BetaFold: A Lightweight Protein Prediction Architecture

Sheikh Shams Azam [1]

Abstract

In this work, we attempt to better understand the challenges of protein folding problem and the intricacies and details of corresponding solutions. We specifically start by studying the leading computational solution in the domain of protein folding, AlphaFold-2 (Jumper et al., 2021), and explore the building blocks of its architecture. We further evaluate the accessibility (bottlenecks and drawbacks) of AlphaFold-2 for general research population without access to terabytes of GPU memory for training and inference. We propose an alternative methodology called BetaFold, that draws inspiration from the design of AlphaFold-2 but tries to alleviate the training hardware bottlenecks by assuming a simpler structure that can be trained on GPU memory as small as 8GB. We further discuss the modeling framework and optimization strategies for learning the BetaFold model, discuss its similarity to the image-to-image translations task, and finally present the results of BetaFold on DeepCov (Jones & Kandathil, 2018) dataset which is a representative dataset of the larger CASP13 (CASP, 2013) dataset, one of the hardest protein folding dataset.

1. Introduction

Understanding protein structure is central to understanding diseases. Determining the 3 dimensional (3D) structure of protein is thus essential in drug discovery and design process (Nero et al., 2018). Traditionally, protein structure is determined through rigorous laboratory experimental procedures which are often extremely time consuming and subject to delays due to various procedural and logistic restrictions. Due to this limitation, the structure of only about 100,000 proteins have been determined so far (wwp, 2019), which represents a very small fraction of the billions of known proteins (Mitchell et al., 2020; Steinegger et al., 2019). Motivated by this set back of experimental protein structure discovery, there has been an increasing interest in developing computational solutions that can predict the 3D protein structure directly using the amino acid sequence. This general problem termed as "protein folding problem" (Dill et al., 2008) has been an important open research problem for over 50 years (Anfinsen, 1973). Dedicated to this effort, Critical Assessment of protein Structure Prediction (CASP) (Kryshtafovych et al., 2021) is an annual challenge seeking solutions towards the protein folding problem. In this paper, we start by studying the architecture of AlphaFold-2 (Jumper et al., 2021), one of the leading solutions proposed in CASP-2014 and the current state of the art in the domain of computational protein folding prediction. We highlight the fundamental contributions of AlphaFold-2 in terms of its model architecture and further dive into its computational bottleneck issues. Specifically, AlphaFold-2 is an extremely large machine learning (ML) model that is trained on 100-200 GPUs for several weeks and further fine-tuned. Even running the inference on AlphaFold-2 requires as much as 40-80GB of GPU memory (DeepMind, 2022). We thus sidestep the computational challenges of AlphaFold-2 and instead focus on its architectural contribution: *AlphaFold-2 depends highly on the prediction accuracy of inter-residue distances often represented using a distance histogram (distogram).*

In this work, we build a smaller network – BetaFold – that tackles the computational bottlenecks associated with AlphaFold-2 by utilizing computationally-tractable networks as well as feature engineering that models the stage 1 (prediction of distogram) of AlphaFold similar to an image translation pipeline. Moreover, the resulting BetaFold model can be trained on GPU with memory as little as 8GB.

The rest of this paper is structured as follows: In Section 2, we discuss the related works in the domain of protein folding problem including a deep dive into the architectural design of AlphaFold-2. Next, Section 3 presents the problem formulation that we concentrate on in this work. Following this Section 4 explains in detail, the modeling framework and optimization strategies for learning the BetaFold model. Section 5 presents the experimental results using BetaFold on the DeepCov (Jones & Kandathil, 2018) dataset. Finally, Section 6 & 7 discuss the conclusion and future work.

[1]School of ECE, Purdue University, West Lafayette, IN 47906. Code: https://github.com/shams-sam/BetaFold.

This is a project report for *ECE695BH, Purdue University* using template from the *Proceedings of the 39^{th} International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022.

