Examining Gated Recurrent Network Architectures in Protein Analysis

Michael Sommer, Donal Naylor, Steven Stalder & Lukas Klein
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Determining the three-dimensional secondary structures of proteins from their amino acid sequences is a central task in computational biology research. Recent advancements in the field of machine learning and especially in natural language processing have led to promising new results in this area. In particular, long short-term memory (LSTM) architectures have been shown to compete very well when compared to Transformers and residual neural networks. We have investigated another type of architecture closely related to LSTMs – the gated recurrent unit (GRU). With our best GRU-based model, we were able to match or even slightly outperform an LSTM-based network in the secondary structure prediction task while converging considerably faster. These results indicate that GRUs might be a viable or even superior alternative to LSTMs in further work.**

I. Introduction

As the fundamental basis of life, proteins are one of the key elements in biology to understand evolution and diseases [1]. Proteins are chains of amino acids folded into a specific structure, determining their function. Obtaining meaningful labels about their 3D structure based on only the sequence requires large amounts of monetary and scientific resources. Thus, deep learning technologies would accelerate biological research extensively by predicting aspects of the structure, or the whole protein shape itself. Just recently this task got worldwide attention when on the first of December 2020, DeepMind and Google AI presented their second version of the AlphaFold model [2] which can estimate the whole 3D structure of a protein with very high accuracy. Nevertheless, subtasks in the field of protein structure prediction remain important since for many research questions only some aspects of the structure are relevant and not the overall 3D appearance of the protein, which is a lot harder to predict. We specialized in the prediction task of the protein's secondary structure. After the discovery of the α- and β-helix structure of proteins by Pauling and Corey [3], determining those secondary structures has been an ongoing field of research [4]. They are the key elements for predicting more detailed structures and the discovery of disease-causing mutations.

Over the last years, the amount of protein databases has grown exponentially with no standardized evaluation method for papers and literature. The first attempt to systematically evaluate deep learning models on protein sequences was introduced by Rao et al. [5] in 2019 with the Tasks Assessing Protein Embeddings (TAPE) datasets, providing benchmarks for five biologically relevant prediction tasks. The secondary structure task is based on protein sequences with a maximum of 25% sequence identity from the Klausen et al. dataset [6] for training and validation, and the CB513 dataset [7] for testing. The target labels are the three possible structural parts an amino acid could belong to: "Helix", "Strand" and "Other". An unlabeled pretraining corpus is available as well, but because of missing resources, we neglected the pretraining since results for comparison are also available for non-pretrained models and improvement with pretraining scales almost linearly for all architecture types.

Due to recent research in the natural language processing (NLP) area, sequence to sequence (seq2seq) tasks can be solved more and more accurately. Feedback neural networks (FNN) are particularly capable for seq2seq tasks and indeed, models based on bidirectional long short-term memory (BiLSTM) [8] networks are leading the ranking on the TAPE data, even ahead of Transformers [9]. However, there is scientific proof that neural networks based on gated recurrent units (GRU) as introduced by Cho et al. [10] achieve almost similar results in protein-related classification tasks with significantly reduced training time and required resources compared to LSTM-based architectures [11], [12]. We transferred these results to the secondary structure seq2seq prediction tasks. First, we are presenting our uni- and bidirectional GRU architectures which we compare to the baseline results of Rao et al. on the TAPE data as well as to our own LSTM networks in terms of performance and efficiency. Secondly, we introduce

a task-specific autoregressive architecture, inspired by the idea of protein-specific models by Yang et al. [13]. Excluding Transformers, this is – to our knowledge – the first time autoregressive models have been applied on the secondary structure prediction task.

II. Models and Methods

For the tokenization of the amino acids, we considered the same two vocabularies used by Rao et al.: IUPAC [14] and UniRep [15]. However, early experiments showed better results with IUPAC, which we continued to utilize for all later experiments. To have equal protein lengths within a batch, all sequences were padded based on the maximum sequence length in the respective batch. For all models and also the masked estimation in the autoregressive classifier we used the cross-entropy loss. Further, stochastic gradient descent (SGD) was applied to minimize the loss, which led to better results than the Adam optimizer. The learning rate was chosen depending on the model. For the training procedure, early stopping was used, with the stopping criterion being an increase in validation accuracy of fewer than 0.001 points after 10 consecutive epochs. To reduce the possibility of an interpolating regime we trained a large model for 150 epochs, which showed no convergence to an interpolation region after overfitting.

