Rational design inspired application of Natural Language Processing algorithms to red shift mNeptune684

Scott Parkinson

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for degree of
Master of Science Mathematics and Statistics[1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1]The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Abstract

Recent innovations and progress in machine learning algorithms from the Natural Language Processing (NLP) community have motivated efforts to apply these models and concepts to proteins. The representations generated by trained NLP models have been shown to capture important semantic and structural understanding of proteins encompassing biochemical and biophysical properties, among other key concepts. In turn, these representations have demonstrated application to protein engineering tasks including mutation analysis and design of novel proteins. Here we use this NLP paradigm in a protein engineering effort to further red shift the emission wavelength of the red fluorescent protein mNeptune684 using only a small number of functional training variants ("Low-N" scenario). The collaborative nature of this thesis with the Department of Chemistry and Biomolecular Sciences explores using these tools and methods in the rational design process.

Dedication

To Jane, Avery, and Rowan for the motivation to pursue knowledge, again.

Acknowledgements

I would like to extend my sincere appreciation to Dr. Maia Fraser, Dr. Roberto Chica, Sandrine Legault, Compute Canada, Arne Fabrituis (for sequence variants), the TAPE team (for code and pretrained models) and the UniRep team (for code and pretrained models).

Contents

List of Figures

List of Tables

Chapter 1

Overview and Motivation

1.1 What we are attempting.

This is a light and relaxed introduction to motivate the subject and goal of this thesis. In this work, we attempt to create a new protein that has never before existed to do something that we want it to do. We aim to improve upon evolution so we may achieve a stated human goal. This effort takes place in the vastness of possible protein space while constrained by the physics of Nature. Specifically, our task is to take an existing red fluorescent protein and increase its maximum emitting wavelength thereby shifting it further into the red light spectrum.

The implementation we lay out is rooted in recent developments in the machine learning approaches that have been so successful in natural language processing (NLP). Consider what happens as the mind attempts to fill the blanks in the following sentence: "Unfortunately my _____ broke down and I was late for _____." People have enough language and life experience to determine several reliable possibilities to fill in the missing words. Adults may consider the word pair "car/work" suitable. Alternatively, students might consider "bike/school" as better choices. Notice importantly, that context matters and the other words in the sentence give this context in addition to the individual's own world perspective and knowledge of language.

Recently, training machine learning models on "fill-in-the-blank" tasks similar to the above have led to significant progress in NLP problems. By training on enormous datasets with powerful computers and appropriate algorithms, the models build a representation of language that captures the structural semantics and contextual understanding of words. This allows the models to generate new text, translate, write in certain styles and act as chatbots, among many other tasks. The results have become surprisingly good and have been useful in many domains.

But our focus is proteins, not language. Proteins are the building blocks of biological life. They are chains of amino acids that are linearly assembled and sub-

sequently fold into a specific 3-dimensional structure inducing certain functionalities. By writing down the order, or sequence, of the amino acids for any given protein based on an alphabet of 20 letters, we can see the analogy that proteins are literally the language of life. The letters representing the amino acids link together to make words or sentences that encode for actual 3-dimensional proteins. These proteins interact together in Nature to ultimately tell a story that is life as we know it.

Due to this correspondence with language, it becomes obvious to at least attempt an NLP approach with proteins. Instead of training the models on language text, they are trained on now widely abundant protein sequences. Instead of learning the concepts of language, the models learn about proteins in a way that allows them to be useful for tasks of interest to protein designers. Instead of a language translation task, we will explore the task of changing, or mutating, a protein sequence to make the protein be a little bit more of what we want it to be.

An additional challenge in our project is that while there exists plentiful sequence data to train the language model responsible for generating the representations, the specific task model (Top Model) to predict red shift only has access to a small amount of data, on the order of 100 sequences, for construction. This is termed the "Low-N" environment and is common to many protein engineering tasks. So, while we can leverage the extensive data available to construct a robust deep learning language model to generate representations for proteins, we will only be using a simple linear Top Model trained with our small number of local samples. To be effective, this will demand the language model representations summarize and encode essential information about proteins both generally and specifically for our task at hand. We will see this is just the case.

This work has been cross-disciplinary with the Department of Chemistry and Biomolecular Sciences and benefits from the insights available from the rational protein design framework. The rational design approach uses extensive knowledge of proteins to recommend sequence changes, often confirmed by detailed computational

models. We will show this framework is highly complementary to the machine learning based process. Insights and experience from rational design help focus the protein search space, eliminate proposal sequences based on known scientific rationale and critically evaluate the final sequence proposals using computational analysis. The computational analysis thus serves as a sanity check, to effectively corroborate the machine learning proposals and verify where the model proves effective - and where it may not be. Our aspiration for these machine learning techniques is to demonstrate their usefulness in order to become a new standard tool available for rational protein design.

With that, we proceed to the more formal exposition.

Chapter 2

Introduction

2.1 Natural Language Processing

Natural Language Processing (NLP) involves using computer algorithms to process human language data to create useful representations available to downstream tasks. These representations, or statistical summaries represented by feature vectors, can be useful in a wide variety of tasks such as translation, social media monitoring, text generation and digital assistants. It has a rich history dating back decades and has recently achieved significant breakthroughs leading to accelerating development [53]. Much of the success is attributable to the availability of increasingly more massive training datasets, decreasing computing costs, access to GPUs and TPUs and the advent of algorithms including the Transformer [55] and Long Short Term Memory (LSTM) [25] recurrent neural networks. The model created using one of these algorithms is referred to as a language model and is then a model that will take as input a sequence of words or tokens and generate a fixed length vector representation of the input in feature space.

Along with the increased capabilities have come increasing model size, complexity and training requirements. For example, the recently released GPT-3 [11] autoregressive Transformer based language model contains 175 billion parameters, was trained on 300 billion tokens requiring an estimated 355 years of GPU compute. However, in line with this scaling is a commensurate increase in qualitative and quantitative task measurements.

Inevitably, given the success, it would be an obvious extension to consider other domains to which NLP algorithms and methods could be applied. At minimum, as merely an engineering exercise, to explore the degree to which this approach could work. If successful and proven to be useful, the trial could then be pushed further. To use the modern NLP framework, the problem space must provide access to significant amounts of data that can be expressible in some notion of language. The world of proteins fulfills these requirements.

Proteins are chains of amino acid residues formed from 20 standard amino acids appearing in the genetic code. These 20 standard amino acids can be expressed with an alphabet by assigning each a letter from the English alphabet. Therefore, a protein sequence written out using this alphabet can be considered as a word or sentence in the language itself. Of course, in the natural world, these protein sequences take on 3-dimensional shapes driven by molecular dynamics, and it is the shape of the protein that give it functionality. The protein sequence itself does not obviously indicate the shape or characteristics of the final form. Additional insight is needed to be able to read the sequence and derive its meaning. Indeed, it is not a skill that the human mind has developed, and this is where machine learning can be applied to expose statistically significant patterns from which meaning can be extracted.

Additionally, significant amounts of data now exists for protein sequences. Due to the ability to sequence genetic material and the significant investments to catalog genomic data, there has been exponential growth in protein sequence data availability (Figure 2.1). Note the adjustment in 2015, shown as a sharp drop, was to identify and reduce highly redundant proteomes. There are now datasets, including PFAM [19] and UniRef50 [52], with tens of millions of known protein sequences that can be justifiably used as the basis to train deep NLP algorithms.

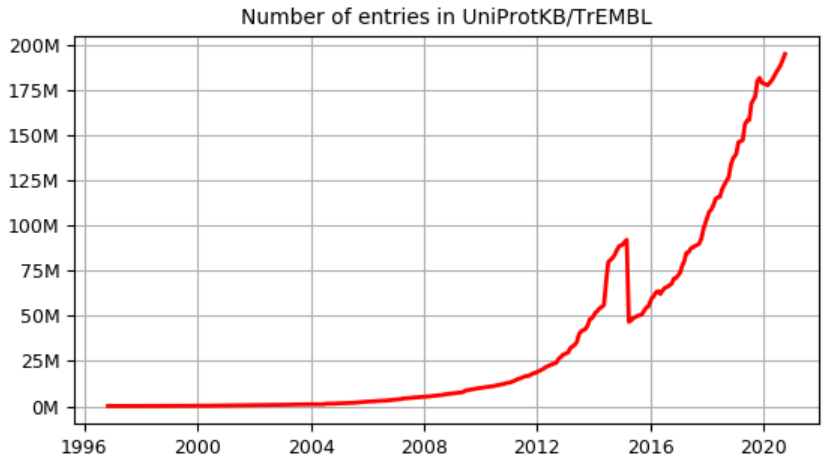

Figure 2.1: Growth of protein sequence availability. Source: UniProtKB.

Almost concurrently, this idea of using NLP techniques on protein sequences was explored and published by several groups who demonstrated the viability of the approach [2][41][42][20][23][34]. Of particular importance are the observations that the model encodings, the statistical summaries of the sequences distilled into a vector representation, contain semantically rich biological information. The model protein embeddings have been shown to encode biochemical properties, organism-level information, secondary structure, evolutionary and functional information, remote homology, information from multiple sequence alignments and higher-order structural features [2][42][54]. These important biological concepts are contained in the encoding and represent some form of knowledge regarding proteins.

This rich biological summary thus becomes potentially useful as a basis for other tasks. These representations have been useful in predicting protein stability, secondary structure, contact maps and mutation effects, and in capturing the information contained in multiple sequence alignments and remote homology [2][41][42]. Additionally, the representations were shown to be valid in predicting the brightness of green fluorescent protein avGFP, a task very similar to the one we are pursuing,

and hence the direct motivation to make use of these representations in our case.

Briefly, the representations from the language model are used as input to another much simpler machine learning model, that we call the "Top Model". Essentially the Top Model sits on top of the language model. The Top Model in our case will be a regularized linear regression model that we train on labeled data, while tuning with cross-validation. Depending on the task, for example, predicting brightness or emission wavelength, different Top Models can be constructed using labelled datasets. Construction of the Top Model is explained in 4.5. The underlying language model, by contrast, is trained on a much larger unlabeled dataset, using an unsupervised task in order to construct representations that capture the universal motifs of proteins. A very simple Top Model can then be used to extract various functional properties from these rich language model representations.

Subsequently, it has been demonstrated that the embeddings from the language models trained in the large data environment can also be useful in the situation where there are only a small number of samples to build a Top Model [9]. This is termed the "Low-N" environment and is the scenario for many protein engineering tasks. It is precisely the case we are in. While built from only a few available sequence samples close to the protein of interest, these smaller and simpler models are constructed on the representations that capture extensive biological information, as noted above. This Top Model can be used to explore the functional landscape of interest *in silico* and predict possible candidate sequences that show gain-of-function [49]. We employ the Low-N framework in this thesis.

It is important to note and acknowledge that this work has only been possible given that two groups, UniRep [2] and TAPE [41] both released access to their code implementations and pre-trained language models on GitHub. Given the resources required to train the language models, this type of research is difficult to extend without the benefit of access to pre-trained models.

2.2 Protein Engineering

Protein engineering can be viewed as a set of methods to accelerate or circumvent evolution to increase the functionality of a protein for some stated goal. As a starting point, it may take an existing protein, make changes to the sequence, subsequently produce the sequences and then evaluate the manufactured proteins for functional improvement. Alternatively, a protein can be designed from scratch, *de novo*, using computational methods to construct a protein that will achieve a specific structure and functionality. We will be focusing on the former in this thesis.

One protein engineering approach known as *mutagenesis* is to introduce random mutations to an initial protein sequence. This method is performed in the lab and requires high throughput screening to analyze the resulting mutated sequences. The high throughput screening is necessary to be capable of analyzing an adequate portion of the protein landscape for reasonable probability of success. When performed repeatedly, using gain-of-function winners in the next round, it is termed *directed evolution*. It is analogous to speeding up Mother Nature.

An alternative approach is to use knowledge of proteins and biochemistry to hypothesize the effect of mutations, known as *rational design*. Rational design will often involve computational tools to evaluate the stability of the hypothesized changes as a guide for choosing which proteins of the hypothesis space to characterize and evaluate. This method is like predicting and then checking Mother Nature.

2.3 Putting it together

The end result of this research is to attempt to create a new protein that will fluoresce at a higher emitting wavelength than those currently available (excluding bacterial based proteins requiring a cofactor to fluoresce). The current highest emitting wavelength protein is mNeptune684 with an emission wavelength of 684nm

falling in the visible light spectrum's red range. Thus, mNeptune684 is aptly considered a red fluorescent protein (RFP) and has been derived from a naturally occurring protein discovered in the sea anemone *Entacmaea quadricolor*. Fluorescent proteins have found extensive use as biomarkers and red fluorescent proteins benefit from lower frequency light that can penetrate mammalian tissue with less damage [43]. As a result, these red fluorescent proteins are beneficial for *in vivo* tissue studies.

As a starting point, we will use mNeptune684 [29] and gather other protein sequences as similar as possible. This small set of proteins, approximately 100, is used to create the Top Model following the Low-N methodology. First, each of the roughly 100 training sequences are fed through the pre-trained language model to output high dimension vectors, the representations, that together form the inputs to fit the linear regression Top Model (Figure 4.1 C). This Top Model will then be used to search for new sequences predicted to have a further red shift in their emission wavelength (Figure 4.1 D).

Rational design will be employed at two points in this process. First, it is used to form a hypothesis based on the interaction of amino acids in three dimensions to reduce the search space to a manageable size that can be exhaustively searched by the machine learning model (Figure 4.1 D). Secondly, the Top Model's final ranked proposals are checked for stability based on computational design calculations (Figure 4.1 E). This computational step confirms whether the machine learning candidates are expected to fold correctly in the natural environment. Together these methods were used to select a final subset of eight proteins to characterize and study in the lab (Figure 4.1 F).