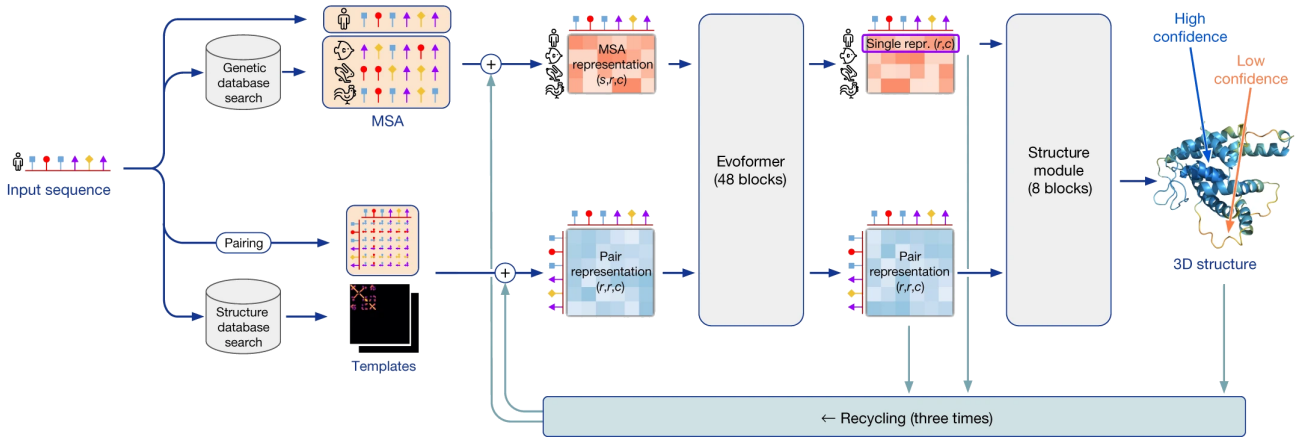

Figure 1. Structure of AlphaFold-2: The input sequence is first fed through the feature extractor to recover the MSA and pairwise residue representations. These features are then fed into Evoformer, whose output are then processed by the structure prediction network. The final structure is iteratively refined by using the feedback loop from structure prediction network to the Evoformer input as shown above.

2. Related Work

2.1. Traditional Solutions to Protein Folding

The traditional solutions in the domain of computational 3D protein structure prediction can be divided into two complementary flavors: one focuses on the physical interaction while the other on the evolutionary history. Among the two, the solution that rely on physical interaction are highly complex and thus do not scale to large protein structures. These solutions suffer from several drawbacks that include statistical approximations, molecular simulation, and inaccurate protein physics models (Brini et al., 2020; Sippl, 1990). On the other hand, the solutions following evolutionary history rely on data driven learning and borrows from the advances in bio-informatics analysis of protein evolution, homology of protein structure, evolutionary correlations. Despite these theoretically driven design principles, the physical interaction and evolutionary history models fail to give high accuracy protein structure prediction in a majority of cases where a close homologous has not been solved experimentally. This limits the real-world application of such solutions.

2.2. AlphaFold-2 (Jumper et al., 2021)

AlphaFold-2 (Jumper et al., 2021) is claimed to be the first computational approach capable of giving high quality (near experimental accuracy) protein folding prediction even in cases where similar homologous are not observed by the model during training. It was proposed in CASP-2014 challenge, as a successor to AlphaFold-1 (Senior et al., 2020), where it outperformed its nearest contemporary by a significantly large margin. While AlphaFold-1 utilized of residual network (He et al., 2016), AlphaFold-2 depends on attention-based transformers (Vaswani et al., 2017). Also, while AlphaFold-1 consisted of two modules trained independently: (i) one for predicting the inter-residue distances in the form of distance histogram or "distogram", and (ii) other for determining the 3D structure from distogram by constructing a potential for mean force (Kirkwood, 1935) that can accurately describe the shape of a protein, AlphaFold-2 is trained end-to-end. Specifically, AlphaFold-2 consists of two separate deep neural network (DNN) based models, each responsible for distorgram prediction and 3D protein structure discovery but trained jointly with a feedback cycle. Apart from this, AlphaFold-2 is also trained on a bigger dataset. The dataset is formed by aggregating several sources that includes the Protein Data Bank (PDB) (wwp, 2019) apart from CATH (Dawson et al., 2017), Uniclust30 (Mirdita et al., 2017), HHblits (Remmert et al., 2012), etc.

The success of AlphaFold-2 is attributed to the fact that the DNN is trained using procedures based on evolutionary, physical, and geometric constraints of the protein structure. The architecture: (i) jointly embeds multiple sequence alignments (MSAs) and pairwise features, (ii) proposes new output representation and associated loss for end-to-end structure predictions, (iii) uses a new equivariant attention architecture, (iv) uses an intermediate loss for iterative refinement of predictions, and (v) learns from unlabelled protein sequences using self-distillation.

AlphaFold-2, as shown in Fig 1, has two main components:

- *Evoformer* is the key building block of the AlphaFold-2 architecture which views the protein structure through the lens of graph inference in a 3D space, wherein graph

edges define residues in proximity and columns of MSA representation encode this information.

- *End-to-End Structure Prediction* module operates on the output of the Evoformer. This module uses geometry aware attention (termed invariant point attention (IPA)) module to learn rotation and translation invariance. It also uses "frame-aligned point error" (FAPE) to compare the structure under different alignments.

AlphaFold-2 also incorporates several other techniques such as noisy self-distillation (Xie et al., 2020) and iterative refinement to further enhance its accuracy. The self-distillation procedure helps utilize unlabelled datasets during training and considerably improves the performance. The Evoformer module uses Bidirectional Encoder Representation from Transformers (Vaswani et al., 2017) (BERT) (Devlin et al., 2019) to predict masked/corrupted MSA sequences and in turn learns subtle phylogenetic and covariation relationships without explicit hard-coding. Next, the Evoformer and structure prediction network undergo an iterative refinement such that the output of the structure prediction network is input through the Evoformer network several times ($3\times$ as per Jumper et al. (2021)) to enhance the structure prediction as shown in Fig. 1.