Inspired by the code implementation of Rao et al., we divided our model into two modules: encoder and classifier. An overview of the architectures can be observed in Figure 1. However, it is by no means a shared architecture: the encoder is only connected to one classification head. All of our models are available in our GitHub repository[1].

We use two baselines for comparison. First, we took the results from the Rao et al. paper where they achieve an accuracy of 0.71 on the task with their three-layer BiLSTM encoder and a hidden layer size of 1024 units. Secondly, we constructed our own LSTM encoder model to make better comparisons between variable architectures and to verify the results by Rao et al. since their information on the LSTM architecture appeared to be inconsistent in the paper, appendix, and code repository. In addition, we aimed for a version of their LSTM model deployed in PyTorch Lightning for our benchmarks, since the original code is in PyTorch version 1.1 and is neither working nor supported anymore [16].

[1]https://github.com/stevenstalder/GRU-Protein-Analysis

A. Encoders

To support the classifier in making an accurate sequence prediction, the encoder module extracts important information from the input embedding and provides it to the classification module. Every token of a sequence is embedded into a 128-elements long vector representation. Afterwards, the embedding is fed into the network (see Figure 1). For the encoder network, several GRU-based architectures were evaluated. The hidden layers are either one- or bidirectional, with a dropout layer on the outputs of each GRU layer. A single linear layer acts as the output to provide a fixed output size of the encoding to the classifier module since a bidirectional output layer would double the output size compared to a unidirectional one. Most of the time stacked GRUs were used with more than one hidden GRU layer to create more complex encodings. For the comparisons, we additionally built LSTM encoders, strongly based on the architecture of the GRU encoder modules.

B. Classifiers

1) CNN Classifier: On top of the encoder output, we have implemented two different classifiers. The first one relies heavily on the convolutional neural network (CNN) that has been used together with an LSTM encoder architecture by Rao et al. This classifier consists of two one-dimensional convolution layers with a ReLU activation and dropout layer between them (see Figure 1). To mitigate the vanishing gradient problem in deep architectures, a batch normalization [17] layer acts as the input layer, and for faster convergence by reparameterization, weight normalization [18] is applied on each convolution layer. The idea of the overall classifier is to filter the encoder output to produce three target class values for each amino acid in the sequence. One notable difference in our implementation compared to the one by Rao et al. is that we make use of dilated convolutions [19] in the first convolution layer of this classification head. By increasing the receptive field of that layer we hoped to capture more context information from the encoder output. Our experiments showed that indeed this change resulted in higher accuracy scores for the overall network.

2) Autoregressive Classifier: The second classifier aims at exploiting the fact that the task at hand is a sequence prediction. It is an autoregressive model that, when predicting the structure of a certain amino acid, takes into account predictions about previous ones. This could be done with a decoder model of a Transformer.

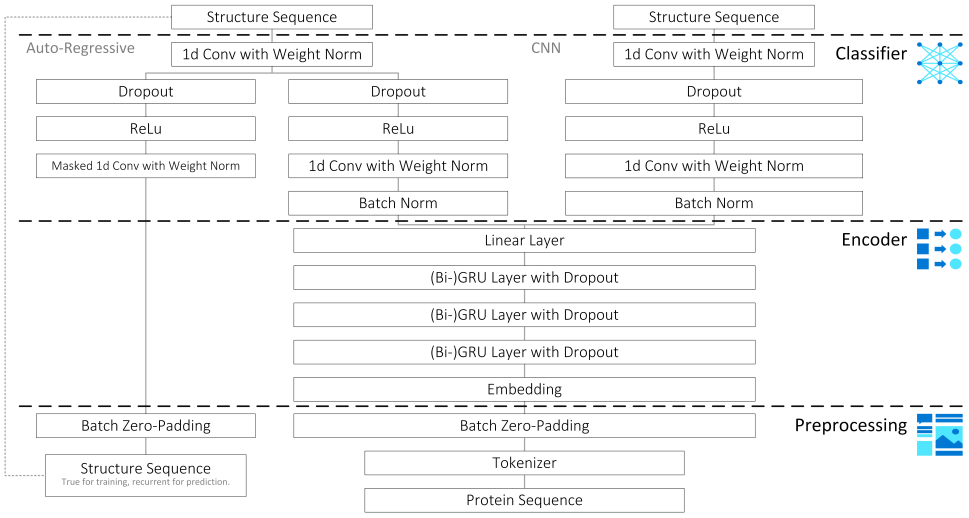

FIG. 1: *Architecture map of the seq2seq classification models. No shared encoder model, only either autoregressive or CNN classifier. Layer sizes depend on the specific model.*

