Yu-Kai Wang, Matthew Uryga
CSCI-4969
Spring 2022

04/27/2022

Final Writeup

1 Distribution of Work

1.1 Yu-Kai (Steven) Wang

(a) Multihead self attention

(b) Outer product mean

(c) Triangular self attention

(d) IPA Module

(e) Parallelization of model so that it can run on any number of GPUs

(f) Dispatching jobs to the DCS cluster

(g) Optimization of evoformer trunk and structure module

1.2 Matthew Uryga

(a) Input feature projections

(b) Evoformer trunk structure

(c) Structure module, excluding the IPA module

  (i) Backbone update
  (ii) FAPE and torsion angle loss
  (iii) etc.

(d) Dataset for batching of crops for training/validation and whole sequences for testing

(e) Model training and validation loops

(f) Model evaluation

(g) Predicted/Ground truth structure visualization

(h) Writeups

1.3 Repository Link

The code for our implementation of Alphafold2 can be found here:
https://github.com/mnuryga/MLBinfCapstone.

2 Method

2.1 Overall Implementation

For the most part, the Alphafold2 supplemental paper was followed closely when implementing the model structure. Some deviations include a reduced breadth of input data (no extra msa information) and some slight variations in the final computation of coordinates. It is also worth noting that the model was not designed to predict anything beyond the ϕ and ψ torsion angles and the locations of C, C_α, and N.

2.2 Parameters

Due to memory, time, and processing power limitations, the parameters of the model were decreased from the specifications in the Alphafold2 paper. Namely, the following parameters were altered:

$$\begin{aligned}
N_{res} &= 256 \longrightarrow 64 \\
N_{clust} &= 16 \longrightarrow 8 \\
c_m &= 256 \longrightarrow 128 \\
c_z &= 128 \longrightarrow 64 \\
c_z &= 128 \longrightarrow 64 \\
c_{MHSA} &= 32 \longrightarrow 16 \\
c_{outer\ prod\ mean} &= 32 \longrightarrow 16 \\
c_{trangular\ attn} &= 32 \longrightarrow 16
\end{aligned}$$

Without these parameter reductions, evaluation of the model would not be possible on the largest test sequences, as the GPUs would run out of memory to hold the entirety of the protein and its representations.

2.3 Tuning of Learning Rate

Because training took a significant amount of time, the learning rate could not be optimally tuned. It was determined through short validation runs that a learning rate of 0.001 would be reasonable. The learning rate was manually halved every 6 epochs.

2.4 Tuning of Dimension Parameters

Through a variety of limited experiments, it was found that decreasing the dimension parameters had a minimal effect on the performance of the model, however this may not have been the case in actuality.

2.5 Training

Training was conducted on the DCS cluster so that the model could make use of 4 GPUs. This substantially decreased the training time from approximately 3.5 hours per epoch to 50 minutes per epoch. In total, the model was trained for 18 epochs.

2.6 Validation

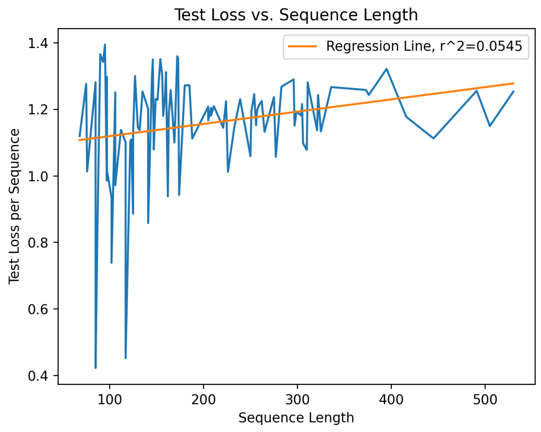
Validation was run after each epoch, and the model training was stopped if the validation loss was greater than the 5-epoch rolling mean of previous validation losses, although this never occurred.

