Deep Learning for Protein Structure Prediction

CMSC 35440 Machine Learning in Biology and Medicine
Amit Pradhan
pradhanak@uchicago.edu

Introduction

Proteins are the sequences of amino acids which perform a variety of functions in living organisms. Determining protein structures is essential to understand the functions of the proteins. Experimental techniques like x-ray crystallography and NMR produce up to 0.5 Å rmsd accuracy but they require a lot of work even for predicting structure of one protein. Because of this problem, a very small fraction of protein structures have been determined till today. Protein three dimensional structure prediction is solely based on the amino acid sequence and the protein folding problem is of great research interest from the last 50 years. Majority of the protein structure prediction methods depend on the known structures of the homologous proteins. In the absence of homologous proteins, these methods don't perform very well. DeepMind group presented a deep neural network based model AlphaFold 2 in the 14th Critical Assessment of Protein Structure Prediction (CASP14) competition. This model demonstrated to predict protein structures with accuracy similar to the experimental results.

AlphaFold 2 is a novel machine learning approach which combines biological and evolutionary knowledge with latest deep learning architectures to solve the protein folding problem. While AlphaFold 2 achieved state of art accuracy in protein structure prediction, the architectural details for the deep learning system used in AlphaFold 2 is not easily available in the literature. Before AlphaFold 2, another deep learning based model[1] came out in 2017 which significantly outperformed all the existing models at that time. This model predicts contact map from amino acid sequences of the protein and then contact map can be used to predict protein tertiary structure (3-d). The model integrates both sequential and evolutionary information to predict protein contact map using two ultra deep neural networks. I have implemented a slightly modified version of this model using tensorflow.

The model predicts protein contact map from primary structures (amino acid sequences), secondary structures and evolutionary information in the form of position specific scoring matrices (PSSM). It models the atomic interactions as an image in which each pixel represents interaction between two atoms. We can then turn the problem into an image semantic segmentation problem and use convolution operations to build a very deep neural network. From the predicted contact map, we can construct the 3-d tertiary structure of the protein. There are solutions readily available to predict 3-d protein structures from the contact map. In this project, I only predict contact map from the protein sequence.

Imagenet brought a revolution in deep learning especially in image classification. Subsequently, significant innovations were made to the imagenet architecture. Residual connections of the resnet made it possible to train very deep neural networks. Latest deep learning techniques like batch normalization, dropout and self attention mechanism are all used in AlphaFold 2. In this project, I plan to train a very deep neural network for protein contact map prediction and predict protein contact maps with very high accuracy. This model is very unique in the sense that it takes both 1-d input (sequence information) and 2-d input in the form of evolutionary information. One innovation in the paper is how we convert a 1-d matrix to 2-d matrix. This operation is not only a mathematical technique but it models the interactions among the atoms using sequential information like primary/secondary structures.

Background

Broadly there are two types of protein structure prediction models available in the literature based on the availability of similar homologous proteins for which the structures have been experimentally determined[5]. Template based modelling uses a solved protein structure as a template to predict structure of homologous protein. However this method is not very effective in some types of protein e.g. membrane proteins. Template-free modelling (ab initio folding) models are used when no similar proteins with solved structure is available. One such method 'fragment assembly' is based on monte carlo and is used for small proteins. However for large scale structural bioinformatics, computational methods are needed to solve protein structures from amino acid sequences. AlphaFold 2 is the current state of art computational method which predicts protein structures with atomic accuracy comparable to the experimental structures for majority of the proteins.

Amino acids in physical contact tend to coevolve across related proteins[5]. For example a mutation which causes one amino acid to grow bigger will survive if another matching protein undergoes contraction to make space available for the former amino acid. So it makes sense to use coevolution analysis from multi sequence alignment of proteins in the protein structure prediction in addition to sequential information like primary, secondary structures and solvent accessibility. There are several variations of neural network based models for protein structure prediction. Deep Belief Networks (DBN) in 2012 and Fully Convolutional Neural Networks (CNN) in 2016 performed well in CASP competitions.

