Transforming Protein Secondary Structure Prediction with Deep Learning: A Novel Approach Using Encoder-Decoder Architectures

Aakash Mor*,Sunisha Arora†
*University of the Arts London (aakashmor@gmail.com)
†University of St. Thomas - Minnesota (sunisha.arora1309@gmail.com)

***Abstract*—Predicting protein secondary structure is a pivotal challenge in structural biology, given its critical role in understanding protein function and designing novel therapeutics. Traditional methods, while informative, fall short of providing the accuracy and efficiency required for real-world applications. This thesis introduces an innovative application of Encoder-Decoder neural networks, originally designed for Natural Language Processing, to advance protein secondary structure prediction. By treating amino acid sequences as structured data akin to language, we leverage embeddings and sequential modeling to capture intricate relationships within the protein chain. We present three novel deep learning architectures tailored for this task and evaluate them on a custom-built dataset derived from the Protein Data Bank. Our approach demonstrates the potential to enhance Q8 accuracy and provide reliable preliminary predictions for experimental biologists. By bridging cutting-edge AI techniques with the demands of structural biology, this work aims to set a new benchmark for computational tools in protein research.**

I. Introduction

Protein secondary structure prediction remains a fundamental challenge in structural biology. Due to the intricate nature of this problem, no known closed-form equation can accurately determine the structure of a protein. It is unlikely that such an equation will ever be formulated. Nevertheless, knowledge of secondary structure is critical, as it significantly influences a protein's biochemical properties and functional mechanisms. With advancements in machine learning, computational approaches have gained prominence as viable tools for enhancing the accuracy of secondary structure prediction. Recent developments have propelled prediction accuracies beyond 80%, yet this remains insufficient given the complexity of molecular interactions governing protein function. This study explores state-of-the-art methodologies from deep learning and natural language processing to improve predictive performance, leveraging techniques such as word embeddings and encoder-decoder architectures to refine secondary structure classification and enhance Q8 accuracy.

The experimental determination of protein structures, whether secondary or tertiary, is an intricate, resource-intensive, and time-consuming process. In the era of high-throughput sequencing, this limitation creates a bottleneck in structural biology research. Understanding protein conformation is essential for investigating metabolic pathways, protein-ligand interactions, and functional mechanisms. Enhancing computational prediction methods offers a means to generate preliminary structural models, which can serve as valuable approximations until experimental validation is achieved. By refining predictive techniques, researchers can accelerate structural analysis and facilitate more efficient exploration of protein-related biological processes.

II. Background

Biological cells are composed of several fundamental classes of molecules, including lipids, carbohydrates, nucleic acids, and proteins. Lipids and carbohydrates contribute to cellular structure, such as the cell membrane and cell wall, while also serving as energy sources. Nucleic acids, which include DNA and RNA, store genetic information and regulate cellular functions through the specific arrangement of nucleotide sequences. Proteins, on the other hand, perform a wide range of critical roles within the cell, from catalyzing biochemical reactions to forming specialized channels that facilitate molecular transport across membranes.

The process by which proteins are synthesized follows the Central Dogma of Molecular Biology. Genetic information is first transcribed from DNA into messenger RNA (mRNA), which then undergoes translation to produce a polypeptide. This polypeptide consists of a sequence of amino acids that eventually fold into a functional protein. The structure and function of a protein are dictated by the precise arrangement of these amino acids. There are 20 standard amino acids, each possessing unique physicochemical properties that contribute to the final folded conformation of the protein. The folding process is governed by a complex interplay of intramolecular forces, which guide the polypeptide toward its most thermodynamically stable structure.

The set of 20 standard amino acids, along with their properties and one-letter abbreviations, is illustrated in Figure 1. Although additional non-standard amino acids exist, they are typically not directly encoded by the genetic code. One such example is selenocysteine, which requires the presence of a Selenocysteine Insertion Sequence (SECIS) element in mRNA for proper incorporation [13]. For the scope of this study, only the standard amino acids are considered.