We will be making use of the insights available from rational design to help inform and guide the machine learning component resulting in an interdisciplinary effort. In effect, we are trying to explore the success of combining machine learning tools with the rational design process. We claim it is a natural fit and can be seen to benefit both approaches. It is also hoped that machine learning will generate novel

insights that can extend the knowledge and understanding of proteins.

Chapter 3

Background

3.1 Machine Learning

In the following subsections, we discuss the machine learning concepts and algorithms used in this work. While there has been previous work incorporating machine learning with protein engineering [54][58], the use of Natural Language Processing (NLP) is a relatively recent, highly promising approach of this kind, and will be the focus of this thesis.

3.1.1 Representation Learning

Representation learning is the underlying foundation for the success or failure of this work. Representations of the input data are learned by extracting features from the data that in turn make it easier to perform a downstream task [8]. In this work, the input data are protein sequences, the representation learned by the language model aims to capture features of interest from the protein sequences in the form of high a dimensional vector, and the downstream task of interest is to predict emission wavelength.

Representation learning seeks to capture the important and relevant patterns and variation of the input data, distilling these into a statistically relevant summary leading to easier downstream task learning. Here "easier" means that a given accuracy can be achieved with high probability on a smaller sample size. In some simple situations, this relationship between sample size and probability of success on the downstream task can be captured by measures of complexity, and conceptually, in this case, advantageous representations are ones which reduce complexity. We will require the language model to represent the input protein sequence data in a manner such that the final observable can be predicted by a simple weighting (linear regression) of the high-dimensional vector representation.

The representation-learning role of the language model in our work is shown in Figure 3.1. As discussed in Section 3.1.4, the language models learn a representa-

tion of the data by training on the amino acid prediction task. Throughout both the mLSTM and BERT Transformer architectures, the protein sequences are transformed through linear and non-linear operations building up a high-dimensional vector that becomes the final representation we use as input to the linear regression Top Model.

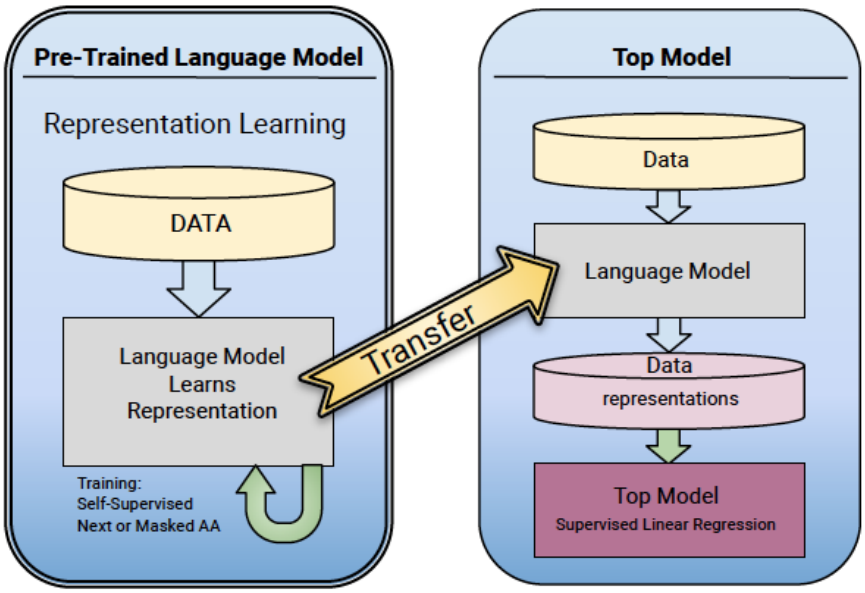

Figure 3.1: Representation and Transfer Learning Summary. First the language model performs a self-supervised task learning useful representations. The language model and learned representations are transferred to the downstream task of supervised emission wavelength prediction using a linear regression Top Model.

There is a dual aspect to the terminology used in representation learning. The word representation can mean both the way of producing the succinct summary, and the summary itself. Stated in our context: it can mean both the language model along with the learned transforms, and alternatively, it may refer to the final high-dimensional vector produced by the language model. Throughout this thesis we may use the two interchangeably, generally referring to the high-dimensional vectors from the language model as the representations. A single protein sequence fed through the

language model is transformed into a high-dimensional vector we term the representation.

3.1.2 Transfer Learning

Transfer learning refers to making use of experience learned on one task as the starting point for another task. The idea is to take advantage of knowledge and understanding that is built up by the first learning task and incorporate this into the design of the second learning algorithm. Technically, the way an algorithm is designed is referred to as its "inductive bias", so we are using experience on the first task to help choose the inductive bias of the second algorithm. One way this can be done is through representation learning, when the representations learned on one task are hoped to be useful for another different task. It has been highly effective in NLP tasks [45] and is at the core of our framework as well.

For this work, the initial task is the training of the language model on amino acid sequence prediction. This language model is then used as the basis to encode the representations used in the subsequent downstream task of supervised emission wavelength prediction. Hence, the knowledge built up by the language model is "transferred" to the task of predicting emission wavelength as shown in Figure 3.1.

3.1.3 Supervised, Self-Supervised and Unsupervised Learning

In our work we will be primarily making use of supervised and self-supervised learning [15]. The distinction has to do with how the target label for training is derived. For supervised learning the label is explicitly provided. In our case, we will have a small number of protein sequences for RFPs that we know the target, emission wavelength, and these sequence/label pairs are the basis of learning. Self-supervised learning will also use a target however the target derives from the data itself. The

language models implement self-supervised learning with a character prediction task where the target, the character to predict (representing the amino acid), comes from the sequence itself. Training proceeds in a similar manner for both supervised and self-supervised learning, in the sense the model will try to predict the target and losses are used to adjust model parameters.

Additionally, there is unsupervised learning [24]. For unsupervised learning there are no labels provided explicitly or derived from the data. Learning attempts to find structure in the data without the use of labels. It may accomplish this by clustering, autoencoders or association. In this work we use unsupervised dimensionality reduction algorithms to aid in visualizing the high dimensional representations.

3.1.4 NLP Algorithms

Here we provide the necessary background to the algorithms and architectures used to train the language models for our work with proteins. In the current modern approach to NLP, there are two distinct families of algorithms employed. The first is the class of Long Short Term Memory (LSTM) recurrent neural networks and we will be using a variant with a multiplicative modification aimed to increase expressiveness, aptly termed a multiplicative LSTM (mLSTM). The second architecture is based on the popular and successful Transformer, an attention-based algorithm.

We use two publicly available frameworks including code and pre-trained models: UniRep [2] and TAPE [9]. The UniRep architecture is mLSTM based, whereas the TAPE framework contains several models including the Transformer based BERT (Bidirectional Encoder Representations from Transformers)[15] in addition to an mLSTM implementation of the UniRep model.

Generally, for NLP applications, language models are first pre-trained on a large dataset using a self-supervised task. Subsequently, for the applicable downstream task, a Top Model is appended to the language model and further training

proceeds using the pre-trained language model weights as the starting point (i.e., transfer learning). Depending on the downstream task, and amount of data available, the pre-trained language model parameters may be frozen or allowed to continue updating during this Top Model training.

Pre-training for the mLSTM and Transformer algorithms proceed in similar, yet slightly different, ways. Both algorithms are pre-trained in a self-supervised manner with a training task to predict the letters of the protein sequence itself. For the mLSTM model, the target letter to predict is the next letter in the sequence given previous letters. Training proceeds linearly from the start of the sequence to the end, each time learning to predict the next character, given what the model has already processed. For the BERT model, the task is to predict a randomly masked 15% of the characters given all the other unmasked characters. In this task, the Transformer can look to unmasked characters both before and after (bidirectional) any character it is attempting to guess during learning.

One of the key challenges for the language model to encode useful representations of proteins is that the linear protein sequence folds into a 3-dimensional shape and this shape is known to affect the protein's functionality. Amino acids close together in 3-dimensional space may significantly affect one another through chemical and physical interactions, yet may be far apart in the input sequence itself. The learning task of predicting a character (amino acid) in the sequence must capture this notion of longer-term sequence dependency. Specifically, to predict an amino acid in the sequence, the model should build some understanding of amino acids that might be close to it in 3-dimensional space. It is this local chemical interaction of amino acids along with their biochemical properties that will give the context to predict what amino acid will best fit in this local environment. Each model architecture has a different route to enabling this notion of context, discussed below.

After pre-training on this basic amino acid prediction task, the language model can transform arbitrary protein sequence input to a representation. This rep-

resentation is a high dimensional summary vector capturing the protein knowledge the model has internalized. It is worth reiterating and emphasizing the impact of this. The language models have learned to encode useful knowledge about proteins by simply performing an amino acid prediction task on lots of protein sequence data. It is the fact that the language model distills such extensive protein knowledge that enables the success of this approach and, in particular, with a small number of labelled training sequences to build the Top Model.

The UniRep and TAPE models are summarized in Table 3.1. Of importance are the significant computational resources involved in training the models. The representation dimension for the UniRep models of 1900 was chosen to be the largest size possible to fit onto the GPU yet still be able to train. Larger dimensions are thought to be more expressive. Notice the number of parameters in the BERT model is roughly twice those of the mLSTMs. Finally, the TAPE repository is of particular appeal since it implements both a Transformer and mLSTM model allowing for direct comparison of the algorithms.

	UniRep	**TAPE mLSTM**	**TAPE BERT**
Architecture	mLSTM	mLSTM	Transformer
Training Data	UniRef50	PFAM	PFAM
Training Seqs	∼24m	∼32.2m	∼32.2m
Training Time	3 weeks	1 week	1 week
Training Computer	4 Nvidia K80	4 Nvidia V100	128 Nvidia V100
Parameters	∼18.2m	∼18.2m	38m
Layers	1	1	12
Heads	-	-	8
Representation Dim	1900	1900	768
Framework	TensorFlow	PyTorch	PyTorch
Training Task	Next aa	Next aa	15% Masked aa
Organization	Harvard	Berkeley	Berkeley

Table 3.1: NLP Model Summary. Essential characteristics for the NLP models used in this thesis.

Additionally, there have been other published efforts to train language models on protein sequences summarized in Table 3.2. These models were not used in this thesis work for various reasons: the models only became recently available, the code has not been publicly released, the codebase does not provide the ability for additional training, or the resources required become too great.

It is important to see the trend towards larger models trained on more data and focusing on the Transformer architecture. The thinking is that, along the success of GPT-3, scaling will lead to better performance; however that may not be the case for any specific task. In particular, the ESM model, larger and trained on more data, does no better on the TAPE fluorescence benchmark [42], a task directly relevant to our engineering task, than either TAPE mLSTM or TAPE BERT.

	ESM	**ProtTrans**	**ProGen**
Architecture	Transformer	Transformer	Transformer
Training Data	Uniparc	UniRef100, BFD	UniParc, TrEMBL
Training Seqs	250m	216m, 2.1b	280m
Training Time	n/a	9.5d/23.5d	2 weeks
Training Computer	128 Nvidia V100	1024 GPUs	256 core TPU
Parameters	42.6m to 669.2m	420m	1.2b
Layers	6-34	30	36
Heads	8	16	8
Representation Dim	768/1280	1024	1028
Framework	PyTorch	PyTorch	TensorFlow
Training Task	15% Masked aa	15% Masked aa	15% Masked aa
Organization	Facebook/NYU	Rostlab/TUM	SalesForce

Table 3.2: Additional NLP Model Summary

3.1.5 mLSTM

The UniRep model [2] chose to use the mLSTM after initial manual explorations and comparisons of LSTM and Gated Recurrent Unit architectures. Here we

explore the details of the mLSTM indicating its possible advantages.

The regular LSTM is an extension to the recurrent neural network (RNN) paradigm and was introduced to overcome issues related to RNNs and their tendency during training for gradients to possibly vanish or "blow up" [27]. The LSTM architecture builds up a representation during training by using a hidden state and a cell state with several gating mechanisms to control the mixing of new and retained information. This mechanism provides for learning long-range sequence dependencies important for protein sequences, given how they fold in 3 dimensions. Usually, the final hidden state is used as the encoded representation for use in the downstream task. However, in this work, following the methodology of [2], for the representation we will be using an average of the hidden states, one for each residue in the protein sequence, thereby integrating information across distant amino acids.

The mLSTM [27] is an effort to improve the expressivity of the LSTM by using the concepts of the tensor RNN and multiplicative RNN (mRNN) [51]. The hidden state of an RNN depends on the weighted input and the learned transition matrix from the previous hidden state. With a tensor RNN, the idea is to have different learned hidden-to-hidden matrices for each input type. In our case, this would result in a different hidden-to-hidden matrix for each of the 20 amino acid types. Generally, to reduce the computational expense of learning many transition matrices, the mRNN simplifies this by learning a factorized transition matrix. Therefore, the mLSTM is a marriage of the LSTM and the mRNN and uses this additional factorized hidden-to-hidden matrix to construct an intermediate state that is then used throughout.

The formulae for the mLSTM are expressed as follows:

$$m_t = (W_{mx} x_t) \odot (W_{mh} h_{t-1}) \tag{3.1.1}$$

$$\hat{h_t} = W_{hx} x_t + W_{hm} m_t \tag{3.1.2}$$

$$i_t = \sigma(W_{ix}x_t + W_{im}m_t) \tag{3.1.3}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_t) \tag{3.1.4}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_t) \tag{3.1.5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(\hat{h}_t) \tag{3.1.6}$$

$$h_t = \tanh(c_t) \odot o_t. \tag{3.1.7}$$

As per the usual LSTM notation, h_t is the hidden state at time step t or in our case residue location t, the various W matrices are all learned weight matrices, σ is the usual logistic sigmoid function, and the gates are i, o, f respectively input, output and forget gates.