Fig 1 below demonstrates the key ideas behind the deep neural network models for protein structure prediction. The input to this model is amino acid sequences. From amino acid sequences, coevolution analysis is performed using multiple sequence alignment (MSA) and coevolution information is fed into a deep convolutional network along with sequence profile information. The Deep convolutional network predicts the contact map. Contact map along with predicted local structures are used to derive the 3-d tertiary structure of the proteins.

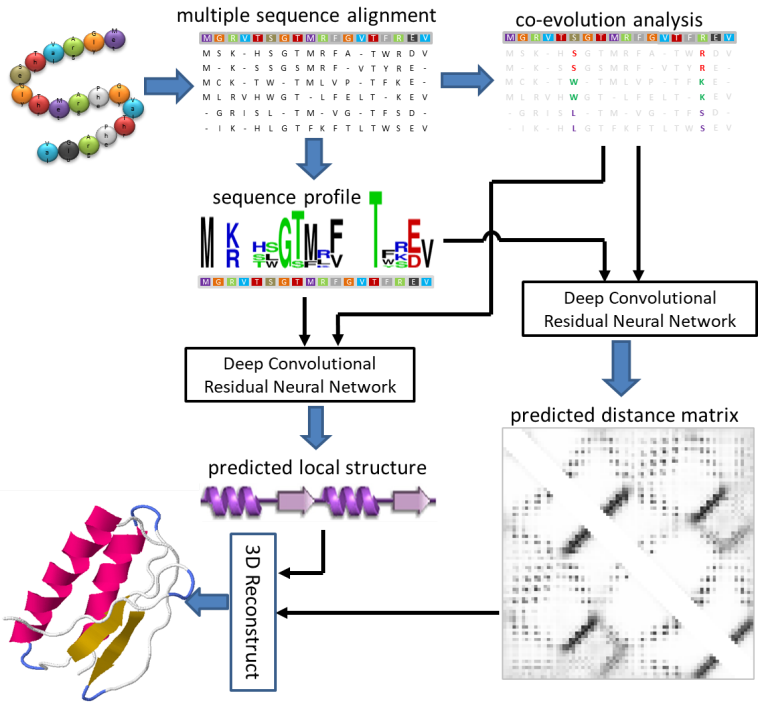

Fig 1: Contact assisted protein folding[5]

Data and Model

Initially I planned to implement the AlphaFold 2 model. Even though the AlphaFold 2 paper provides a high level overview of the model, I could not figure out the exact details of the deep learning model used in it. Instead they provide web service based solutions which can be used to predict protein structures on an ad-hoc basis. However we can use their github repository to deploy a protein structure prediction solution. But just deploying their solution as a black-box model won't provide any insights about the model. So I decided to implement another deep learning based model by Prof Jinbo Xu from Toyota Technological Institute Chicago (TTIC). This group has published multiple computational models, especially machine learning based models on protein prediction and their model RaptorX-Contact ranked first in CASP12 and performed very well in CAMEO[5]. I also took inspiration from the github source code[7] which implements the ideas from the paper[1] using tensorflow keras. This implementation trains a variation of the model proposed in paper[1] with few layers and does not take evolution information into consideration.

