ECM3401: Final Report

Attentive Neural Protein Structure Prediction

Prediction of protein tertiary structure via the translation of amino acid sequences into internal torsional angles.

Jack Marsh
College of Engineering, Mathematics and Physical Sciences
The University of Exeter

1 May 2019

Abstract

Proteins are abundant in virtually all biological processes within the cells of, not just humans, but all living organisms. Sequences of amino acids undergo a spontaneous transition into their final 3-dimensional structure, called folding. Structure determination is important as the structure dictates the function of the protein. This report devises a novel method of protein structure prediction that draws on recent successes in the fields of Neural Machine Translation and Natural Language Processing, applying them to the field of Bioinformatics. The proposed method is a deep attentive encoder-decoder network that translates embedded ngrams of amino acid sequences into torsion angles, feeding them into the pNeRF algorithm for a Cartesian representation. The results are then benchmarked against the contestants of CASP13.

I certify that all material in this dissertation which is not my own work has been identified.

Contents

1 Introduction

Proteins are ubiquitous in a vast number of biological processes in the cells of all organic organisms. They have various functions ranging from antibodies that bind to foreign particles, such as viruses and bacteria, enzymes that act as catalysts in chemical reactions within cells or messengers that transmit signals to coordinate biological processes [1]. They also provide structural support for cells and can transport and store atoms and molecules within them. Computational models of protein folding are vital as the structure of a folded protein dictates its function. Protein structure prediction is an active field of research that remains elusive, captivating the minds of many researchers for over half a century. The fundamental principles of folding have practical applications in the exploitation of the advances in genome research, in the understanding of different pathologies and in the design of novel proteins with special functions, just to name a few.

The costly nature of experimental methods of protein structure determination forms the necessity to develop accurate and efficient computational models instead. Since its inception in 1994, Critical Assessment of protein Structure Prediction (CASP) has run every 2 years for precisely this reason. The competition is conducted in a double-blind fashion; neither the predictors nor the organizers and assessors know the structures of the target protein at the time the predictions are made. This provides the benchmark for comparing the predictive methods and is how the model devised in this report will be assessed.

Currently, no explicit sequence-to-structure mapping exists that is scalable. Molecular Dynamics simulations are accurate but inefficient and Template Modelling is efficient but inaccurate where no close homolog exists. However, Google DeepMind entered CASP for the first time this year with AlphaFold. Much to the surprise of the regular contestants, this deep neural network approach trounced the competition by a statistically significant margin. The model devised in this report intends to capitalise on the success of deep neural networks applied to protein structure prediction, taking inspiration from AlphaFold and other deep learning approaches. The objectives of this research are to conduct a *de novo* predictor that is faster than Molecular Dynamics simulations and more accurate than *ab initio* Template Based Modelling.

The report frames the task of protein structure prediction as a Neural Machine Translation Problem, taking an embedded representation of a sequence of amino acids that have been split into ngrams and translating them into a sequence of discretized internal torsional angles, with an attentive encoder-decoder. The predicted sequence is then fed into the parallelised Natural extension Reference Frame (pNeRF) algorithm to convert the dihedral angles into Cartesian coordinates. The paper shows the promise of sequence-to-sequence models for protein folding, benchmarking the results against those of the contenders of CASP13.

2 Background

2.1 The Folding Process

Under physiological conditions, a protein undergoes a spontaneous transition called folding. Proteins are linear polymers built from a chain of 20 possible amino acids that consist of backbone atoms and a side chain, which differentiates them. The side chain determines the hydrophobicity, charge and polarity of the amino acid, which are all vital physiochemical features in the folding process. There are three distinct stages to the folding process, identified by the structures they create.

Primary structure is the first stage. Chains of amino acids are linked by peptide bonds, referred to as polypeptides. The polypeptide backbone is a repeating sequence along the polypeptide chain. Peptide bonds form through the nucleophilic addition-elimination reaction between the carboxyl group of one amino acid and the amino group of another. X-ray diffraction has found that these peptide bonds are rigid and planar (Figure 2). Thus, for a pair of adjacent amino acids, six atoms lie in the same plane. As discussed later, this provides a useful simplification to the tertiary structure representation. The start of a polypeptide will always be a nitrogen atom and the end of a polypeptide will be a carbon atom; the N-terminus and C-terminus, respectively. Once in the chain, each amino acid is referred to as a residue.

The next step in the folding process is typically towards two motifs, α-helices and β-sheets. Secondary structure is primarily determined by backbone interactions such as hydrogen bonding. α-helices are formed by the hydrogen bonding of the backbone into spiral shapes where the hydrogen bonds run up and down, stabilising the structure. There are two kinds of β-sheets. When the N-terminus and C-terminus are aligned in the same direction it is a parallel β-sheet and when they alternate it is an anti-parallel β-sheet, or β-turn.

Finally, the tertiary structure of a protein is also stabilised by hydrogen bonds but new interactions are now introduced to the folding process. The major driving force is the polarity of a residues side chain. Polarity determines how hydrophobic or hydrophilic the residue is. The hydrophobic effect dictates that in aqueous environments, such as cells, non-polar side chains are hydrophobic meaning they tend to cluster in the centre of globular proteins, away from the aqueous surroundings. On the other hand, polar side chains favour interactions with the surrounding solution and so tend towards the surface of a protein [2]. There are five polar amino acids that can carry a charge indicated by the residues pKa value that are split into two groups, basic and acidic. Basic bear a full positive charge while acidic bear a full negative charge, at the normal physiological pH. If an acidic residue comes into close proximity with a basic residue during the folding process, ionic bonds form as they have opposing charges. The van der Waals force is an attractive force that balances the repulsion that increases when two non-polar atoms approach each other. As a result, there is a distance at which repulsive and attractive forces precisely balance, stabilising the hydrophobic core of proteins [3]. Disulphide bonds are sulphur-sulphur covalent bonds that form between the thiol groups of two cysteine residues. These are very strong intra-molecular bonds that play an important role in determining the structure of a protein. Section 4.2 displays how the physiochemical properties are implicitly encoded in the input representation via an embedding in vector space.

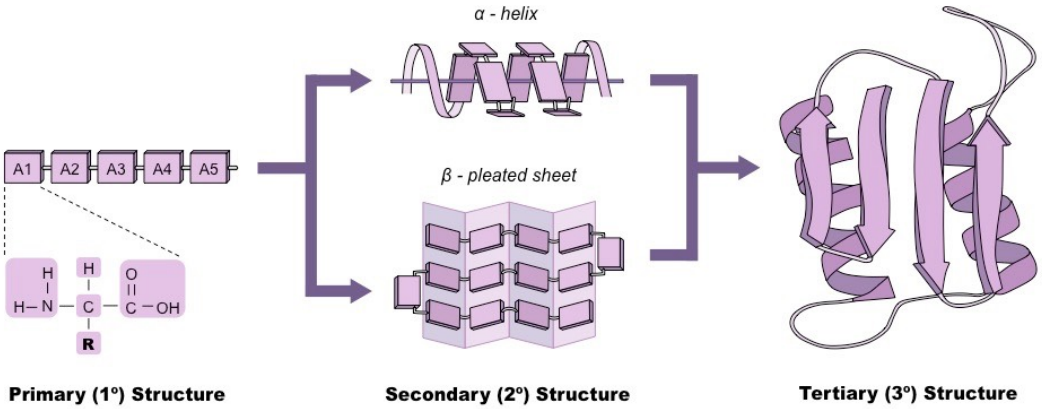

Figure 1: The Folding Process

2.2 Bioinformatics Background

Before 1963, various types of polypeptide chain configurations had been proposed, notably α-helices. However, there was no analytical method of writing these configurations until the physicist, G. N. Ramachandran, devised the Ramachandran Plot (Figure 4, Figure 13) [4]. The peptide bond between the carbonyl C and N only exists in two discrete configurations, trans and cis isomers. 99.9% of peptide bonds adopt the trans isomer with cis isomers reflecting trans isomers at exactly π radians, denoted by the ω angle. The ϕ and ψ angles are the angle of rotation about the single bonds $N - C_\alpha$ and $C_\alpha - C$ respectively. Plotting ϕ and ψ for all residues in a chain show the allowed confirmations of proteins whilst also indicating secondary structure [5]. It is clear from the Ramachandran plots that not all rotations are allowed, due to steric hindrance.

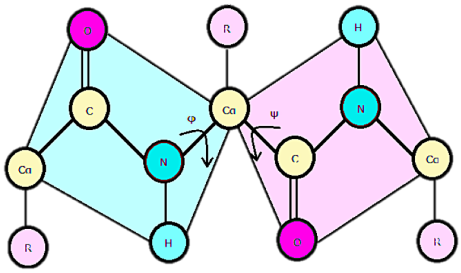

Figure 2: Peptide Bond

In 1973, Anfinsen conducted a Nobel prize-winning experiment that lead to what is now referred to as the Thermodynamic Hypothesis. The hypothesis states that the three-dimensional structure of a native protein in its normal physiological milieu (solvent, pH, ionic strength, presence of other components such as metal ions or prosthetic groups, temperature and other) is the one in which the Gibbs free energy of the whole system is lowest; that is, that the native confirmation is determined by the totality of inter-atomic interactions and hence by the amino acid sequence, in a given environment [6]. Throughout the 1950s, Anfinsen performed a series of unscrambling experiments in support of his eventual hypothesis, that showed primary structure determines the confirmation of a protein [7, 8]. Denaturing agents were mixed with Ribonuclease to create the denatured enzyme by breaking its 4 disulphide bonds. When removing one of the denaturing agents the enzyme refolded to an inactive scrambled enzyme with incorrect disulphide bonds. Upon adding back trace amounts of the denaturing agent, the incorrect disulphide bonds broke causing it to eventually refold to its native state. This showed that for small globular proteins in their standard physiological environment, native structure is determined purely by the proteins amino acid sequence [9].

Several experimental methods are currently used to determine protein structure, including X-ray crystallography and NMR spectroscopy. In most cases, experimental data is not sufficient and bond lengths and angles are often added manually resulting in structures comprised of a balanced mixture of experimental observation and knowledge-based modelling. As these experimental structures are used as the training data it is important to be aware of this.

Upon determining protein structure from X-ray crystallography, the protein is purified, crystallised and subjected to an intense beam of X-rays [10]. The crystal diffracts the beam into a pattern, showing the distribution of electrons in the protein. The electron density map can be interpreted to provide information about the location of each atom in the chain. Although X-ray crystallography shows detailed information about the protein, crystallisation can be difficult.

Flexible proteins do not form useful electron density maps like rigid proteins because crystallography requires many molecules to be aligned in the same orientation [11]. The X-ray resolution is a metric that measures the accuracy of crystallographic structures. The training data for the proposed model consists of proteins whose structure was determined via X-ray crystallography. This is discussed further in section 4.

These methods are expensive and time consuming which means great care must be taken when allocating resources. Given this expense, determining protein structure *in silico* is preferable.

3 Related Research

3.1 Anton

Molecular Dynamics (MD) simulations can be used to model molecular systems such as proteins with an atomic level of precision. Many biological processes occur on millisecond timescales yet MD simulations on this scale lie beyond the reach of current general-purpose technology, by several orders of magnitude. D.E. Shaw Research is a biochemical research company engaged in the design of novel algorithms and machine learning architectures for high-speed MD simulations of proteins and other macromolecules.

Anton is their specialised, massively parallel machine designed to accelerate MD simulations [12]. It simulates atomic motion over a period of time according to classical physics. At each discrete time step the force on each particle due to other particles is computed and the net force is used to update each particles position and velocity. The forces computed are bond, van der Waals and electrostatic forces. The decrease in simulation time comes from specialised hardware that outperforms the expected speedup predicted by Moores Law.

Although Anton has enabled the longest MD simulations to date by a very large factor, its performance is dependent on the size of the system being simulated. For instance, the protein BPTI took 18.2 μs/day to simulate, which has over 17,000 particles [13]. Despite these speed ups, Anton does not scale to long sequences or slow folding proteins. The Folding@home project found NTL9, the slowest-folding protein folded *ab initio* by MD simulation. NTL9 has a folding time of 1.5ms which means Anton would take approximately 3 months to simulate this singular protein [14].

3.2 I-Tasser

A typical template-based modelling procedure involves two major steps: finding proteins with sequences similar to known structure(s), and building 3D models using the detected homologues as structural templates. There were notable improvements from CASP11 to CASP12 in the template-based modelling category with two servers from the Zhang group, Zhang-server and QUARK, outperforming the rest of the servers in a statistically significant manner [15]. They placed second in CASP13 when looking at overall results.

Prior to CASP13, I-TASSER (Zhang-server) has consistently ranked first in CASP competitions [16]. Its name comes from its hierarchical approach of: Threading, ASSEmbly and Refinement. Threading refers to the bioinformatics procedure of identifying template proteins from structure databases. PSI-BLAST matches a query sequence to evolutionary relatives from the database. A sequence profile is created from Multiple-Sequence Alignments (MSA) and used to predict secondary structure using PSIPRED. This is then fed through seven state-of-the-art threading

programs with the top templates from each selected for assembly. Structural assembly involves continuous fragments being excised from the templates and used to build confirmations of well-aligned segments, with unaligned regions being built *ab initio*. The fragment assembly is performed by a Monte-Carlo simulation, with this process being iteratively performed for refinement.

Template-based methods suffer when sequence similarity falls below 20% as the *ab initio* modelling is highly inaccurate.

3.3 AlphaFold

Googles DeepMind entered CASP13 with AlphaFold, their first attempt at protein structure prediction that achieved surprisingly unprecedented results [17]. DeepMind placed themselves a comfortable distance from the second place predictor (the Zhang group) in the free modelling category, which focusses on modelling novel protein folds.

AlphaFold uses MSAs and profiles generated from HHBlits [18] and PSI-BLAST as its input, and a SoftMax over discretized spatial ranges as its output, predicting a probability distribution over inter-residue distances. The distribution from a deep residual convolutional neural network trained on a non-redundant database of proteins selected from the PDB is used as a statistical potential function that is minimised to generate the protein fold. The potential is normalised using a learned reference state and combined with a more traditional physics-based potential. Minimisation of the combined energy function uses L-BFGS gradient descent to yield high accuracy folds.

As the paper is yet to be released, at the time of writing, a more comprehensive analysis of AlphaFold is not possible.

3.4 Google Neural Machine Translation

Away now from computational protein folding models to Google Neural Machine Translation (GNMT), the algorithm that inspired framing the task of protein structure prediction as a NMT problem [19]. In 2016 Google switched their translate service over to the GNMT program to overcome the weaknesses of their conventional phrase-based translation systems.

The new model consists of a deep LSTM network with 8 encoder layers and 8 decoder layers using attention and residual connections. The bottom layer of the encoder network is a bidirectional LSTM, with subsequent layers being unidirectional. The decoder layers are all unidirectional. The attention mechanism connects the bottom layer of the decoder to the top layer of the encoder. Words are divided into sub-words, called wordpieces that help remedy problems attributed to the translation of rare words by providing a balance between character-delimited models and word-delimited models. When decoding the output during inference, a beam search technique is utilised that employed length-normalisation.

Using a human-rated side-by-side comparison as a metric, they show that their GNMT system approaches the accuracy achieved by average bilingual human translators.

4 Model

4.1 Dataset

The Research Collaboratory for Structural Bioinformatics (RCSB) Protein Data Bank (PDB) is a provider of 3D structure data for large biological molecules (proteins, DNA and RNA) typically

obtained via X-ray crystallography, NMR spectroscopy, or, increasingly, cryo-electron microscopy [20]. The search interface enables complex queries of combinations of specific categories. The specific query for this dataset is: sequence identity less than 30%, X-ray resolution less than 2.0Å, chain length between 200 and 400 residues, macromolecular type being protein with no DNA/RNA, singular chains in both the biological assembly and assymetric unit and a singular entity. The query returns a list of PDB IDs (Appendix A), which are 4-character unique identifiers of entries in the PDB. The BioPython python library uses these IDs to retrieve all the necessary sequence and structure information.

As mentioned previously an objective of this paper is to devise a novel approach to *de novo* protein structure prediction. For this reason, sequence similarity must be low so as to not rely on homologous proteins from the training data when making predictions, constructing a more explicit sequence-to-structure map instead. X-ray resolution below 2.0Å is specified to ensure only well-resolved proteins are included as this implies that the backbone and side chains are clear and hence a more exact protein structure will be learnt by the model. The chain length is above 200 to ensure that clear secondary structure appears in each protein and below 400 to reduce the amount of padding per protein. As the scope of this research is on protein structure prediction the macromolecular type is limited to protein only. Likewise, only singular chains are used as this limits the task solely to the prediction of structure and avoids complexities that arise from ligand binding or docking.

Comprehensive statistical analyses of amino acid sequences in proteins have been conducted by many biologists over the years. As seen in the following section, to qualitatively assess the input embeddings, various physiochemcial residue indexes were used. As a measure of polarity Zimmerman *et al.* took a rough approximation of the electric force due to the different side chains acting on their surroundings [21]. The hydrophobicty index for each residue was obtained by Tanford by collecting solubility data [22]. The normalised van der Waals volume index for each residue was obtained by Fauchere [23]. Likewise, normalised frequencies of secondary structural motifs were obtained by Chou and Fasman from their computerised Chou-Fasman predictive method [24].

4.2 Representations

4.2.1 Input Representation

Natural Language Processing systems traditionally treat words as discrete atomic symbols. The encodings are arbitrary, providing no useful information to the system regarding any relationships that may exist between the individual symbols. Representing words in this way leads to data sparsity and requires more data in order to successfully train statistical models. To combat this, vector space models embed words in a continuous vector space where semantically similar words appear close to one and other. Embeddings depend on the Distributional Hypothesis, which states "linguistic items with similar distributions have similar meaning" [25].

