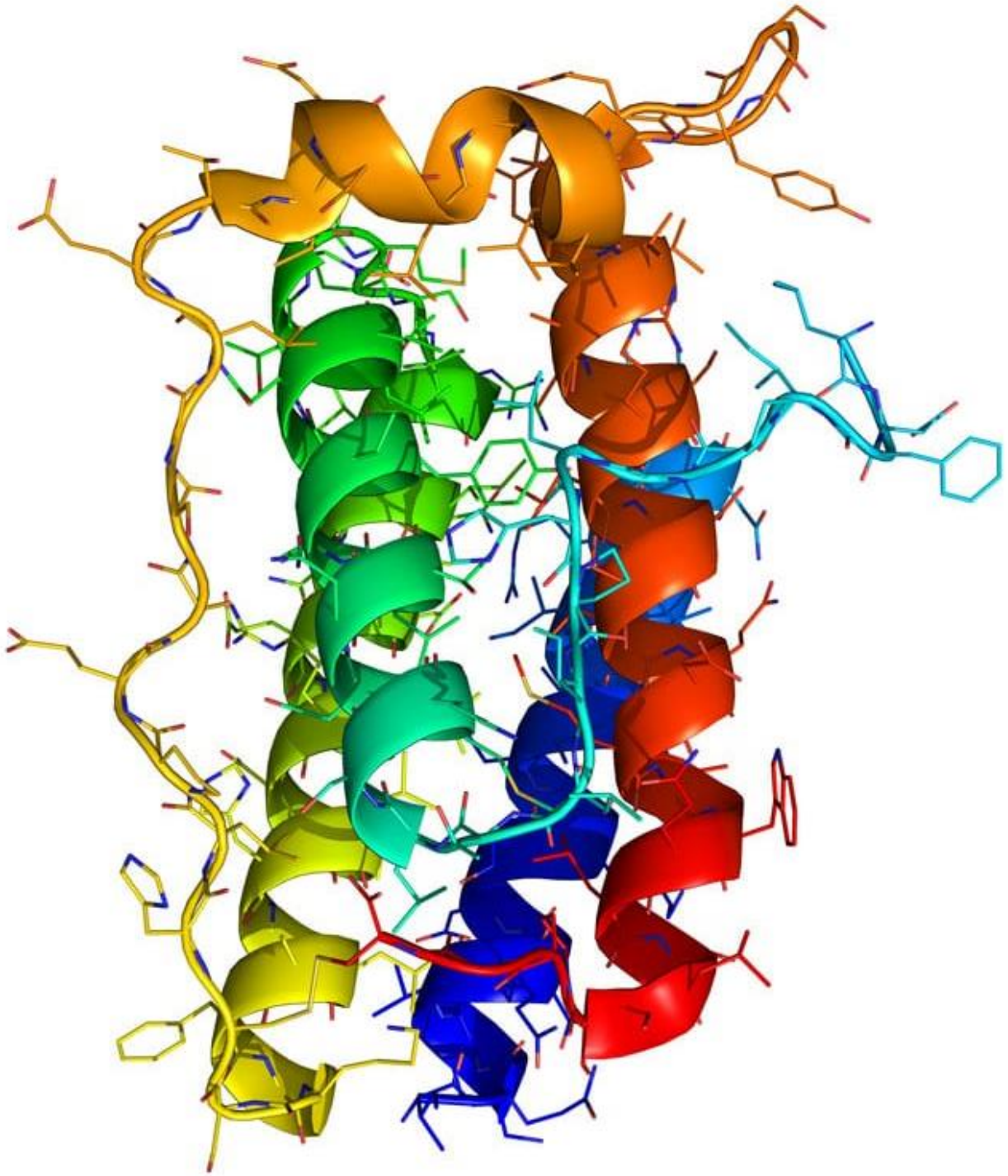
