8-state Protein Secondary Structure Prediction

William Hogan
Dept. of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
whogan@ucsd.edu

Luke Rohrer
Dept. of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
lrohrer@ucsd.com

Vincent Nguyen
Dept. of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
vvn012@ucsd.edu

1 Introduction

For our final project, we will use a deep learning model for protein secondary structure prediction. A protein, at its core, is a sequence—a linear chain of amino acids connected by peptide bonds. There are four distinct levels of protein structure: primary structures, secondary structures, tertiary structures, and quaternary structures. The primary structure is simply the amino acid sequence ordered in the polypeptide chain. The secondary structure refers to repeating sub-structures on the actual polypeptide backbone chain. The tertiary structure is the 3D structure of monomeric and multimeric protein molecules. And, finally, the quaternary structure is the 3D structure that consists of the aggregation of two or more polypeptide chains that operate as a single unit [16]. We will focus on leveraging deep learning to accurately predict 8-state protein secondary structures. We will do this by recreating and proposing several extensions to the experiment described in *Prediction of 8-state protein secondary structures by a novel deep learning architecture* [18].

2 Motivation

To provide motivation for this problem we will address the following questions: (1) why proteins? (2) why protein structures?, and, lastly, (3) why predict protein structures?

2.1 Why Proteins?

Proteins are extremely important macromolecules that are used by our bodies in many ways. They provide a wide range of essential functions. For example, proteins can serve as catalysts, speeding up biochemical reactions within our body. The production of adenosine triphosphate (ATP), an organic compound that provides energy to living cells, is made possible by proteins. Proteins can also function as a transport. Hemoglobin is an example of a protein that carries and delivers oxygen throughout our bodies. Proteins also help protect our bodies from pathogens. Antibodies and antigens, essential to an immune system, consist of proteins. There are many other functions proteins provide, all of which make them an essential building block of life.

2.2 Why protein structures?

Protein structure determines function. By studying protein structure, we can start to understand how a protein works. Knowledge of the function and structure of one protein can often be generalized to other structurally similar proteins [19]. Single mutations within a protein can cause complex diseases.

Understanding the position of the mutation, the structure of the mutated protein, and how the mutation affects protein function can lead to a deeper understanding of diseases on a molecular level.

2.3 Why predict protein structure?

Currently, it's no easy task to analyze the structure of a protein. Protein structure determination has not enjoyed the same drop in costs compared to protein sequence determination. As of 2018, the cost to determine a protein sequence was $1,000, while the cost to determine a protein structure was $100,000 [17]. This disparity in cost has lead to a massive imbalance in available data. There are roughly 140 million sequences stored in the UniProtKB database compared to 150,000 protein structures stored in the Protein Data Bank [12]. Protein structure prediction arrives in this context—researchers have access to protein sequence information but lack access to protein structure information. Protein structure prediction fills in the gap and this is where deep learning comes in.

Deep learning can predict the structures of unknown proteins by drawing on the existing data of known protein sequences and their corresponding structures. With an accurate protein structure prediction model, researchers can eschew expensive tests and obtain valuable structural insights to deepen understanding of a particular protein. The goal of this project is to improve the accuracy of these predictive models.

3 Background

When it comes to predicting secondary protein structures, there are two main tasks. Q3, or the 3-state prediction problem, involves characterizing secondary protein structures as belonging to one of these three categories: helix (H), strand (E), and coil (C). In 1983, Kabsch and Sander [8] developed finer-grained categories with *Define Secondary Structure of Proteins* (DSSP), or *Q8*, a dictionary consisting of eight distinct protein structures. The Q8 categories are: 3_{10} helix (G), α-helix (H), π-helix (I), β-strand (E), bridge (B), turn (T), bend (S), and others (C) [18].

To encode each amino acid residue in the input sequence, we will start by representing it using the 48-dimensional vector described in [20]. We will then transform this feature vector into the 50-dimensional encoding mentioned in [18]. From there, we will explore various experiments involving different concatenations and augmentations of this baseline representation.

Here we will provide a brief overview and discuss the commonalities between some of the many recent published works involving secondary protein prediction. Successful model architectures tend to have designated components that address the following key techniques : modeling local dependencies between adjacent residues, modeling long-range dependencies between far-apart residues, and capturing relationships between structure features and labels.

Deep Convolutional Neural Fields (DeepCNF) aims to extend Conditional Neural Fields (CNF) by integrating Conditional Random Fields (CRF) with deep (rather than shallow) neural networks[14]. Here, local dependencies are modeled by CNFs [15] and feature-output relationships are handled via the depth of the network. While performance of DeepCNFs has been competitive with other state-of-the-art architectures, an important limitation here is the lack of dedicated structure to modeling long-term dependencies.