The word2vec Skip-gram model is a computationally-efficient predictive model for learning word vector representations that are good at predicting nearby words [26]. More formally, given a sequence of training words $w_1, w_2, w_3, \ldots, w_T$, the objective of the Skip-gram model is to maximise the average log probability:

$$argmax \frac{1}{T} \sum_{t=0}^{T} \sum_{\substack{-c<j<c \\ j \neq 0}} \log P(w_{t+j}|w_t) \tag{1}$$

Where $2c$ is the size of the context window with w_t as the center, T is the sequence length. The basic Skip-gram formulation defines $P(w_{t+j}|w_t)$ using the SoftMax function:

$$P(w_O|w_I) = \frac{exp({v'}_{w_O}^{\top} v_{w_I})}{\sum_{w=1}^{W} exp({v'}_{w}^{\top} v_{w_I})} \tag{2}$$

Where v_w and ${v'}_w$ are the input and output vector representations of w and W is the vocabulary size. This formulation is impractical as the cost of computing $\nabla \log p(w_O|w_I)$ is proportional to W. For this reason, Negative Sampling was used instead. Negative Sampling distinguishes the target word w_O from draws from a noise distribution $P_n(w)$ using logistic regression, where there are k negative samples for each data sample and $\sigma(x) = 1/(1 + exp(x))$. Each $\log P(w_O|w_I)$ is replaced with the following.

$$\log \sigma({v'}_{w_O}^{\top} v_{w_I}) + \sum_{i=1}^{k} \mathbf{E}_{w_i \sim P_n(w)} [\log(-{v'}_{w_O}^{\top} v_{w_I})] \tag{3}$$

This report takes inspiration from ProtVec, which conceptualises that nature uses certain molecular languages to describe biological sequences, such as proteins [27]. Asgari and Mofrad proposed the use of a dense distributed representation of biological sequences to discover the functions encoded within them. In order to train these distributed representations, the amino acid sequences must be broken into subsequences of length n (i.e. biological words). Typically, n-gram modelling in protein informatics uses an overlapping window of residues. However, ProtVec generates n lists of shifted non-overlapping biological words instead, resulting in a corpus consisting of $n \times T$ sequences. The vector representation of the biological words in these sequences are learnt through the Skip-gram neural network. To train the embedding vectors a vector size of 128 is considered and a context window of 20 for values of $n = 1, 2, 3$. Thus, each n-gram is presented as a 128-dimensional vector.

To qualitatively analyse the distribution of n-grams within the vector space, TensorFlows TensorBoard Projector is utilised. All n-grams are projected from 128-dimensional space to a 2-dimensional space using t-Distributed Stochastic Neighbour Embedding (t-SNE) [28]. The physiochemical properties of hydrophobicity, polarity and van der Waals volume were analysed alongside the structural motifs of α-helix, β-sheet and β-turn; using the physiochemical properties from Section 4.1. As there are 20 amino acids, it follows that there is a maximum of 20 1-grams, 400 2-grams and 8000 3-grams in each respective corpus vocabulary (not including start, end or pad tokens). For each n-gram in each vocabulary, the physiochemical properties were averaged based on the individual residues that make up the n-gram and TensorBoard colours them according to this averaged value.

For the small number of 1-grams, TensorBoard is limited to unlabelled categorical colouring, which is of limited value (Appendix B.1). The opposite holds for 3-grams, where the larger vocabulary size meant meaningful clusters failed to form, apart from 3-grams containing cysteine separating themselves from the rest of the vocabulary (Appendix B.3). Figure 3 displays that physiochemical and structural information is well-encoded implicitly for the embedded 2-grams. Polar uncharged 2-grams cluster together (lower-right), typically away from hydrophobic 2-grams (upper-left). Likewise, larger van der Waals volumes favour the upper-right of the spherical t-SNE embedding. As the hydrophobicity, polarity and van der Waals volume affect the secondary structure formation, observing the conservation of clusters is promising. Consequently, structure is implicitly encoded aswell. β-sheets favour the opposite space to β-turns and α-helices encompass

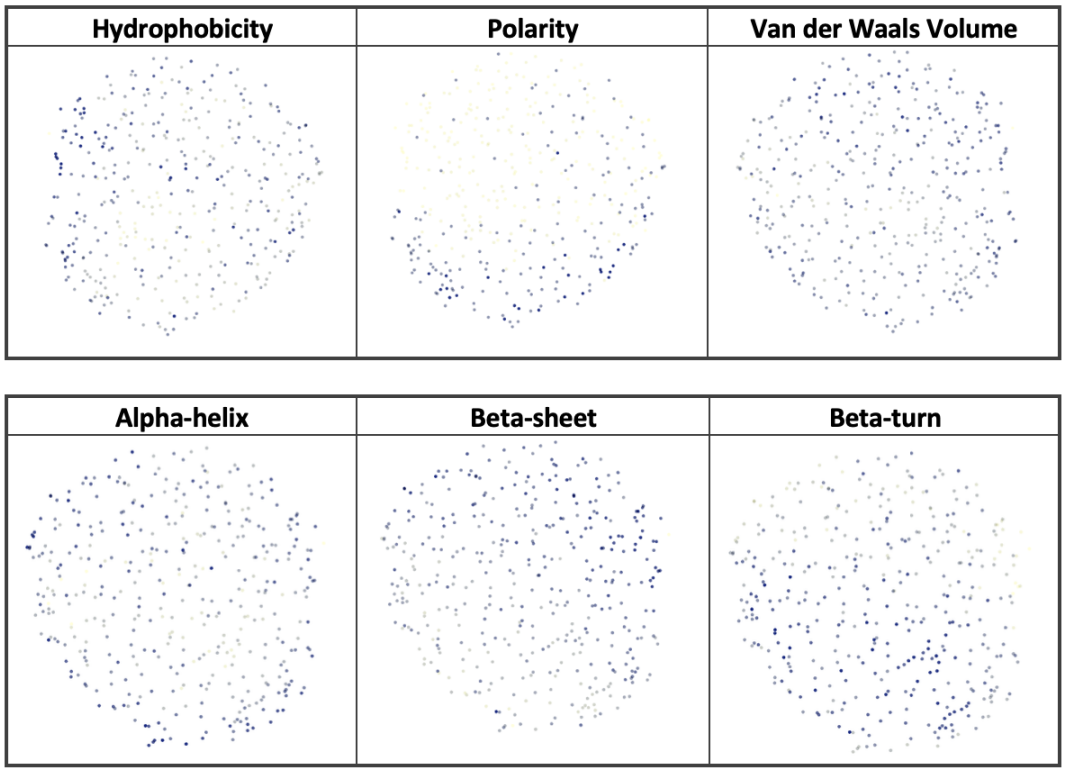

Figure 3: t-SNE 2-gram Embedding

the outer region. As α-helices have one hydrophobic side and one hydrophilic side, on average half the residues would be hydrophobic and half polar, which is exactly what the embedding displays. Therefore, Figure 3 shows a suitable encoding of biological information for the input representation via an embedding of 2-grams.

4.2.2 Output Representation

Polymers can be mathematically represented by the Cartesian coordinates of their atoms, or by the internal coordinates given by a sequence of bond lengths, angles, and torsions of adjacent bonded atoms. Both parameterisations have their advantages and disadvantages. Spatially proximal residues distant in sequence will be more readily observed in Cartesian space than via their internal coordinates. However, internal coordinates offer a consistent local representation of the structure, whose numerical representation does not deviate from arbitrary global rotations as it would in Cartesian space. Moreover, the internal coordinates for a given protein, folded to its native state, will always be the same.

As the bond lengths and angles are fixed, the proteins can be represented solely via their torsions. Furthermore, 99.9% of residues adopt the trans isomer so the ω torsion can be disregarded. This leaves the ϕ and ψ angles as the only variable components of the internal coordinate system. As a result of this, the output representation consists of discretised tuples of the ϕ and ψ angles. Discretisation creates a discontinuous approximation of the continuous torsional rotations that facilitates construction of the output vocabulary. For simplicity, and time considerations, the angles are discretised to 1 decimal place of their value in radians. This simplification does introduce a maximum error of 0.05 radians (2.86°) in both the ϕ and ψ rotations of each residue in a sequence. However, it has the added benefit of keeping the input and output vocabularies at a comparable

size. More importantly, the main advantage of using this parameterisation is that the predicted torsions for each residue can only be drawn from a distribution of legal angles. Disallowed regions generally involve steric hindrance between the side chain and main chain atoms. Hence, it will never be the case that a pair of ϕ and ψ angles are drawn from a blank region of the Ramachandran plot (Figure 4) and hence steric hindrance of adjacent residues will never occur. These regions are sterically disallowed for all amino acids except glycine which is unique in that it lacks a side chain.

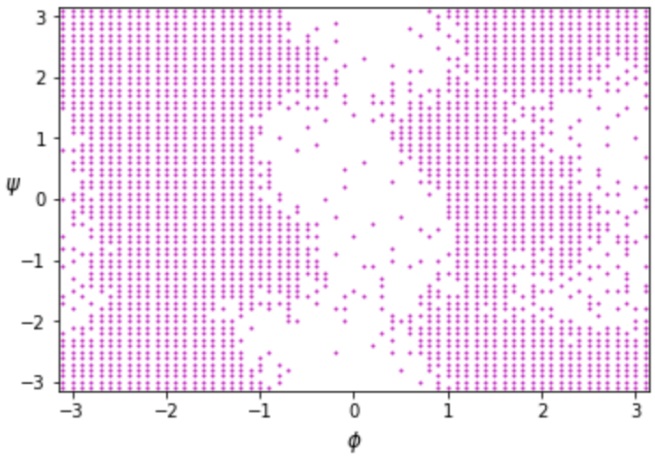

Figure 4: Ramachandran Plot

4.3 Data Pipeline

Both the input and output sequences are tokenised using their associated vocabularies, shown in step 5 of Figure 5, whereby each n-gram and torsion pair are mapped to integers from their resepective vocabularies. After tokenisation, sequences shorter than the maximum length are padded with zeros in step 6, ensuring that each sequence fed into the network is of the same length. To read data efficiently it is helpful to serialise the data and store it in a set of files. A TFRecord is a simple format for storing a serialised sequence of key-value pairs. For each protein in the dataset, the PDB Code, the tokenized padded amino acid sequence and the tokenized padded torsion sequence are stored.

Achieving peak performance requires an efficient input pipeline that delivers data for the next step before the current step has finished, ensuring optimal use of both the CPU and GPU. Google Colabatory is a free-to-use Jupyter notebook environment that runs entirely in the cloud, allowing free access to a Tesla K80 GPU, with 12GB of RAM, in 12 hour sessions. TensorFlows tf.data API builds efficient input pipelines by framing them as an ETL process:

1. **Extract**: Read data from persistent storage (Google Drive mounted to Google Colab).
2. **Transform**: Use CPU cores to parse and perform pre-processing operations on the data such as shuffling and batching.
3. **Load** Load the transformed data onto the GPU that executes the model.

A naïve synchronous implementation results in the GPU sitting idle as the CPU prepares the data and subsequently the CPU sitting idle as the GPU trains the model. Pipelining overlaps the pre-processing with the model execution of a training step. Data is prefetched through the tf.data API, decoupling the time data is produced from the time it is consumed.

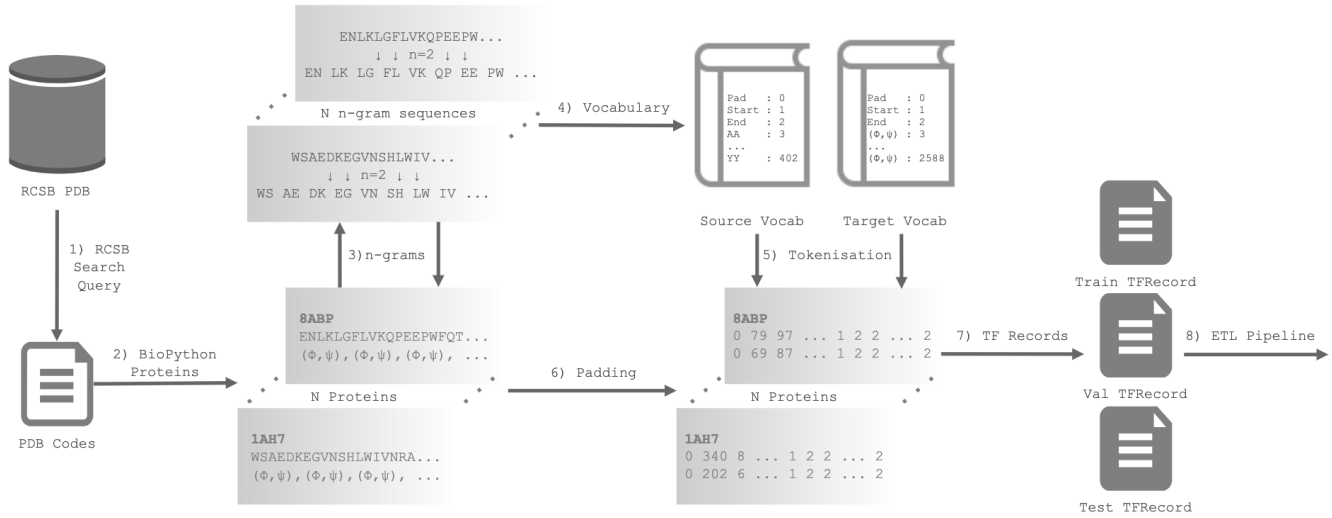

Figure 5: Preprocessing Pipeline

4.4 The Network

The initial, rather vanilla, implementation of the neural network model consists of two Recurrent Neural Networks (RNN) that act as an encoder-decoder pair and an attention mechanism [29]. The encoder encodes the variable-length input sequence of tokens into a fixed-length vector representation, and the decoder maps that representation back to a variable-length sequence of output tokens (variable through padding). RNNs are a class of neural network that perform well on sequential data, evident through speech recognition and machine translation tasks [30, 31]. They scan through data from left to right, sharing parameters between each timestep. This ensures that predictions at an arbitrary timestep receive information from all previous states. In order to incorporate information from future states, a Bidirectional RNN (BRNN) was implemented for the encoder [32]. BRNNs have been shown to be significantly more effective than their unidirectional counterparts, especially where context is vitally important [33]. In principle a large enough RNN should be sufficient to generate sequences of arbitrary complexity. In practice however, standard RNNs are unable to store information about past inputs for very long [34]. Hence, both the encoder and decoder RNNs are gated with Long Short-Term Memory (LSTM) units because this RNN architecture is designed to be better at storing and accessing information than standard RNNs [35]. They also help remedy the vanishing gradient problem that plagues conventional Backpropagation Through Time (BPTT).

The intuition behind this design is that the network may be able to learn long-range dependencies between residues distant in sequence, that find themselves close in Cartesian space. Consequently, these dependencies would be analogous to the bonds that form to give proteins their structure. Illustrative examples of this include the long-range dependency that results in two cysteines forming a disulphide bond or the more short-range dependency of hydrogen bonds forming secondary structural motifs. It is hoped that, as the physiochemical information is well-encoded in the input representation, this should be entirely possible.