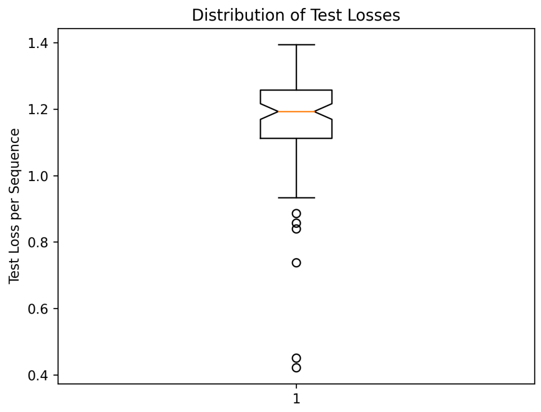
3 Results

3.1 Test Loss

The calculated average test loss was 1.171323 per sequence.

Below are a couple of plots detailing a few more statistics with the test loss:





From the plots above, it appears that the average loss per sequence does not have a significant correlation with sequence length - the least squares regression line has a r^2 value of 0.0545

4 Visualization

The predicted and true protein structures were visualized by plotting the coordinates of the C_α atoms in 3D-space.

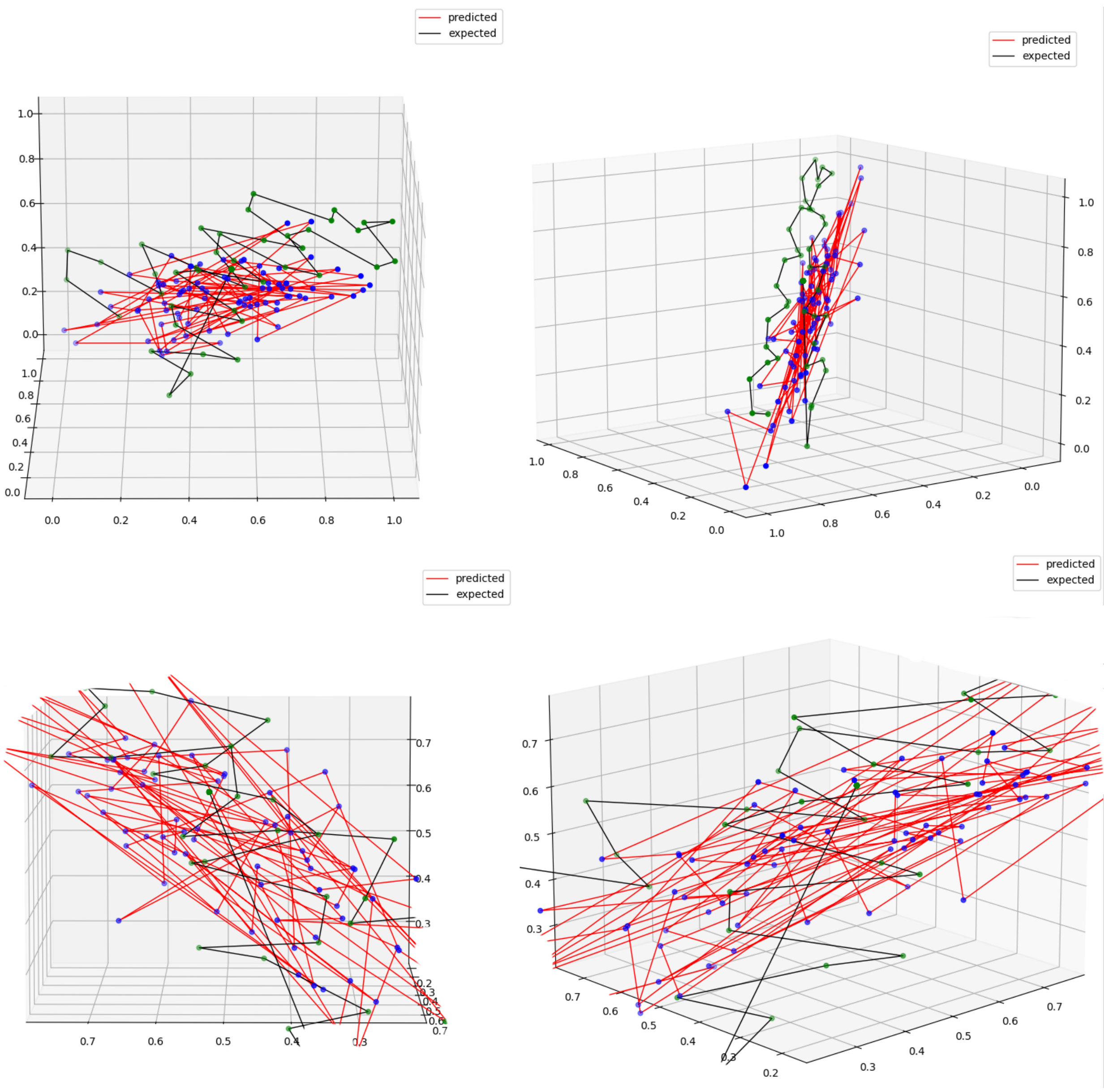
Below are three sequences that were chosen based on their average loss per sequence (the maximum, median, and minimum average loss sequences).