2.2.1. Computational Bottleneck

The main challenge to the accessibility of AlphaFold-2 to general research population is its computationally intensive architecture and training. It is an extremely large model that requires huge compute resources (100-200 GPUs) and several days (over a week) to train and further fine-tune.

A key step in AlphaFold-2 is the feature generation from an amino-acid sequence. While the entire feature generation framework of AlphaFold-2 is not made public, the work (Jumper et al., 2021) mainly focus on explaining the importance of multiple sequence alignments (MSA). The architecture depends on feeding the evolutionary and homologous information about a residue sequence in the amino-acid to the Evoformer network by embedding the multiple-sequence alignments (MSA) of the residue in the input features. However, this step is a major computational bottleneck since MSA features are fed in the form of covariance or precision matrices which are extremely large in size when considering protein sequences of length over a few hundreds. Jumper et al. (2021) themselves admit that the entire dataset before preprocessing is over 2TB and calculation of MSA and feeding it in the Evoformer is a preprocessing mammoth.

In summary, we understand that the bottleneck of AlphaFold-2 is two fold: (i) features generated from MSA and associated higher order cross-residue metrics are extremely large to process, and in turn (ii) the associated model to process these is even larger (over 21 million parameters). In order to overcome these bottlenecks, we propose BetaFold, where we consider simpler feature space by ignoring MSAs and thus have the flexibility of employing smaller networks to model the problem.

2.3. Image-to-Image Translation

There have been numerous works in the domain of image-to-image translation (Ronneberger et al., 2015; Liu et al., 2019) and it's a very well studied domain. Most solutions in the domain consider either a bottleneck network with transpose convolutions (Long et al., 2015) for up-sampling the reduce representation or a dilated convolution (Yu & Koltun, 2015) without a reduction is representation space. In this work we consider the dilated convolutions over the bottlenecks because the bottleneck architecture often lead to information bottleneck as well. In contrast dilated convolutions help improve the extraction of features a different resolutions. This is of particular importance to us in the case of distogram prediction since the long range distances are the most important in estimating the 3D structure of the protein (Jumper et al., 2021).

3. Problem Definition

We start by pointing out that protein processing involves operating on a feature-map generated from an amino-acid sequence of length L. The size of this feature-map is $\mathtt{L} \times \mathtt{L} \times \mathtt{C}$, which results from stacking $\mathtt{C}$ features together generated from inter-residue interaction of $\mathtt{L}$ elements in the amino-acid sequence. Similarly, the target distogram to be predicted is a $\mathtt{L} \times \mathtt{L}$ matrix (symmetric) where each element $\mathtt{l}_{\mathtt{ij}}$ is the distance between residue $\mathtt{i}$ and $\mathtt{j}$ in the amino-acid sequence. As a result, the problem of distogram prediction can be seen as an image-to-image translation (similar to semantic segmentation, depth estimation, etc.) problem wherein the input is an "image" of size $\mathtt{L} \times \mathtt{L} \times \mathtt{C}$ while the output is and "image" of size $\mathtt{L} \times \mathtt{L} \times \mathtt{1}$. By contrast, a conventional image-to-image translation takes an input image of size $\mathtt{H} \times \mathtt{W} \times \mathtt{3}$ if the image is RGB or $\mathtt{H} \times \mathtt{W} \times \mathtt{1}$ if image is gray-scale, where $\mathtt{H}$ and $\mathtt{W}$ are the height and width of the image respectively. We utilize this similarity in modeling BetaFold architecture.

Motivated by AlphaFold-2, the solution for protein folding problem can be broken down into three main stages: (i) feature extraction from the amino-acid sequence, (ii) prediction of distogram using the input features generated in step (i), and (iii) discovering the 3D protein structure using the distogram from step (ii). In this work, BetaFold focuses on step (i) and (ii) since they are more readily adaptable to low compute solutions. The inclusion of step (iii) in BetaFold training and corresponding end-to-end training is left to a future version of this work. Please note that solutions

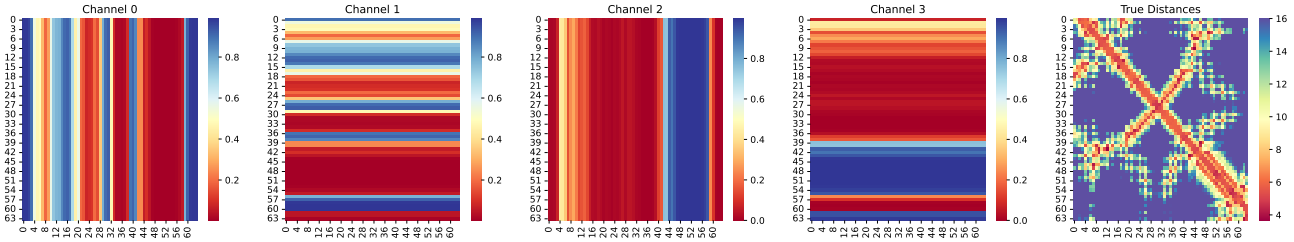

Figure 2. Input Features and Target for BetaFold: Left four subplots visualize the first 4 channels (features) computed from the protein sequence. In total we compute 57 features similar similar to (Jones & Kandathil, 2018). A visualization of all the 57 features can be seen in Fig. 6 of the Appendix A. The final subplot (rightmost) shows the target distogram for the corresponding protein sequence.

adapted from the domain of manifold learning (Elgammal & Lee, 2004) could readily lend themselves to solutions that generate the 3D structures from 2D distograms.