4.4.1 Optimisation

Stochastic gradient-based optimisation is a method of minimising an objective function $J(\theta)$ parameterised by a models parameters $\theta \in \mathbf{R}^d$ through updating the parameters in the opposite

direction to the gradient of the objective function $\nabla_\theta J(\theta)$ w.r.t θ [36]. The learning rate η determines the size of the steps taken towards some (often local) minima. Stochastic Gradient Descent (SGD) is an iterative method for stochastically approximating gradient descent optimisation that performs a parameter update for each training example x and label y:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta) \tag{4}$$

SGD performs frequent updates with high variance that cause the objective function to fluctuate, enabling it to jump to new, potentially better, local minima. However, this also reduces the likelihood of converging to the global minima as overshooting it is common. For this reason, mini-batch gradient descent is performed to reduce the variance of parameter updates, leading to a more stable convergence. Additionally, TensorFlows highly optimised matrix operations compute gradients w.r.t mini-batches with greater efficiency. The dataset is shuffled before batching to improve the generalisability of the model. Batch sizes of 64 were found to improve efficiency and generalisability most, while operating comfortably within the limits of the hardware. The new update equation is given by Equation 5, where b is the batch size: :

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta; x_{i:i+b}, y_{i:i+b}) \tag{5}$$

Adam is an alternative method for efficient stochastic optimisation [37]. The name Adam is derived from adaptive moment estimation, as the method computes individual adaptive learning rates for different parameters from estimates of first and second moments. Again the aim is to minimise an objective function $J(\theta)$ w.r.t its parameters θ. However, Adam also updates exponential moving averages of both the gradient m_t and the squared gradient v_t, where the hyperparameter $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these moving averages. The moving averages themselves are estimates of the 1^{st} moment and 2^{nd} raw moment of the gradient, or the mean and uncentered variance respectively. Thus, the update rule is given by:

$$g_t = \nabla_\theta J(\theta) \tag{6}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{7}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{8}$$

$$\eta_t = \eta \sqrt{1 - \beta_2^t} / (1 - \beta_1^t) \tag{9}$$

$$\theta_{t+1} = \theta_t - \eta_t \frac{m_t}{\sqrt{v_t + \varepsilon}} \tag{10}$$

As m_t and v_t are initialised to vectors of zeros they are biased towards zero, especially initially or when decay rates are small. To counteract this, bias-corrected first and second moment estimates are computed:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{11}$$

$$\theta_{t+1} = \theta_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}} \tag{12}$$

The authors propose default values of: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. Learning rates that are too small lead to slow convergence while learning rates that are too large hinder convergence, causing the loss function to fluctuate around some minimum and potentially diverge. Figures 6

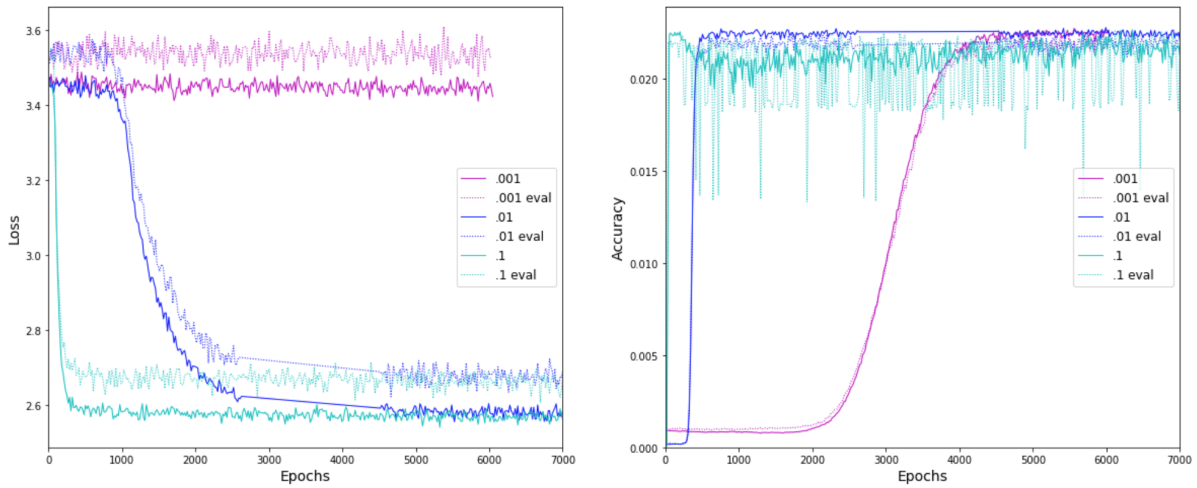

Figure 6: SGD Optimisation

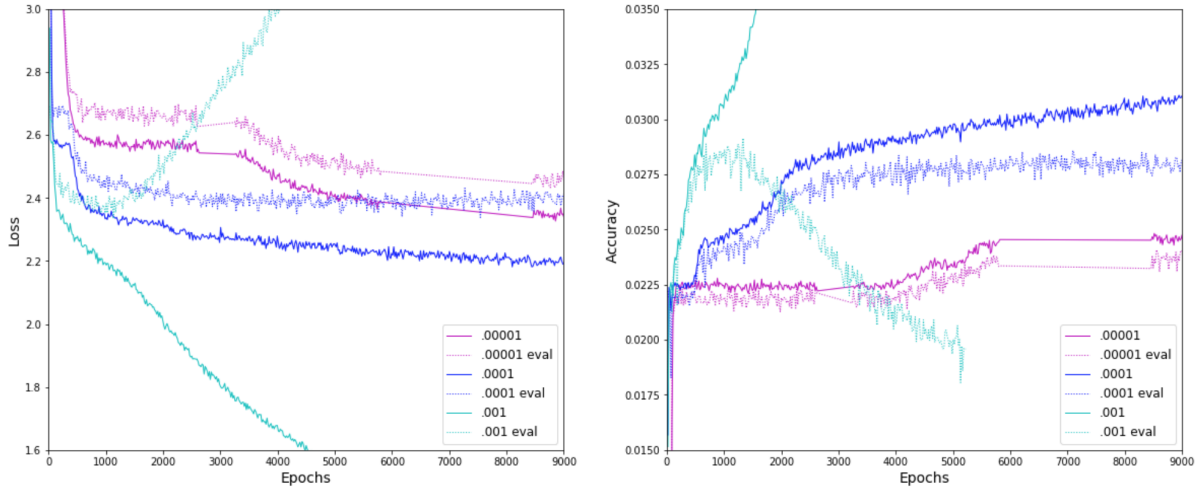

Figure 7: Adam Optimisation

and 7 show experimentation into various learning rates for both SGD and Adam. A comparison between them reveals the more suitable of the two optimisers.

A key challenge of minimising highly non-convex error functions with neural networks is avoiding converging to any of the suboptimal local minima. Moreover, it has been argued by Dauphin *et al.* that non-convex error surfaces in high-dimensional spaces generically suffer from a proliferation of saddle points [38]. It is notoriously hard for SGD to escape saddle points as they are usually surrounded by a plateau of the same error in all directions where the gradient is approximately 0 everywhere. They also state that the ratio between the number of saddle points and the number of local minima increases exponentially with the dimensionality of the function. The results in Figures 6 and 7 are contingent with this insight as SGD struggles to converge while Adam manages to overcome this, converging closer to the optimum without plateauing in either accuracy or loss. As the learning rate of 0.00001 begins to diverge, 0.00005 was chosen as the learning rate for the network from this point on as it falls evenly between the best two values from Figure 7. The Literature Review states that other optimisers such as RMSProp would be analysed, however for time considerations this was omitted as Adam performed best.

4.4.2 Residual Connections

Theoretical and empirical evidence indicates that the depth of neural networks is crucial for their success [39]. With deeper networks, the integration of low-level to high-level features is enriched. Google showed that for NMT systems to achieve good accuracy, both the encoder and decoder have to be deep enough to capture the subtle irregularities in the source and target languages. This decision was made after the success of Sutskever *et al.* from stacking LSTM layers for various NMT problems [40, 41]. In computer vision, models with more than ten convolution layers outperform shallow ones, advancing the state of the art in pattern recognition [42]. The intuition being deeper models possess greater representational power, enabling a hierarchy of concepts to form. However, as depth increases, so too does training time and training difficulty, likely due to exploding/vanishing gradients. Figure 8 displays this well as deeper encoders and deeper decoders converge slower; per both epoch and unit time. Simple stacked LSTM layers in both the encoder and decoder work well up to 4 layers, but 8 layers and beyond saw downgrades in performance.

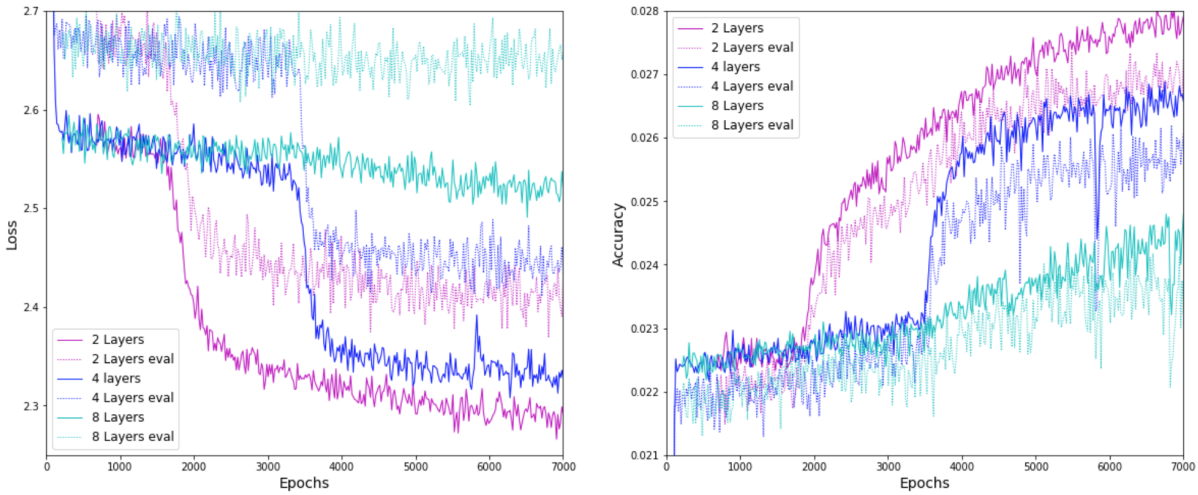

Figure 8: Deeper Networks

The success of simple stacking follows the law of diminishing returns, to the point that by continually increasing the depth, performance can even begin to decline on the training set [43]. As hierarchical composition is, in principle, more powerful, the decline encountered in practice is attributed to the problem of vanishing gradients. The suggestion that adding more layers would decrease representational power is contradictory. Residual networks resolve the problem of training deeper architectures by allowing units to copy their inputs on to the next layer unchanged, via skip connections. For each unit, the output $y(x)$ is given by Equation 13 where the input x is summed to a residual $F(x)$ [44]:

$$y(x) = F(x) + x \tag{13}$$

It is argued that this improves the error flow by decreasing the length of the gradient propagation path [45]. For this reason residual connections were introduced. Input from the lowest LSTM layer is summed element-wise to the layers output and then fed to the subsequent layer.

Training time for 8 residual layers was far too long for the available time of the project to show any reasonable representational benefits from increased depth. Two residual layers did not show much improvement in loss and a decrease in accuracy over the same time-frame. However, 4 residual layers showed improvement in loss but again accuracy began to diverge. As it is expected

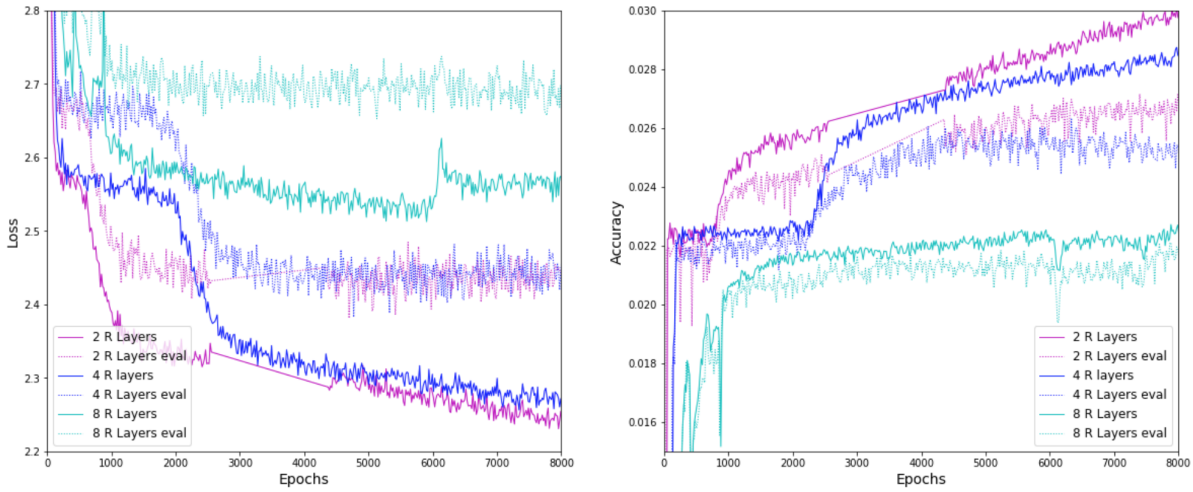

Figure 9: Residual Connections

that deeper representations perform better, a network with 4 residual layers for both the encoder and decoder was chosen, in the hope that divergence in accuracy can be alleviated by the following experiments. Although the Literature Review touched on GNMT, it did not include Residual Connections.

4.4.3 Gradient Clipping

Much like residual connections attempt to resolve a problem attributed to vanishing/exploding gradients, so too does gradient clipping. As introduced by Bengio, the exploding gradient problem refers to the large increase in the norm of the gradient during training [46]. Pascanu *et al.* hypothesised, through a geometrical interpretation, that when gradients explode so too does the curvature along some direction(s), leading to a steep wall in the error surface [47]. Upon a gradient descent step at the cliff edge, an optimiser will be forced to jump across the valley perpendicular to the wall, which could result in a lack of necessary exploration.

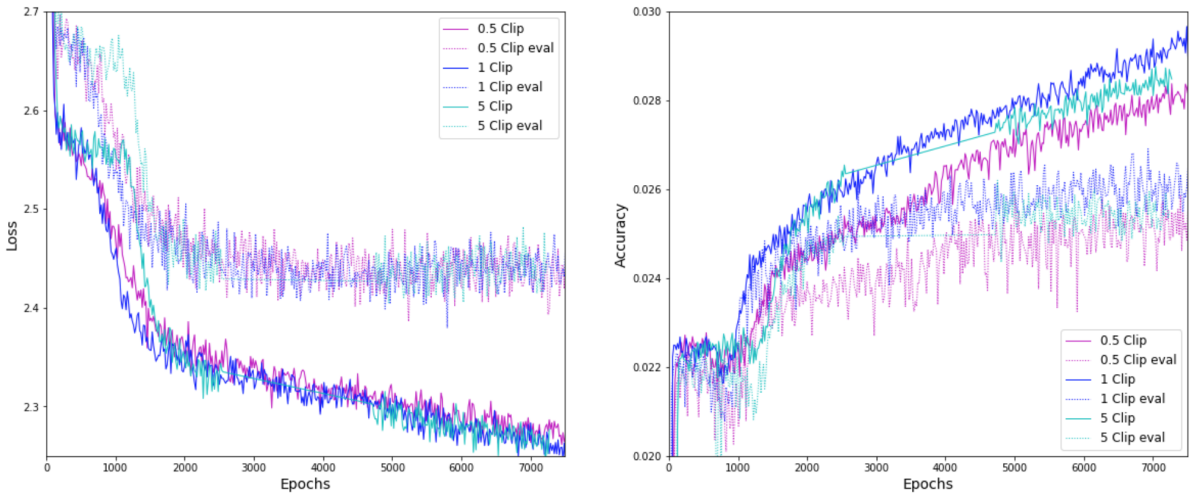

Figure 10: Gradient Clipping

Clipping gradients above a given threshold avoids exploding gradients by trimming the mag-

nitude of the step in the direction that is perpendicular to the edge of the cliff, whilst keeping the direction unchanged; allowing optimisers to more freely explore such valleys. Thus gradient clipping implements a simple form of second-order optimisation in the sense that the second derivative is also proportionally larger in those exploding gradient regions [48]. Gradients are clipped by the threshold values of 0.5, 1 and 5. The results show that the effect of gradient clipping is limited, although subtle improvements in accuracy were realised when gradients were clipped to 1. The previous Literature Review made no mention of gradient clipping so this has been added since the time of it's writing.

4.4.4 Dropout

Overfitting occurs when the neural network is so closely fitted to the training set that it fails to generalise to the validation set. Examples of overfitting exist in all previous experiments where it is clear that the validation loss begins to diverge from the training loss. The dataset was split 60:20:20 during pre-processing, allowing cross-validation of the results.

With limited training data, as in this case where using well-resolved proteins drastically reduces the dataset size, some arbitrary relationships between inputs and outputs learnt by the many non-linear activations within the network may well be the result of sampling noise. The relationships will exist in the training set but not in the general case, and hence display an example of overfitting. Dropout is a regularisation strategy for neural networks whereby units are dropped at random and thus their connections along with them [49]. Dropout acts as an approximate method of combining exponentially many different neural network architectures efficiently [50]. A range of drop values were experimented with to find the best at preventing complex co-adaptions on the training data. The dropout operator is only applied to non-recurrent connections. By not using dropout on the recurrent connections, the LSTM can benefit from dropout regularisation without sacrificing its valuable memorisation ability [51]. The Literature Review did not reference regularisation strategies, so dropout has been added for this report.

Dropout introduced the largest improvement to the accuracy, achieving a peak value of 2.8%.

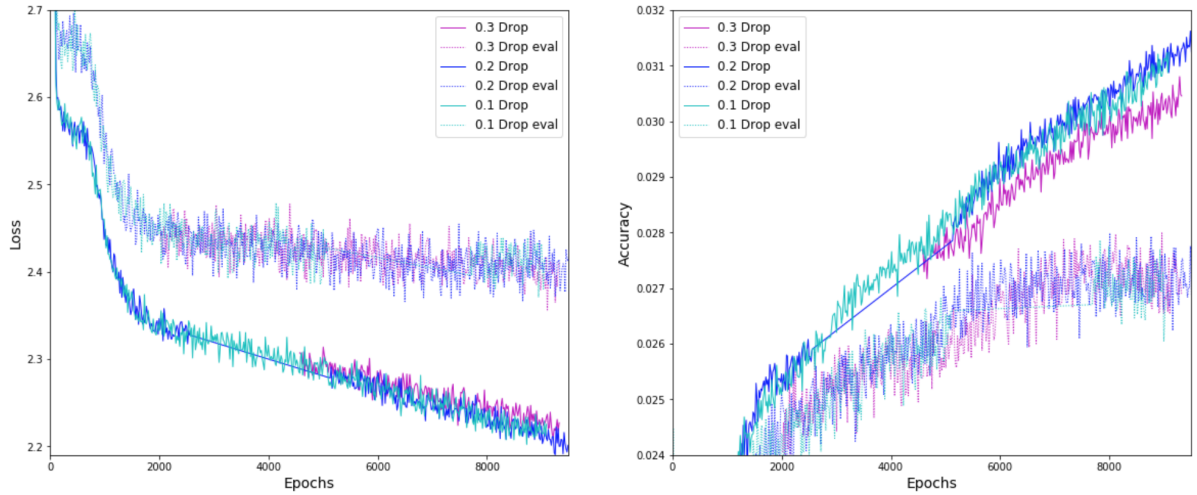

Figure 11: Dropout

4.5 Attention

Upon investigation into the limitations of encoder-decoders, Cho *et al.* showed that NMT performs well on relatively short sequences but performance degrades rapidly as the length of the sequence increases [52]. Carrying on from this, Bahdanau *et al.* went on to conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of the encoder decoder architecture [53]. They proposed an attention mechanism that allowed them to search for parts of the source sequence that are relevant to predicting a target symbol, achieving state-of-the-art performance in NMT tasks. Attention has also shown promise in computer vision problems, such as image captioning [54].