For each protein displayed below, there are two 3D-plots: one without reordering and one with reordering. Reordering was implemented by starting with the first C_α in a sequence and selecting the next C_α by smallest Euclidean distance. This is repeated for the entire sequence.
This creates a less cluttered plot that allows the similarities/dissimilarities to show more clearly.

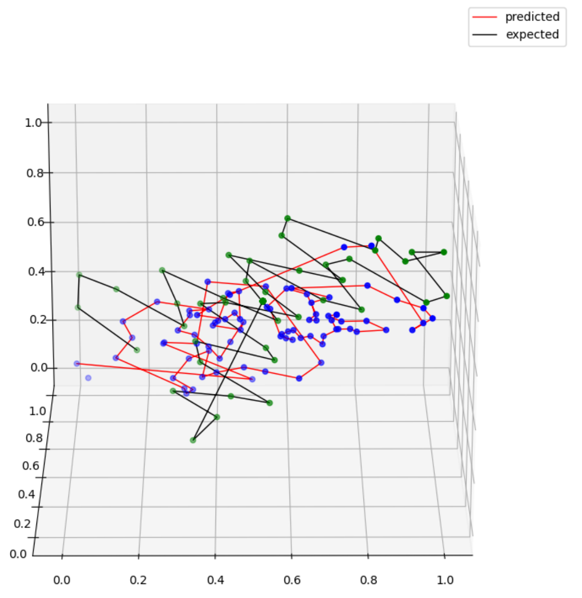
4.1 Best Prediction

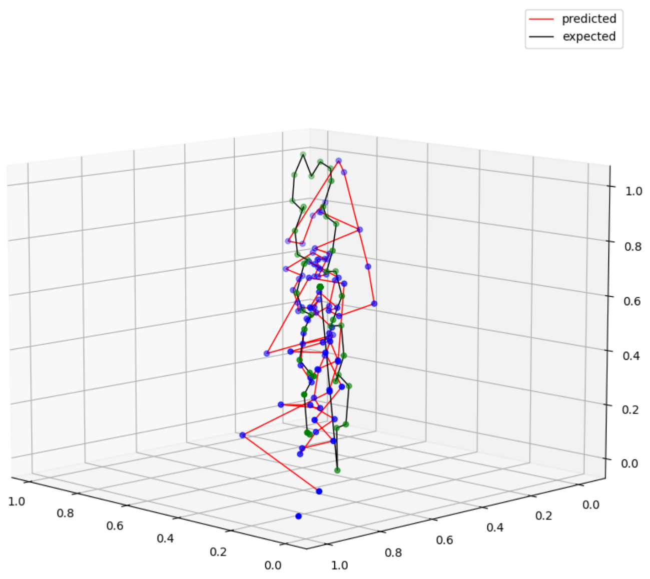
Loss per sequence: 0.42200.