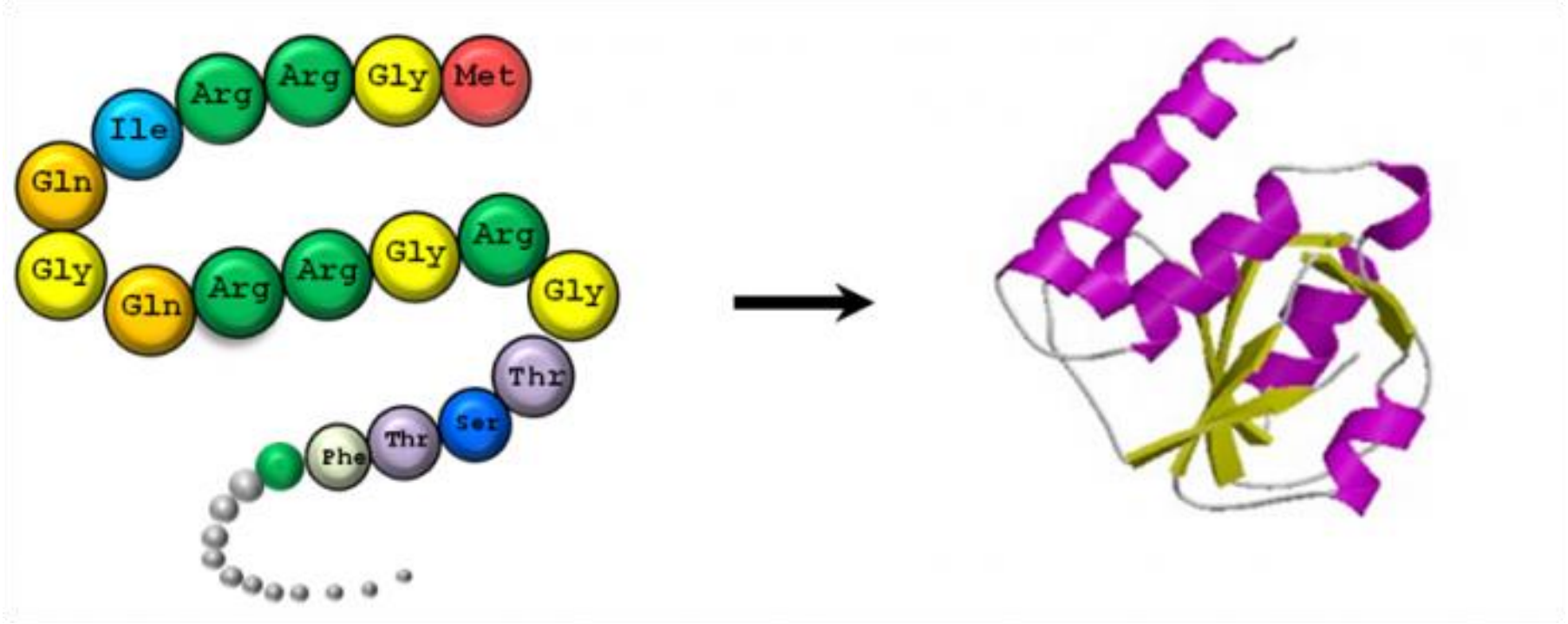