The major predicaments for applying machine learning on protein structure prediction is the unavailability of standardized datasets with train/test/validation splits. Since protein structure prediction requires multiple sequence alignments, we need to query the protein database (PDB) to find similar proteins and perform MSA to derive evolutionary information. Similarly secondary structure prediction and other sequential information like solvent accessibility data need to be extracted. All these preprocessing steps require a significant amount of feature engineering. These preprocessing steps need to be performed early to make the data available for the machine learning model. Otherwise these steps may pose a bottleneck for machine learning models and precious compute power will be wasted waiting on the data pipeline. Proteinnet[6] provides a standardized dataset for machine learning on protein structure prediction. It provides sequential information (primary and secondary structures), tertiary structure, multiple sequence alignments (MSA), position specific scoring matrices (PSSM) along with standard train/validation/test split for building machine learning models. Proteinnet data is based on the biennial CASP assessment to carry out blind fold predictions on recently solved protein structures but not yet published in public domain, to enhance protein structure prediction models. Data from CASP 7 to CASP 12 is provided but I have used CASP11 data for the project as CASP12 dataset is incomplete in proteinnet. The datasets are available in two formats i.e. text format and tensorflow specific TFRecord format. Each file contains protein data including protein id, primary structure, evolutionary data (pssm), tertiary information (coordinates of N, C-alpha, C-beta) for each amino acid. As suggested in the paper, I have used beta carbon coordinates to calculate the distance between each pair of amino acids. Atoms with an euclidean distance less than 8Å are suggested to form a contact[1]. The data files also contain a mask (one bit indicator) for each amino acid which basically tells if the tertiary coordinates for the amino acid is present or not.

From Proteinnet data, I have used proteins with length in between 16 and 128. I didn't include large proteins primarily due to computational reasons. As we will predict contact map which will be a max_length * max_length matrix for each protein, increasing max_length by a factor of 2 will increase the memory requirements by a factor of 4. We also have to pad smaller molecules with a large number of zeros. I found choosing the max_length of 128 gives about 56,000 proteins which is sufficient for deep learning. Choosing a larger max_length will also increase the model parameters size and reduce the batch_size which can be trained on a machine constrained by GPU memory (Google colab in this case has about 16GB video memory).

Data Preprocessing

I have modified the text and tensorflow parser to make them compatible with the latest version of python and tensorflow. I downloaded CASP 11 data from proteinnet in tensor file format. Since this dataset does not contain secondary structure information, I downloaded a json file (i.e. PDB entries) which contains secondary structures (8 classes) of proteins. Proteins with length between 16 and 128 are selected. Among these proteins, proteins whose secondary structures are not present in the PDB entries are further filtered out. After applying the above filters, we get 56,355 proteins in total. I prepared a train/test split from the data in a 7:3 ratio. For each protein, the primary structure is converted into one-hot vector representation. Proteins with length less than 128 are zero-padded which gives us a 128 * 20 matrix for the primary structure of each protein. Similar way, secondary structure is converted into a 128 * 8 matrix for each protein. Since both primary and secondary structures provide sequential information, we stack these two matrices to give us a 128 * 28 matriz for each protein. This matrix is called X_seq which is one of the inputs for our model. Evolutionary information is also zero-padded which results in a 128 * 21 matrix. Proteinnet provides 3-d coordinates for N, C-alpha, C-beta atoms of each amino acid in the protein. Using the coordinates of C-beta carbons, we calculate the distance between each pair of amino acids. Amino acids with distance less than 8Å form contact. We convert the distance matrix to contact map matrix which contains 0 and 1 (0 - no contact, 1 - contact). Contact map is 128 * 128 matrix for each protein. Using the mask information provided (a bit representation indicating whether 3-d coordinates for an amino acid is known or not), we create a 128 * 128 mask matrix which will be used in the calculation of the loss and accuracy of the model. If the 3-d coordinates for an amino acid are not known, the values of the row and column corresponding to the amino acid are set to 0 in the mask matrix and 1 otherwise. The mask matrix also contains 0s for padded rows and columns. So this matrix provides a way not to penalize the model for predictions corresponding to the padded entries or entries whose 3-d coordinates are not known. I have created chunks of 1000 proteins and saved data to npz files. The data loader will read each file one-by-one and load only a fraction of the data into memory. Otherwise if we try to load the whole dataset into memory, we will run into memory issues. Now we have all the information to build a model.