However, Transformer-based approaches to the problem have already been tried before, and our encoder experiments showed potential in replacing them with alternative and simpler architectures. Thus, we decided to instead test an approach based on the convolutional PixelCNN [20]. Originally a generative model for images, we adapted it to use two inputs for predicting a sequence: the amino acid encoding, and the determined structure from the preceding part of the sequence. During training, a previously predicted structure is represented by the ground truth (target output) sequence, and masked convolutions are used to ensure that predictions are based only on past information. This way kernel weights can be learned for all sequence elements in parallel. During inference, the input structure sequence is seeded randomly, then run through the model, and the first part of the result is used as input again to predict the next part, continuing until the end. Optimally, this would mean that we would be running the model once for each amino acid, but because of runtime constraints, we had to settle for predicting structures in groups of 50 amino acids at a time (i.e. one run of the model provides the structure for amino acids 1-50, which is then used as an input to predict 51-100, etc.).

III. RESULTS

To verify the results from Rao et al. and to unify our baselines we have built our models on the basis of some of their architectural choices related to the number of layers and hidden layer sizes. Our own BiLSTM-based encoder module therefore also consists of three layers of size 1024 while the corresponding CNN classifier matches their hidden output channel size of 512. However, as we are using SGD rather than the Adam optimizer [21] with significantly increased learning rates (see Table 1), we have tuned other hyperparameters in the model. This BiLSTM model achieves a test accuracy of 0.7063, which reaches the rounded accuracy of 0.71 that Rao et al. report in their paper. For the comparison of unidirectional models, we created and optimized a unidirectional LSTM (UniLSTM) model inspired by the best performing unidirectional GRU (UniGRU) architectures. Similar to the bidirectional variant, three LSTM layers were stacked with a dropout rate of 0.2 and a hidden layer size of 1024. However, we enlarged the hidden output channel size to 2048 which quadruples the number of parameters of the CNN but also increases performance dramatically, leading to a test accuracy of 0.6781. For efficiency measurements, we look at the total amount of parameters as well as the number of epochs required for convergence. The BiLSTM model converged after 41 epochs and the UniLSTM one after 31. Furthermore, the BiLSTM consists of 69.3M parameters, and the UniLSTM consists of 44.6M. Both models were trained on an NVIDIA 1080 GPU provided by the ETH Zürich Leonhard Cluster. The BiLSTM model took 14.0h and the UniLSTM 11.9h to train on these GPUs.

For the GRU-based encoder models, several architectures and combinations were tried. As for the LSTM based models, the bidirectional variants clearly have the upper hand against unidirectional ones, even when doubling the hidden layer size of the unidirectional models to

	Encoder		Classifier		Full Model				
	Layer Size	Param.	Layer Size	Param.	Total Param.	LR	Epochs	Val. Acc.	Test Acc.
UniLSTM/CNN	1024	23.6M	2048	21.0M	44.6M	0.01	31	68.51%	67.81%
BiLSTM/CNN	2*1024	64.0M	512	5.3M	69.3M	0.1	41	71.99%	70.63%
UniGRU/CNN	1024	18.2M	2048	21.0M	39.2M	0.01	30	68.44%	67.62%
BiGRU/CNN	2*1024	49.1M	512	5.3M	54.3M	0.1	**27**	**72.28%**	**70.68%**