Protein Structure Prediction

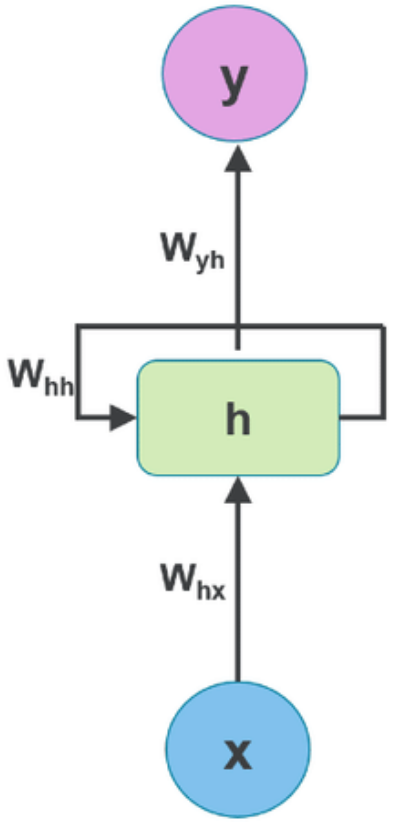
YOMNA GAMAL

Objective

Use deep learning to predict the 3D structure of a protein using just the amino acid sequence.

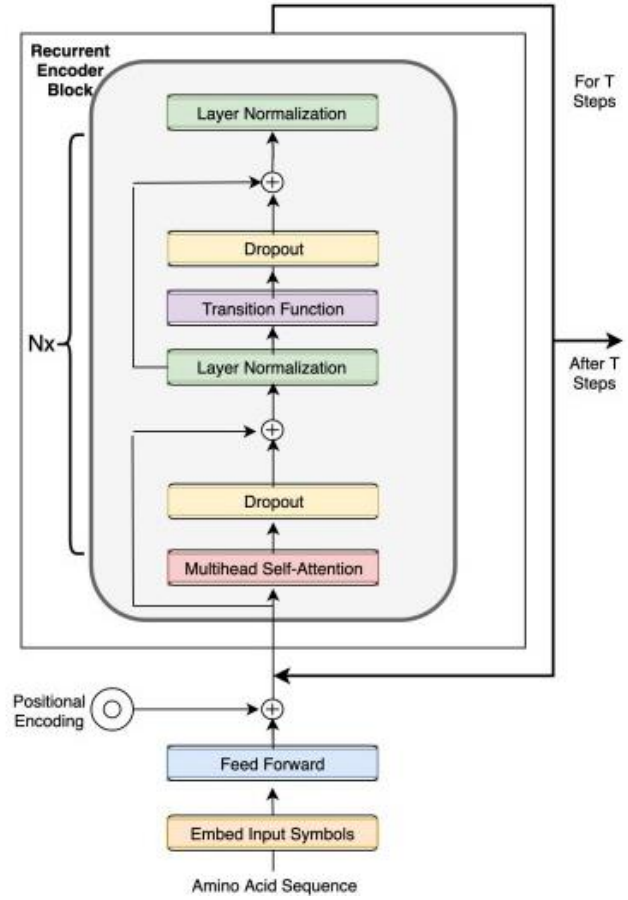


Methods



RECURRENT GEOMETRIC NETWORK (RGN)

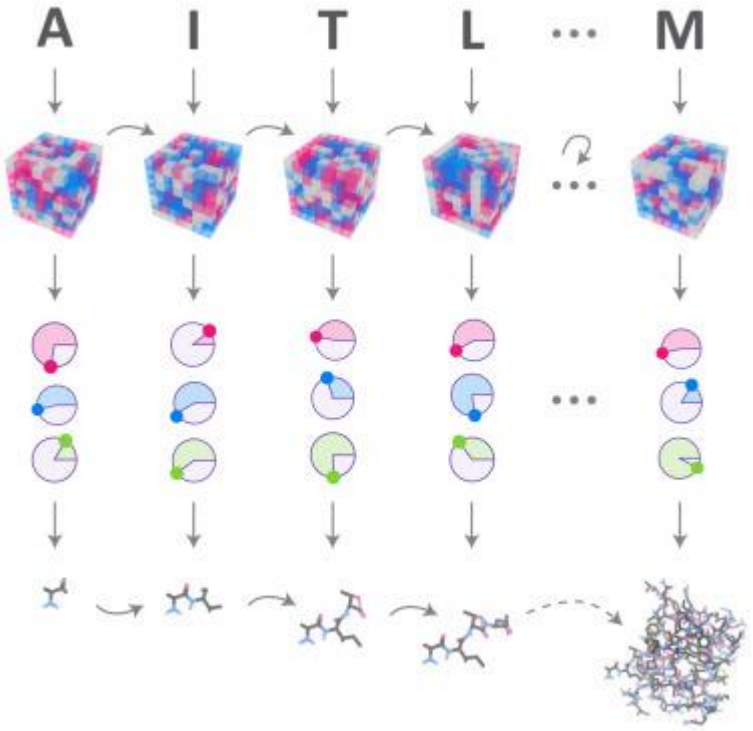
UNIVERSAL TRANSFORMING GEOMETRIC NETWORK (UTGN)



RGN

End-to-End Differentiable Learning of Protein Structure paper introduce an approach based entirely on machine learning that predicts protein structure from sequence using a single neural network. The model achieves state-of-the-art accuracy when published (in 2019), and does not require co-evolution information or structural homologs. It is fast, making predictions in milliseconds.

It based on RNN.



RNN vs Transformers

RNNs takes inputs sequentially, making it harder to parallelize.
Transformers take the entire sequence at once as an input.

Transformers are less susceptible to exploding / vanishing gradient problems found in training **RNNs**.