Protein Inter-residue Distance problem is to predict the 2D pairwise distance matrix (distogram) from the protein sequence. Because of the structure of distogram matrix it can be seen that it is symmetric matrix, however it does not help the overall distogram prediction task. While we drew comparison of distogram prediction to the image-to-image translation task, there are several difference among the two: (i) the number of channels in the protein prediction task can be incredibly high if including MSA features, (ii) visualization of input frames for protein problem would not be as intuitive as on image tasks, (iii) image augmentation techniques such as rotation, scaling, or flipping cannot be applied on protein input features, (iv) while local coherence is more important in images, 3D protein structure reconstruction depends highly of accuracy of elements away from diagonals (meaning we require a wider context window in convolution operator) which are found to be harder to predict. Due to this complexity of distogram, several works instead study the problem of contact prediction. Contact prediction turns the problem of true distance prediction (regression) to a bin prediction (i.e., classification task). In this work we compare the performance of BetaFold on both contact prediction as well as real-valued distance prediction.

4. BetaFold Model

4.1. Feature Extraction

The first step of modeling the BetaFold is the feature generation process. As mentioned earlier the AlphaFold-2 model does not list the exact features used in the feature extraction phase. One of the features readily discussed in AlphaFold-2 is the MSA features, which as discussed in Section 3 as computationally expensive to calculate and store for sequences of large sizes. The main reason for this is two folds: firstly, MSA features would first require downloading a large dataset of protein sequences. AlphaFold-2 repository (DeepMind, 2022) mentions the required size for saving these datasets to be over 2TB, secondly, the features themselves includes higher order alignment metrics such as co-variance and precision matrices which can often have channels as high as 400 individually. To put this into perspective, a network that can do dilated convolutions without bottleneck on a 57 channel wide input with a depth of 128 takes over 4GB of GPU space in or experiments if we keep the batch sizes below 4. As the batch size is increased the corresponding matrices to be processed become intractable. Consequently, we skip MSA representations since we do not have the computational capabilities of processing a 400 channel input. We also note that that reduced complexity of the BetaFold network as a result of this design choice makes it possible run both the training and inference on the network using an 8GB GPU. The overall training time is also reduced several folds. This would help the reproducibility of this work as well as followup works using this architecture.

Owing to these limitations, we turn towards Jones & Kandathil (2018) who propose a more tractable feature generation pipeline. Among the several proposed features, Jones & Kandathil (2018) evaluate the utility/redundancies of these features towards distogram prediction and finally identify seven features (Yang et al., 2020; AlQuraishi, 2019) that are found to be complementary and most informative. These include: (i) sequence profiles, (ii) secondary structure predictions, (iii) solvent accessibility predictions (McGuffin et al., 2000), (iv) co-evolutionary signal predictions (Seemayer et al., 2014), (v) FreeContact (Kaján et al., 2014) (vi) contact potential from MSA (also used in AlphaFold-1), and (vii) Shannon entropy from the alignment. Each of these features has a intuitive meaning, for example: the secondary structure represents the whether a residue occurs in a helix, strand, or coil. Similarly, solvent accessibility predicts if a residue is hydrophobic or not thus signifying if it faces inwards our outwards. Next, the co-evoulutionary signals capture the co-variance between all pairs of residue positions. Similarly, the contact potential and Shannon entropy