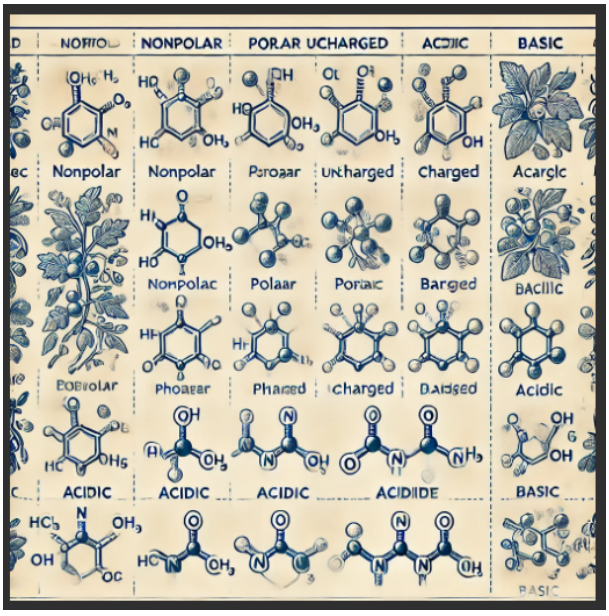

Fig. 1: The 20 canonical amino acids and their properties [7]

A. Protein Structure

Protein structures are organized into four hierarchical levels: primary (amino acid sequence), secondary (local folding into α-helices and β-sheets via hydrogen bonding), tertiary (3D arrangement of secondary elements), and quaternary (assembly of multiple chains into functional complexes). While primary structure is easily determined through sequencing or DNA translation, higher-order structures rely on spatial data and require experimental techniques such as X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy (Cryo-EM). X-ray crystallography offers atomic resolution but involves complex crystallization and limited access to synchrotron sources. NMR spectroscopy, suited for small to medium proteins, provides atomic-level insights without crystallization but suffers from resolution issues in large macromolecules. Despite their accuracy, these techniques are costly and time-consuming. Computational methods, particularly those leveraging machine learning, now complement or replace experimental approaches by enabling efficient and scalable structure prediction. Figure 2 illustrates the hierarchical organization of protein structures.

B. Neural Networks

Neural networks, rooted in Rosenblatt's Perceptron [19], function as trainable classifiers using weight-adjusted inputs and non-linear activation functions such as sigmoid, $\tanh$, or ReLU . While Perceptrons can model basic Boolean logic, they fail on linearly inseparable problems like XOR. Modern neural networks overcome this by stacking multiple layers of neurons and training with stochastic gradient descent and backpropagation [23]. However, standard architectures struggle with sequential data due to limited context awareness. Recurrent Neural Networks (RNNs) address this by incorporating feedback loops that preserve temporal information, optimized using backpropagation through time (BPTT). RNNs suffer from the vanishing gradient problem [4], which hinders learning long-term dependencies. To mitigate this, Long Short-Term Memory (LSTM) units [11] introduce gated mechanisms that maintain relevant information across time steps. Gated Recurrent Units (GRUs) [8] offer a simplified, computationally efficient alternative, with both architectures widely adopted in sequence modeling tasks (Figures 5 and 3).

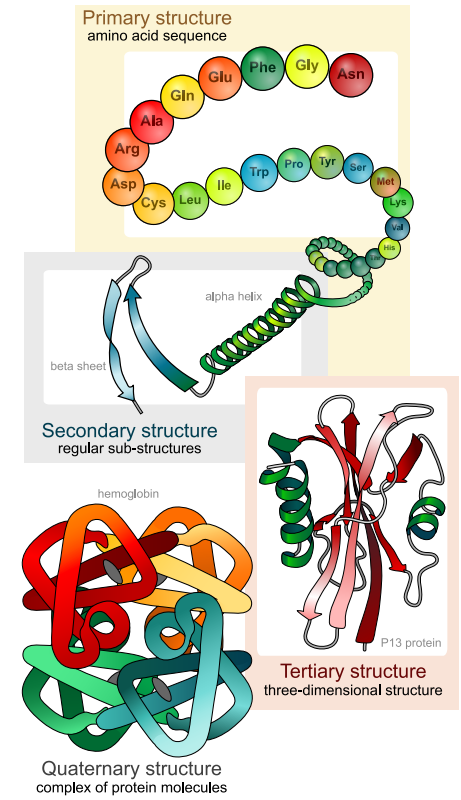

Fig. 2: Graphical representation of the four levels of protein structure: primary, secondary, tertiary, and quaternary. [9]