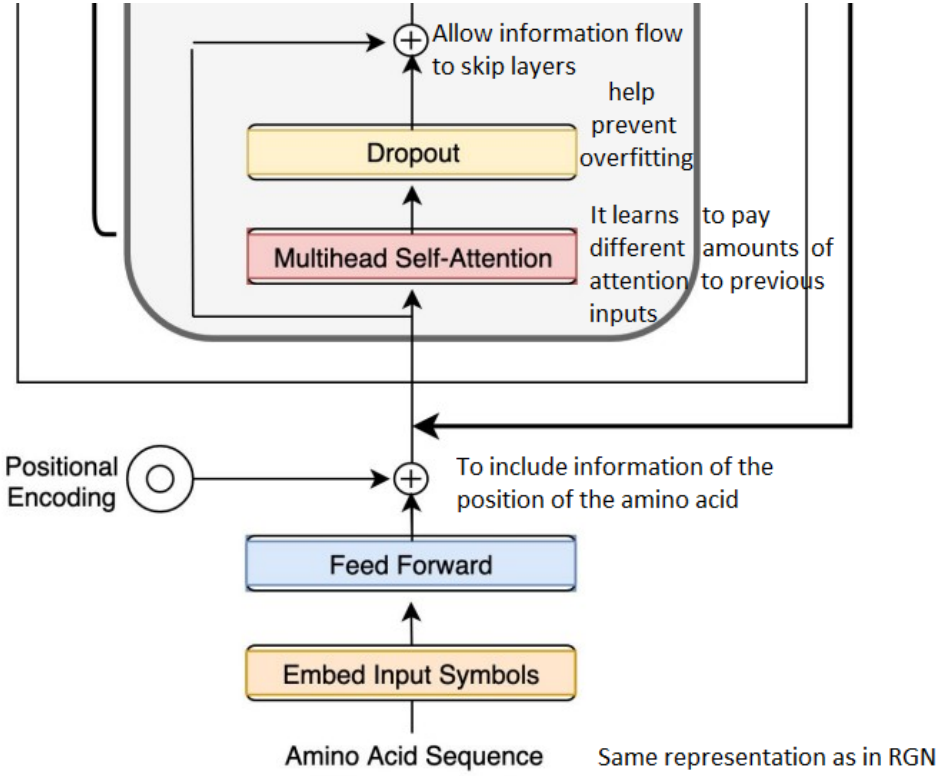
RNNs are unstable (different initialization produces very different results).

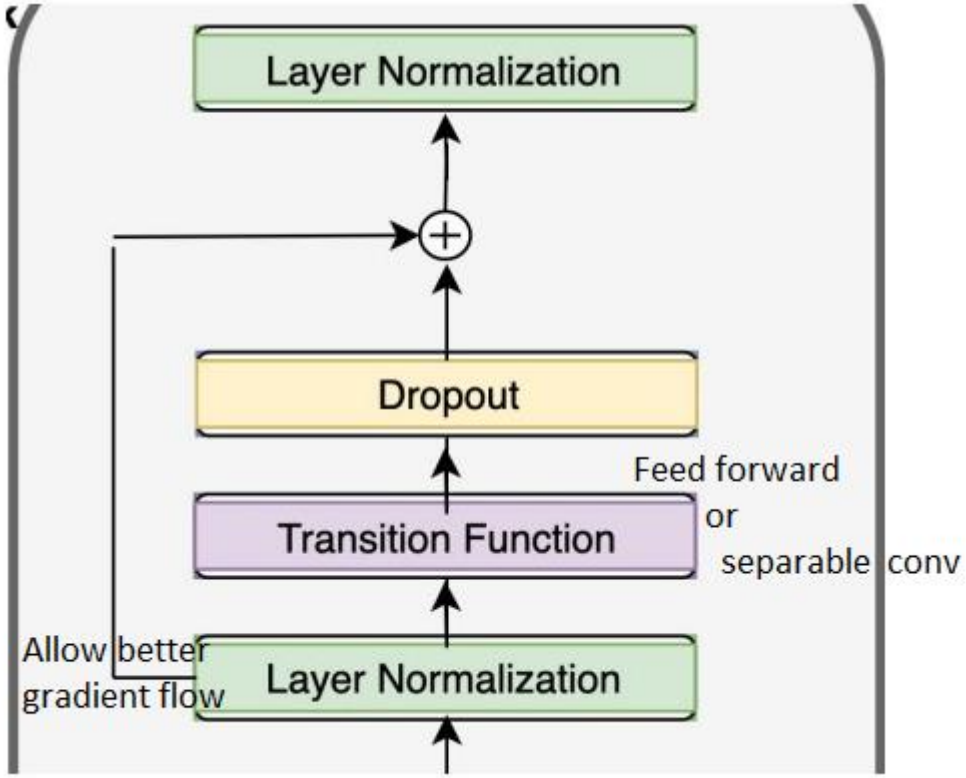
In **RNNs**, Information fades as time step increases.
Transformers are better at creating internal states that considers global dependencies between input and outputs.