Here, m_t is the new multiplicative concept introduced into the standard LSTM model, viewed as an intermediate state. This intermediate state is seen as a revised hidden state that depends on the input x_t (multiplied by the learned weight matrix W_{mx}) and multiplied element-wise by the previous hidden state itself multiplied by a learned weight matrix W_{mh}. It is this intermediate state m_t that is then flowed through the regular gated LSTM. Moreover, this intermediate state m_t affords the mLSTM more expressiveness since it depends on the input residue and is therefore context-dependent.

3.1.6 Transformer

The Transformer [55] is a recent deep learning sequence-to-sequence architecture capable of capturing long-range dependencies through the use of attention. Attention is a mechanism allowing the model to develop context-dependent representations based on other characters in the sequence. The model can learn how much attention to give to other sequence characters and use this to weight representations in a form of mixing.

Notably, the Transformer introduces parallelization to NLP in contrast to

LSTMs, generally required to backpropagate through time with an inefficient loop. This ability to train in parallel allows the Transformer to scale to increasingly massive datasets and ushered in a new era of NLP progress. On the downside, Transformer based models are increasingly large, expensive and challenging to train.

The TAPE implementation of the Transformer is based on BERT: Bidirectional Encoder Representations from Transformers. The BERT model allows the encoder to consider sequence locations (amino acid residues) both before and after the current position of interest. Generally, 15% of the sequence is masked out for the language modelling training task. This allows the model to build up representations considering, or attending to, any other sequence location. In terms of proteins, this could allow the Transformer to learn to incorporate 3-dimensional structure through the attention mechanism applied across the entire protein sequence.

The heart of the Transformer is scaled dot-product attention, expressed in (3.1.8). Here Q, K, V are learned matrices representing the concepts of query, key and value. Effectively, attention is a probability distribution over the representations at each layer of the model, for each sequence position, that learns how best to weight the sequence representations for the purposes of mixing those representations. Each sequence position, amino acid residue in our case, flows through the architecture in parallel building a representation that is contextualized based on all other sequence amino acids representations. The scaling factor is simply the dimension of the representation, in our case 768. Also, as shown in Table 3.1, the implementation we use has 8 heads and 12 layers. The architecture of the Transformer is shown in Figure 3.2.

$$\text{Attention } (Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.1.8}$$

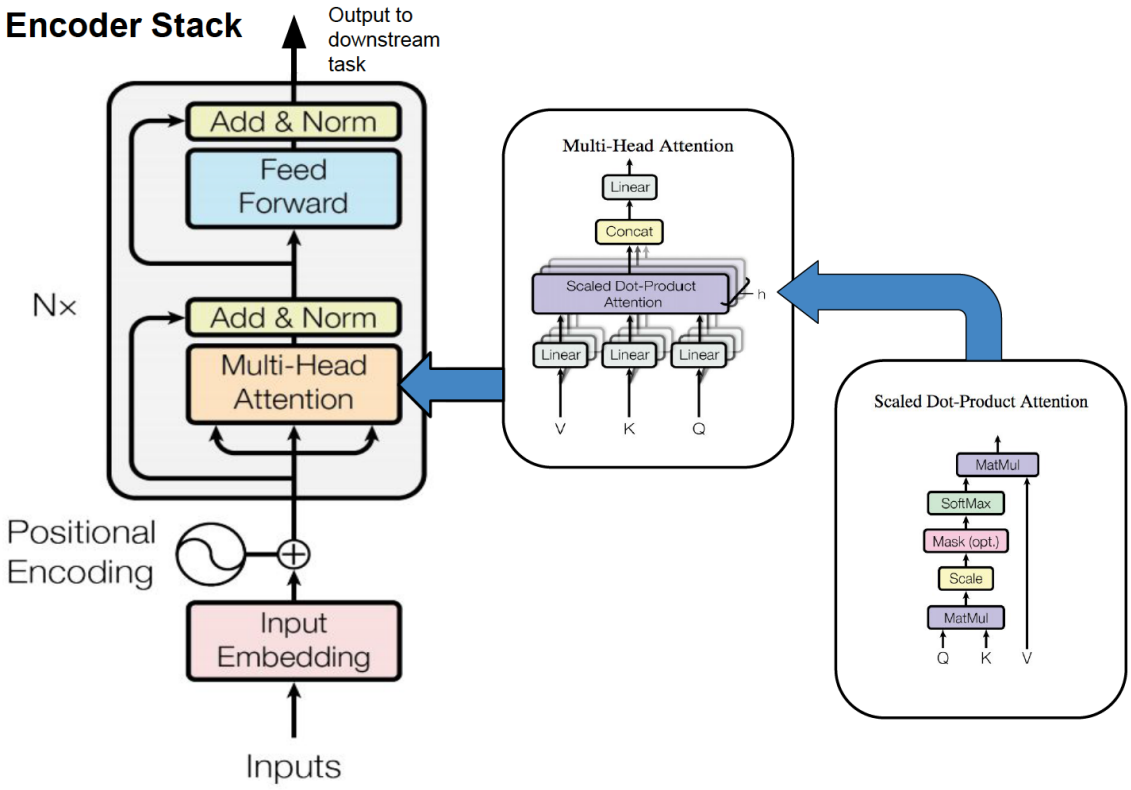

Figure 3.2: Transformer Architecture for the Encoder Stack. The TAPE BERT implementation uses 8 heads and 12 layers. Image amalgamated from [55].

For the final protein sequence representation that will feed into the Top Model, we take the average for the representations of all residues in the sequence from the final layer. This is identical to the methodology for the mLSTM for generating the final representation.

As noted, the Transformer can be difficult to train and several important deep learning techniques are necessary for success including: multiple layers, residual connections, layer normalization, label smoothing and the Adam optimizer. In our work we benefit from access to a pre-trained Transformer model in the TAPE repository.

3.2 Low-N Framework

Once we have access to representations that are obtained from the trained language model, we can use those representations as a starting point to build a task-specific downstream model referred to as a "Top Model". In our case, we will use the vector representations generated from the pre-trained language model on our training data to construct a Top Model that will predict emission wavelength. Higher predicted emission wavelength over baseline is the same as an expectation of red shift, our goal. This construction of the Top Model is a supervised model building process requiring labelled data. See Figure 3.1.

One of our challenges is there is not a significant amount of data available to build the Top Model. This is generally true for protein engineering efforts where the datasets can be small and expensive to create, often only consisting of dozens of sequences. This is coined the "Low-N" framework [9]. A second complication is that the published datasets generally only contain the "winners". During lab experiments, especially mutagenesis, only promising candidates showing potential for improved functionality are retained, studied further and characterized. Poor performing sequences are not worth the investment of time and resources to study and are generally discarded. Hence, there exists a bias towards winners posing difficulties for building reliable Top Models that adequately represent the fitness landscape.

In effect, the Top Model's purpose is to create a vehicle to explore the protein landscape in the vicinity of our protein of interest, mNeptune684. Protein landscapes are very rough and uneven and have been called "holey" due to an abundance of non-functional sequences [44]. In many cases, simply changing one amino acid may prevent a protein from properly folding or exhibiting the functionality we seek. Without some poorly performing sequences as part of the training variants, the model generated fitness landscape may be artificially smooth and less predictive.

The fact that we only have access to a small number of labelled training

sequences limits the class of Top Model we can construct. When using on the order of 100 sequences, it is not feasible to build a reliable deep neural network, for example. For this environment, we are generally limited to linear regression models. Effectively, we are finding a weighting of the high dimensional vector representation created by the language model to predict our quantity of interest.

The Low-N environment also means it is not possible to effectively train back into the language model in an end-to-end fashion. The language models have comparatively too many parameters to attempt this and will quickly lead to memorizing the small Top Model training set and may generalize poorly. It would be possible to train back into the language model and fine-tune the representations given enough data, but that is not our situation. This distinction will arise when comparing published results of the language models allowing backpropagation into the language model. This is explored further in Section 4.2.

A word on expectations given how difficult the task is. The Low-N approach's feasibility was demonstrated by way of retroactive study [9] including predicting brightness of a green fluorescent protein, a task closely related to ours. The paper shows that when building a Top Model with approximately 100 sequences, they achieved an average hit rate of 12.5%. The hit rate is the number of sequences that show improvement over the baseline sequence (the wildtype sequence), in this case from the top 100 ranked candidates from the test set. For machine learning practitioners accustomed to success rates well above 90%, a hit rate of 12.5% may seem like a low success rate, not of much value. However, being well above random, these rates leads to significant cost reductions and time savings and is a step forward from the protein engineering perspective.

It is the Top Model that will be used to explore the fitness landscape and to make predictions about the emission wavelength of a protein sequence in order identify potential gain-of-function sequences. With a Top Model, there are several approaches to explore protein space, and it is important to appreciate the vastness

of this protein space. Our starting protein, mNeptune684, is 244 amino acids long and each position could be mutated to any of 20 amino acids. Combinatorially, this results in 20^{244} possible protein sequences of length 244, which well outnumbers the atoms in the universe, a number far too large to search exhaustively.

One approach to navigate this enormous search space is to limit possible mutations to only specific locations and/or certain amino acids. Another method is to use simulated annealing where we randomly walk through protein space keeping and building on candidates predicted to be good compared to baseline [9]. It could also be possible to consider the "inverse regression" problem and use the Top Model to work backwards to identify sequences that produce the highest output values [32][31]. Our approach will be to use rational protein design insights to exhaustively search a targeted region of protein space informed by knowledge of protein dynamics. This approach is detailed in Section 4.6.

3.3 Proteins

Due to this work's interdisciplinary nature, it is helpful to have a base level understanding of proteins and protein engineering, in particular rational design and computational protein design (CPD). We give that simplified background in the following subsections.

3.3.1 Proteins and Sequences

Proteins are biomolecules made up of chains of amino acid residues and are necessary for life. Generally, we will focus on a single chain of amino acid residues, which are called polypeptides. There are 20 standard amino acids, each made from amine and carboxyl groups and a specific side chain. The instructions to construct proteins are contained in DNA and proteins are synthesized in the ribosome. Once a protein has been created, it folds into a 3-dimensional shape, its conformation. It is this conformation or shape that dictates the function of the protein.

Proteins are often discussed in terms of their different structures. The primary structure is simply the amino acid sequence itself. Local sections of the protein may take on characteristic patterns such as α-helix or β-sheets and these are referred to as the secondary structure. The overall 3-dimensional shape is the tertiary structure and controls the basic functionality of the protein. When several protein molecules group together, this is referred to as the quaternary structure. Our work will mostly involve the primary and tertiary structure.

With the advent of increasingly sophisticated genetic sequencing tools, particularly stemming from to the Human Genome Project, it became possible to quickly and inexpensively obtain protein sequences. This led to exponential growth in the number of sequences available and they have been amalgamated into various public databases, including UniProtKB [52] and PFAM [19] (see Figure 2.1). It is important to note that while there are hundreds of millions of sequences, a much smaller num-

ber of those have been characterized in terms of their function or structure. In many cases, there are only guesses as to how a sequence may behave with these guesses coming from examining similarities between sequences.

In fact, exploring the evolutionary record is a well-established method to understand the nature and relationship of proteins. It is possible to trace the evolutionary record back to a common ancestor and then study the descendants' proteins, noting similarities or differences. This can often give a clue to the protein of interest's functionality and structure if already known in one of the organisms. This is called *sequence homology*. This is generally approached by studying sequence similarity by aligning proteins and building multiple sequence alignments.

Multiple sequence alignments (MSA) can help derive contact maps and, subsequently, the protein structure. When noting differences in sequence alignments, if the occurrence of mutations at positions is correlated, this can be an indication those positions are in closer proximity in 3-dimensional space. This is the case as a mutation in one location often triggers mutations in other nearby locations.

3.3.2 Red Fluorescent Proteins

Red fluorescent proteins (RFPs) are an important class of proteins used extensively in biomedical research as biomarkers and for fluorescence imaging [43] [38]. They were derived from sea anemones and corals and subsequently enhanced through significant protein engineering efforts [36]. One advantage of RFPs is they absorb and emit longer wavelength light spectrum energy with the resulting benefit that the light can penetrate deeper into mammalian tissue and is less damaging. This is the case since hemoglobin has a high absorption rate of light below 650nm [50]. It remains a goal to further red shift the wavelength emission of these proteins. This is the goal of the present research.

RFPs are so-called β-barrels and can be understood as taking the shape

of a tin can with a filament in the centre (see Figure 3.3). It is this filament, the chromophore, that fluoresces and emits red shifted light. The chromophore results from the maturation of three residues in an autocatalytic process [56]. Red FPs have been studied extensively and there is a significant amount of scientific understanding regarding these proteins [17].

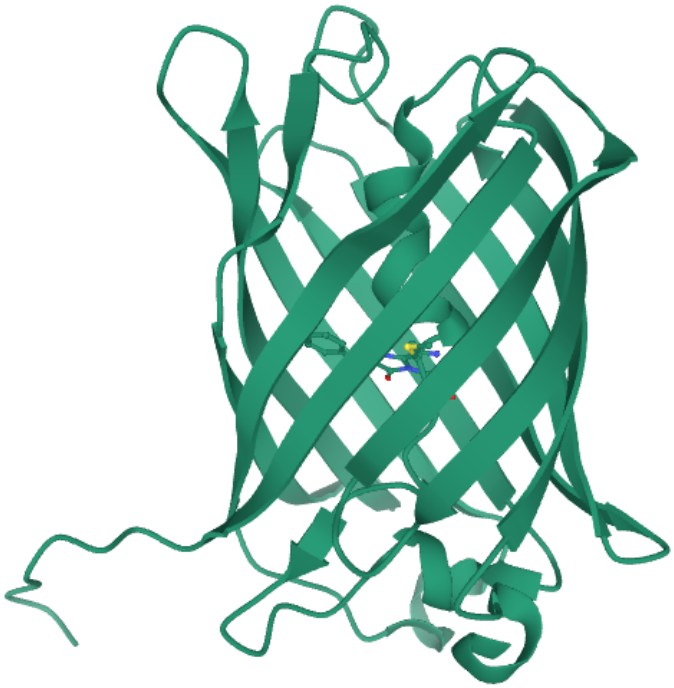

Figure 3.3: RFP mNeptune684 3-dimensional rendered structure. The chromophore is shown as a stick figure in the centre. Source: Protein Data Bank 5YT1.