Next-Step Conditioned Deep Convolutional Neural Networks describe a novel chained architecture that combines a language model over the sequence of secondary structures with a convolutional model [5]. Here, next-step conditioning refers to the concatenation of the current sequence input with the previous structure labels, which is then used as input. This technique, alongside residual connections in the network, are effective in capturing long-term dependencies between residues. Additionally, one dimensional sliding filters in the convolutional layers help capture local context. Next-Step Conditioned Networks have been shown to perform well in this task and achieve very high precision scores.

Convolutional, Recurrent, and Residual Neural Networks (CRRNN) are the current state-of-the-art models and consist of the following components: a local block of 1D convolutional networks acting as a feature detector to model dependencies of adjacent residues, three stacked BGRU layers that utilize a bidirectional RNN with residual connections to capture context and long-range interactions

between residues, two fully connected layers, and a final softmax output layer [18]. By including a processing pipeline for longer-ranged dependencies, CRRNN was able to outperform DeepCNF.

Our proposed implementation will take into account the above findings and will be primarily based on the CRRNN architecture, the details of which will be discussed further in sections 5 and 7.

4 Goals & Metrics of Success

We have two specific goals for this project:

1. Recreate the results of *Prediction of 8-state Protein Secondary Structures by a Novel Deep Learning Architecture* [18].
2. Propose a new model that outperforms the model described in the aforementioned paper.

Table 1 shows current state-of-the-art performance in protein secondary structure prediction. We will consider our project successful if we can (a) reproduce these results and (b) improve upon them.

It is widely believed that there exists a theoretical limit to the accuracy of protein structure prediction. In practice, perfectly differentiated structures (e.g. helices, strands, etc.) do not exist. There is no such thing as a "perfect" helix that has clear start and stop boundaries—structures and boundaries blend between categories. It is estimated that secondary structure prediction cannot exceed 88% accuracy for Q3 classification since there is about 12% variation in secondary structure assignment [11]. Our success metrics will be considered in this context.

Table 1: SOTA Protein Secondary Structure Prediction.

Model	Training Data	Test Data	Q3 Accuracy	Q8 Accuracy
CRRNN [18]	TR12148	**CB513**	86.1%	**71.4%**
		CASP10	84.2%	73.8%
		CASP11	82.6%	71.6%
		CASP12	85.3%	68.7%
eCRRNN [18]	TR12148	**CB513**	87.8%	**74.0%**
		CASP10	85.9%	76.3%
		CASP11	83.7%	72.9%
		CASP12	87.3%	70.7%

For this project, we focus exclusively on CB513 as the testing dataset and Q8 as the prediction problem. Details will be provided in the following sections.

5 Ideas on How to Improve Architecture

We will use PyTorch to implement the original architecture as displayed in Figure 1. Once we confirm the model is working and our results match those obtained by the authors, we will attempt to improve the model by experimenting with different architectures.

To improve on the original paper's model, we will attempt the following experiments:

1. Replace the original model's GRUs with LSTMs. The authors originally chose GRUs to reduce the total number of parameters in the model. We believe replacing them with LSTMs will increase training time but result in better model performance.
2. Use contextualized embedding. The original paper uses a basic feed-forward neural network layer to embed one-hot encoded proteins into a dense 22-feature vector. We believe we can improve upon this method with contextualized embedding. We will use ProtVec[2]. ProtVec provides contextualized representations of proteins and amino acid sequences.
3. A recent paper proposes learning a parameter for residual connections, [4]. Since the original paper's model include residual connections, one of the key ideas we want to try is adapting ReZero's [4] method.

4. A stretch goal is to try WaveNet's recurrent CNN model on this problem. There appears to be a lot of overlap with audio wave generation and protein prediction, and it's possible a recurrent CNN model could preform very well in this domain.