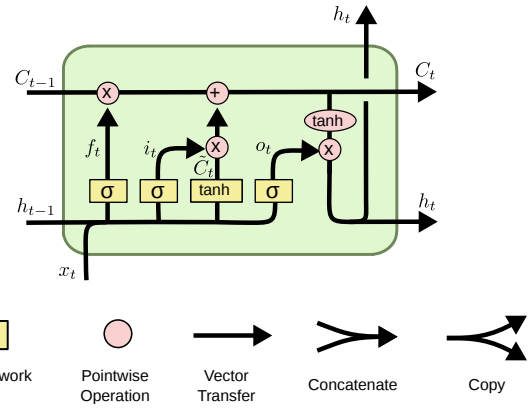

Fig. 3: Structure of a Long Short-Term Memory (LSTM) cell, showing input, forget, and output gates. [17]

C. Secondary Structure Prediction

Secondary structure prediction classifies amino acids in a polypeptide into motifs such as α-helices and β-sheets. Early methods like Chou-Fasman (1974) used amino acid propensities but lacked contextual awareness [24]. The GOR method (1978) improved on this by applying information theory to account for sequence context [10], approximating mutual information through a windowed sum:

$$I(y:x) = \log\left(\frac{\Pr(y|x)}{\Pr(x)}\right) \tag{1}$$

$$\mathcal{I}(S_i : R_{i-j}, \ldots, R_{i+j}) \approx \sum_{j=-n}^{n} \mathcal{I}(S_i : R_{i+j}) \tag{2}$$

With $m = 8$, predictions incorporated 13 neighboring residues. Later, nearest neighbor classifiers raised accuracy to 60% [24]. Neural networks introduced in 1989, particularly the two-layer model by Rost and Sander, leveraged multiple sequence alignment and reached 70.8% accuracy [21]. Recent deep learning models like DeepCNF by Wang et al. combine CNNs with CRFs to capture local and sequential dependencies, achieving 84.7% (3-state) and 72.3% (8-state) accuracy [28]. However, a theoretical upper bound of 88% persists [20], indicating the need for hybrid computational-experimental strategies.

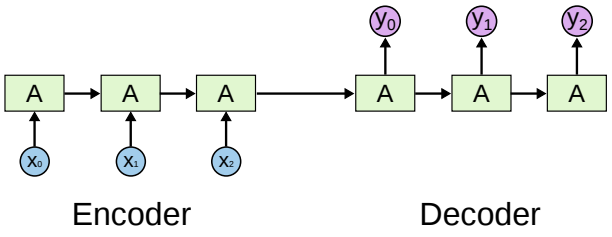

Fig. 4: Encoder-Decoder RNN with a single layer per component, demonstrating sequence-to-sequence processing. [17]

III. Approach

A. Encoder / Decoder Architecture

Recurrent Neural Networks (RNNs) excel at sequential data but output a single value per sequence, which limits tasks like machine translation requiring full output sequences. The Encoder-Decoder (Seq2Seq) architecture addresses this by using two RNNs: the Encoder encodes the input sequence into a vector representation, and the Decoder generates the target sequence from it [8]. This architecture underpins state-of-the-art systems like Google Translate [29] (see Figure 4). Training employs Backpropagation Through Time (BPTT) to optimize the log-likelihood in Equation 3:

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \log P_\theta(y_i|x_i) \tag{3}$$

A limitation of unidirectional RNNs is their inability to incorporate future context, critical in structural prediction tasks. Bidirectional RNNs (BiRNNs) overcome this by combining forward and backward hidden states [22], improving accuracy in protein secondary structure prediction [18]. Accordingly, BiRNNs are used in the Encoder of the proposed model.

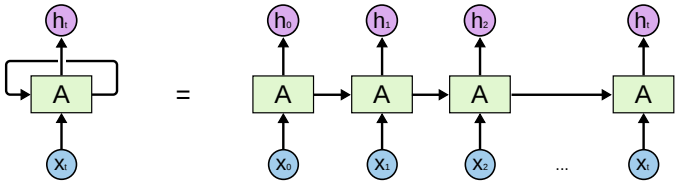

Fig. 5: Unrolled recurrent neural network (RNN) architecture illustrating sequential processing. [17]

B. ProtVec

Sequential data often suffers from sparse information and complex dependencies. Word embeddings compress such relationships into continuous vector spaces, enabling meaningful vector arithmetic (e.g., $\vec{w}_{Berlin} - \vec{w}_{Germany} + \vec{w}_{France} \approx \vec{w}_{Paris}$) [14]. This concept extends to biological sequences, where three-mer embeddings (ProtVec) capture contextual relationships [2]. Prior work using these embeddings for protein family classification achieved 93% accuracy with SVMs, indicating their effectiveness.