UTGN

Keep everything from RGN architecture, except the way the internal states are represented. Replace the Bi-LSTM with a Universal Transformer Encoder to get faster and easier training with good results.

UTGN Architecture





dRMSD Loss

It is calculated by creating the distance map of predicted and experimental structure. Then find the distance between the distance maps. Perform backpropagation to update and optimize weights.

This loss function is **translationally** and **rotationally invariant**. It is differentiable.

Comparison Between Official Paper Results

Template Based Modeling

Model	dRMSD (Å)	TM score
RGN	17.8	0.200
UTGN-FF	17.6	0.198
UTGN-SepConv	**17.1**	**0.208**

Free Modeling

Model	dRMSD (Å)	TM score
RGN	19.8	0.181
UTGN-FF	19.4	0.174
UTGN-SepConv	**18.1**	**0.194**

DataSet

PROTEINNET

- Train / validation set contains all the PDB structures except:
  Structures less than 2 residues.
  >90% of structures were not resolved.
- Contains mask records for missing residues in PDB structure.
- Multiple logical chains are combined.
- Methods for making TF records efficiently.

SIDECHAINNET

- It directly extends ProteinNet.
- Methods for loading and batching SidechainNet data efficiently in PyTorch.
- Methods for generating protein structure visualizations from model predictions.

RGN – Implementation Used

Its official code was using ProteinNet “for dataset” and implemented using TensorFlow.

The loss was used is distance-based root mean square deviation (dRMSD).

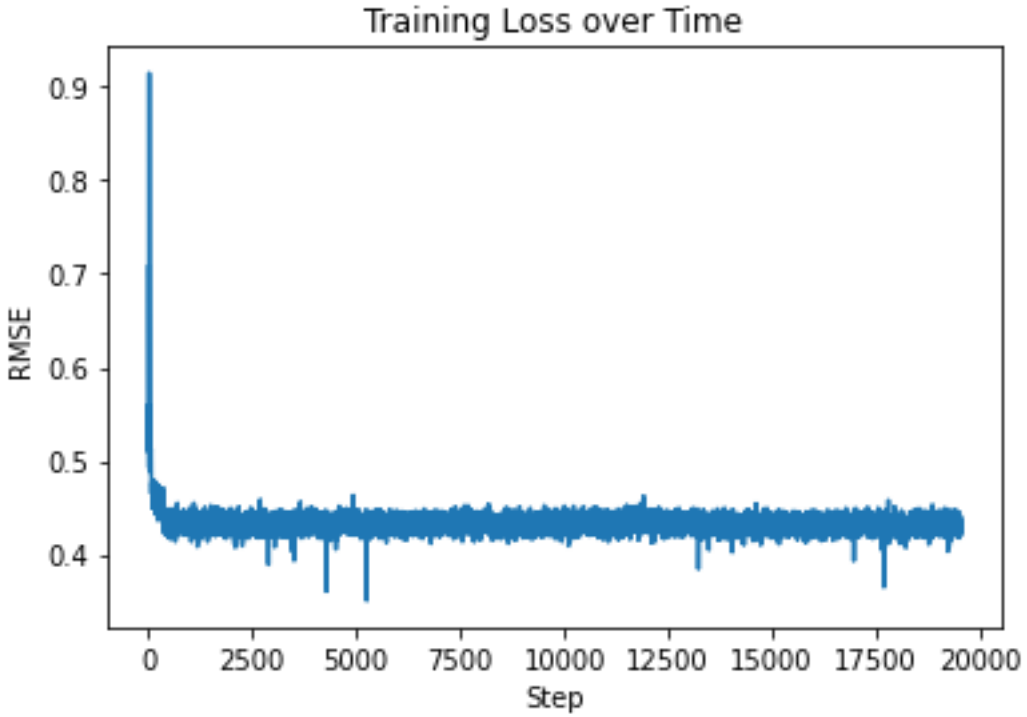
Since official code is old and I couldn’t run it, I use another implementation from SidechainNet tutorial.