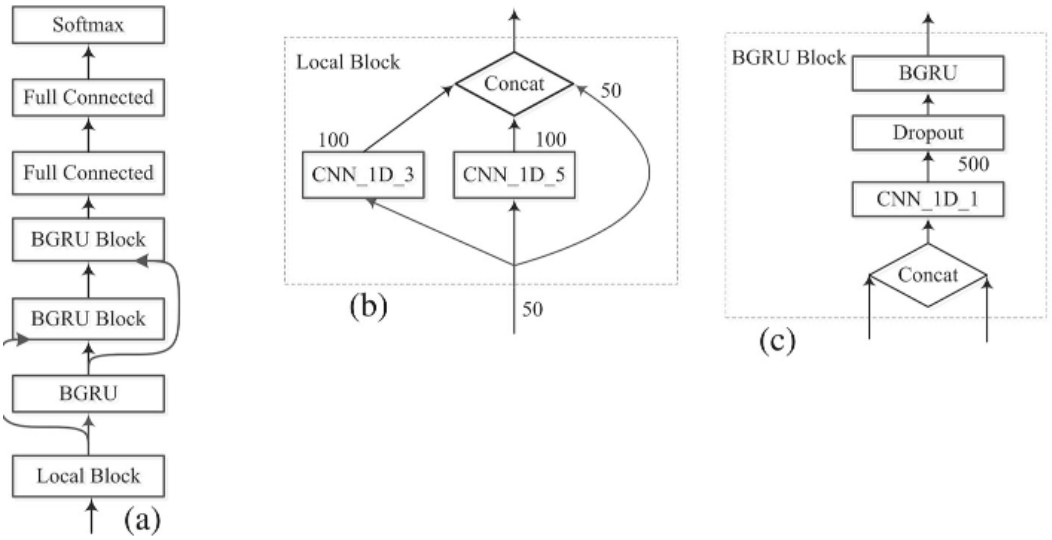


Figure 1: Original architecture proposed in [18]

6 Required Computational Resources

Protein structure prediction is a relatively low-resource problem. Protein sequences and their corresponding structures are represented by strings of characters. These strings do not require significant GPU resources to process. In [18], the authors train and test their model a single NVIDIA GeForce GTX 1080 Ti GPU with 12GB memory. We suspect that the 11GB GPUs available to us via UCSD's DSMLP cluster will be sufficient for our problem.

7 Dataset

We use two of the publicly available datasets used in *Prediction of 8-state Protein Secondary Structures by a Novel Deep Learning Architecture*, CB513 [3] and TR5534 [7]. CB513 consists of 513 protein sequences for evaluation. Each sequence consists of some number of amino acid residues that are then padded to 700 residues. For our training and validation split, we use TR5534 [7] which consists of 5926 sequences with similar format to CB513. Note that the name TR5334 is given by [18], but is sourced from [7]. We do not know the reason why there are 5926 sequences instead of 5534. All data can be downloaded from [20].

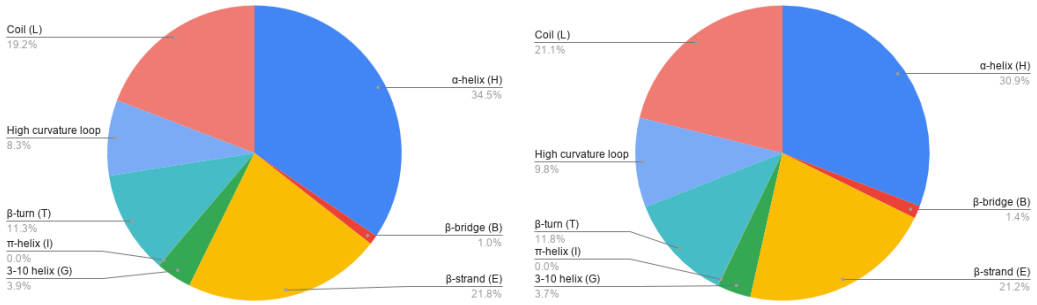
It is important to note that secondary protein structure data is inherently skewed. Some structures, like alpha-helices and beta-strands, naturally occur much more frequently than other structures, like pi-helices and beta-bridges. Figure 2 depicts the distribution of our train (TR5534) and test (CB513) datasets. This highly skewed data presents an added challenge to training our deep-learning models.

7.1 Input Features

Here, we will describe how the data is formatted for input into the model. Consider one protein as one sample. Then, each protein has 700 acids that make of the sequence and each acid has 51 features. Overall, one sample in the data has size $[700, 51]$.

7.1.1 Dense Embedding

Mentioned previously, there are 20 amino acids that build the protein. In the data, there are some proteins that have unknown labels, which are then labeled "X". Additionally, each protein is padded



(a) TR5534 dataset (b) CB513 dataset

Figure 2: Distribution of protein secondary structures.

to 700 acids, so another protein for padding is needed. Overall, there are 22 proteins. This is first represented as a 22 dimensional one hot encoding, but an embedding layer is applied to create a 22 dimensional dense embedding.

7.2 Positional Scoring Matrix

The positional scoring matrix, or PSSM, is calculated for an entire sequence at once. For example, say you have a 27 sequence protein. The PSSM would be of shape $[27, 21]$. Then, PSSM$[i, j]$ would mean "how likely is position i in the sequence to be acid j?" Note that in [18], the PSSM only has 20 columns while in the PSSM from [7] has 21 columns. We suspect that in [7], there is a column that represents unknown acids. This is one of the discrepancies between our implementation and [18].