TABLE 1: *Results on SS3 supervised prediction task (Encoder/Classifier).*

2048. It shows how important the bidirectional view on the protein sequences is for encoding. Nevertheless, we wanted a comparison to the UniLSTM model. The best performing UniGRU model contains three GRU layers of size 1024 with a dropout rate of 0.2 and the same CNN classifier architecture as the UniLSTM but achieves a slightly lower test accuracy of 0.6762. It has the least amount of parameters (39.2M) and converges after 30 epochs. For the bidirectional GRU (BiGRU) model the architecture is very similar to the BiLSTM with three bidirectional layers of size 1024 and a dropout rate of 0.2, a learning rate of 0.1 as well as a hidden output channel size of 512 for the CNN classifier. The model converged after 27 epochs with a total parameter size of 54.3M and a test accuracy of 0.7068. The training time on the GPU was 14.1h for the unidirectional model and 10.2h for the BiGRU. The faster convergence of the BiGRU model shows that a more complex bidirectional model with higher capacity does not necessarily have to lead to longer training times which is the case for the LSTM-based networks.

For all models, we found a relatively small batch size of 4 to work better than larger values for which the loss would be computed with respect to tens of thousands of amino acid predictions at a time. Furthermore, for the CNN classifier module, dilated convolutions in the first layer improved the overall performance of all tested models.

As we see in Table 1, our GRU-based models are able to match the performance of the LSTM-based models, even though they require fewer parameters. Most notably, the BiGRU achieves the overall highest test accuracy while converging in only two thirds of the amount of epochs that the LSTM required (see Figure 2). Both the BiGRU as well as the BiLSTM are also able to match the highest accuracy that Rao et al. reported on the secondary structure prediction task without pretraining [5].

We found that our autoregressive model did not perform as well as expected, being far behind all the others in validation and test accuracy. There could be multiple reasons for this, the most likely being that the current architecture overfits to the ground truth sequence it receives as input during training and learns to assign too much weight to it, which is detrimental when, during inference, the sequence is seeded randomly. It is also possible that finer autoregressive steps (predicting only one amino acid at a time instead of 50) could yield increased performance.

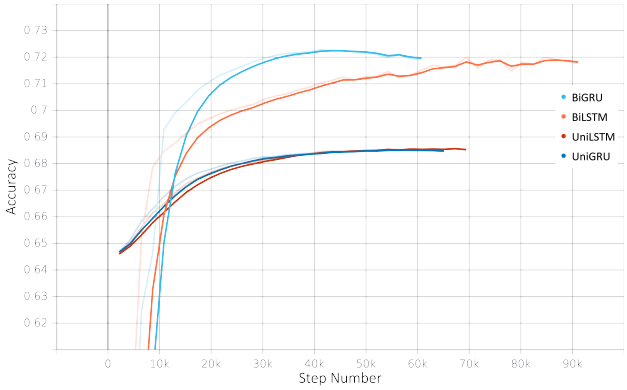

FIG. 2: *Validation accuracy graphs over training steps (smoothed) for the four models in Table 1.*

IV. DISCUSSION

Our results suggest that for the secondary structure prediction task, GRU-based networks are a valid or even superior alternative to LSTM-based models. Unfortunately, our more experimental autoregressive model requires further work to compete with the state-of-the-art models. Due to time constraints and for reasons of comparability, we have kept our model architectures relatively straightforward. However, we acknowledge that although our best models were able to match the highest scores by Rao et al., more complex recurrent network architectures could exceed their capabilities. Nonetheless, our results should provide valuable insight for future work in secondary structure prediction and other protein-related tasks. An important next step would certainly be to determine the effectiveness of GRU-based models when utilizing self-supervised pretraining before

the supervised training that we have covered. Rao et al. have reported that they pretrained their models for one week on four NVIDIA V100 GPUs [5]. For such resource-intensive procedures, the faster learning rate of the GRU could be an important factor. The same argument can also be made for really large networks with hundreds of millions of parameters that are very competent but also require a long time to train.

V. Summary

We have successfully implemented a GRU-based encoder architecture that together with a CNN-based classifier is able to match the best performance by Rao et al. for the secondary structure prediction task without pretraining and even slightly outperform our own LSTM architectures. In addition, we observed that our bidirectional model was converging a lot faster than the identical LSTM architecture. This confirms that previous research results from the protein sequence prediction domain also apply to the secondary structure prediction task.

The attempt to solve the task with an autoregressive classifier was not as successful as hoped. Further research has to be done here, which should not be underestimated since autoregressive models show a strong performance in sequence generating tasks. Otherwise one has to fall back to proven transformer decoder based architectures to benefit from autoregressive behavior. For FNN encoders, task augmentation seems to be the next reasonable step to further improve performance.