The RFP mNeptune684 [29] is currently the longest emitting wavelength RFP (not requiring cofactor biliverdin for chromophore maturation) and will be our starting sequence to attempt further red shift. mNeptune684 is a monomeric protein published in 2016 derived from the sea anemone *Entacmaea quadricolor*. It was discovered using mutagenesis of mNeptune along with structure-based rational design and has excitation wavelength 604nm, emission wavelength 684nm, brightness 1.17

and quantum yield of 0.03 [28]. Its crystallized structure was determined using X-ray diffraction to a resolution of 2 angstroms, available in the Protein Data Bank with identifier 5YT1.

3.3.3 Protein Engineering

Protein engineering is the attempt to create proteins to achieve some stated goal. The two main approaches are directed evolution and rational design.

Directed evolution attempts to mimic evolution by randomly mutating the protein sequence and subsequently examining the effects for improvement. The hope is that, by chance, there will be success in creating a protein that enhances the functionality of interest. Any gain-of-function variants can then be the starting point for additional rounds of mutagenesis to continue the process. This repetition captures the notion of directing the evolutionary process. Several methods include random mutagenesis, error prone PCR and cassette mutagenesis. These techniques have been employed extensively to enhance the functionality of many fluorescent proteins [43].

Ideally, the directed evolution involves some form of high-throughput screening technique in order to scale efficiently. For example, Fluorescence-Activated Cell Sorting (FACS) can be used to characterize on the order of 10^5 sequences as with the green fluorescent protein avGFP [47]. When it is possible to characterize ample data, directed evolution can be combined with machine learning to explore local fitness landscapes *in silico* and can be an effective combination to guide follow-on rounds [10]. We make use of the dataset from [47] in order to explore and compare the effectiveness of our NLP models in Section 4.2.

An alternative to directed evolution is to use knowledge of the protein structure and biochemistry to predict the residue locations most likely to affect functionality, termed *rational design*. Hypotheses can be formed to evaluate these sites of interest and explored with site-directed mutagenesis or computational models. Using

structure as a guide for rational design of RFPs has been used effectively to improve brightness, maturation and photostability [17]. Rational design was also used to increase quantum yield in the red FP mRojaA [37].

Computational Protein Design (CPD) seeks to find protein sequences expected to stabilize to a given model structure [1]. Given a set of atomic coordinates for the backbone chain of the protein structure, sequences varying from the baseline may be evaluated for stability *in silico* using a potential energy function. To accomplish this evaluation, first a backbone template must be established, usually from a previously crystallized structure. Next, a library of side chain configurations, called *rotamers*, is chosen along with a potential energy function to evaluate the sequence and configurations. Finally, since solving for the minimum energy rotamer configuration is NP-hard, a stochastic method to search the possible rotamer configurations such as Monte Carlo simulation is used. Together these steps provide a computational approach to solve the inverse folding problem: what sequences are expected to fold to a given structure.

CPD is a powerful technique [5] and has been previously used in protein engineering efforts to extend the emission wavelength of RFPs [12].

In this work, CPD will provide an additional check on candidate sequence proposals from the machine learning model. CPD will verify any proposal has preferable energetics and is expected to conform to the desired backbone structure of mNeptune684. We will be using an ensemble of backbone structures, providing for the more natural variance of final shapes the protein may conform to and this is termed *multistate CPD*.

Chapter 4

Methodology

4.1 Introduction

The goal of this thesis, as described above, is to successfully identify protein sequences that are predicted to exhibit red shift relative to mNeptune684. Our approach uses the high dimensional representations from the trained language model as the input to a linear regression Top Model to predict emission wavelength. This follows the methodology of [9] and is summarized in Figure 4.1. Specifically, we chose to use the TAPE mLSTM evoTuned language model as described in 4.2 and 4.3. The representation from the language model was chosen to be the average of the hidden representations of each amino acid, as in [2]. Labelled data for 113 sequences similar to mNeptune684 form the Low-N sequences used to generate the representations subsequently used to fit the Top Model. The Top Model is fit using regularized linear regression with 10-fold cross-validation using the RidgeCV function from sci-kit learn [39]. Subsequently, we use this combined TAPE mLSTM evoTuned and Top Model to predict emmission wavelength for a rationally informed subset of protein space. The predictions are then ranked along with a final computational protein design step confirming stability. In the end 8 sequences were chosen to physically make in the lab and study.

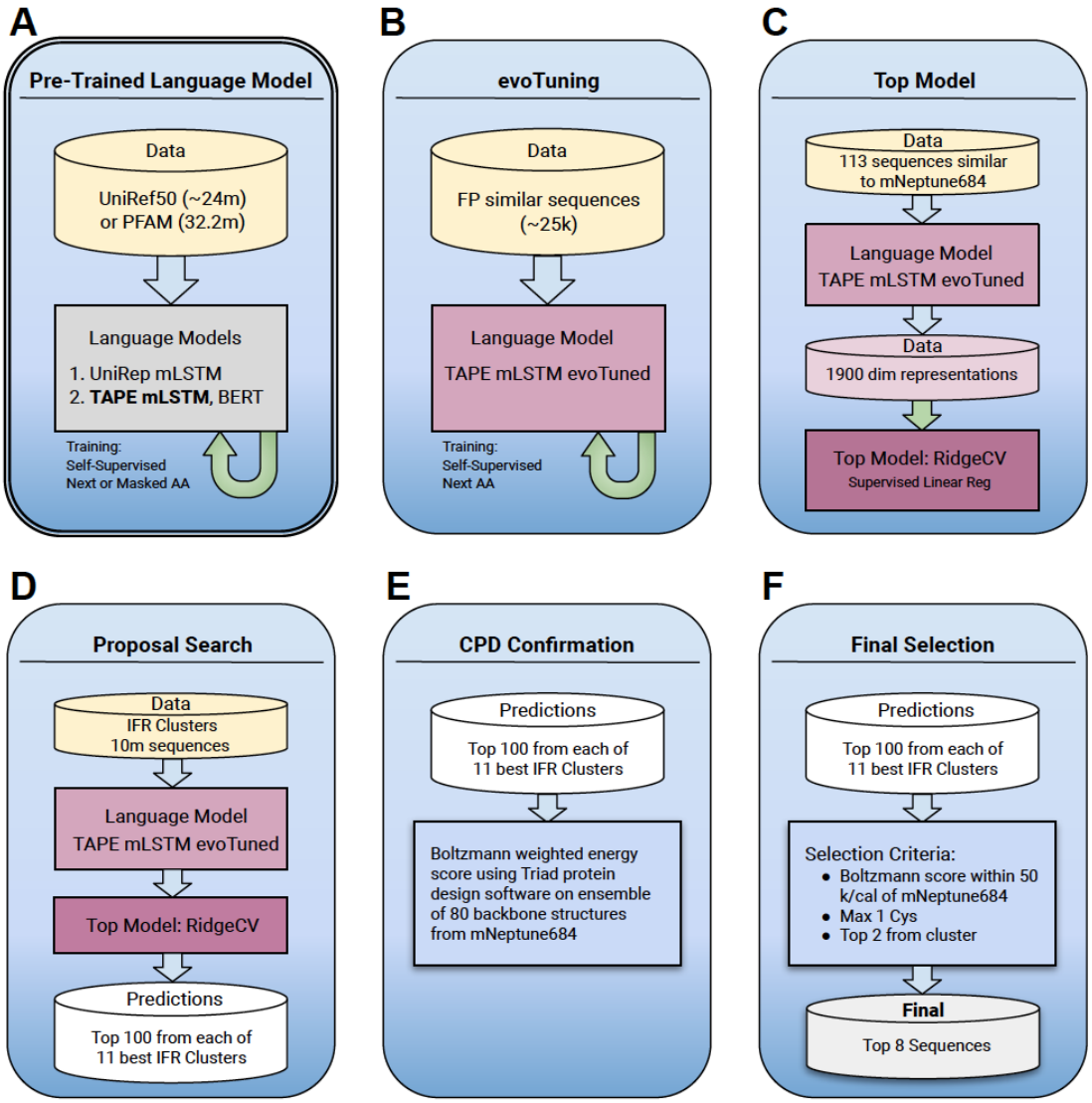

Figure 4.1: Full Pipeline Overview. A) Pre-Trained language models from UniRep and TAPE, trained with self-supervised next or masked token prediction. B) EvoTuning of pre-trained TAPE mLSTM using evoTuning dataset of $\sim$ 25k FP homologs. C) Creation of linear regression RidgeCV top model using representations from fixed TAPE mLSTM evoTuned language model. D) Exhaustive IFR cluster emission wavelength prediction. E) Computational Protein Design confirmation energy calculation. F) Final sequence selection.

4.2 Language Model Choice

In this work, we have access to two pre-trained language model frameworks: TAPE and UniRep. TAPE provides several pre-trained models and a suite of tasks to compare different architectures across the varied protein tasks. UniRep provides an implementation of the mLSTM model. Due to the extensive resources required to train the language models, as previously noted, it is beneficial to start with another organization's computing investment.

The final step of our pipeline involves physically constructing and studying the protein sequences in the lab that the Top Model recommends. The cost of this final step will limit the number of sequences we produce to eight. As such, it will be necessary to choose one language model over the others with some justification. We do not have the budget to comparatively produce sets of proteins from each language model to create and study to evaluate model effectiveness. We first examine the language models available in TAPE to determine whether the mLSTM is better than the Transformer (BERT) for our fluorescence protein engineering task. Subsequently, we compare the versions of mLSTM from TAPE and UniRep.

Comparison of Transformer and mLSTM

It was shown in TAPE [41] in their fluorescence landscape prediction protein engineering task (Task 4) to predict brightness for the fitness landscape of avGFP trained on Sarkisyan data [47], that the Transformer and mLSTM showed similar performance. The Transformer achieved Spearman's ρ of 0.68 and the mLSTM was 0.67 (see their Table 2) [41]. This was an indicator that both models should be considered for our similar protein engineering task.

The first effort is to recreate the published TAPE fluorescence engineering task results. We used TAPE version 0.4 release [41], which is a PyTorch reimplementation of the original published work containing some variation in approach to the

originally published TensorFlow code. We were able to match the paper results in order to establish baselines.

For TAPE BERT we used a batch size 32, learning rate of 1e-5, warmup 50, seed 1210 employing our programmed version of ValuePredictionHeadwithAttention as the Top Model to match the original paper. Under this setup, the resulting Spearman's ρ was 0.676 which compares to the published 0.68.

For TAPE mLSTM (UniRep), we used batch size 16, learning rate of 1e-5, warmup 50, seed 42 and again employing our ValuePredictionHeadWithAttention achieving Spearman's ρ of 0.677, so slightly better than the published 0.67.

Having established baselines to the published results, it is important to observe that loss gradients were backpropagated the full depth of the language models during training. Given the TAPE Sarkisyan [41] dataset's size, with 21,446 sequences in the training set, this is appropriate to the task. However, our task is in the Low-N environment, and with only on the order of 100 sequences to train the Top Model, it is no longer appropriate to backpropagate the training signal into the language model. In our effort, the representations that are produced by the language model will be frozen when training the Top Model. Therefore, it is necessary to check this effect on the baselines.

To understand the effect of training only the Top Model, we employ the same framework; however, all parameters are frozen for the language models. Only the parameters for the Top Model will be trained and hyperparameters have been adjusted to achieve the best result.

For TAPE BERT with batch size of 100, learning rate of 1e-3 with 50 warmup steps, seed 1210 and patience of 10. For this Spearman's ρ drops to 0.565 from 0.676, a significant decrease of .101. Performance is greatly reduced when training is confined to the Top Model.

For TAPE mLSTM with the frozen language model we use a batch size of 100, learning rate of 1e-4, seed 42 and patience of 10. With this setup Spearman's ρ

drops to 0.623 from 0.677, a decrease of 0.054.

Linear regression Top Models using sci-kit learn's RidgeCV [39] were also performed on each language model with fixed language model representations (i.e., no further training of the language model). This linear regression uses 10-fold cross-validation with regularized weight penalization for fitting. Using RidgeCV for the Top Model reflects the final setup that we will employ in our Low-N framework. TAPE BERT produced Spearman's ρ of 0.608 and TAPE mLSTM was 0.625.

Results are summarized in Table 4.1. From the results, it is clear that depending on the type of Top Model selected and whether gradients are propagated into the language model, the mLSTM achieves Spearman's ρ values at least as high as for BERT. This is strong evidence that when the language model's representations are fixed, the mLSTM outperforms the BERT architecture on the fluorescence task.

Training Mix	TAPE BERT	TAPE mLSTM
Published	0.680	0.670
Re-creation	0.676	0.677
Frozen LM	0.565	0.623
RidgeCV	0.608	0.625

Table 4.1: Spearman's ρ model summary for TAPE Fluorescence Protein Engineering Task.