Each time the model generates a target symbol it searches for a set of positions in the input sequence where the most relevant information is concentrated. Attentions distinguishing feature is that it no longer encodes the input sequence into a single fixed-length vector. Instead, the input is encoded to a sequence of vectors and chooses a subset adaptively while decoding the output.

The probability is conditioned on a distinct context vector C_i for each target symbol $\hat{y}_i$. The context vector C_i depends on a sequence of annotations $(n_1, \ldots, n_T)$ where each n_i contains information about the whole input. The context vector C_i is computed as a weighted sum of the annotations where $e_{ij} = a(h_{i-1}, n_j)$ is an alignment model that scores how well the inputs around position j match outputs at position i.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})}, \quad C_i = \sum_{j=1}^{T} \alpha_{ij} n_j \tag{14}$$

Bahdanau successfully applied Equation 14 to jointly translate and align words. Luong drew from this idea to design two novel types of attention based models: a global approach in which all source words are attended and a local one whereby only a subset of source words are considered at a time [55]. The former resembles Bahdanau but the latter is referred to as Luong attention.

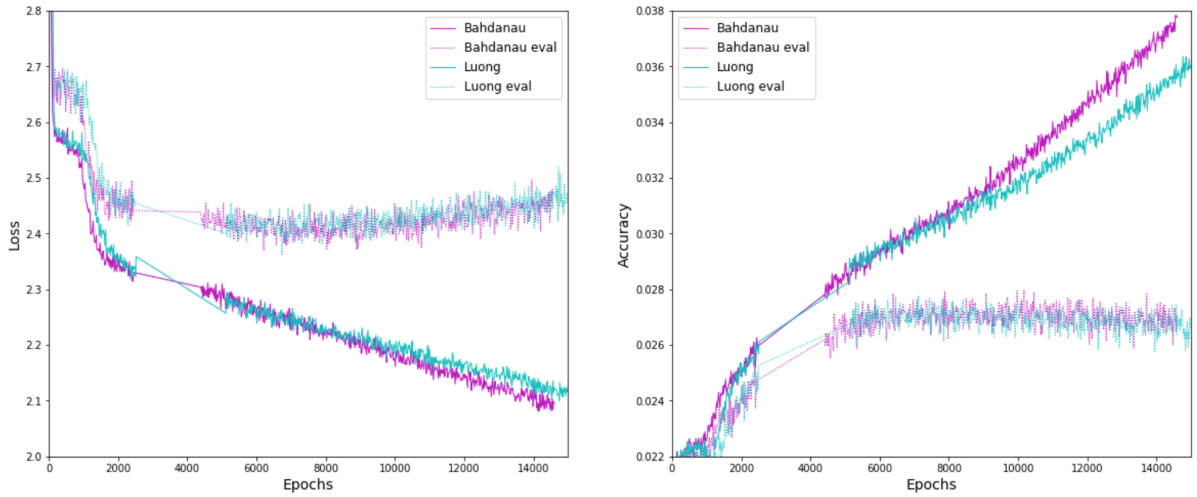

Figure 12: Bahdanau vs. Luong

The global attention of Bahdanau has the drawback that it has to attend all source tokens for each target word. To address that deficiency Luong *et al.* took inspiration from the attempt at image caption generation task by Xu *et al.* that explored the tradeoff between soft and hard attention models. Soft attention refers to global attention, placing a weight over each and every

input. Hard attention, on the other hand, attends to one token at a time. Bahdanau attention was used for all previous experiments. Figure 12 shows that the more global Bahdanau attention performs better at this task than Luong's attention. However, ungratifying experimentation was carried out in this instance in hope of performance benefits. In the Literature Review it was not specified which attention mechanism would be used but as Bahdanau performs slightly better it was consequently kept as the model's attention mechanism.

5 Implementation

The final implementation of the model consists of a deep LSTM network with 4 encoder layers and 4 decoder layers. The encoder layers are all bidirectional with 128 units in each and all the decoder layers are unidirectional. Residual connections are used for both the encoder and decoder. A Bahdanau attention mechanism connects the lowest decoder layer to the highest encoder layer with an attention size of 128. The Adam optimiser was used for training with a learning rate of 0.00005. Batch sizes of 64 are fed into the network. Dropout is applied to the non-reccurrent connections at a drop rate of 0.3. The gradients are clipped at 1.0. Figure 13 shows a diagram of the network.

Early stopping was used after 7000 training steps, providing a model at maximum performance. The minimum loss reached by the final implementation on the validation set is 2.39 with an accuracy of 0.028. The Literature Review intended to use the mean-squared error as the loss metric. However upon further consideration this was seen as a rather naïve metric and so sparse categorical cross entropy was used instead.

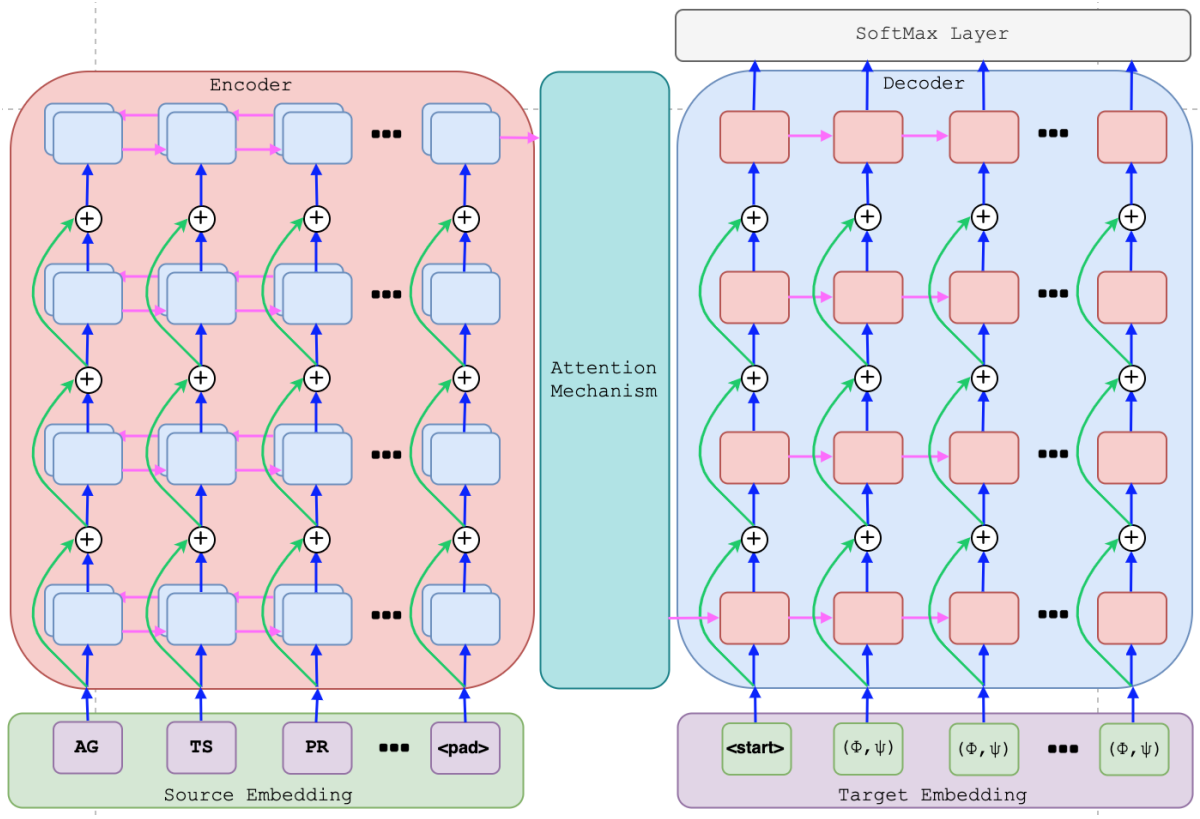

Figure 13: The Model

5.1 Inference

The encoder-decoder model learns a conditional distribution over a sequence conditioned on another sequence $P(\hat{y}_1, \ldots, \hat{y}_{Ty}|x_1, \ldots, x_{Tx})$. To find the closest sequence of torsions for a given sequence of n-grams this conditional probability is maximised. This is typically done with a beam search over a greedy search in order to generate an entire sequence of word that maximises the joint probability as opposed to just the most likely word. Therefore, to calculate the probability of a predicted sequence $\hat{Y}$, the decoder decomposes the joint probability into ordered conditionals.

$$P(\hat{Y}) = argmaxP(\hat{y}_1, \ldots, \hat{y}_{Ty}|x_1, \ldots, x_{Tx}) = \prod_{t=1}^{T} P(\hat{y}_t|y_1, \ldots, y_{t-1}, C) \tag{15}$$

The beam width B specifies the number of torsion to be considered at each step. With larger B more possibilities can be explored; potentially leading to higher accuracy at the expense of computation time. A beam width of 10 was used.

5.2 Torsion Space to Cartesian Space

Efficient conversion from torsion space to Cartesian space is necessary in order to construct PDB files for the predicted proteins. Natural extension Reference Frame (NeRF) is commonly used to switch between parameterisations [56]. Although NeRF can be parallelised to process multiple polymers simultaneously, it is not parallelisable along the length of the polymer. AlQuraishi derived a mathematically equivalent algorithm to overcome this deficiency, pNeRF [57].

A summary of NeRF facilitates understanding of pNeRF. For a given sequence of internal coordinates (encoded via a triplet of bond length, angle and torsion), linear polymers are constructed sequentially by NeRF from one end of the polymer to the other. Coordinates are computed atom by atom in a special reference frame, which, using an affine transformation derived from the three previous coordinates, gives the actual position of the atom. Requiring the previous three atoms to place the next does not allow parallelisation along the length. Thus, pNeRF fragments the polymer into M segments and converts each to Cartesian space independently before reassembling the fragments with M affine transformations.

As the model predicts a coarse-grained protein structure, a basic PDB file is created using just the backbone atoms. For each backbone atom in the chain the Cartesian coordinates are written in the PDB file. A quick pass through Rosetta adds the side chains to the file.

5.3 Global Distance Test Total Score

Global Distance Test (GDT) Total Score (TS) is a measure of similarity between two protein structures with identical amino acid sequences but varying tertiary structures. This metric is more accurate than RMSD that suffers from sensitivities to outlier regions of a structure that is otherwise reasonably accurate. For example, given an almost perfect match with an unstructured tail comprising a 20Å difference between the two models, this difference will dominate the RMSD, even though the difference may be limited to a small fraction of the structure.

CASP use GDT_TS measurements as their major assesment criteria, where the higher the GDT_TS is, the better the prediction. The basic algorithm is to iteratively align two structures under a set of different cutoffs, and then tally the fraction of residues which fall within the cutoffs. The resulting scores end up in the range of 0.0 to 1.0, with 1.0 being a perfect match.

6 Results

It is evident from the results that given the limited hardware and training time, the model does not perform well when compared to state of the art methods. AlphaFold averages a GDT_TS score of 59.65, whereas the implementation of this paper averages a GDT_TS of around 0.06. It is clear from Figure 14 that improved performance is solely contingent on the frequency of α-helices within the protein. 2XBG is largely comprised of β-sheets, 2DHO has a balanced mixture of each and 4X2R is mainly α-helices. Thus, the respective GDT_TS scores of 0.03, 0.08 and 0.105 are understandable. Every single protein predicted by the model contains an α-helix down the entire length of the chain. The reason for this is analysed in the following section.

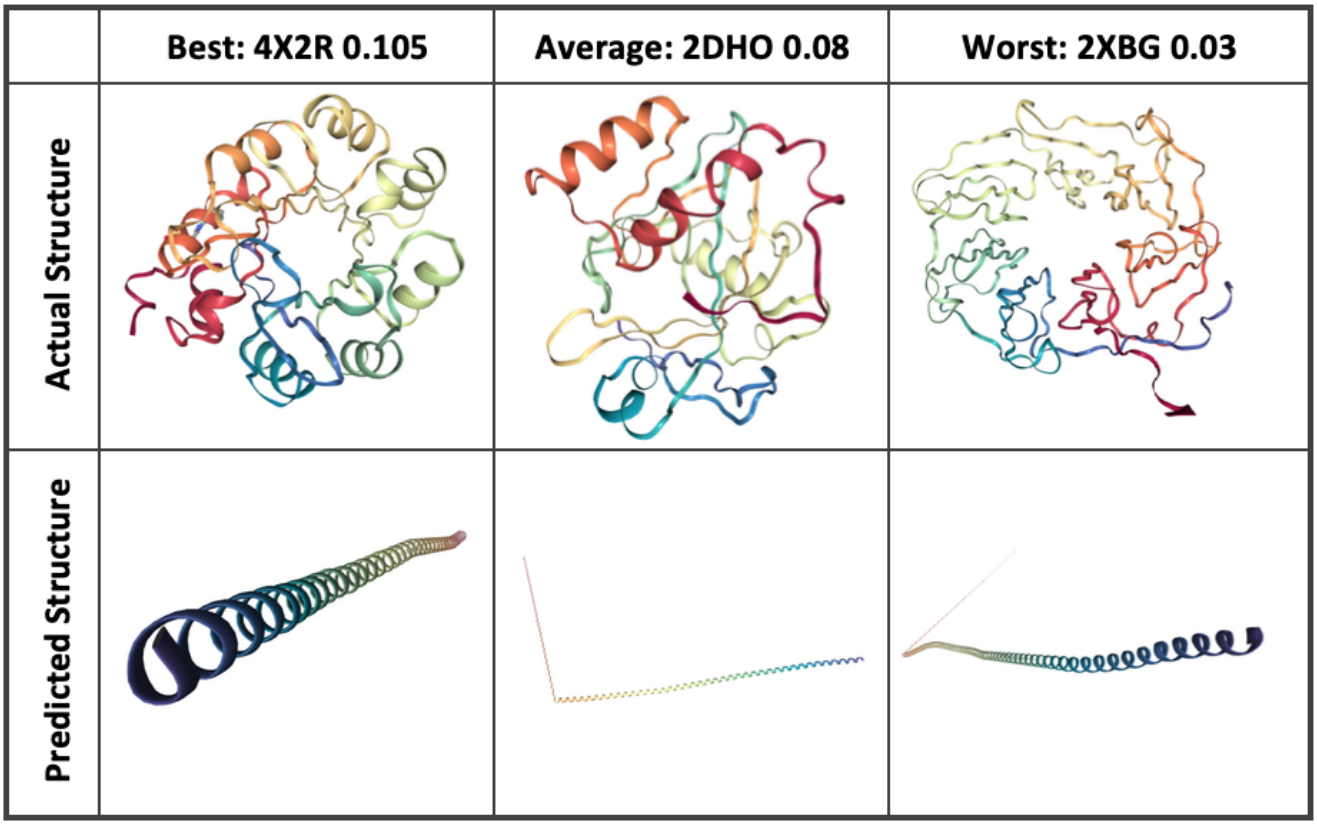

Figure 14: Results

A scatter plot of the summed difference between the real torsions and predicted torsions against the length of the protein chain for each protein shows a strong correlation between the length of the protein and the accuracy of the prediction. Each point is a protein from the test set. It was only noted upon analysis of the test data that some proteins fell below the minimum sequence length of 200. Despite the search query, which stated only singular chains between 200 and 400 residues should be included, proteins with multiple chains were returned by the query. This meant that some proteins were much smaller than the 200 residue threshold which severally skewed the GDT_TS results. The smaller the protein, the easier it is to align to a segment of the target protein, especially when it is a simple α-helix. Figure 14 shows the best, average and worst proteins that fall within the proper range of protein length.

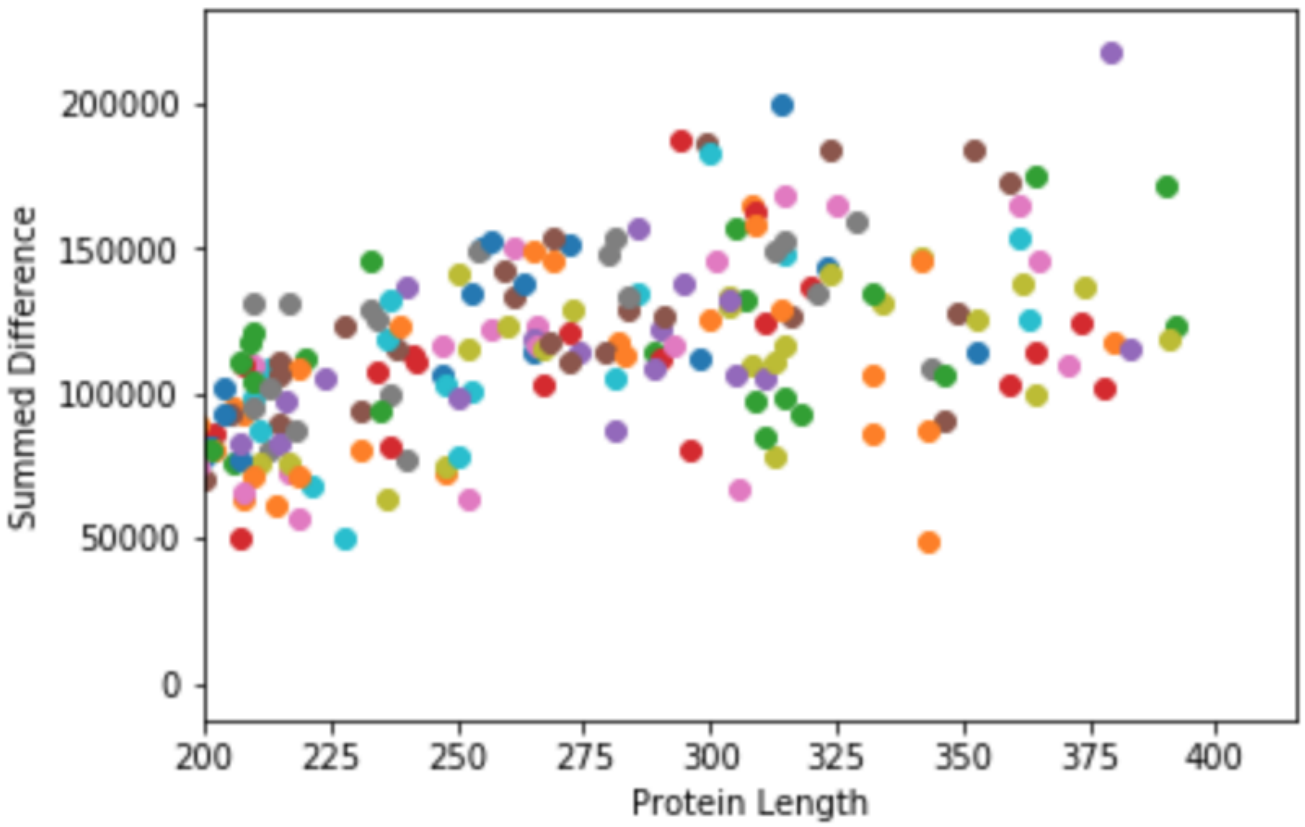

Figure 15: Length Analysis

7 Analysis

7.1 Angle Distributions

The most frequent ϕ and ψ tuple is $(-1.1, -0.7)$ with a total count of 26896 occurrences in the corpus (Figure 16). The output vocabulary consists of 2588 distinct torsional tuples. Thus, for each residue in a sequence there are 2588 possibilities for that residue. The maximum decoder sequence length is 400 because every sequence is padded to this value. Simply choosing the most frequent (ϕ, ψ) pair for each residue results in an accuracy of 2.6% as $\frac{26896}{400*2588} = 0.026$, which is roughly the maximum accuracy achieved in Section 4. Inspection into the predicted angles from the decoder demonstrates that the majority of predicted tokens come from this region on the Ramachandran plot.

