 2019. Advances in protein structure prediction and design. *Nature Reviews Molecular Cell Biology* 20, 11 (2019), 681–697. https://doi.org/10.1038/s41580-019-0163-x

[Lin et al.(2022b)] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022b. A survey of transformers. *AI Open* 3 (2022), 111–132. https://doi.org/10.1016/j.aiopen.2022.10.001

[Lin et al.(2022a)] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. 2022a. Evolutionary-scale prediction of atomic level protein structure with a language model. *bioRxiv* (2022). https://doi.org/10.1101/2022.07.20.500902 arXiv:https://www.biorxiv.org/content/early/2022/10/31/2022.07.20.500902.full.pdf

[Lobanov et al.(2010)] Michail Yu Lobanov, Eugeniya I Furletova, Natalya S Bogatyreva, Michail A Roytberg, and Oxana V Galzitskaya. 2010. Library of disordered patterns in 3D protein structures. *PLoS Computational Biology* 6, 10 (2010), e1000958.

[Lobanov and Galzitskaya(2011)] Michail Yu Lobanov and Oxana V Galzitskaya. 2011. The Ising model for prediction of disordered residues from protein sequence alone. *Physical biology* 8, 3 (June 2011), 035004.

[Merriam-Webster([n. d.]]words-in-english Merriam-Webster. [n. d.]. https://www.merriam-webster.com/help/faq-how-many-english-words

[Mikolov et al.(2013)] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[Mucchielli-Giorgi et al.(1999)] Marie-Hélène Mucchielli-Giorgi, S Hazout, and P Tuffery. 1999. PredAcc: prediction of solvent accessibility. *Bioinformatics (Oxford, England)* 15, 2 (1999), 176–177.

[Paszke et al.(2019)] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[Petsko(2000)] Gregory A. Petsko. 2000. The Grail problem. *Genome Biology* 1, 1 (2000), comment002.1. https://doi.org/10.1186/gb-2000-1-1-comment002

[Rives et al.(2021)] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. 2021. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences* 118, 15 (2021), e2016239118.

[Rose et al.(2011)] Daniel A. Rose, Reecha Nepal, Radhika Mishra, Robert Lau, Shabn Gholizadeh, and Brooke Lustig. 2011. Novel Application of Qualitative Predictors: First Stage Calculation of Solvent Accessible Residues Using Complementary Protein Sequence Entropy and Other Parameters. *Proceedings of the 22nd International Workshop on Database and Expert Systems Applications* (2011), 70–74.

[Rother and Krzycki(2010)] Michael Rother and Joseph A Krzycki. 2010. Selenocysteine, pyrrolysine, and the unique energy metabolism of methanogenic archaea. *Archaea* 2010 (2010).

[Sen et al.(2005)] Taner Z Sen, Robert L Jernigan, Jean Garnier, and Andrzej Kloczkowski. 2005. GOR V server for protein secondary structure prediction. *Bioinformatics* 21, 11 (2005), 2787–2788.

[Staudemeyer and Morris(2019)] Ralf C Staudemeyer and Eric Rothstein Morris. 2019. Understanding LSTM–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586* (2019).

[Strauss(2019)] Benjamin Strauss. 2019. *Predicting Switch-like Behavior in Proteins using Logistic Regression on Sequence-based Descriptors.* Master's thesis. San Jose State University, 1 Washington Square, San Jose, CA 95192.

[Szegedy et al.(2015)] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper With Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Tian et al.(2021)] Hao Tian, Xi Jiang, and Peng Tao. 2021. PASSer: prediction of allosteric sites server. *Machine learning: science and technology* 2, 3 (2021 2021), 035015.

[Van Der Spoel et al.(2005)] David Van Der Spoel, Erik Lindahl, Berk Hess, Gerrit Groenhof, Alan E. Mark, and Herman J. C. Berendsen. 2005. GROMACS: Fast, flexible, and free. *Journal of Computational Chemistry* 26, 16 (2005), 1701–1718. https://doi.org/10.1002/jcc.20291 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.20291

[Vaswani et al.(2017)] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[Wang et al.(2010)] Zhiyong Wang, Feng Zhao, Jian Peng, and Jinbo Xu. 2010. Protein 8-class secondary structure prediction using conditional neural fields. In *2010 IEEE International Conference on Bioinformatics and Biomedicine (BIBM).* IEEE, 109–114.

[Zhou et al.(2018)] Yuan Zhou, Udit Gupta, Steve Dai, Ritchie Zhao, Nitish Srivastava, Hanchen Jin, Joseph Featherston, Yi-Hsiang Lai, Gai Liu, Gustavo Angarita Velasquez, et al. 2018. Rosetta: A realistic high-level synthesis benchmark suite for software programmable fpgas. *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (2018), 269–278.