Overall, it can be said that through the proof of the high performance and efficiency of GRUs in the secondary structure seq2seq prediction task, protein analysis took another step forward in terms of democratizing complex technologies to a broader scientific community by lowering the computational resources to use them.

References

[1] C. Yanofsky, V. Horn, and D. Thorpe, "Protein structure relationships revealed by mutational analysis," *Science (New York, N.Y.)*, vol. 146, no. 3651, p. 1593—1594, December 1964. [Online]. Available: https://doi.org/10.1126/science.146.3651.1593

[2] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, K. Tunyasuvunakool, O. Ronneberger, and R. Bates, "High accuracy protein structure prediction using deep learning," *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction*, Nov 2020. [Online]. Available: https://predictioncenter.org/casp14/doc/CASP14_Abstracts.pdf

[3] L. Pauling and R. B. Corey, "A proposed structure for the nucleic acids," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 39, no. 2, pp. 84–97, 1953. [Online]. Available: http://www.jstor.org/stable/88806

[4] Y. Yang, J. Gao, J. Wang, R. Heffernan, J. Hanson, K. Paliwal, and Y. Zhou, "Sixty-five years of the long march in protein secondary structure prediction: the final stretch?" *Briefings in Bioinformatics*, vol. 19, no. 3, pp. 482–494, 12 2016. [Online]. Available: https://doi.org/10.1093/bib/bbw129

[5] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song, "Evaluating protein transfer learning with tape," *bioRxiv*, 2019. [Online]. Available: https://www.biorxiv.org/content/early/2019/06/20/676825

[6] M. S. Klausen, M. C. Jespersen, H. Nielsen, K. K. Jensen, V. I. Jurtz, C. K. Soenderby, M. O. A. Sommer, O. Winther, M. Nielsen, B. Petersen *et al.*, "Netsurfp-2.0: Improved prediction of protein structural features by integrated deep learning," *Proteins: Structure, Function, and Bioinformatics*, 2019.

[7] J. A. Cuff and G. J. Barton, "Evaluation and improvement of multiple sequence methods for protein secondary structure prediction," *Proteins: Structure, Function, and Bioinformatics*, vol. 34, no. 4, pp. 508–519, 1999. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-0134%2819990301%2934%3A4%3C508%3A%3AAID-PROT10%3E3.0.CO%3B2-4

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[10] 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

[11] A. Vazhayil, V. R, and S. KP, "Deepproteomics: Protein family classification using shallow and deep networks," 2018.

[12] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[13] J. Yang, I. Anishchenko, H. Park, Z. Peng, S. Ovchinnikov, and D. Baker, "Improved protein structure prediction using predicted interresidue orientations," *Proceedings of the National Academy of Sciences*, vol. 117, no. 3, pp. 1496–1503, 2020. [Online]. Available: https://www.pnas.org/content/117/3/1496

[14] I. J. C. on Biochemical Nomenclature (JCBN), "Nomenclature and symbolism for amino acids and peptides," *European Journal of Biochemistry*, vol. 138, no. 1, pp. 9–37, 1984. [Online]. Available: https://febs.onlinelibrary.wiley.com/doi/abs/10.1111/j.1432-1033.1984.tb07877.x

[15] E. Alley, G. Khimulya, S. Biswas, M. Alquraishi, and G. Church, "Unified rational protein engineering with sequence-based deep representation learning," *Nature Methods*, vol. 16, 12 2019.

[16] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song, "tape," https://github.com/songlab-cal/tape#tasks-assessing-protein-embeddings-tape, 2020.

[17] J. Bjorck, C. P. Gomes, and B. Selman, "Understanding batch normalization," *CoRR*, vol. abs/1806.02375, 2018. [Online]. Available: http://arxiv.org/abs/1806.02375

[18] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep

neural networks," *CoRR*, vol. abs/1602.07868, 2016. [Online]. Available: http://arxiv.org/abs/1602.07868

[19] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.07122

[20] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," *CoRR*, vol. abs/1606.05328, 2016. [Online]. Available: http://arxiv.org/abs/1606.05328

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980