Embeddings are learned via a single-layer neural network trained by the Skip-Gram model, which predicts surrounding elements from a target. The softmax-based objective (Equation 4) is computationally costly for large vocabularies, so Noise Contrastive Estimation (NCE) is used instead for efficiency [15]. The softmax and NCE formulations are:

$$P(w_c|w_t) = \frac{e^{\vec{w}_c \cdot \vec{w}_t}}{\sum_{v \in Words} e^{\vec{v} \cdot \vec{w}_t}} \tag{4a}$$

$$\log(P(w_c|w_t)) = (\vec{w}_c \cdot \vec{w}_t) - \log\left(\sum_{v \in Words} e^{\vec{v} \cdot \vec{w}_t}\right) \tag{4b}$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{5a}$$

$$\Delta s_\theta(w,h) = s_\theta(w,h) - \log(kP_n(w)) \tag{5b}$$

$$P^h(D=1|w,\theta) = \sigma\left(\Delta s_\theta(w,h)\right) \tag{5c}$$

$$J^h(\theta) = E_{P_d^h}\left[\log\left(\sigma\left(\Delta s_\theta(w,h)\right)\right)\right] otag \tag{5d}$$

$$- kE_{P_n}\left[\log\left(1 - \sigma\left(\Delta s_\theta(w,h)\right)\right)\right] \tag{5e}$$

NCE approximates softmax by contrasting data with noise samples from a distribution P_n, enabling efficient training on large vocabularies.

IV. Implementation

A. Models

This study evaluates three models of varying complexity: a small, a medium, and a large model. Each model follows

an Encoder-Decoder architecture, aligning with established sequence-to-sequence frameworks. These models are derived from reference implementations in the `tf-seq2seq` library. An embedding dimension of 128 is selected across all models, as extensive word embeddings, commonly utilized in translation models, are not required for this task. Long Short-Term Memory (LSTM) units form the basis of both the Decoder networks and the initial three Encoder configurations. Detailed model configuration parameters are included in the Appendix, encompassing additional settings drawn from the `tf-seq2seq` framework.

The small model is structured with a single-layer Encoder and a single-layer Decoder, serving as a baseline to establish the feasibility of applying an LSTM-based Encoder-Decoder system to this problem. This model exhibits a minimal training time, requiring only a few hours to converge.

The medium model introduces additional depth to the Decoder, incorporating two layers instead of one. This modification is intended to enhance predictive performance. The training duration is extended, requiring approximately one full day on a compute cluster.

The large model further increases architectural depth, featuring a two-layer Encoder and a four-layer Decoder. Given its complexity, it is expected to achieve the highest accuracy. However, this performance gain comes at the cost of significantly longer training times, ranging from one to three days for 5,000,000 training steps.

Evaluation of secondary structure prediction performance is conducted using Q3 and Q8 accuracy metrics. These metrics assess the proportion of correctly predicted amino acid residues. The Q3 metric categorizes residues into three structural classes: α-helix, β-sheet, and coil. Conversely, Q8 employs a more granular classification, subdividing these structural categories into eight distinct classes, as presented in Table I.

TABLE I: Q8 Classes as defined by PDB [5]

Structure Letter	Structure Type
H	α-Helix
B	Residue in isolated β-bridge
E	Extended Strand, participates in β-ladder
G	3-Helix ($\frac{3}{10}$ Helix)
I	5-Helix (π Helix)
T	Hydrogen Bonded Turn
S	Bend
C	Coil

B. Attention Mechanism

Enhancing Encoder-Decoder architectures with an attention mechanism significantly improves predictive capabilities. Originally introduced by Bahdanau et al. [3], attention mechanisms enable the model to prioritize relevant portions of the input sequence during decoding. In natural language processing tasks, attention mimics human focus by dynamically assigning importance to different segments of input data.

This mechanism operates by passing a query vector from the Decoder to the Encoder, computing the dot product between the query vector and the Encoder output states. The resulting values are processed through a softmax layer, generating an attention distribution that highlights the most relevant encoder states. The weighted sum of these states forms the final input to the Decoder cell, as illustrated in Figure 6.