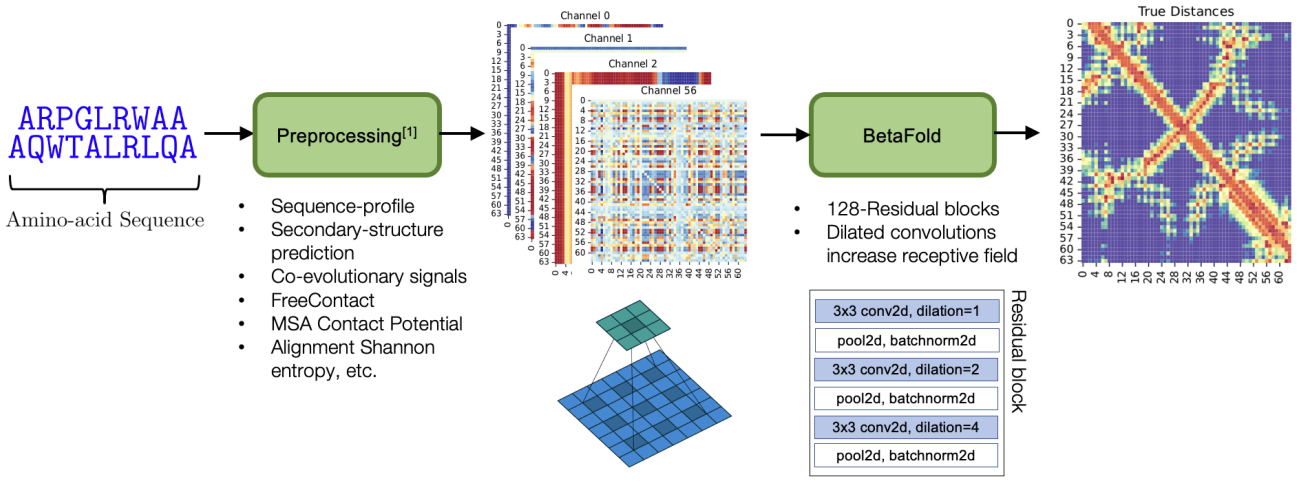

Figure 3. Architecture of BetaFold: During step 1 the protein sequence is converted into the "image" wherein each channel is the extracted feature such as sequence profile, MSA contact potential, etc. This "image" is then fed through and image-to-image translation pipeline with 128 residual blocks, each comprising dynamically increasing dilated convolution layers to increase the receptive field of subsequent convolution layers. The final output is an distogram (distance-histogram) "image".

can be calculated using MetaPSICOV package (Jones et al., 2015). In total, we end up with 57 features, all of which are stacked together to get an input of size $\mathtt{L} \times \mathtt{L} \times 57$ where $\mathtt{L}$ is the length of the protein sequence.

4.2. "Image"-to-"Image" Model

We next summarize our findings discussed in the previous sections that lead to our final model design. Firstly, we detailed in the previous section that the input consists of an "image" (feature-map) of size $\mathtt{L} \times \mathtt{L} \times 57$ while the expected output is a "image" (distogram) of size $\mathtt{L} \times \mathtt{L}$. Secondly, we also discussed in Section 3 the similarity of this input-output framework to an image-to-image translation problem. Thirdly, we discussed in Section 2.3 that image-to-image translations have the choice of bottleneck network with transpose convolutions or dilated convolution network. As discussed earlier, we choose the latter since, (i) bottleneck networks often lead to information loss which we wish to minimize, and (ii) dilation convolutions help encode features with both short and long spatial dependencies which we wish to encourage to capture the long range distance predictions in the target distogram. The effect of dilated convolutions can also be readily seen other works that require to capture long-term dependencies (Oord et al., 2016).

Using these design philosophies, BetaFold consists of a network with 128 residual blocks with dynamic dilation, i.e, each residual block consists of vanilla convolution with no dilation followed by a dilated convolution layer such that dilations are exponentially increased from 1 to 2 to 4 and then reset to 1 and repeated for every subsequent block. Each convolution block has 64 filters. Also, at the input block we encourage a channel mixing of the 57 feature channels using 64 maps of 1×1 convolution. Finally, at the output block we consider a channel compression to meet the dimensionality requirements of the target distogram. A detailed architecture of Betafold can be seen in Fig. 3.

4.3. Optimization and Training

In order to train the network we consider two different objectives: (i) contact prediction, and (ii) true-distance prediction (both discussed in Section 3. While (i) is trained using the binary cross entropy loss and optimized for accuracy, we consider two different losses in (ii), namely: (a) `log_cosh` (i.e., logarithm of hyperbolic cosine), and (b) `inv_log_cosh`. We start with `log_cosh` loss, given by:

$$\log\cosh(y, \widehat{y}) = \log(\cosh(\widehat{y} - y)), \tag{1}$$

where y is the true output and $\widehat{y}$ is the predicted output of the BetaFold network. $\mathtt{log\_cosh(x)}$ is approximately equal to $x^2/2$ (i.e. squared loss) for small x and $\mathtt{abs(x)} - \mathtt{log(2)}$ (i.e., absolute loss) for large x. Essentially, this means that `log_cosh` works mostly like the mean squared error, but will not be strongly affected by the occasional wildly incorrect prediction (tensorflow, 2022). We train using `log_cosh` and find that the training does not yield good results as shown in Fig. 7 & 8 of Appendix B. We find that the main for the bad performance is the failure of `log_cosh` function to optimize for shorter distances. As a result we instead use the `inv_log_cosh` loss proposed by

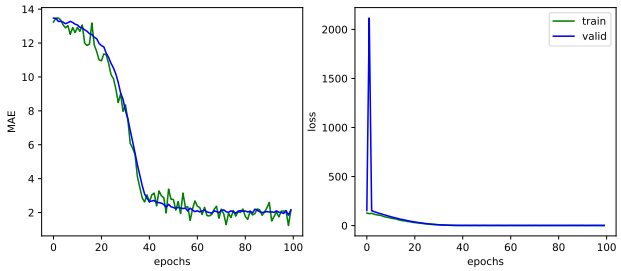

Figure 4. Training curves when BetaFold is optimized using `inv_log_cosh`: (left): shows the mean-absolute error (MAE) between actual and predicted true distances, and (right): shows the corresponding `inv_log_cosh` loss optimized by the network.