The typical secondary structure from this particular region tends to be α-helices; as opposed to β-sheets that occupy the top-left region. This is likely the reason that the model predominantly predicts α-helices over any other structure. The decoder simply learns to place the more frequent torsions along the sequence length as this achieves maximal accuracy without having to learn any difficult dependencies.

The model does manage to decode the initial torsions correctly, such as the start token. Additionally, it manages to learn where the end token is but fails to apply the appropriate padding, instead placing end tokens until the end of the protein. This is promising and means that given more training time, further concepts may be able to be learned by the model.

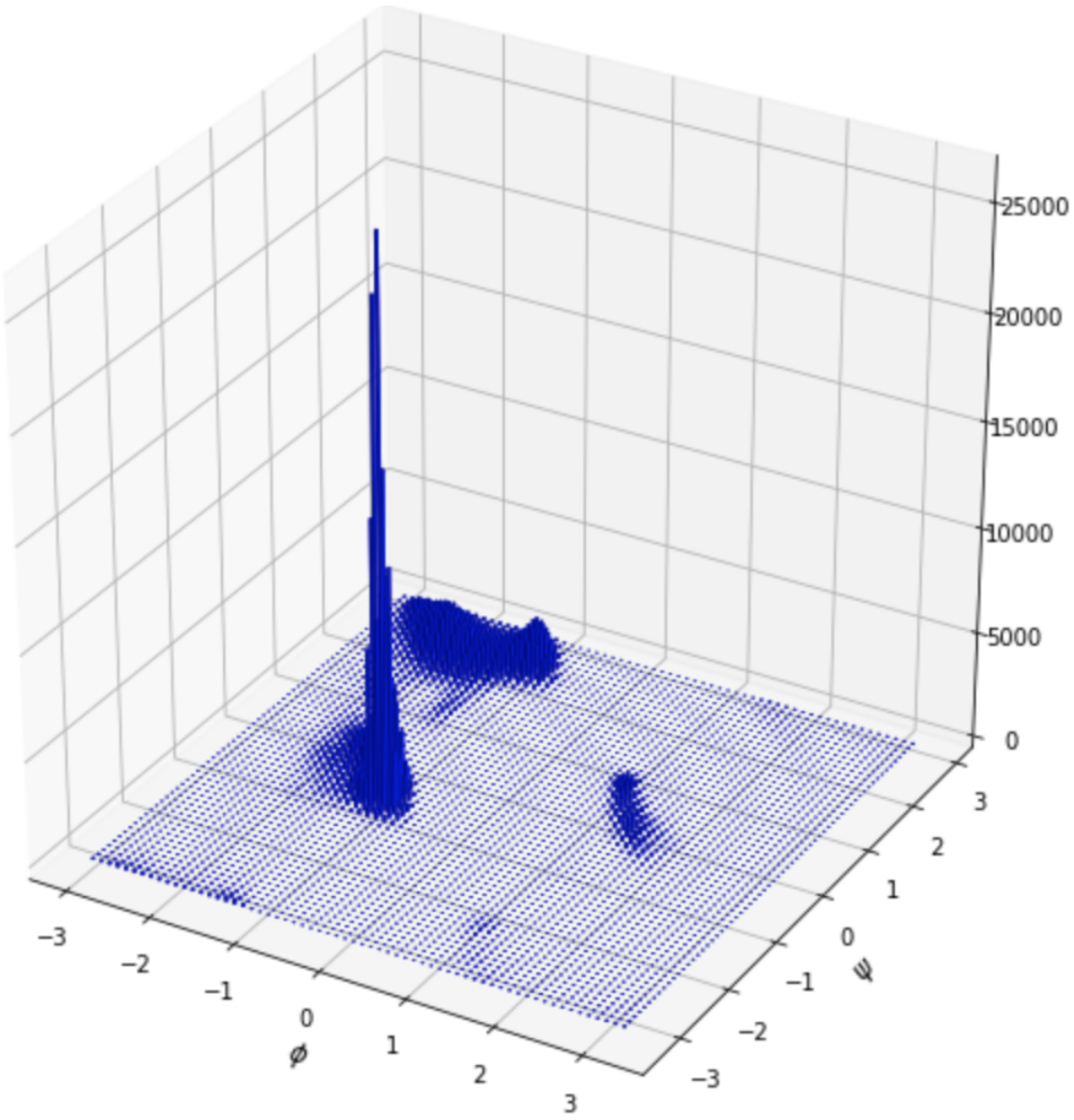

Figure 16: Ramachandran Density Plot

7.2 Reconstruction

While assessing the accuracy of pNeRF a detrimental characteristic of the algorithm was encountered. Figure 17 shows Cartesian reconstructions of proteins using pNeRF from their actual internal coordinates, instead of the predicted ones. It is clear that although the ϕ and ψ angles are correct and secondary structure is conserved, there are various deviations along the chain, resulting in an overall reduction in GDT_TS even for the true internal coordinates.

After conferring with AlQuraishi as to where this discrepancy came from, it became clear that idealised bond lengths and angles were not being used in the naïve pNeRF algorithm. It was assumed in Section 4.2.2 that the bond lengths and angles were fixed. In reality, these mean values of the bond lengths and angles were obtained by Engh and Huber[58]. More recently however, it has been shown that bond lengths and angles are drawn from distributions unique to each residue. Furthermore, these distributions change subject to secondary structure [59]. The

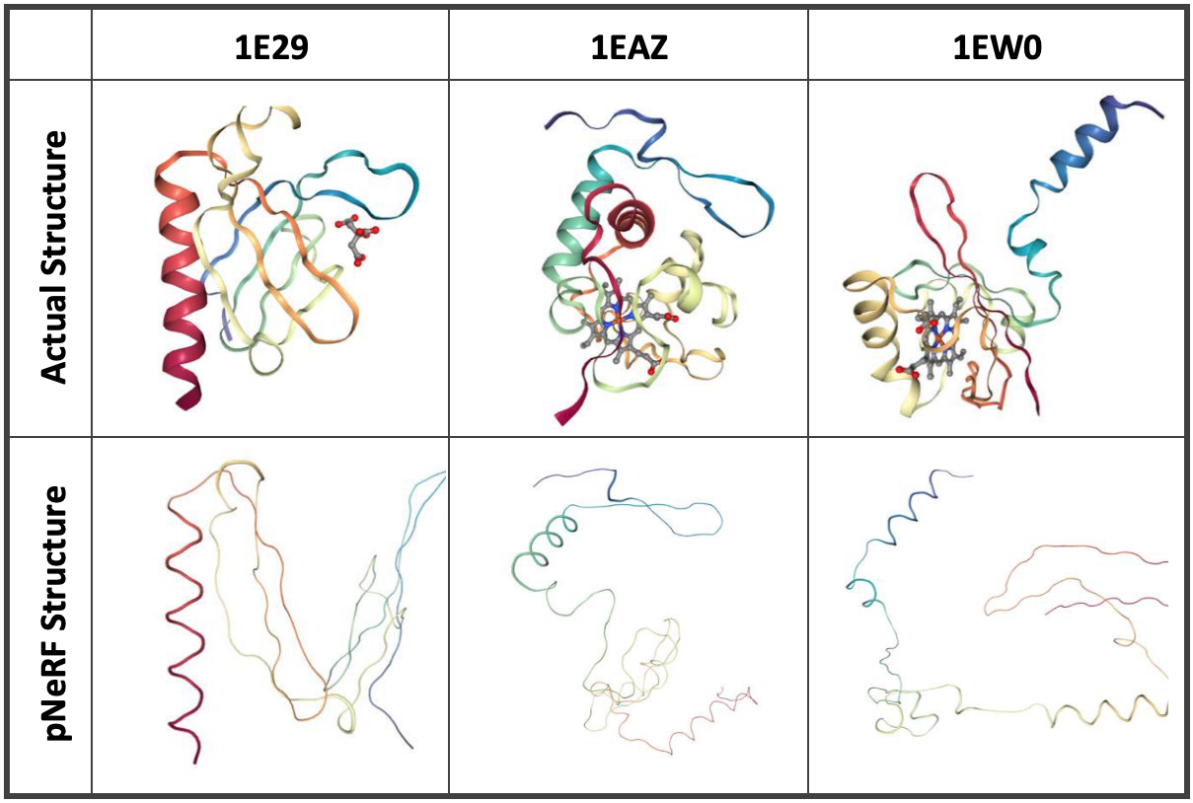

Figure 17: pNeRF Real Reconstruction

pNeRF algorithm simply uses the overall mean bond lengths and angles, regardless of residue and secondary structure. Even if the neural network was as accurate at predicting the internal coordinates as the state of the art methods, this discrepency in the pNeRF algorithm would limit the performance of the model in terms of the GDT_TS.

7.3 Attentiveness

The x-axis and y-axis of each plot in Figure 18 correspond to the n-grams in the source sequence and the generated internal coordiantes, respectively. Each pixel shows the weight α_{ij} from Equation 14 of the annotation of the j-th source n-gram for the i-th target torsion. Unfortunately, the attention mechanism fails to learn any useful information. This is likely due to the length of the proteins. Figure 18 show that the attention is either concentrated in too small a region or spread too thinly over the whole input sequence at each decoding time step. Initial experiments with shorter proteins ranging from 50 residues to 100 residues showed that when decoding a target angle with a particular type of secondary structure, the model could attend to regions in the source sequence of the same secondary structure. However, the size of the dataset for proteins between 50 and 100 residues is limited even further as most proteins are larger than 100 residues in length. A longer training time may be needed in order for the attention mechanism to become fully effective. Alternatively, a larger attention size may offer improvements.

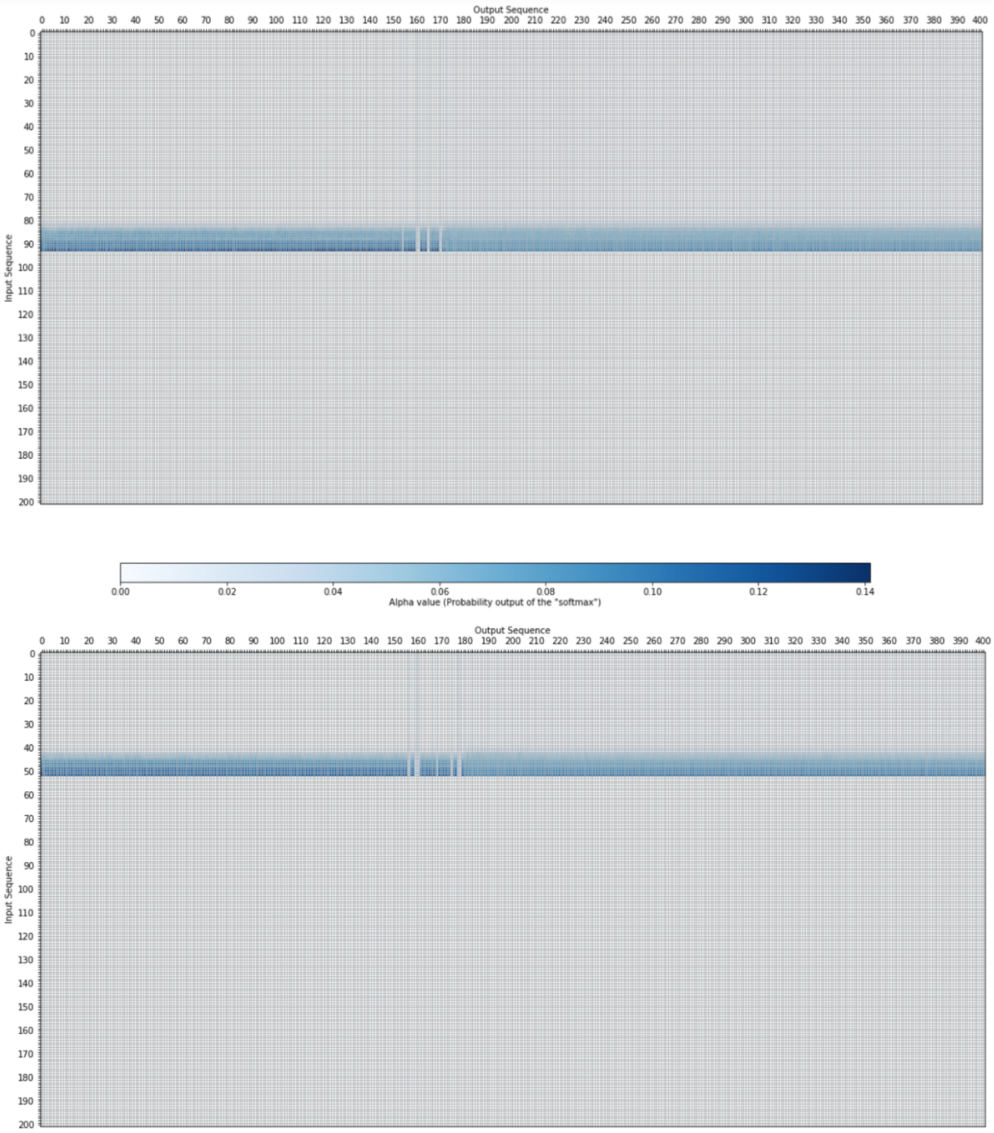

Figure 18: Attention Maps for 41Q6 (top) and 6QPR (bottom)

7.4 A Landscape Comparison

Proteins fold in high-dimensional energy landscapes through myriads of confirmations [60]. The energy landscape theory of protein folding is a statistical description of a proteins potential surface [61]. It suggests an overall topography of a rugged funnel, with a bias towards the native structure, as the most realistic shape of a proteins energy landscape. To ensure folding occurs on biologically relevant timescales, the overall gradient of the funnel must be steep when compared to the roughness caused by local minima [62]. However, the high frequency of deep accessible minima still creates a highly non-convex funnel, littered with saddle points.

The native structure of the protein is found at the global optima of its energy landscape, which correspondingly is well-predicted near the global optima of the objective function of the neural network. Whether the similarities cease here is unknown, yet certainly intriguing. Especially in light of research into loss surfaces showing networks that create funnel-like landscapes [63]. Choromanska *et al.* suggested that energy/optimisation landscapes of deep neural networks resemble the energy landscape of zero-temperature Gaussian spin glass [64]. Given that some neural networks

hint at similarity with glassy spin systems from statistical mechanics [65, 66, 67, 68], it is promising then that protein folding has often been modelled using spin-glasses [69, 70, 71].

8 Future Work

AlphaFold used the second-order gradient descent optimisation algorithm, L-BFGS. L-BFGS is of the quasi-Newton family of optimisation methods, which approximates the Broyden-Fletcher-Goldfarb-Shanno algorithm using a limited amount of computer memory. It uses an estimation of the inverse Hessian matrix to steer its search through variable space, as opposed to the Jacobian matrix used in standard SGD or Adam optimisation. Newton methods have a dampened phase and a quadratically convergent phase, meaning they generally require fewer iterations to converge when compared to stochastic optimisation methods. However, this comes at the trade-off of compute time per iteration. If it holds that the error landscape of the loss function resembles the energy landscape of the protein folding process, it would be interesting to see the effect of using this second-order optimisation method to attempt to better navigate the funnel.

Moreover, upon researching the ways in which gradient descent operates on highly non-convex error landscapes, a deep interest in this area of Machine Learning was formed. An in depth analysis of these high-dimensional error surfaces would be an interesting avenue of future research that may reveal insights that ultimately lead to a better engineered approach to the problem. Great examples of this are: [72, 63, 73]. Personally, visualising the intricacies of neural networks helps build a more intuitive understanding of the complex ways in which they work.