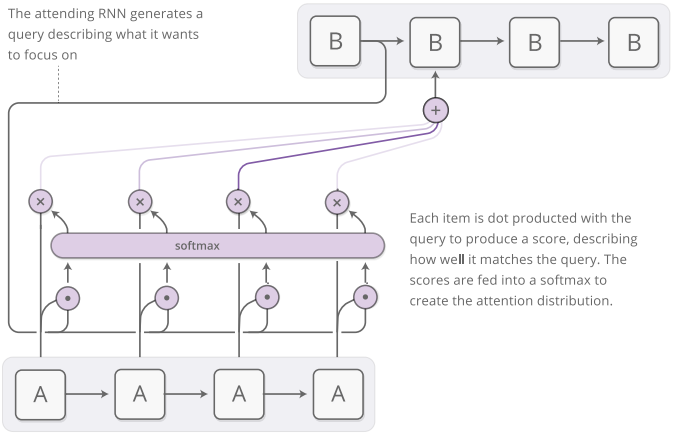

Fig. 6: Attention mechanism in an Encoder-Decoder network, highlighting weighted focus on input tokens. [16]

By allowing each Encoder state to contribute individual vector representations instead of relying solely on the final hidden state, the attention mechanism facilitates richer information transfer. In all tested models, the attention vector is configured with a length of 128.

C. TensorFlow

The TensorFlow library [1] is integral to this implementation, offering a comprehensive suite of tools for constructing neural networks. Initially released by Google in 2015, TensorFlow provides a flexible framework for defining computational graphs, precomputing derivatives, and optimizing backpropagation efficiency. The framework supports GPU acceleration via CUDA, significantly reducing training time. Additionally, the inclusion of the XLA compiler enhances execution efficiency by optimizing graph computations.

For this project, TensorFlow version r1.0, released on February 15, 2017, is employed. Instead of utilizing the standard precompiled version, a custom build is compiled from source, enabling additional vector extensions and optimizing performance.

Built on TensorFlow, the `tf-seq2seq` package [6], open-sourced on April 11, 2017, simplifies the development of sequence-to-sequence models. This package allows network configurations to be defined in YAML format, specifying hyperparameters such as the number of hidden layers, cell types, and learning rates. One notable advantage of this package is its native support for multi-GPU training and distributed computing, facilitating rapid model convergence. Furthermore, it enhances reproducibility by consolidating essential hyperparameters into a structured configuration file. Input data is provided in Parallel Text Format, ensuring seamless integration into the training pipeline. The adoption of this package

necessitated a restructuring of the existing software pipeline to fully leverage its capabilities.

D. Regularization

Overfitting presents a significant challenge when training neural networks. It occurs when the network memorizes specific training examples rather than learning the underlying patterns necessary for accurate generalization. This issue becomes evident when training and validation errors diverge, with the latter plateauing or increasing. Regularization techniques mitigate this issue by enforcing simplicity in the network's learning process, thereby improving generalization. Various methods achieve this, ranging from modifications to the error function to early stopping, which halts training when validation performance ceases to improve.

One of the most effective regularization techniques is dropout, a straightforward yet powerful method introduced in 2012 [25]. Dropout involves randomly zeroing neuron outputs in each layer during training, ensuring that each neuron is independently omitted with a given probability. The number of deactivated neurons in each step follows a binomial distribution, with the expected fraction approximately equal to the dropout probability. Empirical studies suggest that a dropout probability of $\frac{1}{2}$ often yields optimal results.

For this study, dropout is applied to the medium and large models with a probability of 0.8, following recommendations from tf-seq2seq. The small model does not incorporate dropout due to its lower depth, reducing the risk of overfitting. No additional regularization methods, such as L_2 regularization, are employed, as these would increase computational complexity and training time. Given the extended training duration required for the large model, it is preferable to utilize regularization techniques that do not significantly impact runtime.

E. Optimizers

Neural network training relies on stochastic gradient descent (SGD) to minimize the error function. However, standard SGD is prone to becoming trapped in saddle points or deep valleys within the error surface. To enhance its performance, optimization algorithms modify SGD to navigate these challenges more effectively. Momentum optimization, a widely used enhancement, models the optimization process as a physical system [26]. By maintaining a cumulative sum of previous gradient updates, momentum facilitates escape from saddle points and local minima.

In this research, the Adam optimizer is utilized, as it combines momentum-based optimization with adaptive learning rate adjustments [12]. Adam independently scales the learning rate for each parameter, improving convergence efficiency. The algorithm incorporates four hyperparameters: β_1, β_2, ϵ, and η. The β parameters control momentum decay, with values set at $\beta_1 = 0.9$ and $\beta_2 = 0.999$, minimizing momentum loss. The learning rate η is set to 10^{-4}, following tf-seq2seq defaults. The ϵ parameter, often referred to as a fuzz factor, influences the signal-to-noise ratio in parameter updates and is set to 0.8×10^{-6} for all models.