Comparison of Transformer and mLSTM on Low-N Recovery Task

Next, we look at how TAPE BERT and TAPE mLSTM perform on the Low-N recovery task [9]. The Low-N recovery task first constructs a Top Model using only 100 sequences and then checks this model's ability to predict, rank and recover the best sequences from the test set. The recovery is defined as the number of sequences in the top 100 ranked predictions that have a result better than the baseline (wildtype, WT). This is also known as the hit rate, alternatively the percentage of sequences that would be gain-of-function if the 100 proposals were produced in the

lab. This is an important metric as ultimately we seek the best Low-N Top Model at ranking predictions and having the highest hit rate in successfully predicting gain-of-function sequences. Due to the cost of creating sequences in the lab it is usual to only produce a small number on the order of perhaps 100. In our case we will only produce eight.

Results were averaged across 1,000 iterations with the 100 training sequences chosen at random for each run. Results are summarized in Table 4.2. It is clear from the results that TAPE mLSTM is better than TAPE BERT for both the base and evoTuned models (evoTuning is fully described in Section 4.3). This is the second piece of evidence that the mLSTM outperforms BERT on a fluorescence engineering task reflective of what we are trying to achieve. The last line of table 4.2 listed as "test set random" is the hit rate of simply choosing 100 sequences randomly from the test set. It reflects the percentage of testing sequences that are better than baseline.

TAPE Model	**Spearman's ρ**	**> WT in Top 100 Ranked**	**Avg# in actual Top 100**
BERT base	0.234	2.4	0.17
BERT evoTuned	0.245	10.0	1.99
mLSTM base	0.297	3.8	0.48
mLSTM evoTuned	0.487	15.3	1.76
random representations	0.0	1.2	0.12
test set random		3.8	0.00

Table 4.2: Low-N Recovery for TAPE models

We pause at this point to note the strong evidence that the language model representations do in fact encode information in a form suitable to be used by simple Top Models, specifically that they do so better than random high dimensional vectors would. Since, with the Low-N framework, we are fitting a linear Top Model with a small number of sequences using a high dimensional representation (small n, large

p) the problem is ill-posed. As a result, even using random representations instead of the Language Model representations, it is possible for a linear Top Model to fit the training data well. What we must show instead is that these Top Models will perform poorly in the recovery task indicating that they generalize poorly. For this, we construct random representations from a 1,900 dimensional normal distribution with the same mean and variance (across each of the 1,900 dimensions) as the 100 randomly chosen training representations. We use these random representations to perform the Low-N recovery task as above and, again, the results are the average over 1,000 repetitions. As shown in Table 4.2 under "random representations", Top Models fit with random representations perform poorly on the recovery task. This gives a high level of confidence that the language model representations are more useful than random for constructing features that can be successfully be fed into a simple linear Top Model.

Comparison of TAPE and UniRep Frameworks

It is also necessary to compare the mLSTM language models' performance from both the TAPE and UniRep frameworks. As above, we perform the TAPE fluorescence task on the Sarkisyan dataset with stop codons removed, training RidgeCV Top Models on both the full data (Table 4.3) and Low-N environments (Table 4.4). Representations from the language models are fixed (i.e., the language models are frozen) in both cases.

Notice when training on the full training set, the base TAPE and UniRep models achieve identical Spearman's ρ results. The TAPE evoTuned model does better than UniRep evoTuned based on Spearman's ρ and actual number ranked in the Top 100. The evoTuning process is described in Section 4.3, and we speculate that our trained TAPE evoTuned model performs better due to our implementation of evoTuning for TAPE. Additional explanation is provided in Section 4.3.

Also, on the Low-N recovery task, the TAPE evoTuned version is slightly better on the number of sequences in the top 100 that are hits. These are gain-of-function proteins that are better than the baseline sequence.

mLSTM Model	**Spearman's ρ**	**> WT in Top 100 Ranked**	**Avg# in actual Top 100**
UniRep base	0.645	20	3
UniRep evoTuned	0.652	35	6
TAPE base	0.645	20	3
TAPE evoTuned	0.675	33	10

Table 4.3: mLSTM models with RidgeCV Top Model on full TAPE Sarkisyan training data (Not Low-N).

mLSTM Model	**Spearman's ρ**	**> WT in Top 100 Ranked**	**Avg# in actual Top 100**
UniRep base	0.297	3.8	0.52
UniRep evoTuned	0.490	14.0	1.80
TAPE base	0.297	3.8	0.48
TAPE evoTuned	0.487	15.3	1.76
random		3.8	0.00

Table 4.4: Low-N Recovery results for mLSTM models.

Finally, we note that the TAPE framework had the advantage of being more robust and incorporated a BERT model allowing a direct comparison of the mLSTM to the Transformer. On this basis, we proceeded to build the final Top Model and generate proposal candidates using the TAPE mLSTM evoTuned language model.

4.3 EvoTuning

EvoTuning is the process of further fine-tuning the language model on evolutionarily similar sequences to our sequence of interest as described in [2]. The idea

is to fine-tune the generalized protein representations trained on tens of millions of sequences to the area of protein space more representative of the proteins we are focused on (FPs). As was shown in [9] and Tables 4.2 - 4.4, evoTuning provides a significant boost in performance and is critical to the success of the Low-N approach.

The UniRep framework already included evoTuned model versions while TAPE did not and therefore we needed to perform the evoTuning ourselves. For this work, the datasets to further evoTune the TAPE base models were not available and had to be recreated to fine-tune the models. We followed the process as described in [2] and [9] with a small number of required differences: having to use phmmer [18] instead of jackhmmer and truncating fine-tune training to 200 amino acids for the mLSTM (vs 280 in the paper) due to memory constraints. Briefly, jackhmmer is an iterative search tool to search sequence databases for sequence homologs and phmmer is the non-iterative version. The process of generating the evoTuning training data and methods are as follows.

1. Assemble FP sequences for search. FP sequences were sourced from FPbase as of April 8, 2020, UniProt IPR011584 and IPR009017 and PFAM PF01353 and G2F. Sequences were then cleaned by checking amino acid letter validity, eliminating duplicates and retaining only sequences with length less than 500. This created a set of 1470 sequences to be used as the seeds for phmmer searching.

2. Use the phmmer server to generate homolog (evolutionarily similar) sequences. Since jackhmmer was unavailable (as was used in the original papers) at the time of production, we only had access to phmmer. We note that jackhmmer is an iterative search equivalent to phmmer for the first round and that phmmer captures all the same mNeptune684 searched sequences as jackhmmer. The search query used for phmmer was:

   ```
   phmmer -E 0.0000000001 --domE 0.0000000001 --incE 0.0000000001
   --incdomE 0.0000000001 --mx BLOSUM62 --pextend 0.4 --popen 0.02
   ```

```
--seqdb uniprotkb
```

3. Consolidate the results from phmmer. The results from phmmer were filtered for valid amino acids, length less than 500, Levenshtein distance from mNeptune684 less than 400, unique sequences and finally adding back any missing sequences from the original search set. This results in a dataset with 24,786 sequences (compared to 32,225 in [2] and 79,482 in [9]).

4. Create train and test splits. Data were randomly split 90% train and 10% test.

Next, this fine-tuning dataset is used to continue training the pre-trained language models. First, the pre-trained model weights are loaded and training continues on this new dataset using the original language model training task. Conceptually we are simply continuing to train but with new data and modified hyperparameters.

For TAPE mLSTM the hyperparameters used for evoTuning were batch size of 128, learning rate of 1e-4 with 500 warmup steps and the AdamW optimizer, seed 42 and patience of 5 epochs if test loss does not continue to decrease. Training finished after 38 epochs resulting in roughly 5800 parameter updates.

For TAPE BERT fine-tuning we used a batch size of 32, learning rate of 1e-4 with 1000 warmup steps and the AdamW optimizer, seed 42 for 50 training epochs for roughly 34900 parameter updates.

As noted in Tables 4.3 and 4.4 our implementation of evoTuning the TAPE mLSTM model slightly outperforms the pre-trained UniRep mLSTM evoTuned model available from [2]. Our evoTuning dataset was derived using phammer and not jackhmmer as they used (since jackhmmer was unavailable at the time of creation) and our resulting dataset was smaller with 24,786 sequences versus 32,225. It is speculated that our smaller dataset may have given our evoTuned version a tighter region of protein space to fine-tune on and this may have generated the slightly better results on the task given. As corroboration, in [9] it is noted that their eUniRep2 model

evoTuned on an even larger data set of 79,482 evolutionarily similar sequences had a lower hit rate compared to their evoTuning on the dataset with 32,225 sequences.

4.4 Dataset Creation

There are several criteria for assembling a reasonable training dataset under the Low-N framework: ∼100 sequences, if possible, that are close to the sequence of interest (mNeptune684) and, ideally, include negative examples to help shape the fitness landscape. Protein engineering tasks often commence with a round of mutagenesis to gather training data, but we will not be performing that step and instead need to assemble the training sequences from various sources. Our training dataset was constructed as follows:

1. **Fluorescent proteins from the eqFP578 lineage** available from FPbase [28]. Sequences available as of April 8, 2020, were downloaded from FPBase and filtered to create a set of proteins with wavelength emission greater than or equal to 578nm and less than or equal to 684 (i.e., RFPs). Subsequently, using FPbase lineage information the root sequence was manually curated and then the data was filtered for eqFP578, the root for mNeptune684. This lineage are all descendants of *Entacmaea quadricolor* (a sea anemone) and were created through various protein engineering efforts. This generated 42 sequences that hoped to give the model insight into the characteristics that may be important to high wavelength emission and the notion of further red shifting as the lineage has developed.

2. **Data from *Mutagenesis of mNeptune Red-Shifts Emission Spectrum to 681-685 nm Sequences*** [29] from Table 1 (excluding eqFP670 and TagRFP675) and Table 2 yielded 15 additional unique sequences. Of particular importance were several sequences that negatively red shifted in comparison to

mNeptune681. Negative training examples are important to help the model reduce false positive predictions.

3. **Data from *Imaging-Based Screening Platform Assists Protein Engineering*** [21]. From the results of the experimental effort that developed mCarmine commencing from mNeptune684, 52 sequences were made available through correspondence with the authors.

4. **Data from *Autofluorescent proteins with excitation in the optical window for intravital imaging in mammals*** [30]. An additional 4 sequences from Table 2.

Subsequently, the datasets were combined using Numbers and stored as a .csv file to be used in our experimental pipeline. At this stage, additional preprocessing was performed to align naming conventions and generate full amino acid sequences as required. The final dataset contains 113 sequences.

Next, we investigate the relationship between the training dataset sequence representations using UMAP [35] as a visualization tool. We use the UMAP algorithm on the high dimensional (1900 dimension) representations of the training sequences using the TAPE mLSTM evoTuned model. UMAP works by first constructing a high dimensional graph representation, assuming the data exists on a high dimensional manifold, and then creates a lower dimensional graph as structurally similar as possible. UMAP provides a way to visualize the high dimensional data, possibly giving insights to any structure.

In Figure 4.2, we colour the representations in the low dimensional embedding by the source of the data and in Figure 4.3 we colour by emission wavelength. This provides a sense of how the data relate to each other in the high dimensional representations. Notice how the data sources are generally clustered together in Figure 4.2 and similarly higher wavelength emitting sequences in Figure 4.3. This gives some

intuition that the high dimension representations are close together for sequences that are more similar and gives a level of confidence that the language model representations appropriately code for this structure.

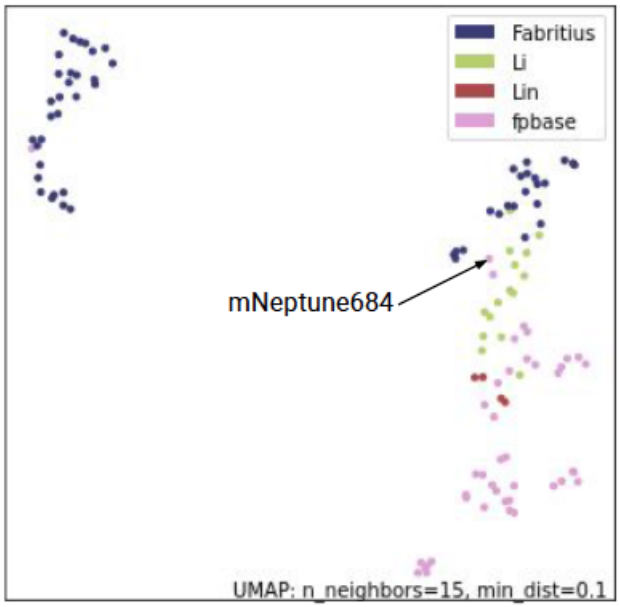

Figure 4.2: UMAP plot of training dataset coloured by data source.

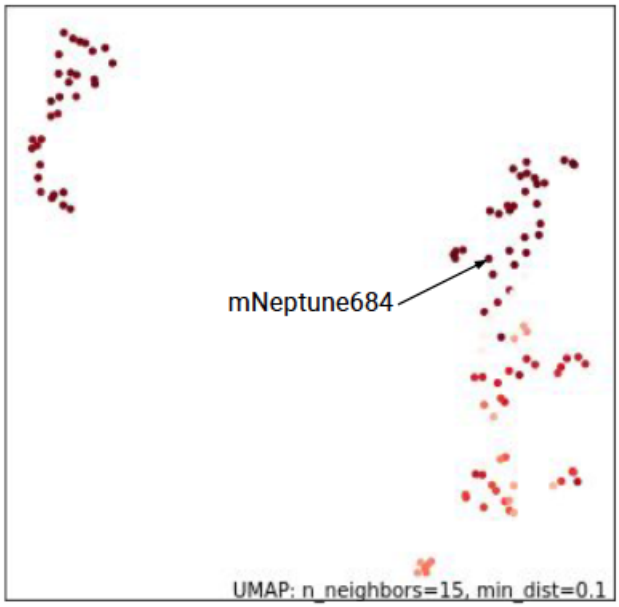

Figure 4.3: UMAP plot of training dataset coloured by emission wavelength, dark red being highest.