After conferring with AlQuraishi about the drawbacks of pNeRF, it became apparent that an improved version of the algorithm could be easily implemented to include the idealised bond lengths and angles. Such an algorithm, pNeRFect, would simply replace the mean bond lengths and angles of each residue in the chain with draws from their respective distributions. Further, secondary structure predictions could be coupled with the model to better draw from these distributions. A quick Monte Carlo simulation would produce a range of idealised configurations that may ultimately improve the GDT_TS.

Other regularisation strategies were experimented with, such as Ridge Regression, Lasso Regression and Elastic Net regression. However, due to time constraints no added benefit was realised from these experiments. In future, a more comprehensive analysis of the effect of combinations of L_1 and L_2 regularisation may reveal added improvements. It is unclear at the time of writing how the application of these strategies would affect performance. It would be interesting to test the impact of these regularisation strategies on both recurrent connections and non-recurrent connections.

As the model only predicts α-helices, regardless of the input sequence, it may be useful to take note of GNMTs approach to rare words. In this case, every torsion pair that is not from the α-helix region of the Ramachandran plot can be considered rare, compared to the α-helix torsions. It may be necessary to even up the dataset to include an equal amount of α-helices, β-sheets and β-turns. Experimentation into the affect of levelling this distribution in the training set is likely to yield performance improvements in the model.

In future attempts, more powerful hardware is a necessity. Utilising Google's ML Engine would enable access to more GPU power as well the number of GPUs. GNMT places each layer of the encoder and decoder on its own specific GPU, enabling a highly parallelised training process. Additionally models with different parameters could be run in parallel to better search the hyper-parameter space which could introduce performance benefits. Google Colab could not handle more

than 2000 proteins as it was limited to 12GB of RAM and preprocessing uses this memory entirely. Again migration to the cloud opens up increased memory such that it would be possible to include more proteins in future, thus increasing the training, validation and testing dataset sizes.

Finally, increasing the depth of the network as well as the number of hidden units, alongside training over a longer time-frame would likely achieve better results. Improved hardware is crucial for this to occur and any future work would have to capitalise on the added benefits from increasing GPU capacity and number.

9 Conclusion

This research devised a novel method of protein structure prediction, taking inspiration from AlphaFold and GNMT, framing the task as an NMT problem. It showed that physiochemical information can be well-encoded in the input representation by embedding n-grams in vector space. It showed that the chosen output representation meant legal structures were more likely to form. The report also includes experimentation into various hyperparameters with the goal of improving performance.

However, the approach of the research failed to materialise to competitive results when benchmarked against the contenders of CASP13. Considering the research was conducted independently, and that CASP predictors have whole teams of researchers behind them, the existence of an actual sequence-to-structure mapping that produces accurate secondary structure is promising, even if it is only one secondary structural motif.

Analysis into the performance of the model was conducted, highlighting distributions of angles on the Ramachandran plot as a limiting factor to performance. When analysing attention it appears that the length of the chains may be another limiting factor in the success of the model. Translating languages will vary rarely take a sentence that is more than 200 words long, if ever. A method of ensuring gradient propogation over longer distances may need to be devised and used in conjunction with the attention mechanism.

The report finishes by discussing the various avenues of future research that would be interesting to explore. From understanding error landscapes and their associations with protein folding funnels to improving the algorithms that facilitated this research, there are many directions this research could be taken in with the aim of improving results.

In conclusion, the model devised does predict structure more efficiently than Molecular Dynamics simulations, yet it fails to reach comparable levels of accuracy to the state-of-the-art methods.

References

[1] Y. Yuedong, G. Jianzhao, W. Jihua, H. Rhys, H. Jack, P. Kuldip, and Z. Yaoqi. Sixty-five years of the long march in protein secondary structure prediction: the final stretch? *Briefings in Bioinformatics*, 19(3):482–494, 2018.

[2] F. Alan. *Structure and Mechanism in Protein Science: Guide to Enzyme Catalysis and Protein Folding*. World Scientific, 1999.

[3] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell, 4th edition*. Garland Science, 2002.

[4] G. N. Ramachandran. Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, 7:95–99, 1963.

[5] G. N. Ramachandran and V. Sasisekharan. Conformation of polypeptides and proteins. *Advances in Protein Chemistry*, 23:283–438, 1968.

[6] C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

[7] C. B. Anfinsen, R. Redfield, Wa. Choate, J. Page, and W. Carroll. Studies on the gross structure, cross-linkages, and terminal sequences in ribonuclease. *Journal of Biological Chemistry*, 2017:201–210, 1954.

[8] C. B. Anfinsen, E. Haber, M. Sela, and F. H. White. The kinettics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *Proceedings of the National Academy of Sciences of the United States of America*, 47(9):13091314, 1961.

[9] F. H. White. Regeneration of native secondary and tertirary structures by air oxidation of reduced ribonuclease. *The Journal of Biological Chemistry*, 236:1353–1360, 1961.

[10] I. Andrea and S. Carmelinda. Protein structure determination by X-ray crystallography. *Methods in molecular biology (Clifton, N.J.)*, 452:63–87, 02 2008.

[11] E. Carpenter, K. Beis, A. Cameron, and S. Iwata. Overcoming the challenges of membrane protein crystallography. *Current opinion in structural biology*, 18(5):581–586, 10 2008.

[12] D. E. Shaw, M. Deneroff, R. Dror, J. Kuskin, R. Larson, J. Salmon, C. Young, B. Batson, K. Bowers, J. Chao, M. Eastwood, J. Gagliardo, J. Grossman, C. Ho, and D. Ierardi. Anton, a special-purpose machine for molecular dynamics simulation. *Communications of the ACM*, 51:91–97, 2008.

[13] D. E. Shaw. Millisecond-scale molecular dynamics simulations on Anton. Technical report, D. E. Shaw Research, 2009.

[14] V. Voelz, G. Bowman, K. Beauchamp, and V. Pande. Molecular simulation of ab initio protein folding for a millisecond folder NTL9(1-39). *Journal of the American Chemical Society*, 132(5):1526–1528, 2010.

[15] A. Kryshtafovych, Bohdan Monastyrskyy, Krzysztof Fidelis, John Moult, Torsten Schwede, and Anna Tramontano. Evaluation of the template-based modeling in casp12. *Proteins: Structure, Function, and Bioinformatics*, 86(S1):321–334, 2018.

[16] A. Roy, A. Kucukural, and Y. Zhang. I-TASSER: a unified platform for automated protein structure and function prediction. *Nature Protocols*, 5:725 EP –, 03 2010.

[17] R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Zidek, A. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, D. Jones, D. Silver, K. Kavukcuoglu, D. Hassabis, and A. Senior. De novo structure prediction with deeplearning based scoring. *In Thirteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstracts)*, 2018.

[18] M. Remmert, A. Biegert, and A. Hauser. Hhblits: Lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature Methods*, 9, 2012.

[19] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144, 2016.

[20] F. Bernstein, T. Koetzle, G. Williams, E. Meyer, M. Brice, J. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank: a computer-based archival file for macromolecular structures. *Journal of molecular biology*, 112(3):535–542, May 1977.

[21] J.M. Zimmerman, N. Eliezer, and R. Simha. The characterization of amino acid sequences in proteins by statistical methods. *Journal of Theoretical Biology*, 21(2):170 – 201, 1968.

[22] C. Tanford. Contribution of Hydrophobic Interactions to the Stability of the Globular Conformation of Proteins. *Journal of the American Chemical Society*, 84(22):4240–4247, 1962.

[23] M. Fauchere, J.and Charton, L. Kier, A. Verloop, and V. Pliska. Amino acid side chain parameters for correlation studies in biology and pharmacology. *International Journal of Peptide and Protein Research*, 32(4):269–278, 1988.

[24] P. Chou and G. Fasman. *Prediction of the Secondary Structure of Proteins from their Amino Acid Sequence*, pages 45–148. John Wiley & Sons, Ltd, 2006.

[25] Z.. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954.

[26] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, abs/1310.4546, 2013.

[27] E. Asgari and M. Mofrad. ProtVec: A Continuous Distributed Representation of Biological Sequences. *CoRR*, abs/1503.05140, 2015.

[28] L. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9, 2008.

[29] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078, 2014.

[30] A. Graves, A. Mohamed, and G. Hinton. Speech Recognition with Deep Recurrent Neural Networks. *CoRR*, abs/1303.5778, 2013.

[31] D. Britz, A. Goldie, M. Luong, and Q. Le. Massive Exploration of Neural Machine Translation Architectures. *CoRR*, abs/1703.03906, 2017.

[32] M. Schuster and K.K. Paliwal. Bidirectional Recurrent Neural Networks. *Trans. Sig. Proc.*, 45(11):2673–2681, 1997.

[33] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM networks. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 4:2047–2052, 2005.

[34] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies, 2001.

[35] S. Hochreiter and J. Schmidhuber. Long Short Term Memory. *Neural Computation*, 9:1735–1780, 1997.

[36] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[37] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[38] Y Dauphin, R. Pascanu, Ç. Gülçehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *CoRR*, abs/1406.2572, 2014.

[39] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. *CoRR*, abs/1507.06228, 2015.

[40] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. *CoRR*, abs/1409.3215, 2014.

[41] T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. *CoRR*, abs/1410.8206, 2014.

[42] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015.

[43] R.K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[44] K. Greff, R.K. Srivastava, and J. Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *CoRR*, abs/1612.07771, 2016.

[45] J. Zhou, Y. Cao, X. Wang, P. Li, and W. Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.

[46] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-term Dependencies with Gradient Descent is Difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994.

[47] R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[48] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. *CoRR*, abs/1212.0901, 2012.

[49] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[51] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.

[52] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR*, abs/1409.1259, 2014.

[53] D. Bahdanau, K. Cho, and Y.Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014.

[54] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[55] M. Luong, H. Pham, and C. Manning. Effective Approaches to Attention-based Neural Machine Translation. *CoRR*, abs/1508.04025, 2015.

[56] J. Parsons, B. Holmes, M. Rojas, J. Tsai, and C. Strauss. Practical conversion from torsion space to cartesian space for in silico protein synthesis. *Journal of Computational Chemistry*, 26(10):1063–1068, 2005.

[57] M. AlQuraishi. pNeRF: Parallelized Conversion from Internal to Cartesian Coordinates. *bioRxiv*, 2018.

[58] R. A. Engh and R. Huber. Accurate bond and angle parameters for X-ray protein structure refinement. *Acta Crystallographica Section A*, 47(4):392–400, Jul 1991.

[59] Wouter G. Touw and Gert Vriend. On the complexity of engh and huber refinement restraints: the angle as example. In *Acta crystallographica. Section D, Biological crystallography*, 2010.

[60] N. Hori, G. Chikenji, S. Berry, and S. Takada. Folding energy landscape and network dynamics of small globular proteins. *Proceedings of the National Academy of Sciences*, 106(1):73–78, 2009.

[61] J. Onuchic, Z. Luthey-Schulten, and P. Wolynes. THEORY OF PROTEIN FOLDING: The Energy Landscape Perspective. *Annual Review of Physical Chemistry*, 48(1):545–600, 1997. PMID: 9348663.

[62] J. Wang, R. Oliveira, X. Chu, P. Whitford, J. Chahine, W. Han, E. Wang, J. Onuchic, and V. Leite. Topography of funneled landscapes determines the thermodynamics and kinetics of protein folding. *Proceedings of the National Academy of Sciences*, 109(39):15763–15768, 2012.

[63] H. Li, Z. Xu, G. Taylor, and T. Goldstein. Visualizing the Loss Landscape of Neural Nets, 2018.

[64] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun. The loss surface of multilayer networks. *CoRR*, abs/1412.0233, 2014.

[65] C. Martin and M. Mahoney. Implicit Self-Regularization in Deep Neural Networks: Evidence from Random Matrix Theory and Implications for Learning. *CoRR*, abs/1810.01075, 2018.

[66] L. Sagun, V. Ugur Güney, and Y. LeCun. Explorations on high dimensional landscapes. *CoRR*, abs/1412.6615, 2014.

[67] A. Barra, G. Genovese, F. Guerra, and D. Tantari. How glassy are neural networks? *Journal of Statistical Mechanics: Theory and Experiment*, 2012(07):P07009, jul 2012.

[68] M. Baity-Jesi, L. Sagun, M. Geiger, S. Spigler, G. Ben Arous, C. Cammarota, Y. LeCun, M. Wyart, and G. Biroli. Comparing dynamics: Deep neural networks versus glassy systems. In *ICML*, 2018.

[69] J. Bryngelson and P. Wolynes. Spin glasses and the statistical mechanics of protein folding. *Proceedings of the National Academy of Sciences*, 84(21):7524–7528, 1987.

[70] Piotr Garstecki, Trinh Xuan Hoang, and Marek Cieplak. Energy landscapes, supergraphs, and "folding funnels" in spin systems. *Phys. Rev. E*, 60:3219–3226, Sep 1999.

[71] V.S. Pande, A.Y. Grosberg, and T. Tanaka. Statistical mechanics of simple models of protein folding and design. *Biophysical Journal*, 73(6):3192 – 3210, 1997.

[72] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah. Activation atlas. *Distill*, 2019. https://distill.pub/2019/activation-atlas.

[73] D. Im, M. Tao, and K. Branson. An empirical analysis of deep network loss surfaces. *CoRR*, abs/1612.04010, 2016.