Model

Figure 2 demonstrates the model proposed in the paper[1] which is implemented with slight modifications. The inputs to our models are sequential information (128*28) and evolutionary information (128*21). Sequential information is passed through a series of residual layers. Figure 3 shows the residual block I have used in the model. I have 24 such blocks which gives us 48 layers and there are two additional conv1d layers without residual connections. As we can see, there are two layers in the residual block. I have two conv1d layers with kernel size = 3. I also use padding ('same') to maintain the length of the sequential matrix at 128 for each conv1d layer. Each conv1d layer is followed by a batch normalization layer, relu activation layer and a dropout layer finally. Batch normalization layers normalize (0 mean and unit variance) the output of the convolution operation so that the values stay in the activation region of the ReLu activation. ReLU activation layer is followed by a dropout layer which provides a way of regularization in the model as it randomly drops some values. Dropout layer is essential to prevent overfitting in the training phase and it is not used in the inference phase. In the residual blocks, input to the first conv1d layer and output of second conv1d layer are added before passing it through the batch normalization, relu and dropout layer. Apart from the 24 residual blocks, I have two additional conv1d layers without residual connections which are primarily used to increase the number of channels to 64 and 128 respectively. After passing the sequential information through the 50 conv1d layers, we get a matrix of 128 * 128 (128 channels and 128 length) for each protein.

We now convert the sequential vector 128 * 128 into 128 * 128 * 256 by a conv1dto2d operation. To convert a 1D matrix into a 2D matrix, we create a blank 2D matrix of shape 128 * 128 first. Then for each position (i, j) (ith

row and jth column) in the 2D matrix, we stack the ith and jth channels from 1D matrix which gives us a 128 * 128 * 256 matrix. Similar operation is performed on the evolutionary information matrix (128 * 21) to give us a 128 * 128 * 42 matrix. We then combine these two matrices (128 * 128 * 256 and 128 * 128 * 42) to obtain a 128 * 128 * 298 matrix which will be fed to the conv2d layers.

Similar to conv1d residual blocks, we have 22 conv2d residual blocks. The only difference here is the use of conv2d in place of conv1d. Conv2d layer uses a kernel of shape 3 * 3 and uses 'same' padding to maintain the shape of the input. Apart from the residual blocks, I have 5 additional conv2d blocks. 4 of the 5 conv2d layers are used to change the number of channels to 256, 128, 64, 4 respectively. The last conv2d layer reduces the number of channels from 4 to 1. The input matrix 128 * 128 * 298 is reduced to an 128 * 128 * 1 matrix by the conv2d layers. The final 128 * 128 * 1 matrix represents logits for contact map of the protein. We use binary cross entropy loss at each position in the 128 * 128 contact map. We use the mask matrix as a weight matrix to calculate a single loss value from the 128 * 128 binary cross entropy loss values. Use of the mask matrix ensures we don't penalize the model for the padded positions and the positions for which tertiary coordinates are not known. Table 1 demonstrates the details of the model.