We also performed principal component analysis (PCA) on the training set language model representations and noted the high correlation between the first principal component and the target emission wavelength. The Pearson R is calculated as -0.63, Spearman's ρ is -0.71 and coefficient of determination R^2 of 0.4. Figure 4.4 demonstrates how the first axis of variation is reflected in the emission wavelength. This is a strong indication the representations statistically encode information relevant to emission wavelength.

Notice how the highest emitting wavelength sequences are clustered together in the upper left of the plot. The first principal component captures the axis of highest variation of the data and appears to capture the concept of emission wavelength.

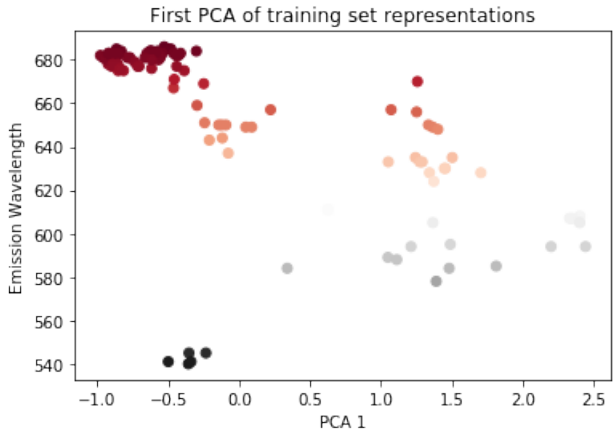

Figure 4.4: First PCA of training dataset language model representations. Sequences are colour coded based on emission wavelength, red being highest.

We revisit the comparison of language model representations to random representations to provide another piece of evidence, this time based on PCA1, suggesting that the language models do encode more useful information than random representations. To demonstrate this, we generate random representations from a 1,900 dimension normal distribution with the same means and variance (across each dimension) as the final training sequences established above. In this case, the PCA1 of random representations fails to show any significant relation to emission wavelength, unlike the language model representations in Figure 4.4. A regression of random representation PCA1 to emission wavelength has R^2 of 0 compared to R^2 of 0.4 for the regression using PCA1 of the language model representations. See Figure 4.5 in contrast to Figure 4.4. This is further evidence that the language model representations encode information relating to emission wavelength.

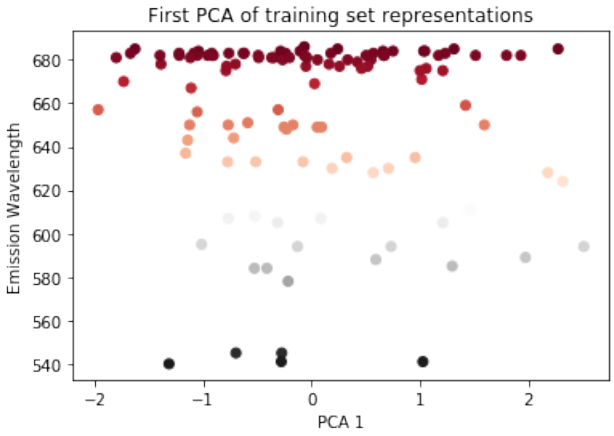

Figure 4.5: First PCA of random representations. The relation is random. Sequences are colour coded based on emission wavelength, red being highest.

4.5 Final Top Model Construction

With the training dataset created and language model for sequence representation selected as TAPE mLSTM evoTuned, we can create our Top Model that will predict emission wavelength and hence determine red shift.

To recap, the procedure is to use the sequences from the training dataset passed through the TAPE mLSTM evoTuned language model that will generate 1900 dimensional vectors for each of the 113 training sequences. Specifically, as the representation, we use the mean of the hidden representations of each amino acid in the sequence after dropping start and stop tokens. These representations are then used as input data to fit a RidgeCV linear regression model with emission wavelength as the target (Figure 4.1 C). RidgeCV is regularized least squares with cross-fold validation. Specific parameters for sci-kit learn's RidgeCV were alpha values ranging from 1e-4 to 100 by 10s, 10 validation folds, fit_intercept = True and normalize = False.

Figure 4.6 shows the fit of the model plotting actual versus predicted emis-

sion wavelength. Points are coloured based on actual emission wavelength.

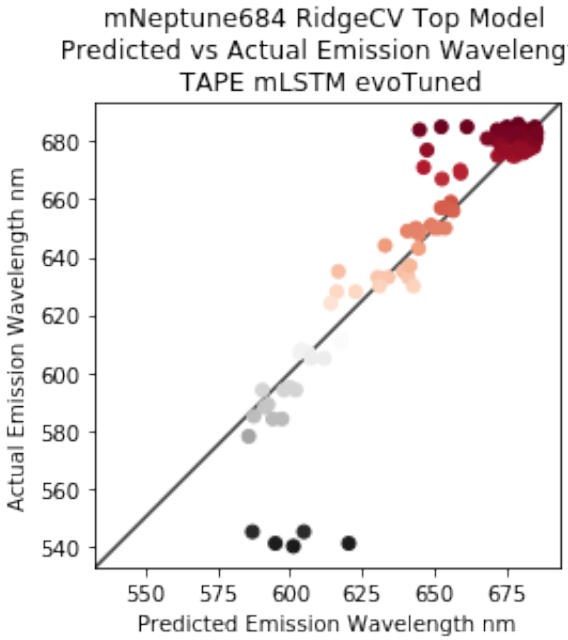

Figure 4.6: Final Top Model: TAPE mLSTM evoTuned.

4.6 Rationally Informed Search Space

Having established the Top Model, we are able to predict the emission wavelength of new protein sequences and subsequently rank them from highest to lowest. This allows us to choose among the top-ranked candidates for a small number of proteins to finally manufacture and study. The challenge now is identifying the protein sequences we would like to input to the Top Model to predict emission wavelength. As mentioned, the size of searchable protein space is vast, and while random searching or simulated annealing approaches [10] have been employed to effect, we will use concepts from rational design. Knowledge of important biochemical and biophysical properties that affect emission wavelength is applied to narrow the search space to one that will be evaluated exhaustively.

As demonstrated in [10], inner-barrel facing mutated residues were more

likely to contain functional non-trivial mutations, mutations that involve greater phenotypic variance and chance of greater impact on protein functionality. In a sense, for these non-trivial mutations, it may be less likely to observe positive results, but when they occur, it can be rewarding. We will be setting our sights towards these more challenging, yet potentially rewarding, locations.

Recall, the chromophore, responsible for fluorescence, is formed by three residues in sequence locations Met63, Tyr64 and Gly65 for mNeptune684. The chromophore can be thought of as the filament inside the β-barrel. Additionally, other amino acids in proximity of, or interacting with the chromophore are also responsible for emission characteristics, while the entire protein makeup dictates the overall global stability of the protein [14].

Our rational design hypothesis makes two assumptions:

1. Residues with side chains directed towards the chromophore have more impact on the chromophore characteristics and therefore greater chance of affecting emission wavelength. Such residues are called *inner-facing residues* (IFRs).

2. When an IFR is mutated, the change will have the greatest impact on those residues closest in 3-dimensional space and whose side chains point towards the IFR's sidechain.

The hypothesis necessitates identifying the IFRs and additional residues that IFR mutations are most likely to affect. Together we will call these *IFR clusters* and will be identified by the IFR sequence number.

To determine if a residue is inner-facing, we consider the α-carbon and β-carbon of the residue. The α-carbon is the residue's carbon atom in the backbone chain that the side chain bonds to, and the β-carbon is the first carbon atom in the side chain. Recall that as the side chain varies it drives the characteristics of that residue. We define an IFR as a residue whose β-carbon is closer to the centroid of

the chromophore than its α-carbon. The centroid of the chromophore is the average of the 3 dimensional coordinates of the atoms comprising the chromophore.

To identify other residues in the IFR cluster, we consider residues where:

1. The distance between the β-carbon of that residue and the IFR's β-carbon is less than 6 angstroms (to capture closeness).

2. The angle between the α-carbon to β-carbon vector of the residue and the vector of the α-carbon of the residue to β-carbon of the IFR must be less than 90 degrees (to captures the concept of "directed towards").

3. We include only the three closest such residues so any cluster contains a maximum of 4 residues (to keep the search space reasonably sized).

For example, residue 158ASN is IFR and has residues 143, 145 and 174 pointing towards the β-carbon of 158ASN. This set of residues 143, 145, 158, 174 forms cluster IFR158. In total, there are 92 IFR clusters resulting in 8,387,300 unique sequences for which to predict the emission wavelength.

To perform the IFR calculations above, the 3-dimensional atomic coordinates for mNeptune684 were downloaded from the Protein Data Bank with identifier 5YT1. Since the residue Glycine does not contain a β-carbon, PyMol [48] was used to replace Glycine with Alanine. This modified structure was then loaded into a Biopython [13] data structure that holds the Euclidean coordinates for each atom in the protein. We subsequently calculate all the pairwise distances between β-carbons creating a contact map. Using the contact map we can identify residues that are within a threshold of 6 angstroms from the IFR of interest and compute if the residue is pointing towards the IFR as described above. Finally, we calculate if the residue is IFR. All results are saved using Pandas into a .csv file.

4.7 Exhaustive Candidate Search with Top Model

Next, using the trained Top Model from Section 4.5 (now frozen), we exhaustively search all possible sequence combinations for each of the 92 IFR clusters identified in Section 4.6 to predict emission wavelengths. Specifically, for the residues in each IFR cluster, each of the 20 standard amino acid mutations is considered. For example, for an IFR cluster with four residue locations, we predict emission wavelength for $20^4 = 160,000$ sequences. All sequences are passed through the TAPE mLSTM evoTuned language model to encode the sequences and this representation is passed through the Top Model for emission wavelength prediction. For each IFR we keep the top 1000 ranked sequences with the highest predicted emission wavelength.

Emission wavelength predictions were made for 8,387,300 sequences taking approximately 5 hours on a single NVIDIA P100 Pascal GPU with 12GB of Ram.

Upon the final ranking of predictions, it was clear some clusters dominated the ranking. Since the input sequence space considers all possible amino acid mutations for a given IFR cluster, the representations tend to be tightly spaced and when a sequence was predicted as strong, it was usually the case that many sequences in the same cluster produced a high predicted output.

To balance the final overall rankings being dominated by single clusters, we ranked the clusters based on the average emission wavelength of the top 5 sequences within each cluster. We retained the top 100 sequences from each of the top ten clusters. The Department of Chemistry and Biomolecular Sciences inspected these top 10 clusters (158, 156, 137, 151, 174, 162, 176, 54, 197, 92) to comment on the quality of the clusters based on knowledge and experience. It was noted that IFR 197 would likely affect emission wavelength as the Arg197 residue interacts directly with the chromophore and has been previously shown to cause a red shift. Also, IFR 143 and clusters containing Asn143, are likely to affect emission wavelength as this residue forms a hydrogen bond with the chromophore.

Three clusters were identified as potentially problematic. IFR clusters 92 and 176 both contain residue Arg92 and this residue is required for the autogenic reactions forming the chromophore. Mutating this residue would likely lead to non-functional proteins. Additionally, it was noted that IFR 151 contains mostly surfaced exposed residues that were not expected to impact emission wavelength.

It was decided then to substitute the next three best IFR clusters. The final top 10 IFR clusters were identified to be (in order of ranking, from highest): 158, 156, 137, 174, 162, 54, 197, 90, 143, 154.

Additionally, we noted that IFR clusters 158 and 197 were predicted to exhibit red shift and both contain residue Asn143. These two IFR clusters could be considered a larger joint cluster given the shared residue. Using the seven residue locations of this combined cluster we searched a reduced sample space using the top 8 (on average) residue substitutions at each residue location from the initial search. This amounted to roughly 2.3 million sequences. The Top Model did predict these combined cluster sequences to have high emission wavelength and so we included the top 100 of these sequences in our final ranked set for consideration.

In total, the proposal candidate list contains 1,100 sequences from the 10 IFR clusters above and combined IFR cluster 158/197.

Dimensionality reduction using UMAP to visualize these 1,100 sequences in addition to the training dataset of 113 sequences is provided in Figure 4.7. Notice clusters 143 and 197 are embedded close to the training set.

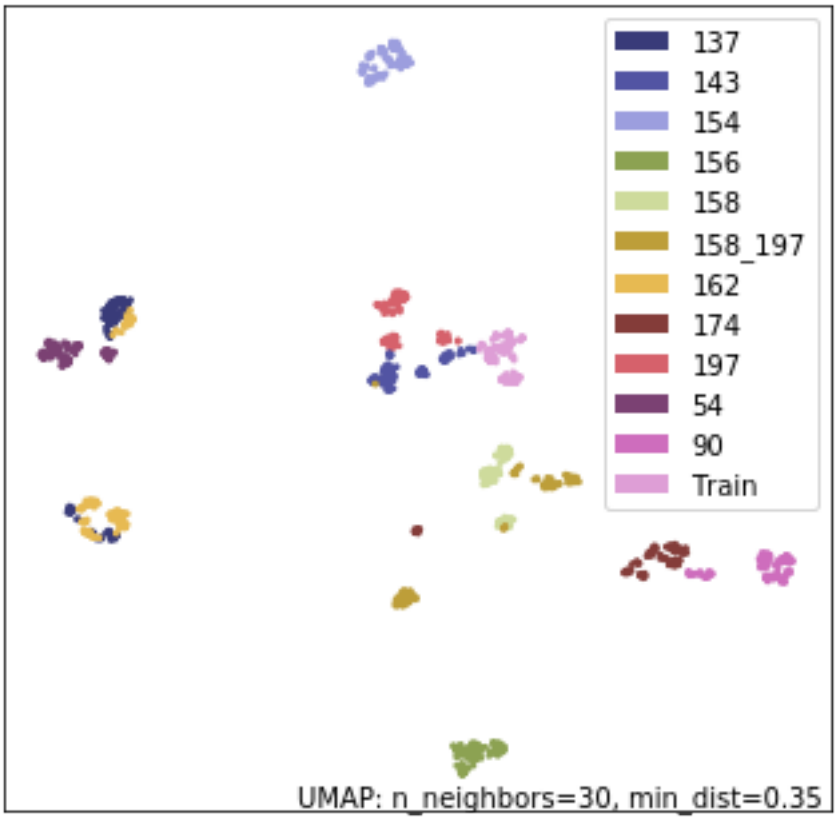

Figure 4.7: UMAP embedding of candidate and training sequences.