7.3 Seven Physical Properties

The authors of [18] obtain seven physical properties of acid residues from [6]. The properties can be seen in Figure 3. For unknown acids, we just use all zeros.

7.4 One dimensional conservation score

Intuitively, this value just determines how important the acid is in maintaining the structure of the protein.

$$R = log(21) + \sum_{i=1}^{21} Q_i log Q_i \tag{1}$$

This value is calculated from the rows of the PSSM where Q_i is the value in the i^{th} column. Note that in [18], the value is calculated over 20 columns because their PSSM has only 20 columns.

8 Methods

Mentioned before, our goal is to improve on the original architecture, which is convolutional recurrent residual neural networks, or abbreviated as CRRNN. In this section, we will explain the base model and then describe the changes and intuitions behind our experimental models.

8.1 CRRNN

The baseline architecture, shown in 1, consists of a three main parts: local block, BGRU block, and fully connected layers.

Name	Ξ[a]	α[b]	υ_v [c]	π[d]	I[e]	α[f]	β[g]
ALA	1.28	0.05	1.00	0.31	6.11	0.42	0.23
GLY	0.00	0.00	0.00	0.00	6.07	0.13	0.15
VAL	3.67	0.14	3.00	1.22	6.02	0.27	0.49
LEU	2.59	0.19	4.00	1.70	6.04	0.39	0.31
ILE	4.19	0.19	4.00	1.80	6.04	0.30	0.45
PHE	2.94	0.29	5.89	1.79	5.67	0.30	0.38
TYR	2.94	0.30	6.47	0.96	5.66	0.25	0.41
TRP	3.21	0.41	8.08	2.25	5.94	0.32	0.42
THR	3.03	0.11	2.60	0.26	5.60	0.21	0.36
SER	1.31	0.06	1.60	-0.04	5.70	0.20	0.28
ARG	2.34	0.29	6.13	-1.01	10.74	0.36	0.25
LYS	1.89	0.22	4.77	-0.99	9.99	0.32	0.27
HIS	2.99	0.23	4.66	0.13	7.69	0.27	0.30
ASP	1.60	0.11	2.78	-0.77	2.95	0.25	0.20
GLU	1.56	0.15	3.78	-0.64	3.09	0.42	0.21
ASN	1.60	0.13	2.95	-0.60	6.52	0.21	0.22
GLN	1.56	0.18	3.95	-0.22	5.65	0.36	0.25
MET	2.35	0.22	4.43	1.23	5.71	0.38	0.32
PRO	2.67	0.00	2.72	0.72	6.80	0.13	0.34
CYS	1.77	0.13	2.43	1.54	6.35	0.17	0.41

[a] Steric parameter (graph shape index)
[b] Polarizability
[c] Volume (normalized van der Waals volume)
[d] Hydrophobicity
[e] Isoelectric point
[f] Helix probability
[g] Sheet probability

Figure 3: Seven physical properties in [6]

8.1.1 Local Block

The purpose of this is to gather local dependencies among the individual acids. To do this, the authors use 1D convolutions layers with kernels 3 and 5. Two output tensor both of size $[B, 700, 100]$ are created from the convolutions which are then concatenated with the original input to create an output tensor of size $[B, 700, 251]$ where B is the batch size. This tensor is then run through the BGRU blocks.

8.1.2 BGRU and BGRU Blocks

The first BGRU maps the 251 dimension into 250 hidden dimension. Since the BGRU is bidirectional, the forward and backward dimensions create two tensors of size $[B, 700, 250]$ and concatenated together create a tensor of size $[B, 700, 500]$.

$$F_1 = \text{Forward hidden states} \tag{2}$$
$$B_1 = \text{Backward hidden states} \tag{3}$$
$$O_1 = concat(F_1, B_1) \tag{4}$$

The output and input to this BGRU are then concatenated together to go into the first BGRU block. Inside the BGRU block, the dimensionality is reduced from 700 to 500, dropout is applied to help generalization, and then the tensor is sent into the BGRU. The 2nd BGRU maps the 500 dimension input into another 500 hidden representation, but unlike the first BGRU, the forward and backward annotations are instead summed together to create a tensor of size $[B, 700, 250]$.

$$O_{2,3} = F_{2,3} + B_{2,3} \tag{5}$$

The process is repeated one more time in a 2nd BGRU block.

8.1.3 Fully Connected Layers

Lastly, there are two fully connected layers. The first layer maps from 500 to 128 and then the second layer maps from 128 to 9. Eight labels for actual secondary structures and one for padding.