(Adhikari, 2020), given by:

$$\text{inv log cosh}(y, \widehat{y}) = \log\left[\cosh\left(\frac{K}{\widehat{y}+\varepsilon} - \frac{K}{y+\varepsilon}\right)\right], \qquad (2)$$

where y is the true output and $\widehat{y}$ is the predicted output of the BetaFold network, K is a scalar multiple to prevent numerical underflow, and ε is a small positive number for numerical stability (i.e., prevent division by zero). The resulting training leads to much better results as can be seen in the training curves shown in Fig. 4. While the training initially is slow (loss value explodes for validation curve), the mean absolute error plot shows that the model learns a good generalization as shown by the overlapping train and test (marked as valid in Fig. 4). However, we get the best performance (see Table. 1) when we invert the label y as K/y and use the regular `log_cosh` loss instead of using the `inv_log_cosh` loss. While such an inversion surprisingly shows good learning performance, there is not much understanding about why it works so efficiently. A similar effect is observed by Adhikari (2020).

4.4. Comment on Knowledge Distillation Goal

While the original proposal for this study was to learn a BetaFold model comparable with AlphaFold-2 in terms of performance using knowledge distillation, we were unable to embark on that journey because of the computation limitations associated with AlphaFold-2 discussed earlier in Section 2, 3, & 4. We would also like to point out that "protein folding problem" is a highly inter-disciplinary problem where domain knowledge of biology, bio-informatics, along with physics would prove vital in building a smaller model on-par with AlphaFold-2. We aim to defer this to a future extension of this work after gaining more domain expertise.

5. Experiments

We next dive into the experimental evaluation of our BetaFold model. We start by going over the DeepCov data used for training, followed by experiments evaluating the performance of different techniques in terms of MAE.

5.1. DeepCov (Jones & Kandathil, 2018) Dataset

The dataset consists of 3456 representative protein sequences further split into 70:30 train-test split. The dataset of 3456 protein sequences do not have a domain homologous in the valid dataset. The sequence length of DeepCov dataset ranges from 50 to 500 residues in training set, and ranges from 50 to 266 in test dataset, respectively. This dataset is also a representative of the Protein Data Bank (PDB) dataset. These 3456 datasets are processed as discussed in Section 4.1: (i) get the FASTA files for each protein sequence, (ii) download the corresponding protein chain from PDB, (iii) clean the chain to remove non-standard residues in the chain, and (iv) finally the pairwise distances are calculated using the true 3D structure of the protein. Adhikari (2020) have generously shared a cleaned version of this dataset also linked our GitHub implementation: `https://github.com/shams-sam/BetaFold`. The total size of the dataset is around 1.1GB.

5.2. Training Setup

We train each model to until its loss saturates. This is usually in the range of 20 to 100 epochs. The models are trained using a batch size of 4 and using Adam (Kingma & Ba, 2014) optimizer with a learning rate of $1e-2$ and $1e-3$ and the best model is chosen. The training is done on a system with 8GB GPU memory and 128GB RAM

5.3. Performance Evaluation

We next evaluate the performance of different models on the basis of chosen metrics Mean Absolute Error (MAE) and accuracy. Please note that the accuracy performance of contact prediction cannot be directly compared with the MAE performance on true distance prediction. We present the results and corresponding hyper-parameters in Table 1.

5.4. Distance Visualization

We further visualize the performance of model #6 from Table 1 by plotting the true and predicted distogram. Even though the predicted and actual distograms are inversions of each other due to the problem formulation as discussed in Section 4.3. The performance comparison in Table 1 and Fig. 5 confirm our hypothesis that the protein folding problem can indeed be tackled as an extension of image-to-image translation framework.

6. Conclusion

In this work, we started by introducing the intricacies of the protein folding problem. We then explored AlphaFold-2, one of the leading computational solutions in the domain. We explained the architectural contributions of AlphaFold-

#	Task	Hyper-parameters	Metric	Performance
1	True Distance Pred.	$\texttt{loss} = \texttt{inv\_log\_cosh}, \texttt{lr} = \texttt{1e} - 2$	MAE	0.27
2	True Distance Pred.	$\texttt{loss} = \texttt{inv\_log\_cosh}, \texttt{lr} = \texttt{1e} - 3$	MAE	0.14
3	True Distance Pred.	$\texttt{log\_cosh}, \texttt{lr} = \texttt{1e} - 2$	MAE	0.30
4	True Distance Pred.	$\texttt{log\_cosh}, \texttt{lr} = \texttt{1e} - 3$	MAE	0.31
5	True Distance Pred.	$\texttt{log\_cosh}, \texttt{lr} = \texttt{1e} - 2$ with inverted distances	MAE	0.08
6	True Distance Pred.	$\texttt{log\_cosh}, \texttt{lr} = \texttt{1e} - 3$ with inverted distances	MAE	0.08
7	Contact Pred.	$\texttt{log\_cosh}, \texttt{lr} = \texttt{1e} - 2$	Accuracy	0.93
8	Contact Pred.	$\texttt{log\_cosh}, \texttt{lr} = \texttt{1e} - 3$	Accuracy	0.92

Table 1. Performance comparison of various formulations of BetaFold with their corresponding hyper-parameters. We consider two tasks: True distance prediction and contact prediction as discussed in Section 3.