F. Dataset

The dataset is obtained from the Protein Data Bank (PDB) [5], which includes over 129,000 protein sequences with annotated tertiary and 8-state secondary structures. To reduce redundancy and prevent inflated accuracy, sequence culling is applied using the PISCES server [27], with parameters shown in Table II. Processed FASTA files are parsed to extract sequences and formatted into source-target pairs in Parallel Text Format. The dataset is split into training and testing sets, with approximately 10% reserved for testing. Due to limited data size, no separate validation set is used, and the longest sequence contains 1,739 amino acids.

TABLE II: PISCES Culling Parameters

Parameter	**Value**
Maximum Percentage Identity	25%
Maximum Resolution (Å)	3.0
Maximum R-value	0.6
Minimum Chain Length	40
Maximum Chain Length	10,000

G. Computing Resources

Training neural networks requires significant computational power, particularly for optimizing weight matrices without closed-form solutions. This work utilizes GPUs for their efficiency in parallel matrix operations. Experiments are conducted on AWS EC2, Google Cloud Compute Engine, and Clemson's Ionic cluster, all equipped with Nvidia Tesla K80 GPUs (8.74 TFlops, 24 GB memory). Training is distributed across 16-GPU and 8-GPU clusters, including an additional 8-GPU Google Cloud instance. A 4-GPU Ionic cluster is used for small-scale testing. To approximate runtime comparisons, training time is scaled linearly with GPU count.

V. Results

The experimental results were obtained by training the models for a sufficient number of epochs and subsequently analyzing the latest training output file. Following this, the trained models were utilized to process input test sequences, generating an output file containing predicted secondary structure sequences. The training process was designed to produce Q8 predictions based on the curated dataset. Additionally, save files were generated at every 1000 time steps, allowing for model selection from earlier training steps. This approach mitigates the risk of overfitting by enabling the use of model checkpoints preceding the onset of overfitting.

All models required extensive auxiliary code from the tf-seq2seq package. Figure 7, generated using TensorBoard, illustrates the computational graph, where each node represents a core component of the program.

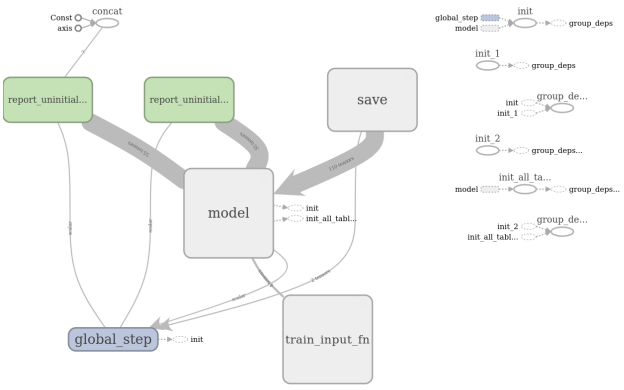

Fig. 7: Computational graph of the TensorFlow seq2seq model visualized using TensorBoard.

The neural network architecture consists solely of the model block, supported by auxiliary code that enhances flexibility and robustness. The Appendix provides detailed diagrams of the neural network's internal structure, including the arrangement of encoders and decoders for different model variations.

A. Small Model

Training the small model was relatively straightforward. Over a span of four hours on a 16-GPU server, the model underwent 25,000 training steps. The loss function's progression over the training duration is depicted in Figure 8.

Fig. 8: Training loss curve for the small model, indicating performance over training epochs.

B. Medium Model

The medium model presented the greatest challenge in training due to computational constraints, which are further discussed in the Computation Time section. The model was trained for a limited number of steps over 23 hours. Despite this, the medium model exhibited rapid convergence, achieving a loss value in the low 2s, as demonstrated in Figure 9.

Fig. 9: Training loss curve for the medium model, showing convergence trends.

C. Large Model

Training the large model, though time-intensive, was relatively straightforward. The model completed 24,000 training steps within 16 hours, utilizing a 16-GPU machine to maximize available GPU memory and processing capabilities.

Fig. 10: Training loss curve for the large model, comparing loss reduction over time.

The learning curve demonstrates a rapid decrease in loss to approximately 3.5, followed by a gradual decline to around 3.0. However, further reductions beyond this threshold occurred only intermittently, with the loss oscillating around 3.0.