8.2 LSTM

Zhang et al. explain that they chose to build their model with BGRUs instead of LSTMs to reduce the number of parameters of their model due to concerns about using too much GPU memory [18]. LSTMs often outperform GRUs but, compared to GRUs, they require roughly 33% more parameters. Knowing this, we attempted a simple experiment by replacing all of the original model's GRUs with single-layer LSTMs as shown in Figure 4. This increased the total model's parameters from 7.74M to 10.3M. We also experimented with double-layer LSTMs but ran into out-of-memory issues.

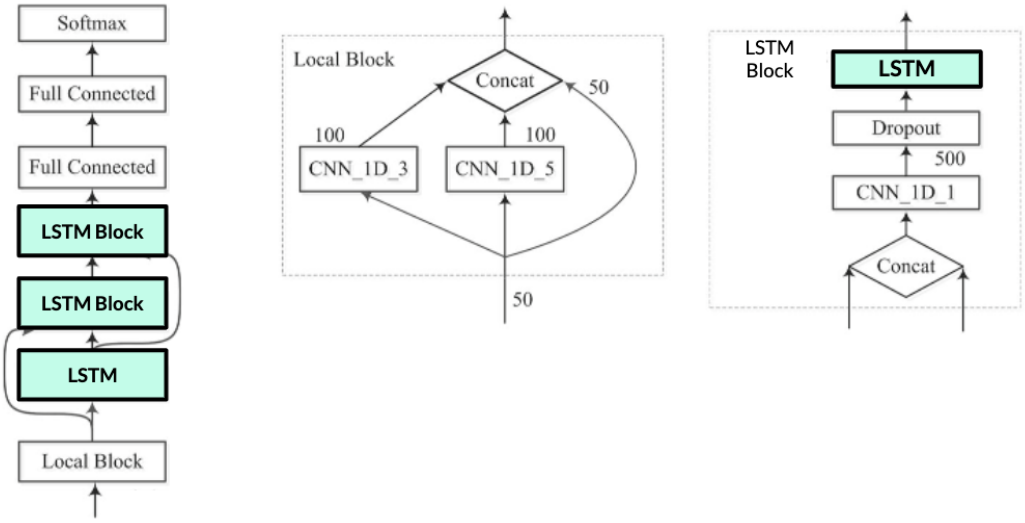


Figure 4: Our LSTM experiment: all the GRUs from the original CRRNN model were replaced with single layer LSTMs.

8.3 ReZero

We also experimented with a ReZero transformer architecture (Figure 5). ReZero is a simple change to the transformer from [13] that allows for faster convergence of deep transformers. Multi-head self-attention is the main cause of poor signal propagation in deep (12+ layer) transformers. ReZero proposes a learned parameter α that replaces layer normalization on each layer. This allows for better signal propagation which results in faster convergence. In our ReZero experiment, we implement a 12 layer transformer with 3 attention heads and a 2,048 dimension feed-forward network. We also experiment with a 6 layer transformer with 10 attention heads. Our design choices with the ReZero model are heavily shaped by the GPU resources available to us. Deeper transformers with more attention heads caused out-of-memory errors.

8.4 Residual Blocks

We take inspiration from the residual blocks in [1]. Audio files can be extremely long, which is why [1] uses dilated convolutions to increase receptive field, but since our sequence length, 700, isn't that long, we opt to use regular convolutions. For our experiment, we use 5 stacks of residual blocks. Taking inspiration from [4], we also test using regular skip connections and learned skip connections. A rough example of our structure can be seen in 6. The local blocks obtained from the 1D convolutions are both run through the residual stacks to try and incorporate more long term dependencies. The local blocks are then concatenated together similar to the base model and everything else remains the same.

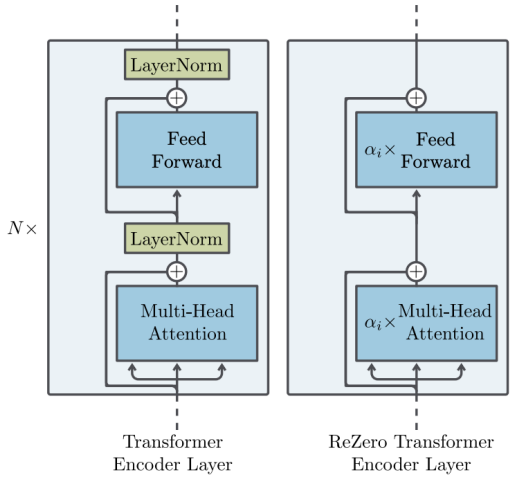


Figure 5: ReZero architecture for the transformer encoder layer.