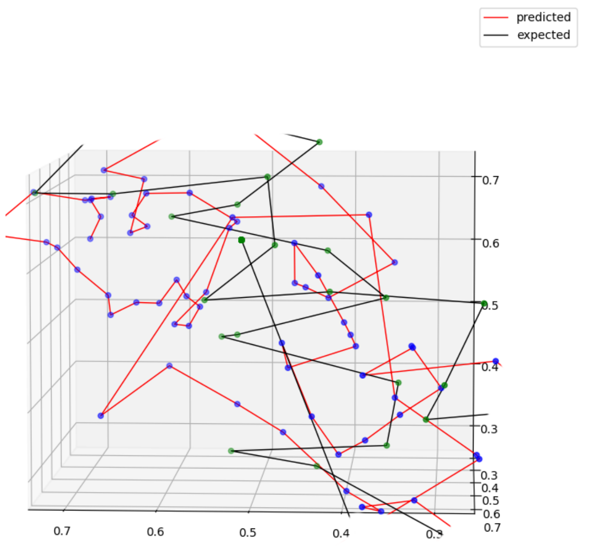
Without reordering:

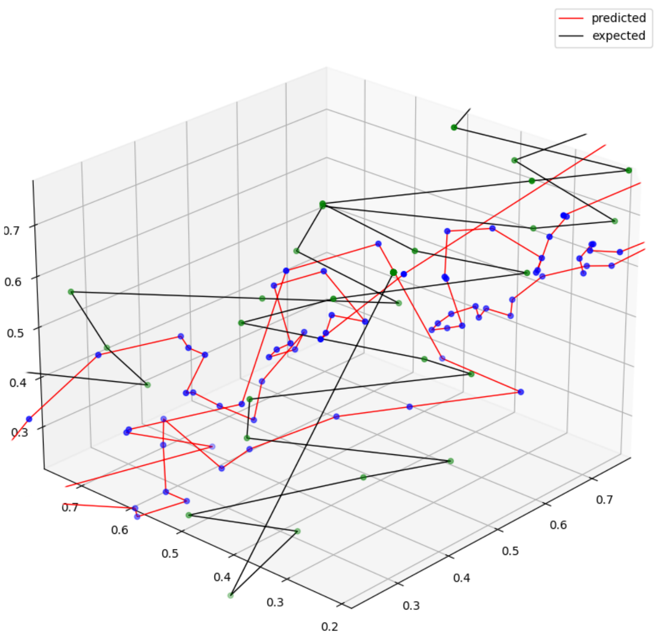


With reordering:





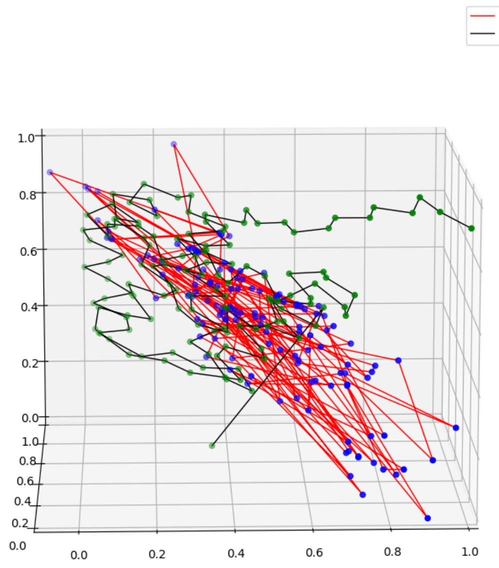


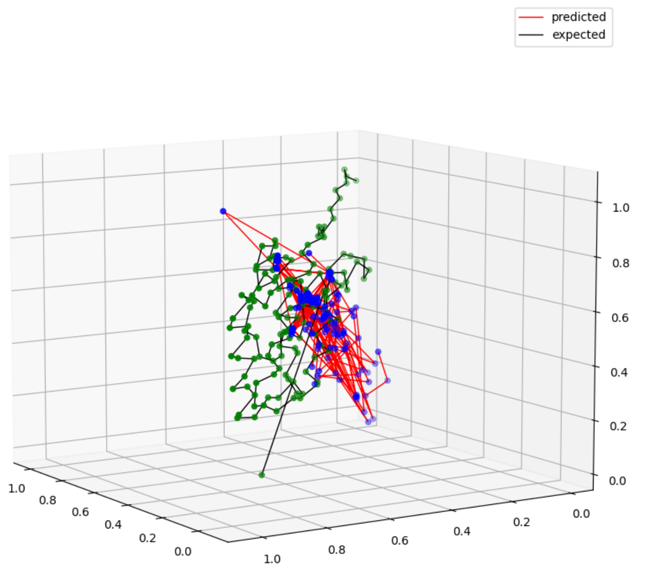


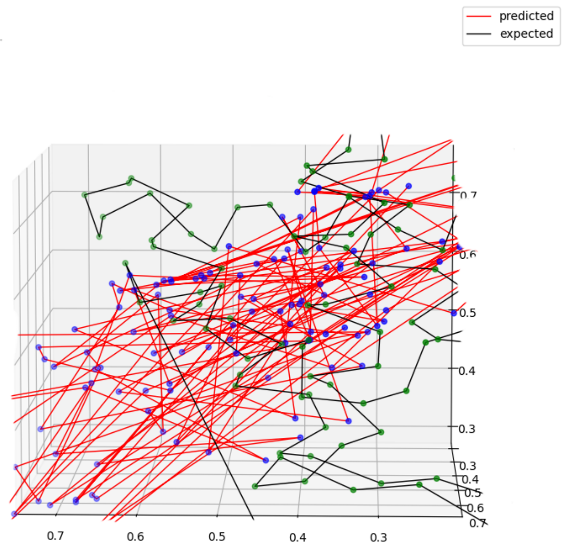
4.2 Median Prediction

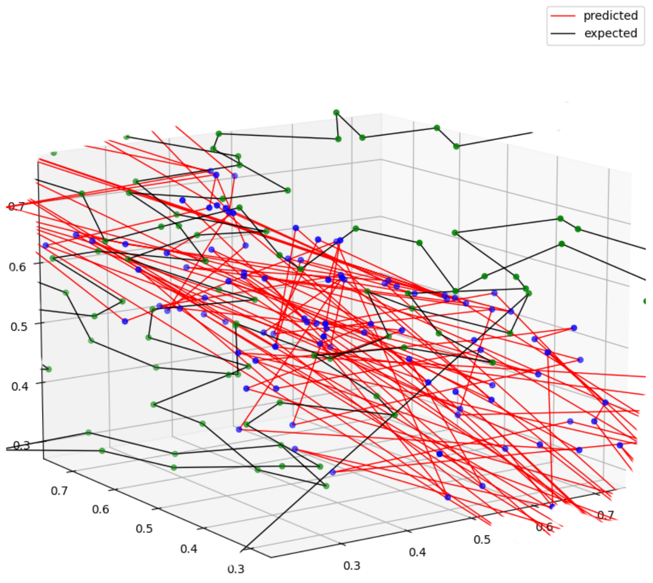
Loss per sequence: 1.17614.

Without reordering:

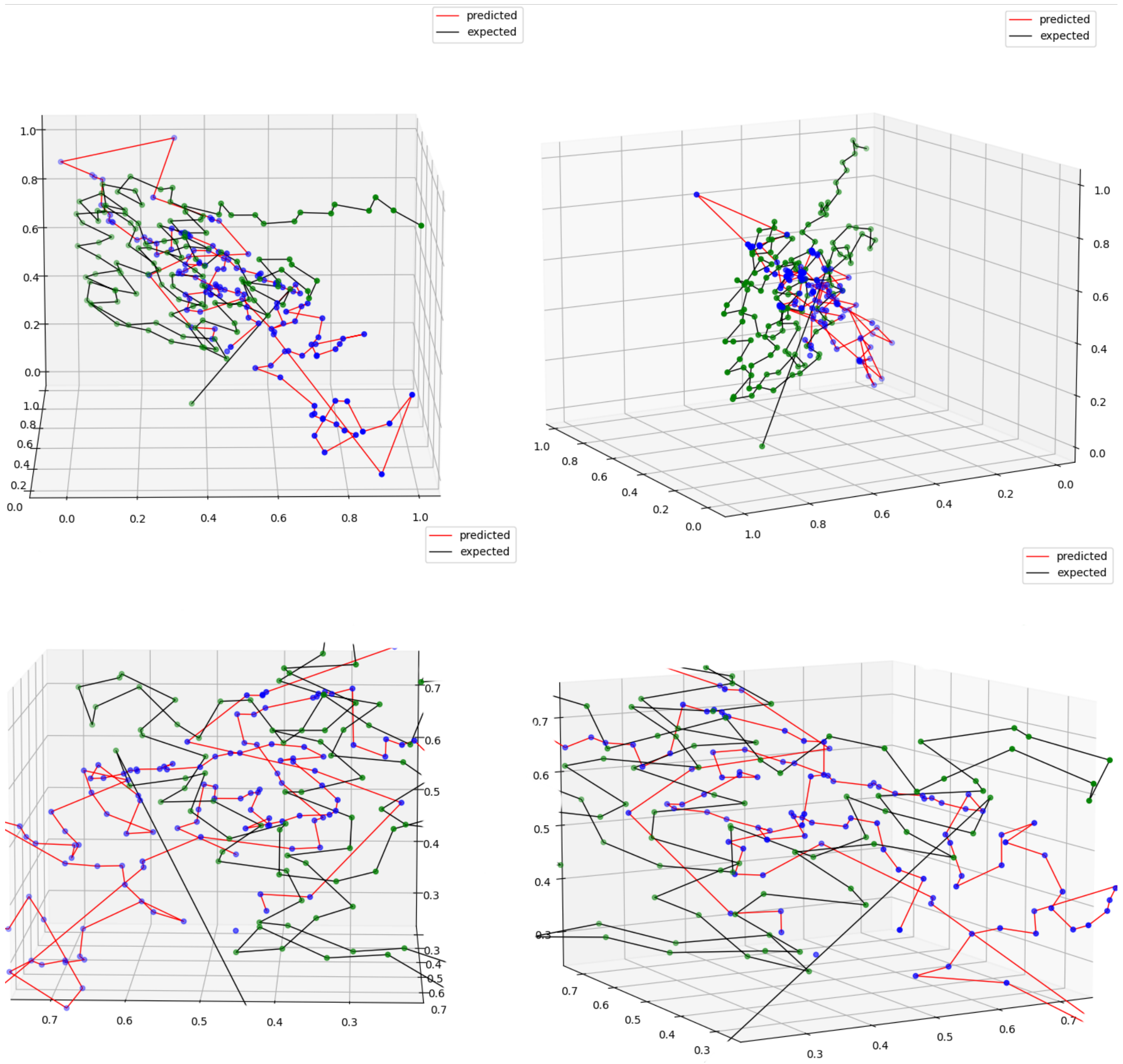








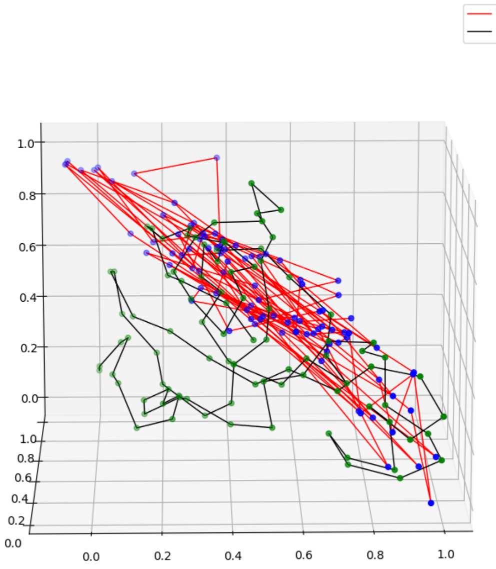
With reordering:

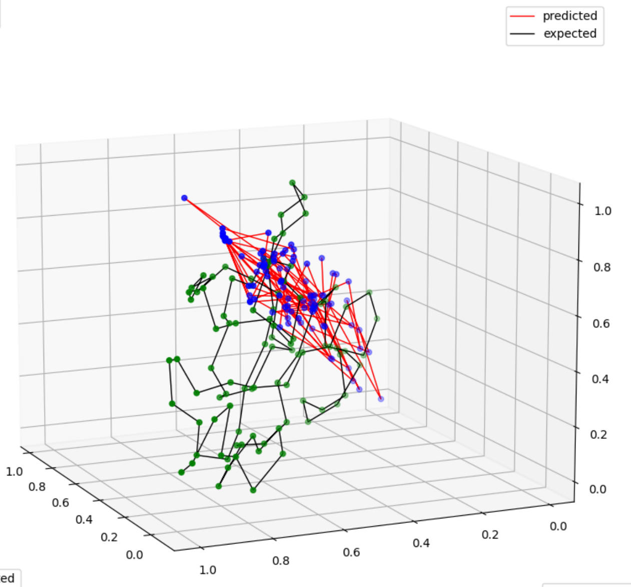


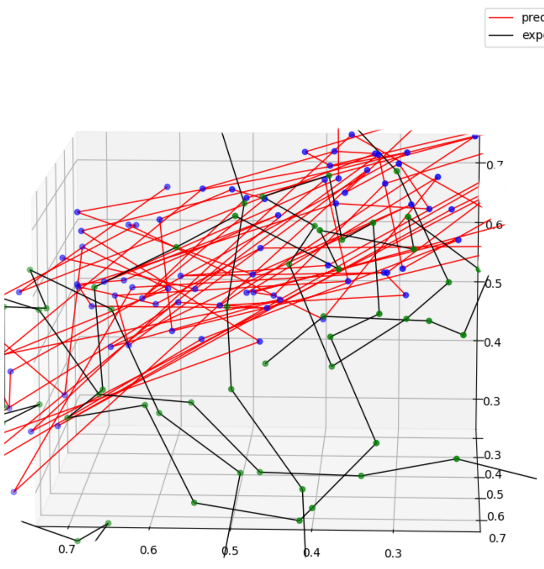
4.3 Poor Prediction

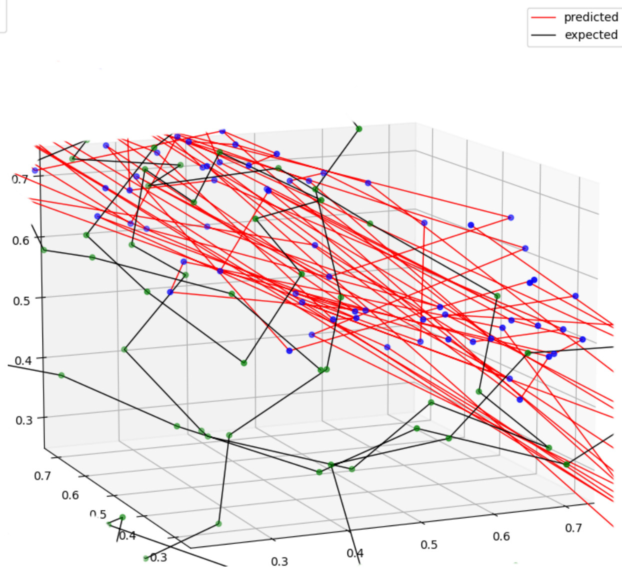
Loss per sequence: 1.39443.

Without reordering:

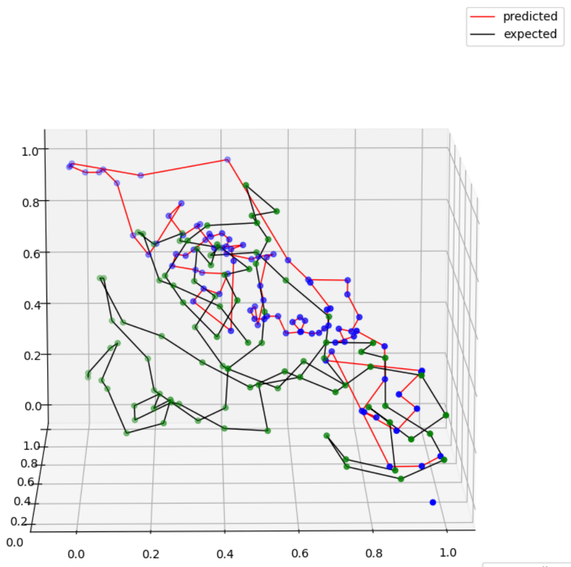


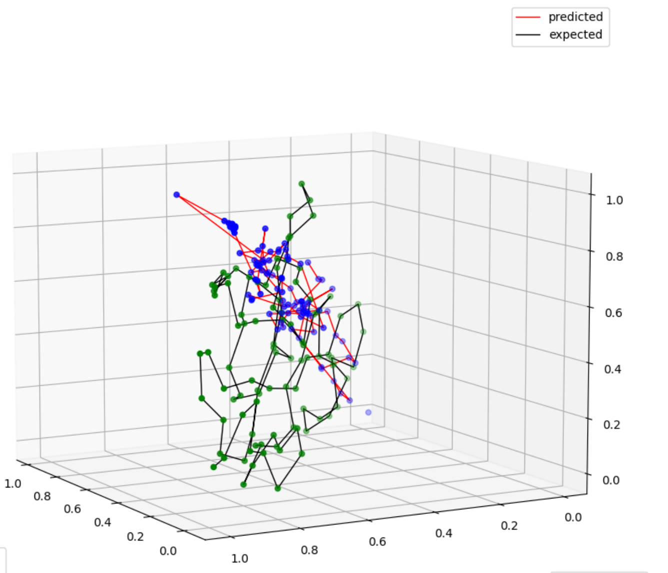


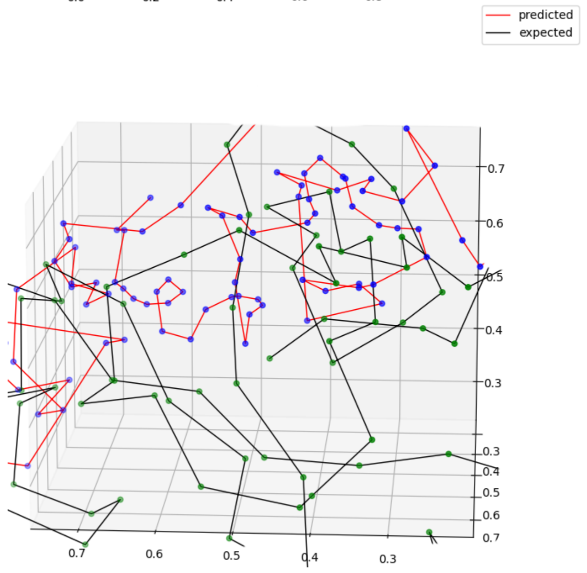


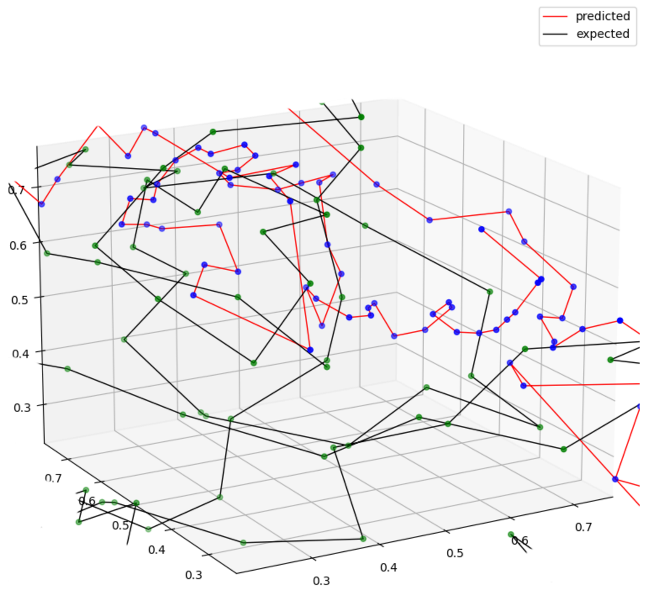


With reordering:









5 Conclusion

5.1 Overall Accuracy

The 3D-plots above make it evident that the current model is not effective at accurately predicting protein structures. When the coordinates are normalized, the predicted coordinates of the C_α atoms line up reasonably well with the actual coordinates, however the order of the atoms is seemingly random. When the order is determined post-prediction with the reordering algorithm described above, the prediction appears to match the expected structure better, however it is still far from ideal.

5.2 Possible Sources of Error

There are several possible causes for the poor accuracy shown above:

(a) The model was not trained for enough time

(b) The model was train with very subpar hyperparameters, leading to very little learning

(c) The reduced dimensionality of the model had a much larger effect on the overall accuracy than was initially expected

(d) There is something fundamentally wrong with the code

Any of these could lead to the inaccurate predictions that the model produces, but most likely it is a combination of all of the above that leads to the poor performance of the model.