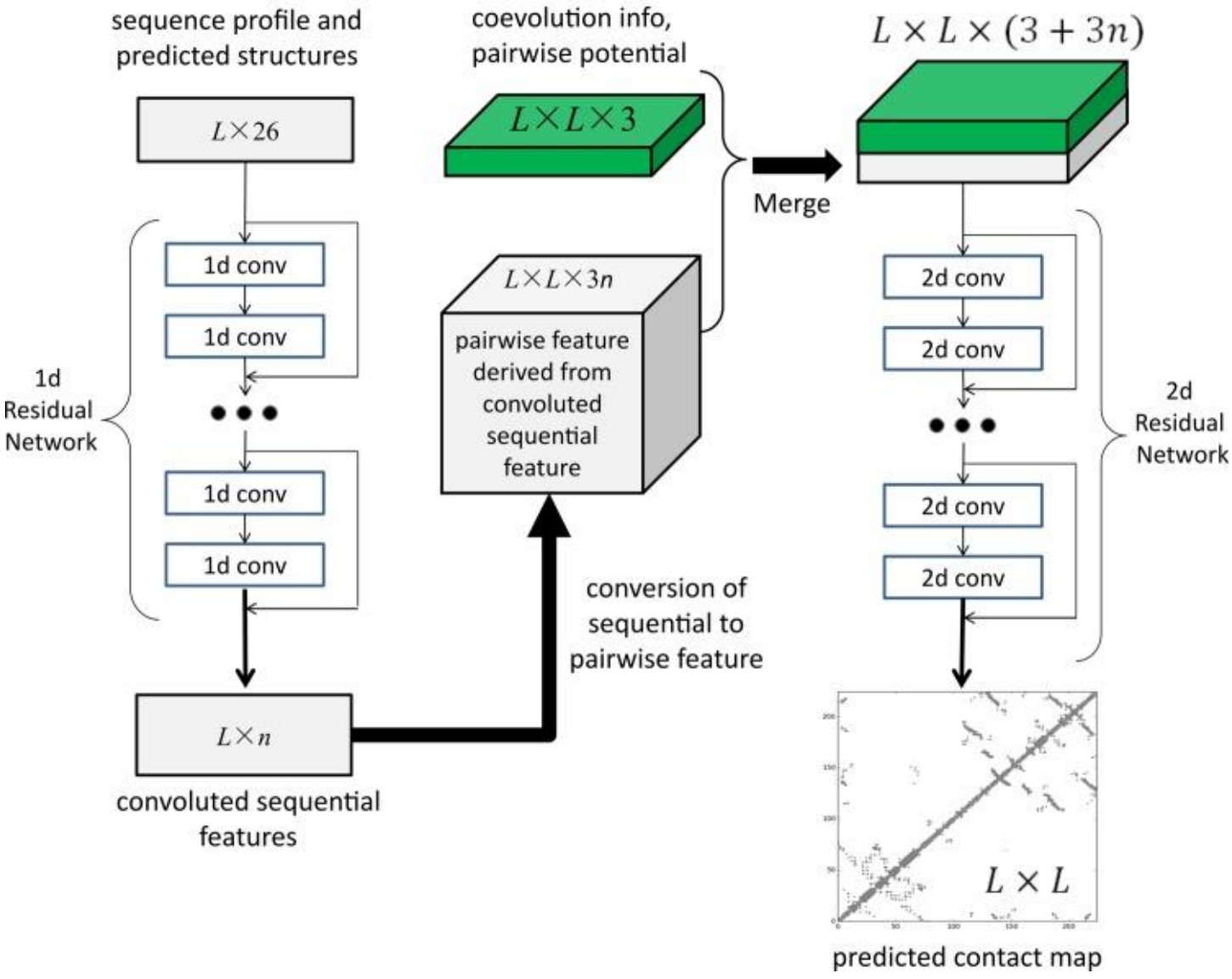

Fig 2: Deep learning architecture proposed in the paper [1]