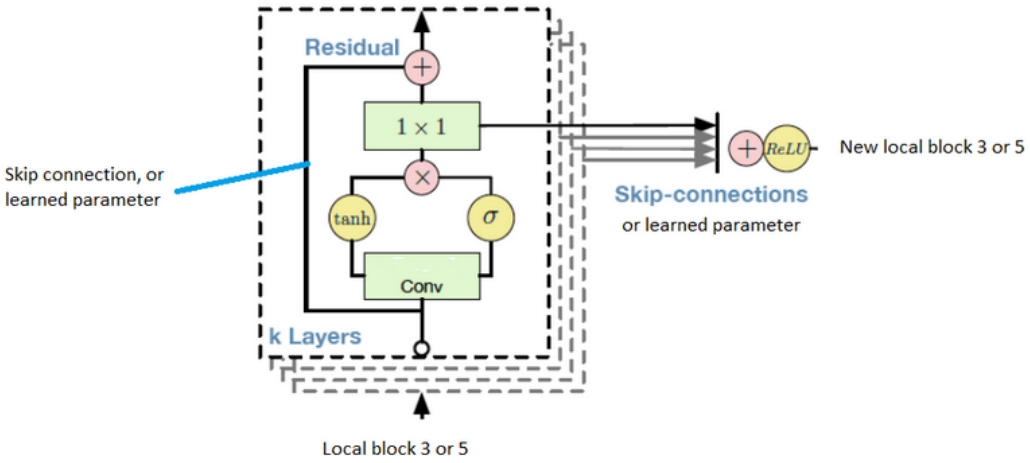


Figure 6: Our modification of the residual block. Image modified from [1]

8.5 ProtVec

Building off the success of Word2Vec [9] in natural language processing, ProtVec aims to create a "data-driven distributed representation for biological sequences" [2]. ProtVec hopes to better represent individual amino acids within a protein sequence by taking into account the local context in which they appear. To accomplish this task, the protein sequences are first segmented into 3-gram "biological words". Next, a two-layer skip-gram network is used that takes in a single amino acid as input and outputs a probability distribution over neighboring context acids. Throughout training, the hidden projection layer learns a 100-dimensional embedding for each 3-gram. At the end of training, each residue in the sequence is therefore represented by a 3-gram and it's associated embedding, $r_i = \{r_{i-1}, r_i, r_{i+1}\}$. One of our experiments here is to replace the dense embedding vector for each amino acid derived from its corresponding one-hot encoding with the ProtVec embedding vector.

8.6 Ensemble

Ensembling is a popular post-training technique in machine learning used to boost overall performance [10]. The main idea here is to utilize a collection of models that, when combined, perform better than the individual models on their own. Specifically, in our implementation, we computed a single probability distribution by taking a weighted average over the softmax outputs of multiple

independently trained models. The final prediction is therefore a simple argmax over that single distribution. As expected, we noticed a substantial performance jump when using this technique.

9 Experiment Design

9.1 Metrics

Accuracy is the sole metric we use for evaluation. Mentioned before, each sequence is is padded to 700, but we ignore the padding when calculating accuracy and performing optimization.

9.2 Train/Val/Test Split

We use 80% data from TR5534 for training and 20% data for validation. We use the entirety of CB513 for evaluation.

9.3 Training

We train for 50 or 100 epochs, but each experiment usually converged before then. [18] mentions using Adam with 0.0004 learning rate with decay, but they don't mention when to decay, so we just use Adam with 0.0004 learning rate.