The other implementation was using SidechainNet “for dataset” and using PyTorch.

In this implementation we can use either RMSE or dRMSD, but dRMSD takes too long for training (3:30 hrs. for only 1 epoch) and due to limitation in using Colab GPU, it was hard to use it so I use RMSE.

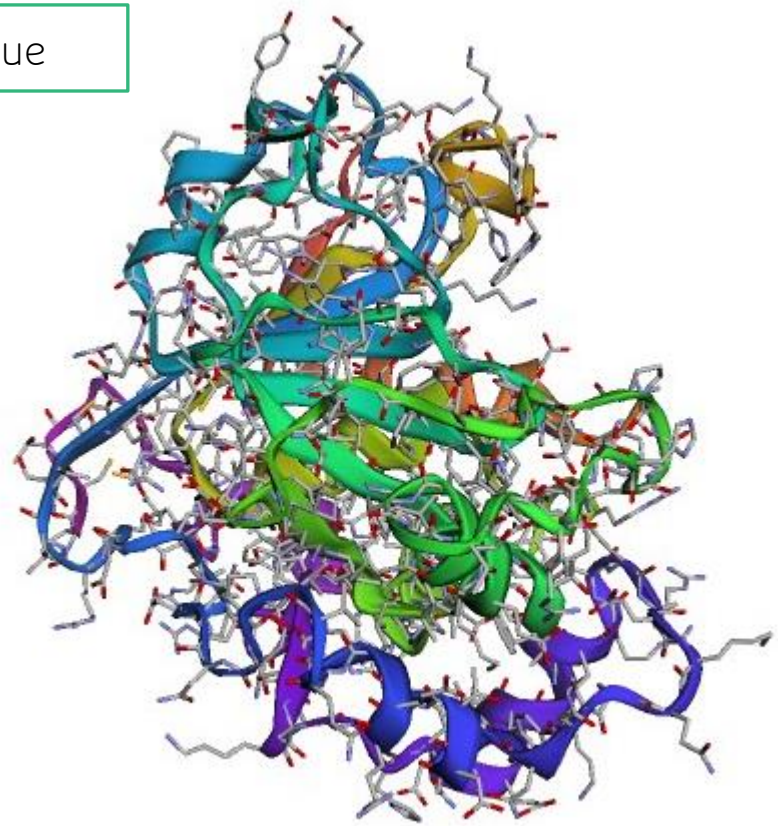
RGN – Run for 25 Epochs Result

Sequences → Angles

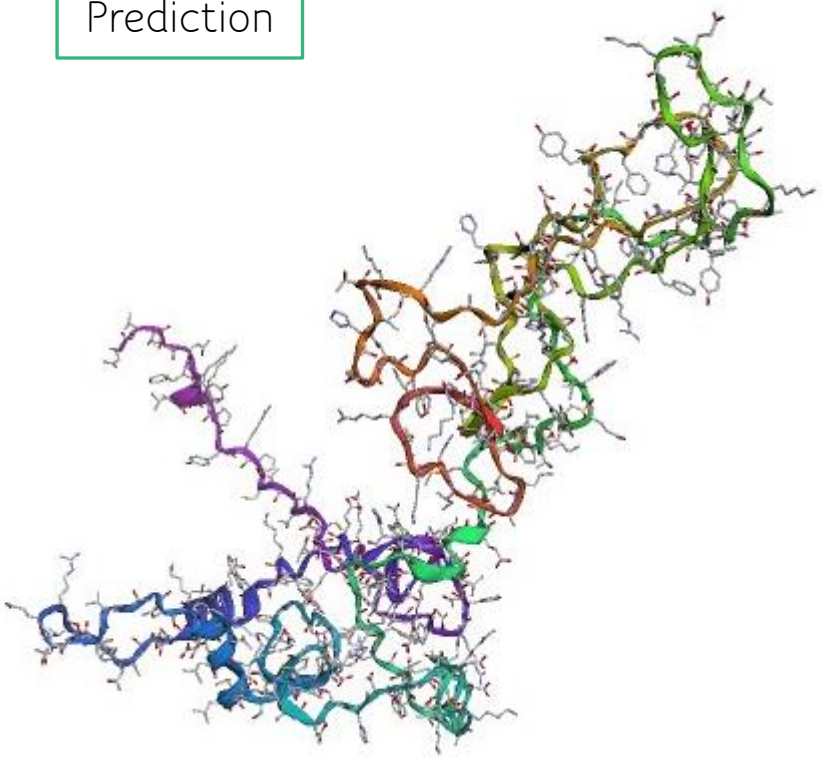


RGN – Run for 25 Epochs Result

True

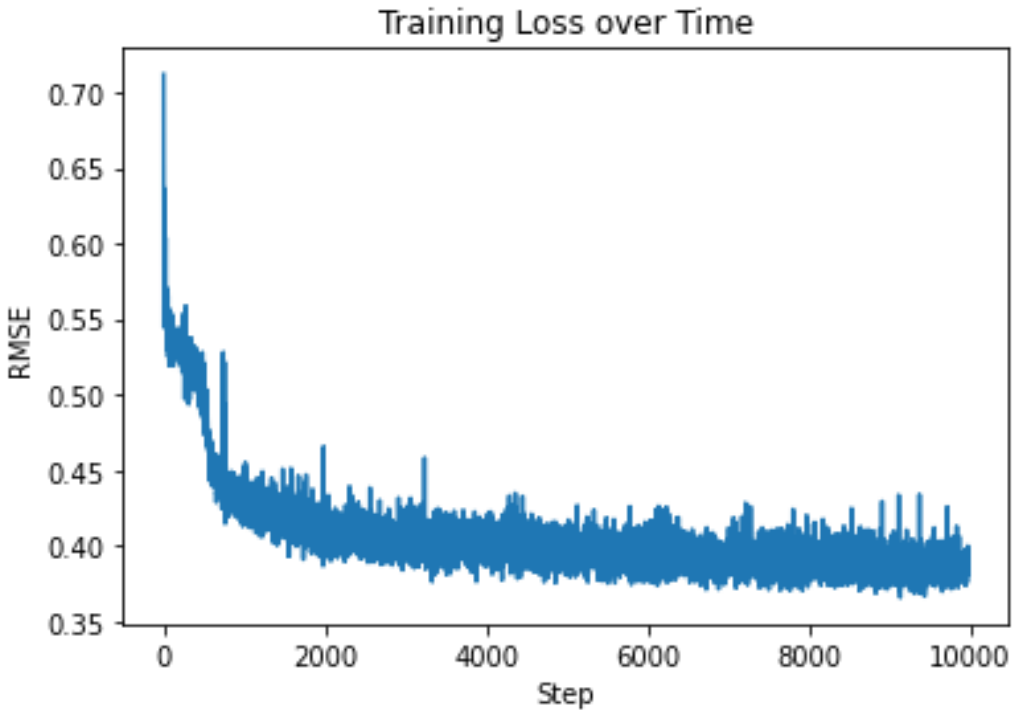


Prediction



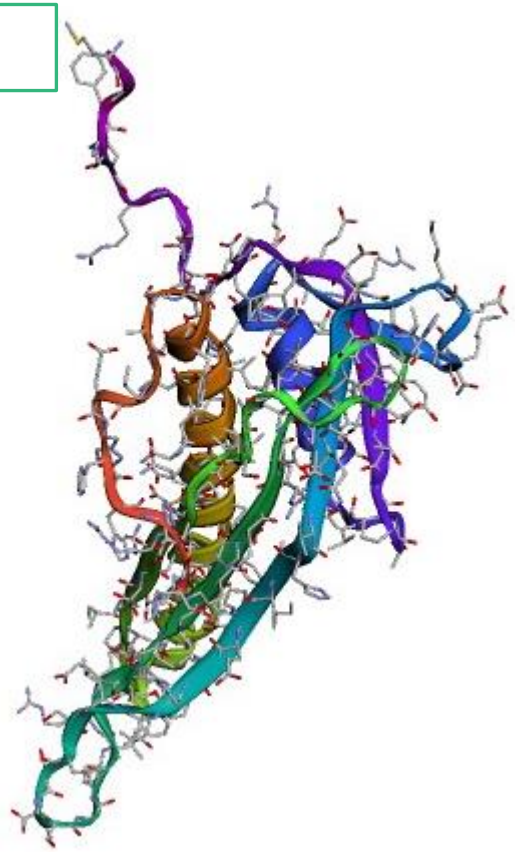
RGN – Run for 10 Epochs Result

(Sequences, PSSMs, Secondary Structures, and Information Content) → Angles

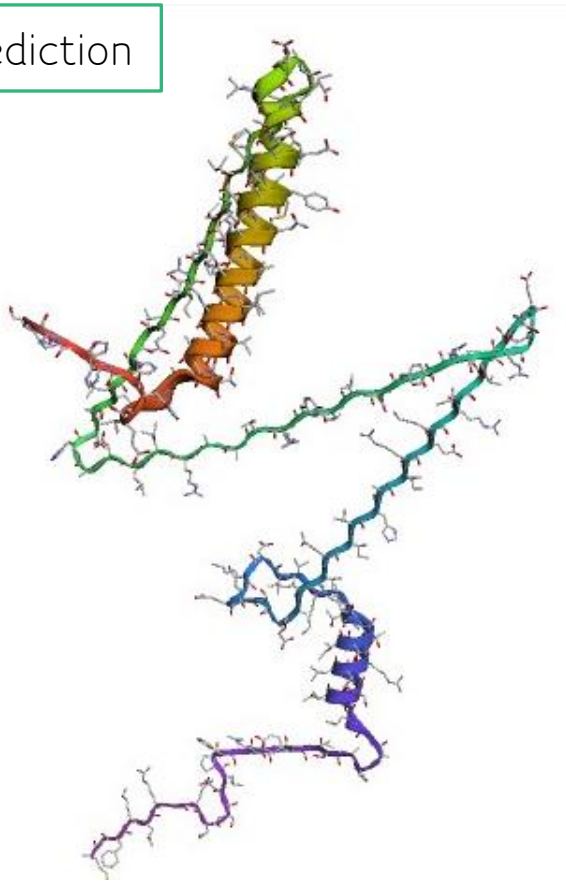


RGN – Run for 25 Epochs Result

True



Prediction



UTGN – Implementation