4.8 Computational Protein Design Confirmation

Next, we use Computational Protein Design (CPD) to further differentiate the sequences by scoring their energetics. Proteins seek to minimize their free energy, and when a sequence is scored to have unfavourable free energy (for a given structure) that sequence is less likely to fold to the required shape. CPD is used to optimize the protein sequence and will indicate incompatible mutations with the structure resulting from clashes based on size, shape or charge of the mutations. CPD will help indicate which of our proposal candidates are more likely to fold to the required shape and can therefore be used to further rank our candidates for final selection.

The list of 1,100 sequences from Section 4.7 was passed to the Department of Chemistry and Biomolecular Sciences for the CPD calculations. They have a well-established pipeline for performing these calculations using the Triad protein design software [40] based on using backbone templates and a library of possible side chain configurations called rotamers. First, an ensemble of 80 backbone structures was generated using molecular dynamics restrained by the electron density of the mNeptune684 crystal structure. This ensemble represents a range of slightly different shapes that the final structure may take, and hence we will want to consider the mutations for each of these different backbones.

Next, for each member of the backbone ensemble, we thread rotamers from a rotamer library attempting to identify the most energetically favourable combination optimizing over 1 million iterations using Monte Carlo simulation. The best free energy is recorded as the score and then averaged across the ensemble using Boltzmann weighting for the final Boltzmann score.

Scoring the potential energy is performed by a physics-based model that considers the interaction energies between individual rotamers and the template as well as pairwise interaction between rotamers. The model accounts for force terms including van der Waals, hydrogen bonding, electrostatic, and terms for implicit solvation. The lower the energy score the better.

Each of the 80 backbone ensemble members required 3 to 6 hours of simulation to calculate the Boltzmann score for the 1,100 sequences. The process can be run in parallel and in total there were roughly 350 hours of computation performed for this step on the compute cluster (Dell PowerEdge R410 1U servers, 2x hex core 2.40GHz Xeon E5645 CPUs with 32 GB Memory).

Finally, the Boltzmann score was used in conjunction with the Top Model ranked emission wavelength predictions to determine a final list of eight sequences to characterize (physically create and study). Final filtering captures the idea of identifying sequences predicted for high emission wavelength by the Top Model and

are energetically favourable as determined by CPD. One limitation with the CPD model is that scoring may be biased for Cys residues (i.e., they are scored well since they occupy a smaller 3-dimensional space) and hence a stipulation of at most a single Cys mutation is allowed.

Final sequence selection criteria for the eight final candidates from the 1,100 candidate proposals is as follows:

1. Boltzmann score within 50 kcal/mol of mNeptune684 Boltzmann score.
2. Does not contain more than 1 Cys residue mutation.
3. Top 2 for the cluster.

After evaluating over 10 million sequences, the final eight sequences selected are (with mutations relative to mNeptune684 PDB 5YT1 ordering):

Mutations	**IFR cluster**	**Intracluster Rank**
W90Y, Q106C, T176S	90	54
Q106C, T176N	90	39
N143T, N158C	143	83
N143A, R197C	143	79
N143A, N158C, L174K	158	68
N143T, E145V, N158C, L174K	158	43
N143S, R197T, L199C, A217M	197	64
N143S, R197C, L199V, A217M	197	48

Table 4.5: Final Eight Sequence Proposals

It was observed that there appeared to be a negative correlation between the Top Model ranking and the Boltzmann score. In particular, for highly ranked sequences in each cluster based on emission wavelength, the Boltzmann score would not necessarily be the lowest. This was verified by a regression model of the Boltzmann score against the predicted emission wavelength resulting in Spearman's ρ of

0.253. Remember, the Boltzmann score is negative for the best sequences and so here the Spearman's ρ is confirming the negative relationship observed. This fact may help target where the Top Model can perform better. In particular, it shows that simply taking the top-ranked candidates from each cluster may not lead to optimal results since those candidates are less likely to take on the desired structure, as reflected in their Boltzmann scores. This is noted by the Intracluster Rank (rank of sequence within the top 100 for their cluster) of the final sequences chosen in Table 4.5. This check of the Boltzmann scores may help to reduce false positive Top Model predictions.

In a sense, the CPD calculations are a reality check on the Top Model predictions. The Top Model may be strong at identifying which IFR clusters show promise and ranking a set of sequences but may not be ideal for differentiating these final candidates. CPD can be seen as providing a second opinion to the Top Model candidates, considering the sequences in a way the Top Model may not capture or cannot statistically differentiate. It is this synergy between the two approaches making them complementary and mutually valuable. The tools of machine learning can be used to construct a model that will predict the wavelength emission of a protein, which is something CPD does not do. While CPD is geared to, and much better at, indicating which sequences will likely conform to the final required shape.

Chapter 5

Results

5.1 Results

The results from the wet lab characterisation are show in table 5.1. We were not successful in producing a gain-of-function result. Of the 8 sequences that were produced only 2 of them fluoresced, with fluorescence less than our baseline sequence. Six of the sequences did not have chromophores that matured as required for fluorescence.

Mutations	**Prediction**	**Actual**
W90Y, Q106C, T176S	694.7	-
Q106C, T176N	695.8	-
N143T, N158C	693.6	655
N143A, R197C	693.8	-
N143A, N158C, L174K	712.1	600
N143T, E145V, N158C, L174K	712.6	-
N143S, R197T, L199C, A217M	699.8	-
N143S, R197C, L199V, A217M	700.1	-
mNeptune684	680.1	675

Table 5.1: Results

Chapter 6

Discussion

6.1 Discussion

mLSTM vs Transformer

With the recent dominance of the Transformer architecture in NLP tasks, it is somewhat unexpected the mLSTM was shown to be the better language model choice for our application. Current thinking suggests the Transformer would provide the better solution, yet this wasn't the case. We explore several reasons why this may be.

An obvious difference between the two model implementations is the size of representation: the mLSTM is dimension 1900 while the Transformer is dimension 768. It is assumed, heuristically, that larger representations will be richer and more expressive [2]. If this is the case, then the 1900 dimension representation in the mLSTM may be capable of providing richer encoding of features affecting emission wavelength that can be linearly exploited by our Top Model.

Another difference may relate to vocabulary size. Recall, there are 20 standard amino acids and so the alphabet used in our language model is limited to 20 characters or tokens. The training task for pre-training the language models, based on predicting single amino acid residues, is limited to these 20 tokens. The usual setup for an NLP task, at which Transformers excel, involves token vocabularies that are significantly larger, on the order of 20,000. It may be that since we are using a small and targeted vocabulary, the mLSTM is as expressive as the Transformer.

Additionally, it is possible the mLSTM captures a better inductive bias for our specific protein engineering task. On natural language tasks, LSTMs have been shown to capture hierarchical structure by learning from the bottom up, relying on effective representations of shorter "children" [46]. It is possible this better reflects the motifs and structural units of a protein over the attention-based mechanism of the Transformer.

We also showed how evoTuning is critical for the success of our Low-N im-

plementation. Recall that evoTuning involves continued pre-training of the language model and is a form of fine-tuning. It could certainly be that case that fine-tuning is more easily performed on the mLSTM than the Transformer. Fine-tuning on the Transformer appears to be very much an art with significantly different results depending on seeds used as well as often arbitrary number of training epochs (often just 3)[16][15]. There does not appear to yet be a standard proven methodology for fine-tuning the Transformer. Given the significance of evoTuning for final results, this difference could give the mLSTM the advantage.

It should be noted we are only discussing the superior performance of the mLSTM on our specific task. On other protein tasks the Transformer has been shown to produce better results than the mLSTM, and also benefits from scaling, echoing results on natural language tasks [41][42]. So, differences may very well be task dependent. Future protein engineering tasks will have to continue to use care in selecting the appropriate model.

The Transformer has been shown to produce better results on tertiary contact map prediction (i.e., predicting 3-dimensional structure) than the mLSTM, especially as the Transformer is scaled [42]. Certainly, knowledge of structure is important to understanding the nature of the protein and will impact functionality but to what degree is not so clear. It was shown for the task of predicting brightness of the green fluorescent protein avGFP using the Sarkisyan dataset, protein structure information was negligible in model performance [22]. Advantages the Transformer may provide in terms of predicting and understanding 3-dimensional shape may not be advantageous for our task. It is possible the mLSTM may do a better job at feature extraction more relevant to emission wavelength.

Also, while the Transformer has been shown to scale efficiently and successfully on some tasks, it does not always lead to better downstream results [20]. In particular they note LSTMs can perform similarly at downstream tasks requiring fewer parameters and resources. As shown in Table 3.2, the ProtTrans model is mas-

sive, approximately ten times larger than the TAPE model we use. This effect was also shown in [42], where massive scaling showed no benefit on the TAPE fluorescence task. This all points to the idea that scaling may not improve prediction for our task, but rather capturing the correct features from protein sequences is paramount. One of the key benefits of the Transformer architecture, scaling, may not be so relevant in this work.

EvoTuning

As noted, evoTuning played a critical role in the success of the Low-N framework. Our implementation of evoTuning for the TAPE language models used a smaller evoTuning dataset than published accounts [9], yet produced better results on our recovery task analysis. Because our evoTuning training set is smaller, it is reasonable to expect that it reflects a tighter region of protein space. It could be appropriate for future tasks to explore the size of evoTuning dataset employed, with the goal of maximizing task-specific representation expressivity.

Why does the Low-N framework work?

The Low-N framework has been shown to be useful for protein engineering. As discussed, the success rests upon the richness of the representations learned by the language model, to such a degree that a simple linear weighting can lead to useful predictions. But why is this framework successful and what role do the Low-N variants play in fitting the Top Model?

To address this, we first note that protein sequences, the inputs to our model, are information rich. It was postulated by Anfissen [6] that protein sequence should fully determine structure; that primary sequence alone should contain enough information to predict the 3-dimensional shape. This was recently demonstrated with the success of AlphaFold in CASP14 [26] where they used protein sequence,

along with Multiple Sequence Alignments, resulting in a significant breakthrough in protein structure prediction. It is also the case that structure is important for functionality, so it is natural to assume the same protein sequence inputs also contain information relating to function. In our exploratory work we have shown this to be true, however, the extent is not fully known.

We drew comparison between protein sequences and natural language by noting they both use an alphabet forming words that have meaning, but protein sequences result from a significantly different process than natural language. Natural languages are created by humans to communicate and share ideas while proteins are physical structures created by Nature through evolution for some function relating to overall environmental fitness. Proteins are naturally occurring chains of amino acids, and while humans can choose to record them as a sequence of letters, this does not imply further similarities with natural language nor that they may have similar fundamental underlying structure. It isn't clear why models effective at NLP tasks should preform well on protein sequences based on surface similarities. AlQuraishi has speculated a "linguistic hypothesis" [4] wherein the space of naturally occurring proteins occupy a special manifold stemming from the evolutionary pressure to reuse working fragments at many scales. This manifold may in turn have hierarchical organization with structure similar to linguistics with a grammar. Under this hypothesis, we can start to understand why the NLP algorithms successful on natural language may be useful on proteins: the underlying structures may have similarities.

The analogy between natural languages and protein sequences was further strengthened in [59]. Here they use bigram analysis on protein domains to conclude evidence of a "quasi-universal grammar" relating to proteins. This grammar, or structure, having potential similarities to linguistic structure, may be enabling the success of the NLP algorithms.

It is hypothesized that the Low-N framework succeeds by two mechanisms [9]. First, by learning to probabilisitically identify protein sequences that are more

likely to exist in nature. Second, the use of a small set of additional proteins similar to the sequence of interest (the Low-N variants) to learn a linear weighting of the representations which can provide predictions greater than baseline (i.e., gain-of-function sequences).

The hypothesis suggest that self-supervised pre-training allows the language model to capture features from the naturally occurring proteins in the large training set, in turn enabling the model to understand and predict which sequences are more natural and likely to exist. This ties in to the manifold concept in AlQuraishi's linguistic hypothesis above. It is conceivable that known proteins, existing in Nature, occupy a lower dimensional manifold of possible protein space. As was shown in [9], there is strong correlation between the language model's log-likelihood of the sequence and the first principal component (PCA1) of the language model's representations, which was in turn correlated to brightness in avGFP. In other words, when the model predicted a higher probability for the sequence existing, it also corresponded to that sequence having brighter fluorescence.

However, while there is a relationship between brightness and PCA1 for avGFP, this relationship is unable to differentiate sequences with brightness greater than baseline (i.e., show improvement). Recall, PCA1 is an unsupervised dimensionality reduction technique that learns a linear weighting of inputs along the axis with highest variation. By using a small number of labelled training examples (the Low-N training variants), a different linear weighting of the same representations may be learned by the Top Model. This change in the learned linear weights, based on the additional information from the labels, enables the Top Model to make predictions that can differentiate sequences greater than baseline.