10 Results

10.1 Training Curves

10.1.1 Baseline

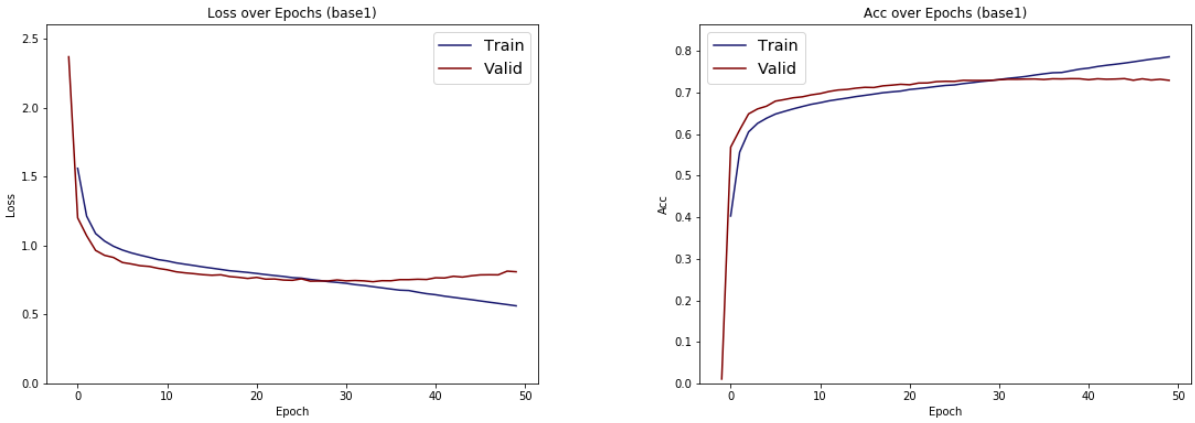


Figure 7: Accuracy and loss curves for the baseline model

10.2 Final Accuracy

In Table 2, we report the final testing accuracy on CB513 for each of our experiments. It is critical to note that the only training we used was derived from TR5534, which is around half the size of TR12148. TR12148 was the dataset used in [18]. We did not have access to the TR12148 dataset. The two top performing models (reporting 71.4% and 74% accuracy respectively), both of which come from our reference paper [18] and one of which being an ensemble model, were both trained using this larger dataset. The results that follow will only consider models trained on exclusively TR5534.

We conducted several experiments using the baseline architecture with only slight variations. As shown in the table as "Our CRRNN", five experiments were conducted where the architecture was

kept constant and the only alteration came in the form of different input features. (This is not entirely true, as the dimension of each feature vector plays a role on the dimension of early layers in our architecture, but this was not determined to be significant.) An additional experiment simply exchanged GRU with LSTM. The results here lay the groundwork for our baseline and allow us to better analyze and identify differences in final performance.

Our top performing single model (without ensembling) uses the same CRRNN architecture proposed in our reference paper, but with LSTM instead of GRU, and achieves 70.33% accuracy. This result outperforms the original CRRNN model, which reports 69.6% accuracy. We were also able to outperform the original CRRNN with our baseline architecture using unmodified input features, as well as just PSSM input features.

The other experiments we attempted generally performed poorly and underachieved our expectations for reasons we will discuss in section 11.

Table 2: Experimental Results on Q8 Protein Secondary Structure Prediction.

Model	**Training Data**[7]	**Test Data**[3]	**Model Variants**	**Q8 Accuracy**
CRRNN [18]	*TR12148*	CB513	None	74%$^{+}$ 71.4%
CRRNN [18]	TR5534	CB513	None	**69.6%**
Our CRRNN	TR5534	CB513	None	70.52%$^{+}$
			LSTM replaces GRU	**70.33%**
			None	70.31%
			PSSM only	69.91%
			ProtVec[2] replaces dense embedding	69.56%
			Dense embedding only	58.19%
			ProtVec[2] only	54.85%
Residual Blocks*[1]	TR5534	CB513	None	69.82%
Residual Blocks[1]				69.57%
ReZero[4]	TR5534	CB513	10 attention heads, 6 layers	47.08%
			3 attention heads, 12 layers	43.37%

*includes learned skip connections $^{+}$ensemble model

Table 3: Confusion matrix for LSTM experiment. Predictions are on the y-axis and ground-truth labels are on the x-axis.

	B	**E**	**G**	**H**	**I**	**L**	**S**	**T**
B	**28**	263	11	81	0	662	41	95
E	5	**15055**	54	251	0	2184	129	338
G	1	148	**973**	763	0	635	25	587
H	1	231	277	**24172**	0	821	41	614
I	0	2	0	24	**0**	1	0	3
L	9	2817	183	707	0	**12374**	506	1324
S	0	876	147	550	0	3603	**1414**	1726
T	0	470	382	1621	0	1742	205	**5593**

11 Discussion

Here we will evaluate our successes and failures, and discuss a few different directions for potential future work.

We were successfully able to recreate the results of *Prediction of 8-state Protein Secondary Structures by a Novel Deep Learning Architecture* [18]. We were also able to create a new model that outperforms the current state-of-the-art. The caveat here, however, is that our top performing model was only able to achieve higher accuracy on SOTA models trained on the smaller TR5534 dataset. When the larger TR12148 is used during training, the current SOTA is still several accuracy percentage points higher than our optimal result.

Despite our successes, we still uncovered numerous limitations and failures throughout our experimentation.

The LSTM experiment produced the best stand-alone performance of any of our experiments. It achieved an accuracy of 70.33% which was only exceeded by the ensemble model (70.52%). In Table 3, we examine a confusion matrix from the LSTM experiment. We notice that the model gains most of its accuracy from predicting the protein secondary structures that are best represented in the data (see Figure 2 for the distribution of secondary structures in the CB513 test set). The LSTM is most accurate at predicting alpha-helices (H) and beta-strands (E)—the two most common secondary structures represented in the data. Conversely, the model does poorly on the two least common secondary structures—pi-helicies (I) and beta-bridges (B). This highlights the need for more balanced data or a different approach to the problem, potentially one that leverages few-shot learning techniques.