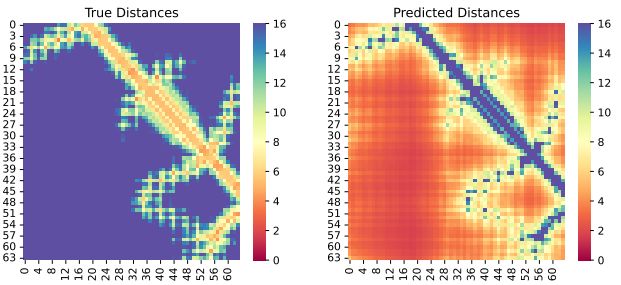

Figure 5. Visualization of true vs predicted distances in the form of a distogram. It can be seen that predicted distogram is a inverted map of the original one to a close extent. The original distogram can be easily obtained by inverting it.

2, followed by its computational bottlenecks. Using the motivation from the study of AlphaFold-2, we first proposed a tractable feature extraction framework such that input size is manageable for smaller memory devices. We next propose BetaFold by exploring the application of image-to-image translation in the distogram prediction problem. We discuss our design choices in terms of convolution operation of choice and the loss function used. We finally presented the results of BetaFold when applied on the DeepCov dataset which is representative of larger CASP13 and PDB dataset. We presented the results both in terms of quantitative metrics (in Table. 1) as well as qualitative assessment (in Fig. 5).

7. Future Work

Even though BetaFold is a step in the right direction, it is by no means ready to be compared to the larger models such as AlphaFold-1 or AlphaFold-2. As a first step, we next wish to explore the application of manifold learning in predicting the 3D structure of protein from the 2D distogram. There are several auxiliary datasets that generally prove useful in these 3D conversion specifically, we need MSA features. As discussed in Section 3 MSA features are indeed very large to store and use directly. As a orthogonal step, we could assess alternative methods such that once MSA is used to train a parametric model that brings the knowledge of MSA in a smaller tractable network. Concepts in the domain of pre-training using transformers or Neuro-symbolic AI could prove useful in this effort. We also wish to further extend BetaFold to be distilled from the larger AlphaFold-2 model as by overcoming the bottlenecks discussed in Section 4.4.

References

Protein data bank: the single global archive for 3d macromolecular structure data. *Nucleic acids research*, 47(D1): D520–D528, 2019.

Adhikari, B. A fully open-source framework for deep learning protein real-valued distances. *Scientific reports*, 10 (1):1–10, 2020.

AlQuraishi, M. End-to-end differentiable learning of protein structure. *Cell systems*, 8(4):292–301, 2019.

Anfinsen, C. B. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

Brini, E., Simmerling, C., and Dill, K. Protein storytelling through physics. *Science*, 370(6520):eaaz3041, 2020.

CASP. 13th community wide experiment on the critical assessment of techniques for protein structure prediction. https://predictioncenter.org/casp13/, 2013. Accessed: 2022-03-30.

Dawson, N. L., Lewis, T. E., Das, S., Lees, J. G., Lee, D., Ashford, P., Orengo, C. A., and Sillitoe, I. Cath: an expanded resource to predict protein function through structure and sequence. *Nucleic acids research*, 45(D1): D289–D295, 2017.

DeepMind. Alphafold github: Running alphafold. https://github.com/deepmind/alphafold#running-alphafold, 2022. Accessed: 2022-03-30.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. arxiv. *arXiv preprint arXiv:1810.04805*, 2019.

Dill, K. A., Ozkan, S. B., Shell, M. S., and Weikl, T. R. The protein folding problem. *Annu. Rev. Biophys.*, 37: 289–316, 2008.

Elgammal, A. and Lee, C.-S. Inferring 3d body pose from silhouettes using activity manifold learning. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pp. II–II. IEEE, 2004.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jones, D. T. and Kandathil, S. M. High precision in protein contact prediction using fully convolutional neural networks and minimal sequence features. *Bioinformatics*, 34(19):3308–3315, 2018.

Jones, D. T., Singh, T., Kosciolek, T., and Tetchner, S. Metapsicov: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. *Bioinformatics*, 31(7):999–1006, 2015.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. **Highly accurate protein structure prediction with AlphaFold**. *Nature*, 596(7873): 583–589, 2021.

Kaján, L., Hopf, T. A., Kalaš, M., Marks, D. S., and Rost, B. Freecontact: fast and free software for protein contact prediction from residue co-evolution. *BMC bioinformatics*, 15(1):1–6, 2014.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirkwood, J. G. Statistical mechanics of fluid mixtures. *The Journal of chemical physics*, 3(5):300–313, 1935.