The explanatory mechanism could be as follows. As noted in [9], "the meaningful information contained in the mLSTM representations are entangled". PCA is an unsupervised technique and [33] shows such techniques are in general unable to disentangle representations, unless they are lucky enough to already incorporate a

suitable inductive bias. This is exactly where the Low-N variants are of use: they help to better extract meaning from the entangled features, in our case with linear weights, under a supervised approach.

To summarize, we noted protein sequences are information rich and they hypothetically have structure somehow similar to linguistic grammar, allowing proven NLP algorithms to learn useful representations. Next we showed how the self-supervised learning of the language model captures the concept of proteins that are more probable to naturally exist and these representations are further correlated with emission wavelength. However, a small amount of supervised learning is needed to learn a better Top Model that can differentiate gain-of-function sequences. Together these concepts help to explain the success of the Low-N framework presented here.

Synergy between CPD and ML

As previously noted, CPD and ML can be viewed as complementary to each other. Their combined use allows fast *in silico* exploration of protein space: ML based models can predict the characteristic of interest and CPD can confirm through stability calculations that the proposals are expected to fold to the required shape. In this work we were able to efficiently explore a rational design hypothesis, evaluating 10 million sequences, reduced to a smaller set of 1100 subsequently evaluated by CPD. This is an order of magnitude above the efficiency of traditional directed evolution.

In our pipeline there are two natural places to include human expertise and computational biochemistry to improve the ML model. Rational design expertise helped to reduce the search space in Section 4.6 and in Section 4.8 we see how CPD lends computational biochemistry insight to evaluation of the proposals. The collaborative nature of this project is echoed in the pipeline interplay between ML, rational design and CPD.

Concurrently with the writing of this thesis, a paper was published echoing

the use of CPD as we employ in our work. In their paper [57], they make use of CPD to improve their previous conception of machine learning applied to directed evolution. They employ calculations of the free energy to aid in identifying protein sequences to use in the construction of their Top Model for a well-established localized protein landscape (4 residues) for protein GB1. They show the use of free energy to identify training sequences greatly improves the success of their Top Model predictions. Conversely, in our work, we already have a small set of labelled data to construct our Top Model, and subsequently employ free energy calculations to aid in selecting final protein sequence proposals from a wide area of protein space.

Future Directions

Our work points to several interesting avenues for future research, including practical and theoretical aspects. On the practical side a better understanding of the evoTuning process would give a higher level of confidence to such an important component of the framework. As we have noted, is the size of the evoTuning dataset important with smaller perhaps better? As well, evoTuning (and fine-tuning in general) with the Transformer needs to be better understood. The dataset used to pre-train the language model is worth considering as well: is bigger better or would a set more specific to organism domain yield better representations [3]? Additionally, there is room to explore appropriate dimensionality reduction techniques on the high dimensional language model representations, to the extant that it aids in constructing better Top Models.

On the theoretical side, deeper understanding of the causal factors driving protein functionality and how to best create useful disentangled representations would be impactful. A possible effective step in this direction, for text, with Transformers has recently been taken with [7]. Our Top Model is created with linear weightings of language model representations that we have no conceptual understanding of. It is

not possible to point to significant weightings of the representations as features that impart understanding to us. The black box is closed. A step towards causal models would help open this up. Currently, only through retrospective biochemistry effort, by studying what sequences are successful and why, is knowledge potentially gained from this implementation.

Another theoretical area of interest would be to better understand where the Top Model makes reliable predictions - and where it doesn't. We saw through the use of CPD that the Top Model makes proposals that don't always have ideal expectation of properly folding to the required shape. How can we create better, more optimal Top Models? Practically, incorporating CPD-calculated free-energy into the regression target could improve Top Model performance.

Chapter 7

Conclusion

7.1 Conclusion

In this work we used the Low-N machine learning framework in combination with rational design and CPD to exhaustively search and rank 10 million protein sequences in an effort to red shift mNeptune684. Our work shows the synergy of ML and CPD and points to ML as being an effective tool for rational protein design saving time and resources. The nature of this cross-disciplinary work shows how the tools and knowledge from one domain can greatly enhance the efforts in the other.

The contributions of this thesis are to demonstrate that the mLSTM architecture performs better than the Transformer BERT architecture on a specific Low-N recovery task in turn raising interesting questions concerning architecture suitability and how this relates to the nature of the protein task. We also confirmed evoTuning is critical for optimal results and that a smaller evoTuning dataset may provide advantages. Finally, we drew a new connection to why the Low-N training variants are required for predicting gain-of-function protein sequences.

While we were not successful in creating any gain-of-function sequences among the 8 produced and studied in the wet lab, we remain optimistic the approach has merit.

Bibliography

[1] James A.Davey. Multistate computational protein design: Theories, methods, and applications, 2016.

[2] Ethan C. Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M. Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods*, 16(12):1315–1322, 2019.

[3] Jose Juan Almagro Armenteros, Alexander Rosenberg Johansen, Ole Winther, and Henrik Nielsen. Language modelling for biological sequences - curated datasets and baselines, 2020.

[4] Mohammed AlQuraishi. Protein linguistics, 2018.

[5] Oscar Alvizo, Benjamin D Allen, and Stephen L Mayo. Computational protein design promises to revolutionize protein engineering. *BioTechniques*, 42(1):31, 33, 35 passim, January 2007.

[6] CB Anfinsen. Principles that govern the folding of protein chains. *Science (New York, N.Y.)*, 181(4096):223–230, July 1973.

[7] Anonymous. Disentangling representations of text by masking transformers. In *Submitted to International Conference on Learning Representations*, 2021. under review.

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives, 2014.

[9] Surojit Biswas, Grigory Khimulya, Ethan C. Alley, Kevin M. Esvelt, and George M. Church. Low-n protein engineering with data-efficient deep learning. *bioRxiv*, 2020.

[10] Surojit Biswas, Gleb Kuznetsov, Pierce J. Ogden, Nicholas J. Conway, Ryan P. Adams, and George M. Church. Toward machine-guided design of proteins. *bioRxiv*, 2018.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[12] Roberto A. Chica, Matthew M. Moore, Benjamin D. Allen, and Stephen L. Mayo. Generation of longer emission wavelength red fluorescent proteins using computationally designed libraries. *Proceedings of the National Academy of Sciences*, 107(47):20257–20262, 2010.

[13] Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 03 2009.

[14] Peter Dedecker, Frans C. De Schryver, and Johan Hofkens. Fluorescent proteins: Shine on, you crazy diamond. *Journal of the American Chemical Society*, 135(7):2387–2402, 02 2013.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[16] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping, 2020.

[17] Matthew G Eason, Adam M Damry, and Roberto A Chica. Structure-guided rational design of red fluorescent proteins: towards designer genetically-encoded fluorophores. *Current Opinion in Structural Biology*, 45:91 – 99, 2017.

[18] Sean R. Eddy and the HMMER development team. Hmmer: biosequence analysis using profile hidden markov models, 2020.

[19] Sara El-Gebali, Jaina Mistry, Alex Bateman, Sean R Eddy, Aurélien Luciani, Simon C Potter, Matloob Qureshi, Lorna J Richardson, Gustavo A Salazar, Alfredo Smart, Erik L L Sonnhammer, Layla Hirsh, Lisanna Paladin, Damiano Piovesan, Silvio C E Tosatto, and Robert D Finn. The Pfam protein families database in 2019. *Nucleic Acids Research*, 47(D1):D427–D432, 2019.

[20] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing, 2020.

[21] Arne Fabritius, David Ng, Andreas Michael Kist, Mutlu Erdogan, Ruben Portugues, and Oliver Griesbeck. Imaging-based screening platform assists protein engineering. *Cell chemical biology*, 25(12):1554, 1561.e8, December 2018.

[22] Sam Gelman, Philip A Romero, and Anthony Gitter. Neural networks to learn protein sequence-function relationships from deep mutational scanning data. *bioRxiv*, 2020.

[23] Michael Heinzinger, Ahmed Elnaggar, Yu Wang, Christian Dallago, Dmitrii Nechaev, Florian Matthes, and Burkhard Rost. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics*, 20(1):723, 2019.

[24] Geoffrey Hinton and Terrence J. Sejnowski. *Unsupervised Learning: Foundations of Neural Computation*. The MIT Press, 1999.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[26] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Kathryn Tunyasuvunakool, Olaf Ronneberger, Russ Batesand Augustin Žídek, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Anna Potapenko, Andrew J Ballard, Andrew Cowieand Bernardino Romera-Paredes, Stanislav Nikolovand, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Martin Steinegger, Michalina Pacholska, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. In fourteenth critical assessment of techniques for protein structure prediction (abstract book), 2020.

[27] Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative LSTM for sequence modelling. *CoRR*, abs/1609.07959, 2016.

[28] Talley J. Lambert. Fpbase: a community-editable fluorescent protein database. *Nature Methods*, 16(4):277–278, 2019.

[29] ZhaoYang Li, ZhiPing Zhang, LiJun Bi, ZongQiang Cui, JiaoYu Deng, Dian-Bing Wang, and Xian-En Zhang. Mutagenesis of mneptune red-shifts emission spectrum to 681-685 nm. *PloS one*, 11(4):e0148749–e0148749, 04 2016.

[30] Michael Z Lin, Michael R McKeown, Ho-Leung Ng, Todd A Aguilera, Nathan C Shaner, Robert E Campbell, Stephen R Adams, Larry A Gross, Wendy Ma, Tom Alber, and Roger Y Tsien. Autofluorescent proteins with excitation in the optical window for intravital imaging in mammals. *Chemistry & biology*, 16(11):1169–1179, 11 2009.

[31] Johannes Linder, Nicholas Bogard, Alexander B. Rosenberg, and Georg Seelig. Deep exploration networks for rapid engineering of functional dna sequences. *bioRxiv*, 2019.

[32] Johannes Linder and Georg Seelig. Fast differentiable dna and protein sequence optimization for molecular design, 2020.

[33] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. A sober look at the unsupervised learning of disentangled representations and their evaluation, 2020.

[34] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R. Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *bioRxiv*, 2020.

[35] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[36] Ho-Leung Ng and Michael Z Lin. Structure-guided wavelength tuning in far-red fluorescent proteins. *Current opinion in structural biology*, 39:124–133, 08 2016.

[37] Antonia Pandelieva, Miranda Baran, Guido Calderini, Jenna McCann, Véronique Tremblay, Sabina Sarvan, James Davey, Jean-François Couture, and Roberto Chica. Brighter red fluorescent proteins by rational design of triple-decker motif. *ACS chemical biology*, 11, 12 2015.

[38] Antonia T. Pandelieva. Increasing the quantum yield of red fluorescent proteins using rational design, 2016.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] Protabit. Triad protein design software suite.

[41] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John F. Canny, Pieter Abbeel, and Yun S. Song. Evaluating protein transfer learning with TAPE. *CoRR*, abs/1906.08230, 2019.

[42] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, 2020.

[43] Erik Rodriguez, Robert Campbell, John Lin, Michael Lin, Atsushi Miyawaki, Amy Palmer, Xiaokun Shu, Jin Zhang, and Roger Tsien. The growing and glowing toolbox of fluorescent and photoactive proteins. *Trends in Biochemical Sciences*, 42, 11 2016.

[44] Philip A. Romero, Andreas Krause, and Frances H. Arnold. Navigating the protein fitness landscape with gaussian processes. *Proceedings of the National Academy of Sciences*, 110(3):E193–E201, 2013.

[45] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, 2019.

[46] Naomi Saphra and Adam Lopez. Lstms compose (and learn) bottom-up, 2020.

[47] Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, Natalya S Bogatyreva, Peter K Vlasov, Evgeny S Egorov, Maria D Logacheva, Alexey S Kondrashov, Dmitry M Chudakov, Ekaterina V Putintseva, Ilgar Z Mamedov, Dan S Tawfik, Konstantin A Lukyanov, and Fyodor A Kondrashov. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 05 2016.

[48] LLC Schrodinger. Pymol the pymol molecular graphics system, version 1.8, schrodinger, llc. November 2015.

[49] Sam Sinai and Eric D Kelsic. A primer on model-guided exploration of fitness landscapes for biological sequence design, 2020.

[50] Andrew M Smith, Michael C Mancini, and Shuming Nie. Bioimaging: second window for in vivo imaging. *Nature nanotechnology*, 4(11):710–711, 11 2009.

[51] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 1017–1024, Madison, WI, USA, 2011. Omnipress.

[52] Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. Uniref clusters: a comprehensive and scalable alterna-

tive for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

[53] Amirsina Torfi, Rouzbeh A. Shirvani, Yaser Keneshloo, Nader Tavaf, and Edward A. Fox. Natural language processing advancements by deep learning: A survey, 2020.

[54] Serbulent Unsal, Heval Atas, Muammer Albayrak, Kemal Turhan, Aybar C. Acar, and Tunca Dogan. Evaluation of methods for protein representation learning: A quantitative analysis. *bioRxiv*, 2020.

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[56] Vladislav V Verkhusha, Dmitry M Chudakov, Nadya G Gurskaya, Sergey Lukyanov, and Konstantin A Lukyanov. Common pathway for the red chromophore formation in fluorescent proteins and chromoproteins. *Chemistry Biology*, 11(6):845 – 854, 2004.

[57] Bruce J. Wittmann, Yisong Yue, and Frances H. Arnold. Machine learning-assisted directed evolution navigates a combinatorial epistatic fitness landscape with minimal screening burden. *bioRxiv*, 2020.

[58] Kevin K. Yang, Zachary Wu, and Frances H. Arnold. Machine learning-guided directed evolution for protein engineering, 2019.

[59] Lijia Yu, Deepak Kumar Tanwar, Emanuel Diego S. Penha, Yuri I. Wolf, Eugene V. Koonin, and Malay Kumar Basu. Grammar of protein domain architectures. *Proceedings of the National Academy of Sciences*, 116(9):3636–3645, 2019.