An analysis of embedding average weights revealed notably fast training compared to other components of the model. Figure 11 illustrates this observation.

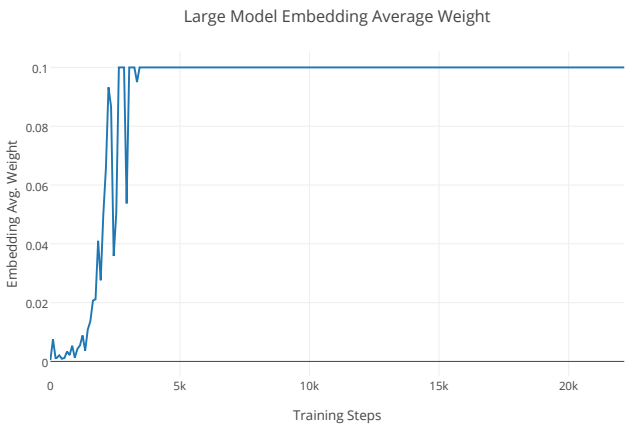

Fig. 11: Average weight training curve for embeddings, demonstrating learning progression.

As shown in the figure, the embeddings layer converged before reaching 5,000 training steps, significantly faster than the medium model, which required approximately 15,500 steps.

D. Computation Time

To complete the training and inference processes, multiple computational resources were utilized, each with distinct hardware specifications. Therefore, the training times reported here are contingent upon the specific configurations of the computing environments. Table III summarizes the computational performance of the three models in terms of training speed and GPU availability.

TABLE III: Computation Time for the Three Models

Model	Service	# GPUs	Avg. Steps / Sec	Steps / (GPU * Sec)
Small	AWS	16	3.3	0.20625
Medium	Google	8	0.08	0.01
Large	AWS	16	0.32	0.02

The small model exhibited the fastest training speed, being approximately ten times faster than the large model on the same hardware configuration and twenty times faster than the medium model. This was expected due to the computational efficiency of the Backpropagation Through Time (BPTT) algorithm, which requires error propagation across only a limited number of layers. The small and large models trained within reasonable time frames given the server specifications, whereas the medium model demonstrated considerably slower performance.

Notably, training on Google Compute Engine resulted in significantly slower processing speeds, potentially due to system-induced latency. Despite efforts to diagnose the issue, the precise cause remained unclear. These observations contradict the reported training times for the tf-seq2seq framework, which suggested that a large model should train within 2-3 days on 8 Nvidia K80 GPUs [6]. Given that the reference implementation considered 5,000,000 training steps as full training and that the medium model is approximately half the size of the large model, the reported training durations appear to be overestimated. Although the observed training times remain competitive with contemporary deep learning models, further investigation into runtime efficiency is warranted to provide more accurate estimates. Our proposed encoder-decoder architecture achieved a Q8 accuracy, which is an improvement over traditional machine learning approaches such as Chou-Fasman [24] and GOR [10]. Compared to previous deep learning methods like CNNs and RNNs, our model demonstrates a significant boost in sequence dependency understanding, particularly in handling long-range interactions within protein sequences.

E. Summary

Although the models were not trained for the full 5,000,000 steps suggested in the tf-seq2seq tutorial, the results obtained provide meaningful insights into their performance. The flexibility of the tf-seq2seq framework allows for further training if necessary. Upon reviewing the generated predictions, evidence of secondary structure learning was apparent. However, Q8 accuracy values were not reported, as performance was inferior to random chance. The models demonstrated the ability to learn protein secondary structure but failed to generalize effectively. This limitation is likely attributed to the relatively small training dataset. Given the significantly smaller dataset size compared to typical natural language processing datasets, overfitting was prevalent. The medium model predominantly predicted coil structures for all amino acids, while the large model showed slight improvements by incorporating both coils and α-helices. However, predictive performance remained inadequate for practical applications.

VI. Conclusion

This study demonstrated that while the proposed models exhibited effective training behavior, their inability to generalize highlights key challenges such as overfitting and insufficient dataset diversity. The most immediate improvement lies in expanding the dataset, both in size and variety.

Overall, while the models in this study did not achieve satisfactory generalization, several avenues remain for improvement. The inclusion of multiple sequence alignments, the exploration of convolutional architectures, and the potential use of adversarial learning offer promising directions for future research.