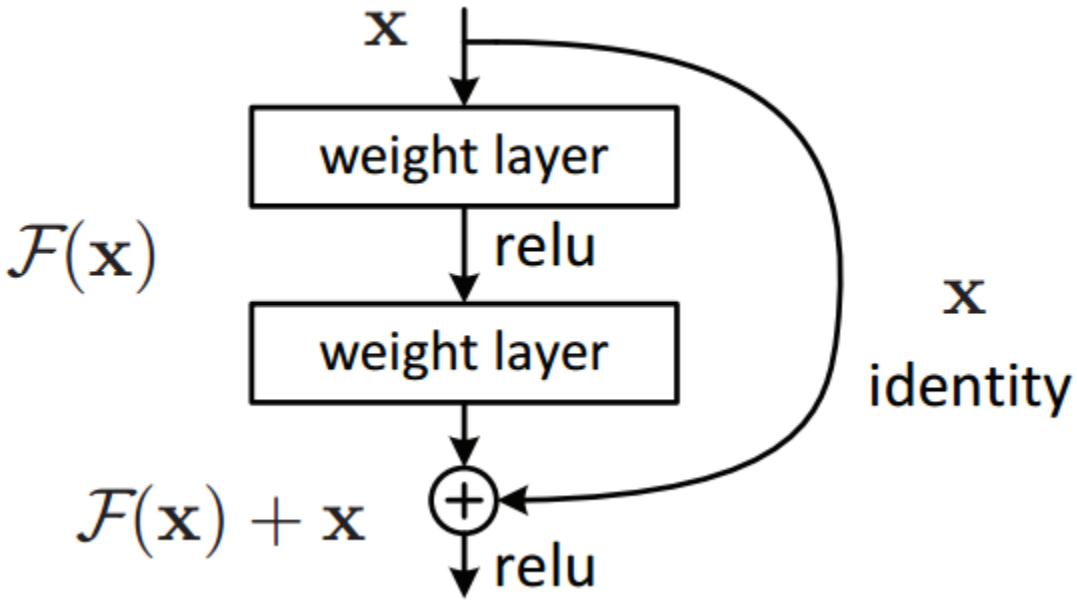

Fig 3[8]: Resnet block used in the model. The weight layer is either a conv1d or conv2d layer, followed by a batch normalization layer, relu activation and finally a dropout layer.

Conv1d layers	50
Conv2d layers	50
Total layers	100
Trainable params	10,512,625
Non-trainable params	20,360
Total params	10,532,985

Table 1: Details of the model

Innovations

The key idea behind this model is to represent the interaction between the amino acids as an Image where a pixel represents interaction between two amino acids. Similar to imagenet, convolution layers are used for predicting the contact map. Use of mask matrix not to penalize the model for the padded positions and the positions for which 3-d coordinates are not known is another key idea in this model.

I think the uniqueness of this deep learning architecture is that the model is able to combine both 1-D and 2-D inputs using two deep learning networks. At heart of the model lies the 1-D to 2-D conversion layer which converts the output of the 1-D residual blocks and evolutionary information into 2-D. The original paper suggests stacking the ith, jth and (i+j)/2 th channels at each position (i, j) to obtain a 2-D matrix from a 1-D matrix. However I could find an efficient implementation to add (i+j)/2 th channel. So in my implementation I

stack only ith and jth while obtaining the 2-D matrix. Intuitively it makes sense since interaction between two positions should be obtained from the channels present at these two positions. I have explained the algorithm to obtain 2-D from 1-D matrices below.

Algorithm to convert from 1-D to 2-D:
input : B * L * n (B: Batch size, L: Length of the protein, n: number of channels)
output: B * L * L * 2n
X1 = tf.ones(B, L, L, 1) * tf.expand_dims(x, axis=2)
X2 = tf.transpose(x, (0,2,1,3))], axis=3)
output = tf.concat([X1, X2], axis=3)

Result

I trained the model in Google Colab which uses the Tesla P100 GPU accelerator. It has about 16 GB GPU memory and I could train with a batch size of 10. Each epoch takes around 1 hour, so running 5 epochs took about 5 hours. I could achieve around 95 % train and test accuracy with the model. Figure 6 shows comparison between contact map prediction vs actual values for 10 proteins in the test dataset.