Unfortunately, ProtVec underachieved our expectations. We suspect that this is due to the fact that there exist unknown amino acids in our dataset labeled with an 'X'. This means that there is not a clear 3-gram embedding, so our input vector for any 3-gram containing an 'X' will be a vector of all 0s. This issue is further compounded by the fact that the 'X' would show up in three consecutive 3-grams. In fact, if every third amino acid was unknown, we would not have any nonzero embeddable 3-grams. A potential workaround to circumvent this issue would be to average over several embedding vectors containing the subset of the 3-gram not including the 'X' or train our own ProtVec embeddings in a similar manner.

The ReZero architecture also performed poorly and we think this is due to a combination of long sequences, limited training data, and the use of only a few transformer attention heads. We were limited by memory due to the constraints of UCSD's DSMLP cluster platform. We noticed that the ReZero model performed better with more attention heads but we could not add more than 10 attention heads without running into out-of-memory errors. We were also limited by a relatively small training dataset. We suspect that gaining access to larger amounts of data would provide a significant and nontrivial performance boost to several of our models.

Additionally, the experiments involving residual blocks underachieved and may have caused the resulting models to overfit. Increasing the receptive field to be larger may be a disadvantage, because the purpose of the local block was to capture local information before the BGRU. This issue may benefit from moving the residual blocks after the local block along the processing chain to offset the larger receptive field.

We feel as though the changes we propose in this paper provide a strong baseline capable of challenging the current state of the art. Looking to the future, extensions to this work should focus on procuring a larger training dataset, getting access to a working memory with greater capacity, and experimenting more cleverly with various protein embedding strategies.

References

[1] Heiga Zen Karen Simonyan Oriol Vinyals Alex Graves Nal Kalchbrenner Andrew Senior Koray Kavukcuoglu Aaron van den Oord, Sander Dieleman. Wavenet: A generative model for raw audio. *arXiv*, September 2016.

[2] Ehsaneddin Asgari. Replication Data for: Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics, 2015.

[3] Zikrija Avdagic, Elvir Purisevic, Samir Omanovic, and Zlatan Coralic. Artificial intelligence in prediction of secondary protein structure using cb513 database. *Summit on translational bioinformatics*, 2009:1–5, Mar 2009. 21347158[pmid].

[4] Thomas Bachlechne, Bodhisattwa Majumde, Huanru Mao, Garrison Cottrell, and Julian McAuley. Rezero is all you need:fast convergence at large depth. *arXiv*, Mar 2020.

[5] Akosua Busia and Navdeep Jaitly. Next-step conditioned deep convolutional neural networks improve protein secondary structure prediction. *arXiv*, Feb 2017.

[6] Anita ZeidlerFelix Schmäschke Jens Meiler, Michael Müller. Generation and evaluation of dimension-reduced amino acid parameter representations by artificial neural networks. *Springer-Verlag*, January 2001.

[7] Olga G. Troyanskaya Jian Zhou. Deep supervised and convolutional generative stochastic network for protein secondary structure protein. *arXiv*, Mar 2014.

[8] Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[10] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *J. Artif. Int. Res.*, 11(1):169–198, 1999.

[11] Burkhard Rost, Chris Sander, and Reinhard Schneider. Redefining the goals of protein secondary structure prediction. *Journal of Molecular Biology*, 235(1):13 – 26, 1994.

[12] Irena Roterman-Konieczna Tomasz Smolarczyk and Katarzyna Stapor. Protein Secondary Structure Prediction: A Review of Progress and Directions. *Current Bioinformatics*, 15:90–107, 01 2020.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[14] Peng J.-Ma J. et al. Wang, S. Protein secondary structure prediction using deep convolutional neural fields. *Sci Rep*, 6, 2016.

[15] Peng J Xu J. Wang Z, Zhao F. Protein 8-class secondary structure prediction using conditional neural fields. *Proteomics*, 11:3786–3792, 2011.

[16] Wikipedia. Protein structure — wikipedia, 2020. [Online; accessed 26-April-2020].

[17] Wang J Heffernan R Hanson J Paliwal K Zhou Y Yang Y, Gao J. Sixty-five years of the long march in protein secondary structure prediction: the final stretch? *Brief Bioinformatics*, 19:482–494, 05 2018.

[18] Buzhong Zhang, Jinyan Li, and Qiang Lü. Prediction of 8-state protein secondary structures by a novel deep learning architecture. *BMC Bioinformatics*, 19(1):293, Aug 2018.

[19] Chengxin Zhang, Peter L. Freddolino, and Yang Zhang. COFACTOR: improved protein function prediction by combining structure, sequence and protein–protein interaction information. *Nucleic Acids Research*, 45(W1):W291–W299, 05 2017.

[20] Jian Zhou. Cb513 dataset.