Its official code was a modification on the RGN official code. So it is also using ProteinNet “for dataset” and implemented using TensorFlow, with same loss used (dRMSD).

This implementation is full of bugs and after some modification it still didn’t work correctly.

I tried to implement it from the beginning using PyTorch but unfortunately it still not working correctly.

Notebook

Final colab notebook, contains all work with PyTorch Models
https://colab.research.google.com/drive/1TbjKAcMeHbKz_mE5GnnVO53mJ5rVUj9z?usp=sharing

Old Colab notebook, contains trials made for working with official codes that uses TensorFlow
https://colab.research.google.com/drive/1PVHuGhD6pDsNCyW3g9WG7G_E2_lDQeBX?usp=sharing

Model weights and result of training
https://drive.google.com/drive/folders/1tjHLCoyx9Cp2bW4iVzQ-_WUKVg_6lCzi?usp=sharing

References

RGN paper: https://www.cell.com/cell-systems/fulltext/S2405-4712(19)30076-6
RGN official code: https://github.com/aqlaboratory/rgn
UTGN paper: https://arxiv.org/abs/1908.00723
UTGN official code: https://github.com/JinLi711/UTGN
Sidechainnet: https://github.com/jonathanking/sidechainnet
ProteinNet: https://github.com/aqlaboratory/proteinnet
Some for implementing UTGN:
https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoderLayer.html
https://github.com/andreamad8/Universal-Transformer-Pytorch
https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/_downloads/transformer_tutorial.ipynb
https://colab.research.google.com/github/PytorchLightning/lightning-tutorials/blob/publication/.notebooks/course_UvA-DL/05-transformers-and-MH-attention.ipynb
https://pythonrepo.com/repo/minqukang-transformer-pytorch-python-deep-learning