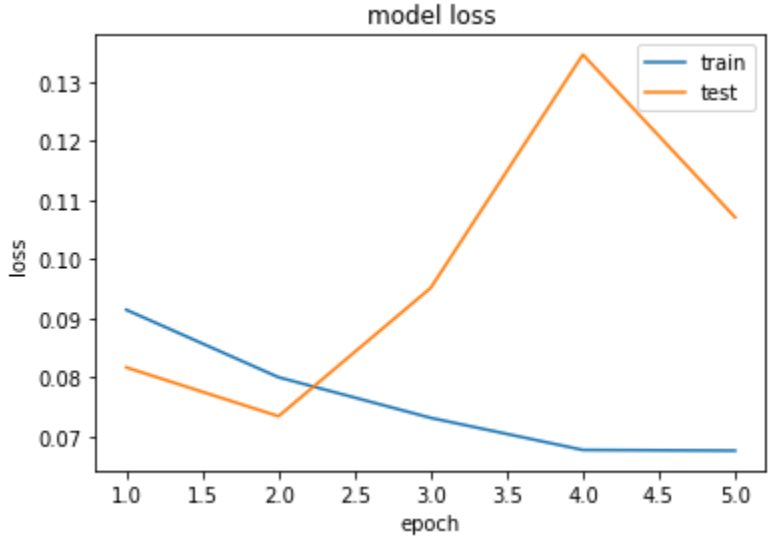

Fig 4: train and test loss vs epoch

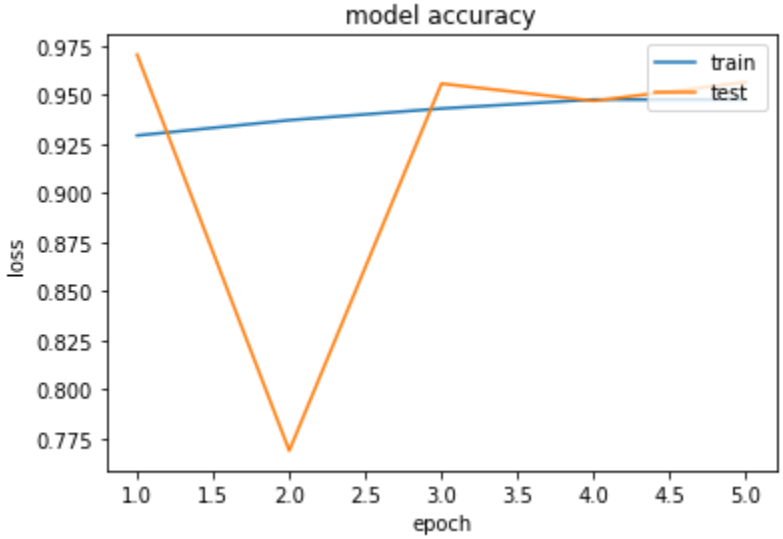

Fig 5: train and test accuracy vs epoch

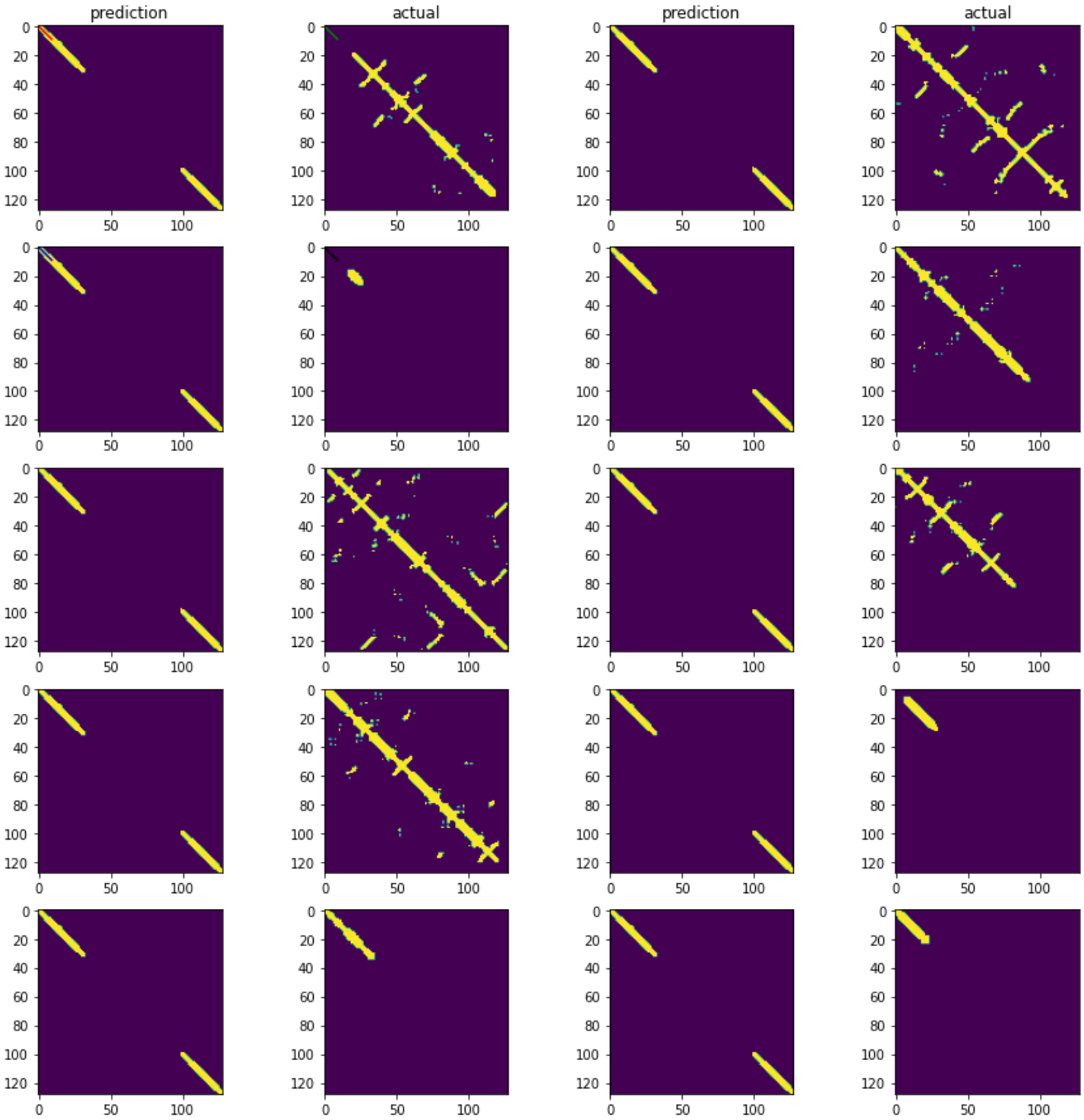

Fig 6: comparison of prediction vs actual for some proteins

Conclusion and Lesson Learnt

From the above results, we can observe that the model achieves close to 95 % train and test accuracy with 5 epochs. Due to compute limitations, I could not run for more epochs. I believe if we could train the model for a sufficient number of epochs, we will get even better model accuracy. However with this work, we demonstrate that deep learning can be used in protein contact map prediction. Latest deep learning based models like AlphaFold 2 in fact have achieved close to experimental accuracy.

Since this model came in 2016, one year before the self attention paper was published. Self attention was a major innovation and is now being used in most deep learning systems. In fact AlphaFold 2 uses self attention. If we have to modify this model to include self attention layers, I believe we can have multi-head attention layers in place of the 1-d convolutional layers. Using the multi-head attention mechanism, we will be able to extract multiple meanings/features from the sequential information (primary and secondary structures).

References