Kryshtafovych, A., Schwede, T., Topf, M., Fidelis, K., and Moult, J. Critical assessment of methods of protein structure prediction (casp)—round xiv. *Proteins: Structure, Function, and Bioinformatics*, 89(12):1607–1617, 2021.

Liu, H., Jiang, B., Xiao, Y., and Yang, C. Coherent semantic attention for image inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4170–4179, 2019.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

McGuffin, L. J., Bryson, K., and Jones, D. T. The psipred protein structure prediction server. *Bioinformatics*, 16(4): 404–405, 2000.

Mirdita, M., Von Den Driesch, L., Galiez, C., Martin, M. J., Söding, J., and Steinegger, M. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic acids research*, 45(D1):D170–D176, 2017.

Mitchell, A. L., Almeida, A., Beracochea, M., Boland, M., Burgin, J., Cochrane, G., Crusoe, M. R., Kale, V., Potter, S. C., Richardson, L. J., et al. Mgnify: the microbiome analysis resource in 2020. *Nucleic acids research*, 48 (D1):D570–D578, 2020.

Nero, T. L., Parker, M. W., and Morton, C. J. Protein structure and computational drug discovery. *Biochemical Society Transactions*, 46(5):1367–1379, 2018.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Remmert, M., Biegert, A., Hauser, A., and Söding, J. Hhblits: lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature methods*, 9(2):173–175, 2012.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Seemayer, S., Gruber, M., and Söding, J. Ccmpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics*, 30(21):3128–3130, 2014.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W., Bridgland, A., et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792): 706–710, 2020.

Sippl, M. J. Calculation of conformational ensembles from potentials of mena force: an approach to the knowledge-based prediction of local structures in globular proteins. *Journal of molecular biology*, 213(4):859–883, 1990.

Steinegger, M., Mirdita, M., and Söding, J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nature methods*, 16(7):603–606, 2019.

tensorflow. tf.keras.losses.log$_c osh.$, 2022. Accessed: 2022-03-30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10687–10698, 2020.

Yang, J., Anishchenko, I., Park, H., Peng, Z., Ovchinnikov, S., and Baker, D. Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences*, 117(3):1496–1503, 2020.

Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

A. Input Features

Figure 6. Input Features and Target of BetaFold (extended): The first 57 (all except the last) subplots visualize the 57 channels (features) computed from the protein sequence similar to (Jones & Kandathil, 2018). The final subplot (bottom-row, 3rd-column) shows the target distogram for the corresponding protein sequence.

B. Training BetaFold using `log_cosh`

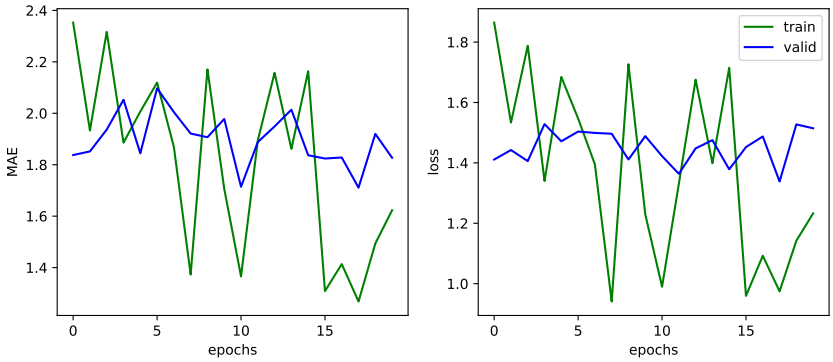

Figure 7. Training curves when BetaFold is optimized using `log_cosh`: (left): shows the mean-absolute error (MAE) between the actual and predicted true-distances, and (right): shows the corresponding `log_cosh` loss optimized by the network with a learning rate of $1e-2$.

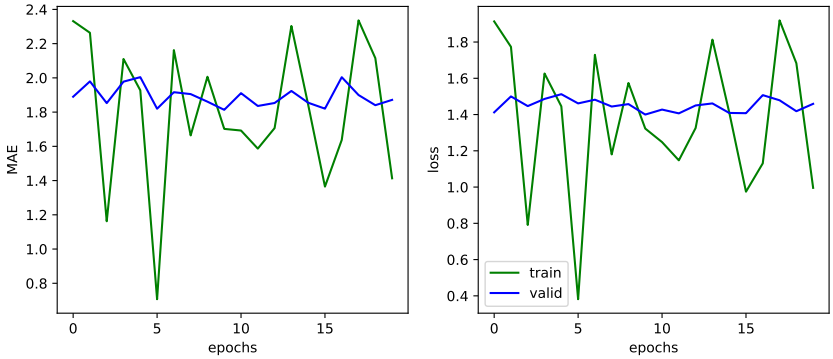

Figure 8. Training curves when BetaFold is optimized using `log_cosh`: (left): shows the mean-absolute error (MAE) between the actual and predicted true-distances, and (right): shows the corresponding `log_cosh` loss optimized by the network with a learning rate of $1e-3$.