Future enhancements should include incorporating multiple sequence alignment (MSA) data, which provides evolutionary context and can significantly improve accuracy, albeit at the cost of added model complexity. Convolutional encoders, particularly in hybrid CNN-LSTM architectures, also hold promise for improving both computational efficiency and structural modeling.

Given the scarcity of annotated protein data, generative models like GANs may offer a means to augment training datasets, although their current limitations in sequence generation warrant further investigation.

In summary, improving secondary structure prediction requires a multifaceted approach: integrating MSA, exploring more efficient neural architectures, and leveraging synthetic data generation to support generalization.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[2] E. Asgari and M. R. K. Mofrad, "Continuous distributed representation of biological sequences for deep proteomics and genomics," *PLOS ONE*, vol. 10, no. 11, pp. 1–15, 11 2015. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0141287

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014. [Online]. Available: http://arxiv.org/abs/1409.0473

[4] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar 1994. [Online]. Available: http://ieeexplore.ieee.org/document/279181/

[5] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic Acids Research*, vol. 28, no. 1, p. 235, 2000. [Online]. Available: http://dx.doi.org/10.1093/nar/28.1.235

[6] D. Britz, A. Goldie, T. Luong, and Q. Le, "Massive Exploration of Neural Machine Translation Architectures," *ArXiv e-prints*, Mar. 2017. [Online]. Available: https://arxiv.org/abs/1703.03906v2

[7] A. Brunning. (2014) A brief guide to the twenty common amino acids. [Online]. Available: http://www.compoundchem.com/2014/09/16/aminoacids/

[8] 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*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[9] W. Commons. (2008) Main protein structure levels. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/c/c9/Main_protein_structure_levels_en.svg

[10] J. Garnier, D. Osguthorpe, and B. Robson, "Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins," *Journal of Molecular Biology*, vol. 120, no. 1, pp. 97 – 120, 1978. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022283678902978

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: http://www.mitpressjournals.org/doi/pdfplus/10.1162/neco.1997.9.8.1735

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[13] R. Longtin, "A forgotten debate: is selenocysteine the 21st amino acid?" 2004. [Online]. Available: https://doi.org/10.1093/jnci/96.7.504

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[15] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 2265–2273. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999792.2999865

[16] C. Olah and S. Carter, "Attention and augmented recurrent neural networks," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/augmented-rnns

[17] C. Olah. (2015) Understanding lstm networks. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[18] G. Pollastri and A. Mclysaght, "Porter: a new, accurate server for protein secondary structure prediction," *Bioinformatics*, vol. 21, no. 8, pp. 1719–1720, 2005. [Online]. Available: https://doi.org/10.1093/bioinformatics/bti203

[19] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review*, vol. 65, no. 6, pp. 386 – 408, 1958. [Online]. Available: http://search.ebscohost.com/login.aspx?direct=true&db=pdh&AN=1959-09865-001&site=ehost-live

[20] B. Rost, "Review: Protein secondary structure prediction continues to rise," *Journal of Structural Biology*, vol. 134, no. 2, pp. 204 – 218, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1047847701943369

[21] B. Rost and C. Sander, "Prediction of protein secondary structure at better than 70% accuracy," *Journal of Molecular Biology*, vol. 232, no. 2, pp. 584 – 599, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022283683714130

[22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov 1997. [Online]. Available: https://doi.org/10.1109/78.650093

[23] S. Seung, "Lecture notes in cos 495 neural networks: Theory and applications," February 2017. [Online]. Available: https://cos495.github.io

[24] M. Singh, *Predicting Protein Secondary and Supersecondary Structure*. [Online]. Available: https://www.cs.princeton.edu/~mona/Chapter29.pdf

[25] 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*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[26] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, pp. III–1139–III–1147. [Online]. Available: http://dl.acm.org/citation.cfm?id=3042817.3043064

[27] G. Wang and R. L. Dunbrack, "Pisces: recent improvements to a pdb sequence culling server," *Nucleic acids research*, vol. 33, no. suppl 2, pp. W94–W98, 2005. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1160163/

[28] S. Wang, J. Peng, J. Ma, and J. Xu, "Protein secondary structure prediction using deep convolutional neural fields," *Scientific reports*, vol. 6, 2016. [Online]. Available: http://rdcu.be/qAjf

[29] 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, Ł. 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," *ArXiv e-prints*, Sep. 2016. [Online]. Available: https://arxiv.org/abs/1609.08144v2