A PDB Codes

1AH7, 1AJ2, 1AL3, 1ALQ, 1AOL, 1ARB, 1ATG, 1AZO, 1B8O, 1BHE, 1BS0, 1BS1, 1C1K, 1C3D, 1C8Z, 1CC4, 1CEO, 1CEX, 1CFB, 1CHD, 1CT5, 1DD9, 1DMG, 1DS1, 1DUA, 1DUW, 1DZF, 1E5K, 1EFD, 1EG2, 1EG3, 1EH6, 1ELV, 1EOK, 1ES5, 1ES6, 1F00, 1F9V, 1FC6, 1FCQ, 1FHU, 1FKM, 1FL2, 1FNF, 1FO8, 1FOB, 1FT5, 1FVI, 1FYE, 1G3P, 1G3Q, 1G55, 1G66, 1G8A, 1GA8, 1GCI, 1GP6, 1GPP, 1GQE, 1GV9, 1GXN, 1H2E, 1H6T, 1HCZ, 1HDO, 1HG5, 1HH8, 1HQ0, 1HXN, 1I1W, 1I5R, 1I60, 1I9G, 1I9S, 1I9Z, 1IAP, 1IHG, 1IJB, 1IN4, 1IO2, 1IOW, 1IPC, 1IQQ, 1IUQ, 1IXH, 1IXK, 1IYN, 1J1T, 1J5X, 1J77, 1J7X, 1JFX, 1JG1, 1JIX, 1JJF, 1JJV, 1JM1, 1JOV, 1JP4, 1JX6, 1JYK, 1K1B, 1K5C, 1K77, 1K7C, 1K7J, 1K7K, 1KA1, 1KEA, 1KG2, 1KGS, 1KI0, 1KJW, 1KQ3, 1KS9, 1KSK, 1KUX, 1KWF, 1KYH, 1KZF, 1LBU, 1LC0, 1LFP, 1LS1, 1LV7, 1LZJ, 1LZL, 1M15, 1M1H, 1M40, 1M4L, 1MIX, 1MJ5, 1ML4, 1ML9, 1MUW, 1MVL, 1N40, 1N70, 1NAR, 1NC5, 1NE9, 1NIJ, 1NKR, 1NMY, 1NNF, 1NNH, 1NTY, 1NUY, 1NWA, 1NZJ, 1O1Y, 1O9G, 1OC7, 1ODM, 1OEW, 1OFC, 1OG1, 1OGQ, 1OHT, 1OI7, 1OIS, 1OUV, 1OXX, 1OZ2, 1P0H, 1P3C, 1P99, 1PG6, 1PHP, 1PRZ, 1PSW, 1PUJ, 1PV5, 1Q0P, 1Q0R, 1Q33, 1Q42, 1QAZ, 1QB0, 1QCS, 1QCX, 1QGQ, 1QNR, 1QNX, 1QOY, 1QQQ, 1QR0, 1QTS, 1QUS, 1QWY, 1R2D, 1R3F, 1R4X, 1R5L, 1R5Y, 1R6X, 1R9L, 1RC9, 1RGP, 1RHS, 1RO2, 1RTQ, 1RU4, 1RVK, 1RWR, 1RYO, 1RZ2, 1RZ3, 1S21, 1S2O, 1S2W, 1S2X, 1S68, 1S8N, 1S9U, 1SDI, 1SFS, 1SGW, 1SO7, 1SQ9, 1SQH, 1SR8, 1SUM, 1SUU, 1SVB, 1SYY, 1SZ7, 1T1G, 1T6C, 1T6E, 1T7V, 1T9H, 1TBF, 1TCA, 1TK1, 1TKE, 1TKJ, 1TL2, 1TOL, 1TQ5, 1TVZ, 1TXL, 1U02, 1U2K, 1U7L, 1U9C, 1UAS, 1UCH, 1UI0, 1UNF, 1US0, 1US5, 1UX6, 1UXY, 1V33, 1V77, 1V9F, 1V9M, 1VAJ, 1VCT, 1VE4, 1VHE, 1VHN, 1VHU, 1VIN, 1VK1, 1VK2, 1VK4, 1VKW, 1VL1, 1VLY, 1VPR, 1VRM, 1VYR, 1W0H, 1W66, 1WCH, 1WCW, 1WD5, 1WDE, 1WER, 1WF3, 1WJ9, 1WMA, 1WNH, 1WQ3, 1WR2, 1WU4, 1WV3, 1WZZ, 1XCL, 1XDN, 1XDZ, 1XFK, 1XGK, 1XIO, 1XKG, 1XKR, 1XOV, 1XQO, 1XRO, 1XTP, 1XZZ, 1Y08, 1Y0K, 1Y80, 1Y8A, 1YFQ, 1YHT, 1YI9, 1YLE, 1YQE, 1YQS, 1YS1, 1YT3, 1YTQ, 1YUL, 1YWF, 1YWM, 1Z2N, 1Z70, 1Z9T, 1ZA0, 1ZA4, 1ZAR, 1ZDY, 1ZGK, 1ZI8, 1ZJR, 1ZM8, 1ZN6, 1ZS9, 1ZSW, 1ZTF, 2A0M, 2A14, 2A6Y, 2A6Z, 2ABK, 2ABS, 2AE0, 2AGK, 2AH5, 2AMH, 2AP1, 2AQW, 2AXC, 2B4L, 2B4W, 2B78, 2B7U, 2B9E, 2B9L, 2BBH, 2BFW, 2BJQ, 2BL9, 2BOG, 2BSY, 2BUE, 2BW0, 2BZG, 2C43, 2C4X, 2C71, 2CB9, 2CI1, 2CIS, 2CIW, 2CNQ, 2CUL, 2CWA, 2CWS, 2CXA, 2CXH, 2CXV, 2CY7, 2CYG, 2CZL, 2D3Y, 2D4X, 2D7J, 2D81, 2DBY, 2DCF, 2DDX, 2DE2, 2DEJ, 2DHO, 2DKV, 2DPM, 2DUL, 2E3N, 2E6M, 2E8B, 2EBN, 2EGU, 2EI9, 2EJ9, 2ERF, 2ETB, 2ETJ, 2F1N, 2F4Q, 2F68, 2F6R, 2FBO, 2FDR, 2FEZ, 2FK8, 2FM9, 2FQX, 2FSJ, 2FSQ, 2FSU, 2FUZ, 2FVY, 2FWV, 2FX5, 2FXU, 2FY7, 2G62, 2G9F, 2GA8, 2GFH, 2GFO, 2GGC, 2GHS, 2GJL, 2GKE, 2GNO, 2GNP, 2GQT, 2GS8, 2GU1, 2GWM, 2GZQ, 2GZS, 2H14, 2H1V, 2H7O, 2HBW, 2HC1, 2HCF, 2HDO, 2HES, 2HHC, 2HJE, 2HL9, 2HLY, 2HQ0, 2HRX, 2HUO, 2HYK, 2I0O, 2I1Q, 2I53, 2I5H, 2I5V, 2I6D, 2II2, 2ILR, 2IM9, 2IMQ, 2IN3, 2IRX, 2IS9, 2IU1, 2IVN, 2IW0, 2IW1, 2IWA, 2IY9, 2J0A, 2J89, 2J8G, 2J8K, 2JAY, 2JFN, 2JFR, 2JH1, 2JKS, 2NLR, 2NRT, 2NSF, 2NWH, 2NXC, 2NXF, 2O0A, 2O0J, 2O0M, 2O2G, 2O2X, 2O5V, 2O7R, 2O8L, 2OBL, 2ODH, 2ODL, 2OF3, 2OG2, 2OG4, 2OI0, 2OJH, 2OKT, 2OLN, 2ONS, 2OO3, 2OPJ, 2OPW, 2OQZ, 2ORP, 2OSA, 2OVJ, 2OXN, 2OYR, 2OZT, 2P0K, 2P17, 2P18, 2P4F, 2P4H, 2P4O, 2P51, 2P6W, 2P8T, 2P9W, 2PB7, 2PC1, 2PES, 2PET, 2PFZ, 2PGE, 2PGF, 2PIA, 2PJZ, 2PLW, 2PNW, 2PQ8, 2PQX, 2PU3, 2PWQ, 2PXX, 2Q0I, 2Q3H, 2Q40, 2Q4M, 2Q88, 2Q8P, 2Q99, 2QDJ, 2QED, 2QFL, 2QGU, 2QHP, 2QHS, 2QHT, 2QIK, 2QK1, 2QLT, 2QMQ, 2QNK, 2QOL, 2QQ8, 2QQI, 2QRU, 2QT1, 2QTF, 2QX2, 2R0S, 2R31, 2R3A, 2R44, 2R4G, 2R58, 2R9F, 2RA8, 2RAU, 2RBK, 2RCI, 2RDI, 2RDQ, 2REU, 2RIK, 2UUR, 2UYO, 2V05, 2V0S, 2V3G, 2V3K, 2V7K, 2VCL, 2VDF, 2VGA, 2VOV, 2VQP, 2VQX, 2VX5, 2VY6, 2VYO, 2W0M, 2W15, 2W18, 2W1N, 2W2R, 2W39, 2W3Z, 2WAA, 2WAO, 2WB0, 2WBF, 2WBN, 2WBQ, 2WF7, 2WFP, 2WHK, 2WHL, 2WJR, 2WKQ, 2WM3, 2WNF, 2WNK, 2WNP, 2WOY, 2WT0, 2WTA, 2WUR, 2WXU, 2X0C, 2X3M, 2X47, 2X49, 2X4L, 2X55, 2X6U, 2X7Q, 2X8X, 2X9O, 2X9Z, 2XB0, 2XBG, 2XGR, 2XIO, 2XJ4, 2XJP, 2XM5, 2XMZ, 2XTP, 2XU3, 2XUZ, 2XWV, 2XXP, 2XY4, 2XZG, 2Y24, 2Y3C, 2Y8D, 2YAS, 2YB1, 2YD4, 2YHA, 2YHC, 2YLH, 2YMO, 2YMV, 2YN0, 2YV7, 2YVT, 2YXX, 2YZQ, 2YZV, 2YZY, 2Z0M, 2Z0Q, 2Z1E, 2Z2N, 2Z2Z, 2Z72, 2Z84, 2Z8L, 2ZCU, 2ZFY, 2ZHJ, 2ZPT, 2ZQ5, 2ZU5, 2ZZJ, 3A0N, 3A1T, 3A72, 3AAP, 3ACH, 3ACX, 3AI4, 3AI5, 3AJD, 3AKE, 3AL2, 3ALJ, 3AMR, 3ATS, 3B49, 3B5O, 3B6E, 3B8B, 3BB7, 3BDI, 3BI7, 3BIY, 3BK5, 3BKB, 3BKP, 3BO5, 3BOE, 3BPT, 3BQK, 3BWH, 3BWX, 3C0S, 3C5H, 3C5V, 3C6A, 3C8U, 3C9Z, 3CGX, 3CH0, 3CH4, 3CHJ, 3CJJ, 3CJM, 3CKK, 3CKM, 3CLM, 3COU, 3CRR, 3CS1, 3CU9, 3CZ7, 3D02, 3D06, 3D1R, 3D30, 3D6J, 3D8R, 3DAS, 3DCY, 3DCZ, 3DDJ, 3DEF, 3DF7, 3DFF, 3DHA, 3DHN, 3DLC, 3DMG, 3DS8, 3DSM, 3DT8, 3DU1, 3DV9, 3DXL, 3DXY, 3E23, 3E2O, 3E46, 3E8T, 3E9O, 3EA6, 3EAF, 3EDH, 3EG4, 3EGA, 3ELB, 3EMR, 3ERB, 3ESQ, 3ESS, 3ETV, 3EVN, 3EVP, 3EW8, 3EWN, 3EXM, 3F44, 3F4S, 3F67, 3F6Y, 3F7W, 3FAN, 3FHG, 3FND, 3FO8, 3FQG, 3FRH, 3FSS, 3FTD, 3FTJ, 3FWK, 3FZG, 3G3T, 3G40, 3G5T, 3G7U, 3G85, 3G91, 3GA7, 3GD0, 3GG7, 3GHA, 3GIR, 3GIW, 3GJY, 3GKJ, 3GMF, 3GMI, 3GMS, 3GNZ, 3GO5, 3GOH, 3GON, 3GPI, 3GQJ, 3GQV, 3GVO, 3H04, 3H0O, 3H14, 3H20, 3H2G, 3H2Y, 3H41, 3H4X, 3H74, 3H75, 3H7C, 3H7I, 3HBN, 3HC1, 3HC7, 3HP7, 3HR8, 3HRA, 3HRG, 3HUT, 3HX3, 3HY3, 3I10, 3I2N, 3I36, 3I45, 3I47, 3I6I, 3I94, 3IDV, 3IEE, 3II7, 3IID, 3IKW, 3ILS, 3IM1, 3IM6, 3IO3, 3IPC, 3IPX, 3IR4, 3IU0, 3IUZ, 3IVF, 3IXL, 3JS6, 3JTM, 3JWG, 3K0B, 3K1Z, 3K7X, 3K8W, 3KA2, 3KB9, 3KBG, 3KCI, 3KFO, 3KG4, 3KHI, 3KOG, 3KRE, 3KSX, 3KV0, 3KVD, 3KZX, 3L23, 3L3F, 3L4E, 3L5I, 3L6A, 3L6G, 3L7Y, 3L9U, 3LCC, 3LEC, 3LHI, 3LHO, 3LI9, 3LJU, 3LKM, 3LLC, 3LMC, 3LO8, 3LOP, 3LP5, 3LPZ, 3LQ0, 3LR1, 3LTI, 3LTJ, 3M1M, 3M1T, 3M31, 3M3P, 3M4R, 3M5Q, 3M66, 3M70, 3M7D, 3MBR, 3MDQ, 3MQZ, 3MST, 3MT0, 3MVS, 3MW8, 3MWQ, 3MZF, 3N0R, 3N35, 3N3U, 3N6T, 3N6Z, 3NBI, 3NE0, 3NNB, 3NO2, 3NOL, 3NUQ, 3NUR, 3NWO, 3NYC, 3O0F, 3O0P, 3O12, 3O15, 3O2T, 3O4P, 3O6N, 3O6P, 3O7W, 3OCJ, 3OHG, 3OKP, 3OOS, 3OQV, 3OSD, 3OSX, 3OU2, 3OXH, 3OYV, 3OZ2, 3P02, 3P28, 3P2H, 3P2W, 3P4L, 3P6L, 3PB6, 3PDT, 3PDW, 3PE6, 3PE7, 3PF2, 3PHS, 3PKV, 3PMM, 3PMS, 3PNN, 3POW, 3PSH, 3PT5, 3PUA, 3PV7, 3PYW, 3Q13, 3Q1C, 3Q1N, 3Q27, 3Q2U, 3Q60, 3Q7C, 3Q7M, 3Q8G, 3QC5, 3QDH, 3QLE, 3QM1, 3QOU, 3QSJ, 3QUT, 3QWL, 3QY7, 3R0V, 3R2G, 3R5T, 3R6D, 3R9M, 3RAY, 3RBS, 3RC1, 3RD5, 3RGA, 3RJU, 3RJV, 3RKG, 3RL5, 3RLE, 3RLG, 3RM3, 3RNL, 3RPW, 3RQ4, 3RR6, 3RZV, 3S0K, 3S25, 3S2J, 3S3Z, 3S44, 3S5Q, 3S83, 3S98, 3S9J, 3SBM, 3SC0, 3SCY, 3SEE, 3SG0, 3SHQ, 3SHS, 3SIG, 3SIL, 3SK9, 3SKX, 3SLR, 3SMV, 3SMZ, 3SRE, 3SU6, 3SVJ, 3SY1, 3T0H, 3T0O, 3T6P, 3TAW, 3TBD, 3TBJ, 3TCQ, 3TDW, 3TEE, 3TGH, 3THI, 3TJM, 3TRD, 3TRQ, 3TT9, 3TTG, 3TU8, 3TUT, 3TVY, 3TWZ, 3U1L, 3U26, 3U3L, 3U7E, 3U9G, 3UC9, 3UEN, 3UF8, 3UGU, 3UJC, 3UMH, 3UQ8, 3UUE, 3UWC, 3V0E, 3V2I, 3V55, 3V5F, 3V68, 3VGI, 3VJ9, 3VMV, 3VN5, 3VNE, 3VRN, 3VTG, 3VUR, 3VWD, 3W06, 3W0F, 3W0O, 3W1E, 3WCX, 3WDQ, 3WE9, 3WH1, 3WND, 3WP4, 3WS7, 3WW9, 3WWA, 3WWC, 3WY8, 3ZBQ, 3ZD9, 3ZGH, 3ZI1, 3ZJ0, 3ZNV, 3ZUZ, 3ZXY, 3ZYP, 3ZYQ, 4A29, 4A3P, 4A3X, 4AC1, 4AF8, 4AK2, 4ANN, 4ANO, 4ASC, 4ATE, 4ATN, 4AVA, 4AWN, 4AX1, 4AYL, 4B0A, 4B29, 4B2F, 4B3X, 4B46, 4B89, 4BGP, 4BJI, 4BMH, 4BN4, 4BPS, 4BZA, 4C5P, 4CBE, 4CCS, 4CCW, 4CD8, 4CDP, 4CFI, 4CH7, 4CIL, 4COS, 4CZG, 4D01, 4D0P, 4D3X, 4D7R, 4D8B, 4DB6, 4DCG, 4DDP, 4DI9, 4DKQ, 4DLM, 4DNU, 4DQA, 4DTH, 4DZ1, 4E14, 4E1B, 4E1S, 4E40, 4EBY, 4ECF, 4EDK, 4EFI, 4EG9, 4EHC, 4EHX, 4EIU, 4EKF, 4EMD, 4EQ3, 4ERN, 4ES7, 4ESM, 4EU0, 4EUO, 4EV1, 4EVF, 4EWE, 4EX6, 4EXQ, 4EYS, 4EZI, 4F06, 4F0J, 4F1V, 4F67, 4F6P, 4FB7, 4FBR, 4FCG, 4FE3, 4FF5, 4FFL, 4FH3, 4FHG, 4FK9, 4FQE, 4FS7, 4FS8, 4FUU, 4G3N, 4G54, 4G75, 4G7L, 4GBM, 4GC3, 4GC5, 4GGC, 4GHN, 4GJZ, 4GNR, 4GSJ, 4GSR, 4GT6, 4GXT, 4H14, 4H1X, 4H27, 4H59, 4H79, 4HBD, 4HD5, 4HD9, 4HLJ, 4HNO, 4HO3, 4HTY, 4HYQ, 4I05, 4I17, 4I1F, 4I8H, 4I8I, 4I90, 4I92, 4IBN, 4IC4, 4IFA, 4IGI, 4IIK, 4IIL, 4IM6, 4INK, 4IU6, 4IUS, 4IXE, 4IYA, 4IYK, 4J0W, 4J37, 4J7N, 4J87, 4J8L, 4J9T, 4JB3, 4JBU, 4JCC, 4JD0, 4JDF, 4JDU, 4JF5, 4JG2, 4JHT, 4JJA, 4JJO, 4JLX, 4JMP, 4JNF, 4JP0, 4JWO, 4JXH, 4JZ5, 4K0N, 4K4O, 4K6N, 4K82, 4KDR, 4KG7, 4KH8, 4KK7, 4KL0, 4KM6, 4KQ9, 4KQP, 4KRU, 4KUI, 4KV7, 4KVF, 4KYQ, 4KZK, 4L0J, 4L7N, 4LAY, 4LEU, 4LG8, 4LGJ, 4LIM, 4LJ9, 4LMP, 4LN7, 4LPQ, 4LQ6, 4LSC, 4LSW, 4LUN, 4LWL, 4LZD, 4LZH, 4M02, 4M0N, 4M5B, 4M5R, 4M7T, 4M82, 4M88, 4M8I, 4M9K, 4M9P, 4MAI, 4MBO, 4MC3, 4MCD, 4MDA, 4ME2, 4ME3, 4MEV, 4MEW, 4MFK, 4MI5, 4MKM, 4MLO, 4MLV, 4MNC, 4MO9, 4MUQ, 4MXD, 4MXP, 4N0D, 4N13, 4N1I, 4N30, 4N4G, 4N67, 4N6W, 4N77, 4N9W, 4NBR, 4NC6, 4NLM, 4NMW, 4NNO, 4NRP, 4NS5, 4NUI, 4NWB, 4NWK, 4NX2, 4NZK, 4NZN, 4O4F, 4OAI, 4OCV, 4OQ1, 4OQV, 4OVK, 4OX3, 4OZU, 4OZW, 4P0M, 4P48, 4P6Q, 4P7Q, 4P7X, 4P98, 4PDY, 4PED, 4PEU, 4PF3, 4PGR, 4PHR, 4PI8, 4PJR, 4PK9, 4PMH, 4PMQ, 4PO3, 4PP4, 4PPR, 4PQI, 4PSC, 4PWO, 4PYH, 4PYR, 4PYT, 4PZ0, 4Q2W, 4Q3J, 4Q68, 4Q8K, 4Q8R, 4Q98, 4QA8, 4QDJ, 4QEK, 4QGS, 4QHQ, 4QJK, 4QMA, 4QMH, 4QPN, 4QPT, 4QQ0, 4QRB, 4QRK, 4QTC, 4QTQ, 4R0J, 4R1B, 4R2H, 4R3F, 4R6F, 4R6Y, 4R75, 4R78, 4R9I, 4R9N, 4R9P, 4RAG, 4RAJ, 4RAX, 4RDB, 4RDC, 4REO, 4RL1, 4RMK, 4RNZ, 4ROR, 4RXV, 4RZA, 4TQX, 4TRK, 4TVE, 4TXD, 4U06, 4U0P, 4U3V, 4U82, 4U89, 4U9O, 4U9S, 4UAR, 4UAV, 4UJ7, 4UML, 4UOB, 4UQX, 4USR, 4UY5, 4UYR, 4UZ1, 4UZG, 4V3J, 4W79, 4W7W, 4W8B, 4WA0, 4WCJ, 4WE2, 4WIQ, 4WJE, 4WKA, 4WPG, 4WPK, 4WQD, 4WQM, 4WT3, 4WTP, 4WUI, 4WY9, 4WZG, 4WZZ, 4X1O, 4X2R, 4X35, 4X7G, 4X9T, 4X9X, 4XA7, 4XAB, 4XDU, 4XEH, 4XEZ, 4XFD, 4XFK, 4XGL, 4XIJ, 4XJ5, 4XKB, 4XKZ, 4XP7, 4XPZ, 4XQ7, 4XRW, 4XSJ, 4XTL, 4XXT, 4XXX, 4XZ0, 4Y0X, 4Y1W, 4Y63, 4Y7U, 4Y9T, 4YAA, 4YAH, 4YBG, 4YGS, 4YI8, 4YLA, 4YLE, 4YN1, 4YN8, 4YOD, 4YOK, 4YWF, 4YWR, 4Z1P, 4Z2N, 4Z51, 4Z9P, 4ZJH, 4ZNK, 4ZP0, 4ZX2, 4ZZ1, 5A0N, 5A3A, 5A4A, 5A67, 5A8C, 5A8I, 5AE0, 5AGI, 5AGR, 5AJG, 5AMV, 5AN4, 5AO9, 5AOY, 5AZB, 5B0T, 5B4Z, 5B5H, 5B5L, 5BOI, 5BP9, 5BPJ, 5BQ1, 5BR7, 5BTB, 5BV8, 5BVR, 5BYK, 5C0P, 5C12, 5C17, 5C1E, 5C2M, 5C30, 5C3U, 5C5G, 5CAD, 5CD2, 5CDZ, 5CET, 5CH8, 5CHH, 5CMM, 5CNW, 5COT, 5COZ, 5CR9, 5CWG, 5CXU, 5CZW, 5D2F, 5D3K, 5D6E, 5D8M, 5DAG, 5DJH, 5DLB, 5DM2, 5DMM, 5DP2, 5DUT, 5DV4, 5DXL, 5DZE, 5E6V, 5E6X, 5E72, 5E7H, 5E7L, 5EC6, 5EDF, 5EH1, 5EJ0, 5EKY, 5EKZ, 5EOV, 5EP0, 5EP2, 5EQV, 5EWY, 5EXJ, 5EZQ, 5F18, 5F1S, 5F7V, 5FAA, 5FBF, 5FEW, 5FFA, 5FSV, 5FTW, 5FUT, 5FXT, 5G3Y, 5G59, 5GGB, 5GHU, 5GI7, 5GJ3, 5GKV, 5GM9, 5GO5, 5GTQ, 5GU6, 5GV1, 5GV8, 5GVA, 5GYC, 5GZA, 5H02, 5H0J, 5H0Q, 5H28, 5H4E, 5H5F, 5H7K, 5H9I, 5HFG, 5HGZ, 5HJM, 5HKX, 5HL3, 5HQG, 5HUS, 5HWA, 5HWI, 5HYA, 5I10, 5I45, 5I8G, 5I9J, 5ICF, 5IDH, 5IDV, 5IGI, 5IHS, 5IHW, 5IK4, 5IKU, 5IMU, 5ITQ, 5IXP, 5J3U, 5J4L, 5J4O, 5J4R, 5JDA, 5JDD, 5JE5, 5JGJ, 5JH8, 5JI7, 5JJ2, 5JJO, 5JJX, 5JMU, 5JO8, 5JOH, 5JP6, 5JUF, 5JUH, 5JVF, 5JVI, 5JYS, 5K4C, 5KCO, 5KIV, 5KO3, 5KO9, 5KQR, 5KTN, 5KWB, 5KZL, 5L4L, 5LB6, 5LF9, 5LFZ, 5LHM, 5LJM, 5LNN, 5LP1, 5LQ5, 5LQ6, 5LS4, 5LSV, 5LT7, 5LTE, 5LW3, 5LW6, 5LY8, 5M04, 5M17, 5M3Q, 5M43, 5M82, 5ME4, 5MEK, 5MJR, 5MK9, 5MLH, 5MLP, 5MLZ, 5MOB, 5MQI, 5MSZ, 5MT2, 5MWA, 5MX9, 5MY5, 5N40, 5N7Z, 5N9B, 5NJI, 5NJL, 5NOA, 5O2D, 5O2X, 5O37, 5O6C, 5O7E, 5OBY, 5ODJ, 5OF2, 5OH5, 5OLT, 5OLU, 5OMY, 5ONK, 5OQ3, 5OUO, 5P9V, 5PHJ, 5Q22, 5QIF, 5SV6, 5T39, 5T76, 5T8C, 5TCB, 5THX, 5TPG, 5TTY, 5TUX, 5TXU, 5U19, 5U5O, 5U69, 5U81, 5UBA, 5UE0, 5UF2, 5UFY, 5UJC, 5UKH, 5UKI, 5ULB, 5UM2, 5UMS, 5UQ6, 5UQJ, 5UQZ, 5UVD, 5UVG, 5UWZ, 5UXY, 5UZ8, 5V7M, 5VBB, 5VEJ, 5VFZ, 5VIA, 5VJW, 5VTL, 5VWM, 5VXV, 5W2F, 5W2I, 5W5B, 5W8E, 5W98, 5WA2, 5WCF, 5WCJ, 5WGI, 5WL5, 5WLY, 5WOL, 5WQW, 5WY0, 5X16, 5X4R, 5X9L, 5XDT, 5XFO, 5XG2, 5XHB, 5XHW, 5XJ5, 5XSO, 5XTU, 5Y1A, 5Y1F, 5Y2S, 5Y3Q, 5Y6T, 5Y7R, 5Y92, 5Y9X, 5YAL, 5YBY, 5YFK, 5YGB, 5YGM, 5YHH, 5YKJ, 5YLT, 5YSC, 5YXM, 5YZP, 5Z4C, 5ZBF, 5ZBZ, 5ZDA, 5ZDM, 5ZGX, 5ZJ0, 5ZNT, 5ZOD, 5ZQA, 5ZQY, 6A42, 6A9W, 6AN9, 6AOK, 6APE, 6AQK, 6AVX, 6B2V, 6B91, 6BA9, 6BC7, 6BG3, 6BGD, 6BGY, 6BIO, 6BKA, 6BM5, 6BNZ, 6BO0, 6BT1, 6BXD, 6BXR, 6C0T, 6C10, 6C35, 6C37, 6CAF, 6CF3, 6CFV, 6CGJ, 6CH1, 6COK, 6D0G, 6D21, 6D46, 6D4K, 6DCH, 6DET, 6DFP, 6DG4, 6DI2, 6DJT, 6E0K, 6E0M, 6EAS, 6ECB, 6ECT, 6EDE, 6EDK, 6EF6, 6EHD, 6EHN, 6EIO, 6EKZ, 6EMS, 6EOU, 6EQE, 6ESK, 6EUG, 6EVU, 6EY1, 6EYG, 6F7B, 6F8B, 6F8P, 6FHO, 6FIE, 6FJ4, 6FM7, 6FMX, 6FNU, 6FSN, 6FSW, 6FUC, 6FZ9, 6G1G, 6G28, 6G7W, 6GEH, 6GKW, 6GW0, 6GZU, 6H20, 6H95, 6HFQ, 6HS5, 6I8Y, 6IBE, 6IEO, 6IH0, 6IJN, 6ISA, 6IYY, 6JDP, 6M9M, 6MBF, 6MD5, 6MFK, 6MIJ, 6MPT, 6MUQ, 6N9M, 6NHX, 6NIO, 6NS1, 6Q4G, 6QE3, 6QPR, 7A3H, 8ABP