[1] Wang, S., Sun, S., Li, Z., Zhang, R., & Xu, J. (2017). Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model. PLoS computational biology, 13(1), e1005324. https://doi.org/10.1371/journal.pcbi.1005324

[2] Jumper, J., Evans, R., Pritzel, A. et al. Highly accurate protein structure prediction with AlphaFold. Nature 596, 583–589 (2021). https://doi.org/10.1038/s41586-021-03819-2

[3] Wikipedia contributors, "Protein," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Protein&oldid=1054273318 (accessed November 19, 2021).

[4] https://www.slideshare.net/ag1805x/ab-initio-protein-structure-prediction

[5] Lecture notes, TTIC 31050 Introduction to Bioinformatics and Computational Biology, https://www.ttic.edu/courses/#cb

[6] AlQuraishi, M. ProteinNet: a standardized data set for machine learning of protein structure. BMC Bioinformatics 20, 311 (2019). https://doi.org/10.1186/s12859-019-2932-0

[7] N. Lakmal Deshapriya, Protein Structure Modeling with Deep Learning (Tensorflow-Keras), https://github.com/lakmalnd/deep-protein-structure-modeling

[8] https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec

Appendix

1. deep-learning-protein-structure-prediction.ipynb: jupyter notebook with model and results
2. text_parser.py: to read proteinnet data in text format
3. tf_parsr.py: to read proteinnet data in tensorflow file format
4. data_loader.py: machine learning model data loader
5. preprocess.py: data preprocessing