User Manual

Adi Avni, Iris Schodl, Shlomit Gila, Avishai Elmakies, Noam Ben Sason

Introduction
Our program's goal is to predict an amino acids sequence from a 3D protein structure. We build three different networks -

- “vanilla” model - convolution based
- attention model - attention and convolution based
- multi headed attention model - multi headed attention and convolution based

How to predict a sequence from a pdb file using one of our model

Requirements
Please see the packages and versions requirements file in the git repository.
Please install the requirements using pip
Very important - tensorflow 2.8!

Input to the command line

```
python3 run.py [model type] [pdb file path] [--prinf_seq]
```

- Model type: N - “normal” model , A - attention model, AMN - multi headed attention model)
- pdb file path: a PDB file path of which you want to predict the sequence of. The pdb file should be with all unknown amino acid (the model does not use them anyway)
- prinf_seq :a flag to give if you want to print the predicted sequence to the screen
- You can also write -H for help

Output
If you chose the flag of print_seq the predicted sequence will be printed to the screen.
There will also be a fasta file of the predicted sequence and a PDB file of the final predicted model in a new directory called 'predicted'.

How to run our training

Requirements
Please see the packages and versions requirements file in the git repository.
Very important - tensorflow 2.8!

you will also need to install and sign in to 'weights and biases' and log in from the command line before running the file.

installation command: pip install wandb
Log in command: wandb login

In the 'train_network_wandb.py' file, please change the ENTITY constant to your wandb user name to see the training result in your account.

Then, run the file 'train_network_wandb.py'.

Additional files and scripts

Network
encoder .py - the vanilla encoder model
encoder_attn.py - the attention encoder model
encoder_mhattn.py - the multi headed attention encoder model
decoder - the decoder model
train_network_wandb.py - the training process

For more information see project report.

RMSD calculations
Var_RMSD.py - calculates the RMSD of a given PDB file compared to the original PDB file (should also be given)
calc_RMSD.py - calculates all the RMSD scores for the three models and all the test samples. Outputs a csv containing all the RMSD scores for all the models and all the test samples.

Plot scripts

- **scatter_plt_seq_loss_vs_mse_loss/create_loss_scatter_plot.py** - plots the loss of the sequence (CCE) versus the loss of the structure (MSE) in our three models, for all proteins in the test set.
- **sequence_logo_files/logo_maker.py** - plots a sequence logo of predicted sequence of a given structure (PDB code - 1dlf) and the original sequence. Each prediction is from a different model. The x axis represents the different positions of the sequence (from 1 to 140) and the y axis represents the counts per amino acids character. The code was used on three different models - multi headed attention model, original model, attention model. For each model, we created a csv with probabilities for each position and each possible amino acids. This csv is read into the variable - prob_df and then is used to create the sequence logo figure using the logomaker library.
- **boxplot_rmsd.R** - plots 9 boxplots each representing the RMSD between original and predicted CDR region (CDR1, CDR2, CDR3) using different autoencoder architectures. It reads three different csv files, one for each different

model - multi headed attention model, original model, attention model. Each csv lists the RMSD for each CDR region.

- model_accuracy.py - a script that calculates the accuracy for sequence prediction. Calcs accuracy for total sequence and also for CDR’s