B Embeddings

B.1 1-Grams

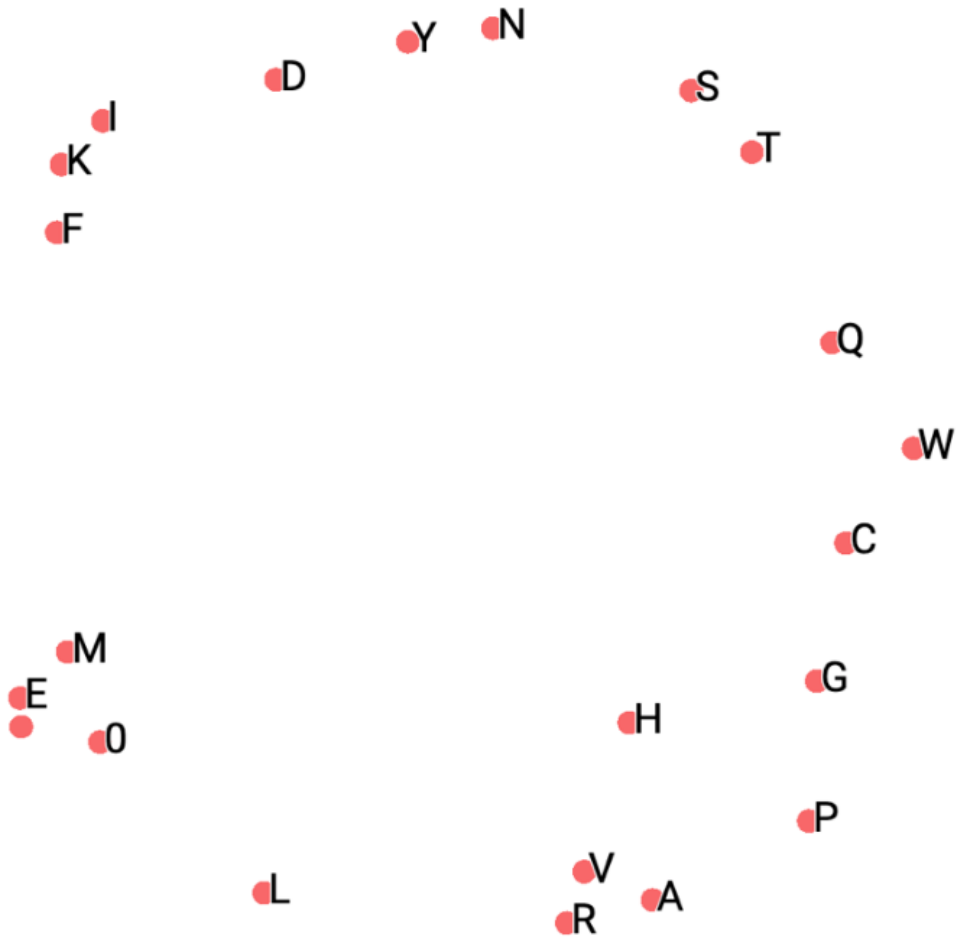

Figure 1: 1-gram

B.2 2-Grams

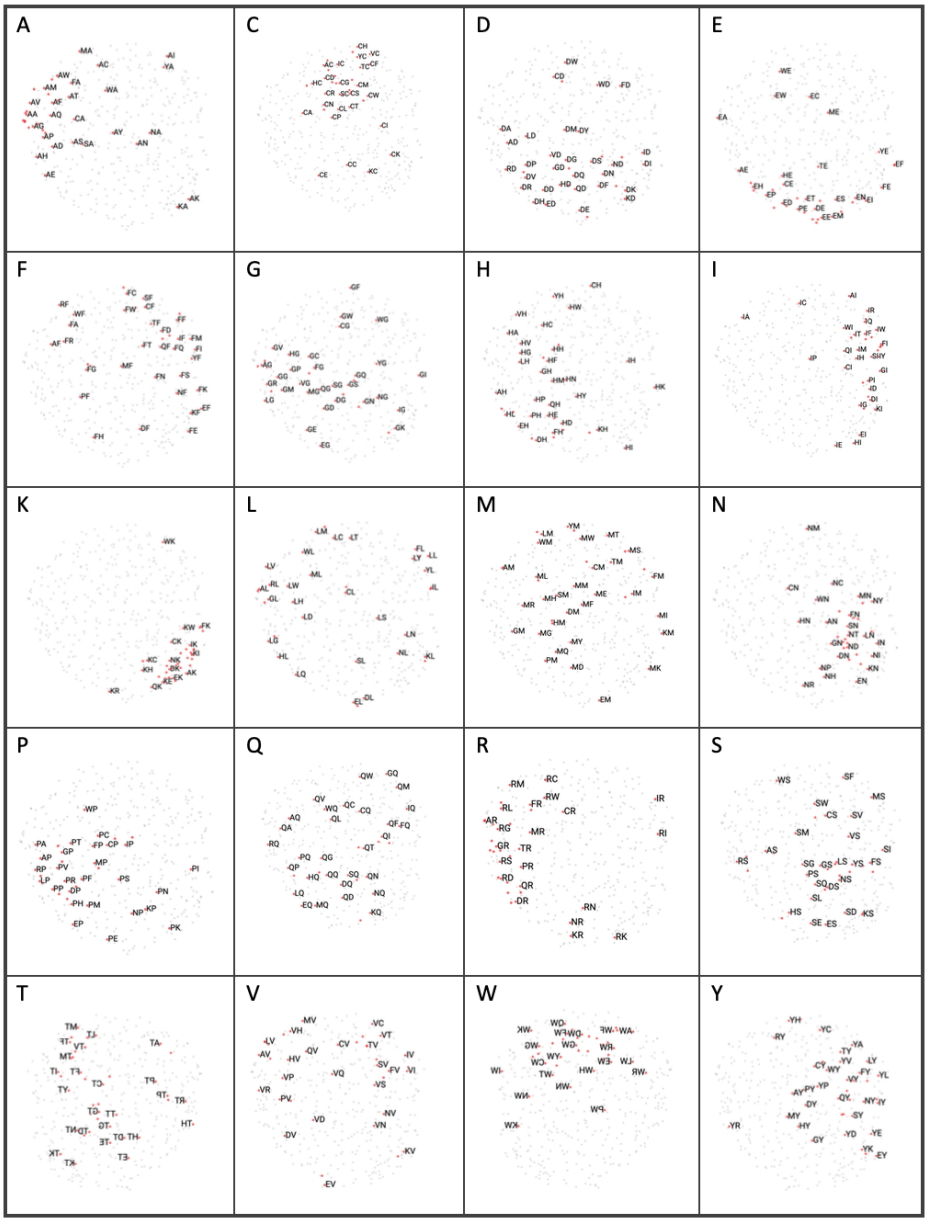

Figure 2: 2-gram

B.3 3-Grams

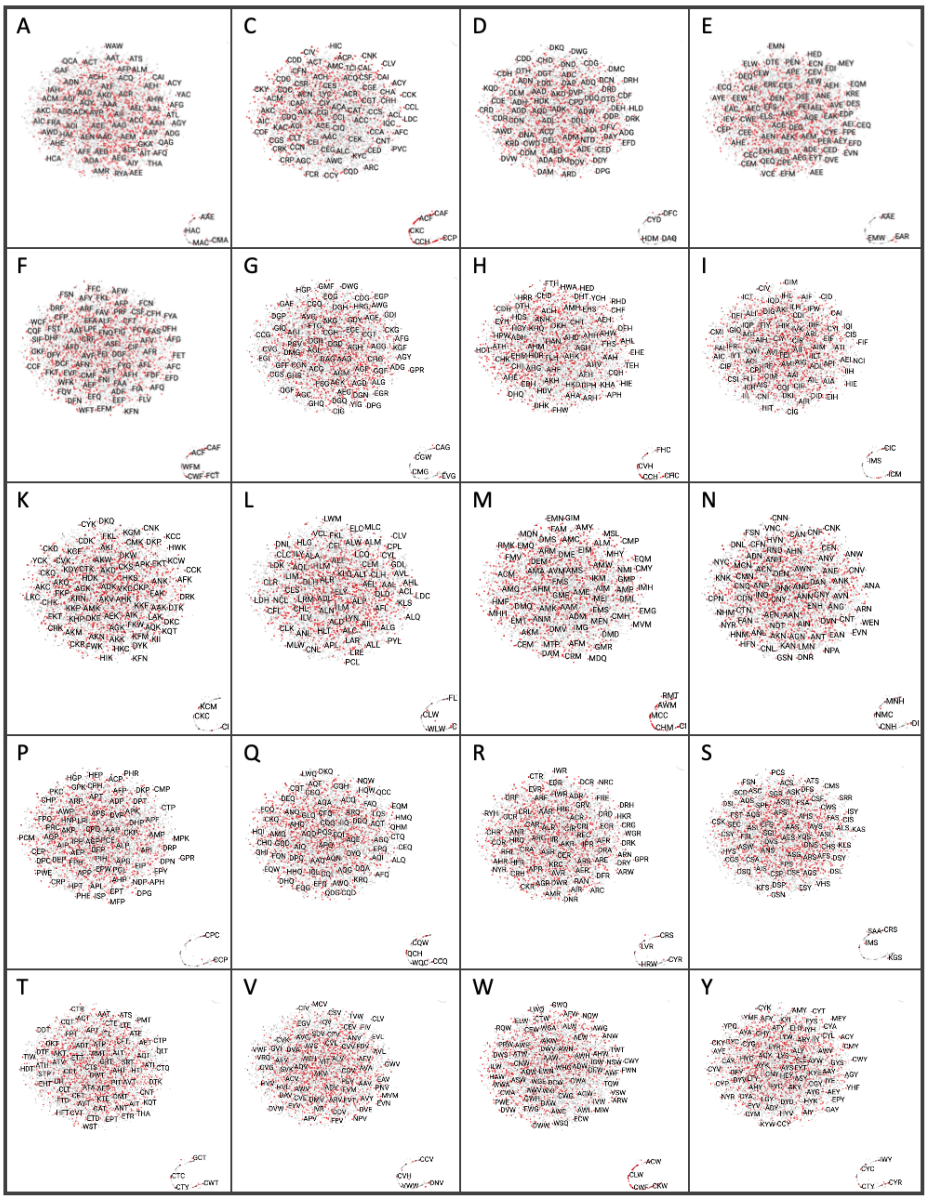

Figure 3: 3-gram