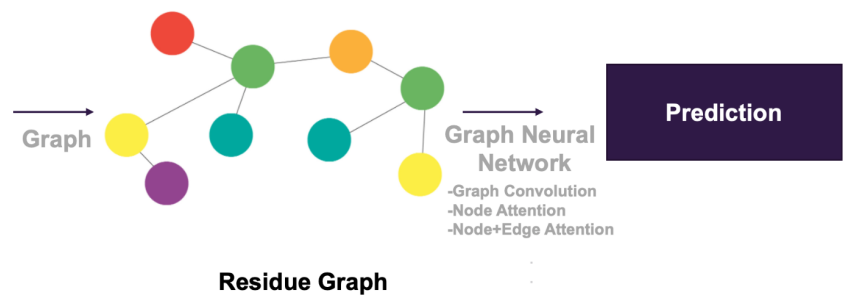


INSTRUCTION FOR PROTEIN-PROTEIN COMPLEXES QUALITY ASSESSMENT USING GRAPH NEURAL NETWORKS

Yanis Tazi



I - INITIAL SETUP :

Step 1: Fork and clone the repository

- Option 1 (recommended) :

- 1. Fork repository from:
https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes
- 2. Clone repository from your Github on the cluster :

```
git clone https://github.com/<YOUR_USER_NAME>/Graph_Neural_Net_Protein_Scoring.git
```

- Option 2 (not recommended):

```
scp -r  
<YOUR_CLUSTER_NAME>@192.69.102.112:/bgfs01/insite/yanis.tazi/Scoring_Project/  
<YOUR_GRAPH_FOLDER/>
```

Step 2 : Create a dedicated environment compatible with the cluster

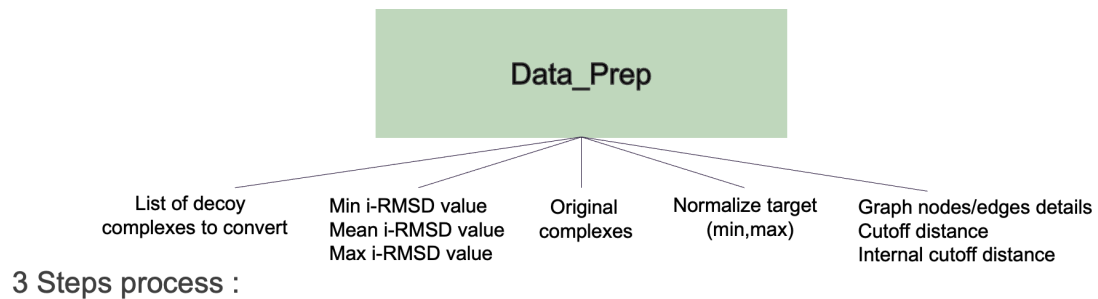
(after step 1 , you should be able to find environment_graph_predictions.yml)

```
conda env create --name <YOUR_ENV_NAME> --file=environment_graph_predictions.yml
```

Step 3 : Activate the dedicated environment

```
conda activate <YOUR_ENV_NAME>
```

II - DATA PREPARATION : FROM PDB FILES TO RESIDUE GRAPHS



1.create_dict_selected_pdb():

create a dictionary with the selected pdb files that follow the above criteria and their corresponding i-RMSD .

Example : {'1JTD_it1_complex_329.pdb': 14.98, '1JTD_it1_complex_359.pdb': 11.385}

2.prepare_pdb_selected_data(): Prepare pdb files and corresponding output (normalized)

3.prepare_hdf5_graphs(): Create the hdf5 graphs corresponding to the inputs



Step 1 (*Optional*) : Get the complete DB5 docking dataset (~600K pdb files)

- Option 1 (recommended) : <https://data.sbgrid.org/dataset/131/>

```
rsync -av rsync://data.sbgrid.org/10.15785/SBGRID/131
```

- Option 2 (not recommended) :

Make a copy from my directory to your new pdb data folder

```
cp -r /bgfs01/insite02/yanis.tazi/data/graph_project_data/131/  
<YOUR_INITIAL_PDB_DATA_FOLDER>/
```

Step 2 (Optional) : Residue Graph Preparation from pdb files :

Fully Automated data preparation pipeline that creates balanced graph datasets from PDB protein complexes and decoys files. It is a 3 step process that:

1. creates a balance dataset of pdb files based on the [min,mean,max] i-RMSD values provided, and on the minimum number of decoys per stage.
2. Prepare a folder with the selected pdb files and their corresponding i-RMSD and input complexes if provided.
3. Create the residue graphs with the nodes/edges based on the node cutoff distance , edges internal cutoff distance with the nodes and edges features provided , creates the normalized i-RMSD and can also add input complexes

Tutorial :

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/blob/main/Graph_Project/Tutorials/Data_Prep_Tutorial.ipynb

OR

Also available in : Graph_Project/Tutorial/Data_Prep_Tutorial.ipynb

Data preparation functions with detailed explanations :

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/blob/main/Graph_Project/Model_Functions/Data_Preparation.py

OR

Also available in : Graph_Project/Model_Functions/Data_Preparation.py

Additional Information :

The user can directly make a copy of all the data from pdb files to .hdf5 graphs and their replicates.

The copy will of course be dependent on the parameters I chose to create the graphs to include i-RMSD (min,mean,max), nodes/edges features and node cutoff distance as well as edge internal cutoff distance,...

```
cp -r /bgfs01/insite02/yanis.tazi/data/graph_project_data/  
<YOUR_INITIAL_DATA_FOLDER>/
```

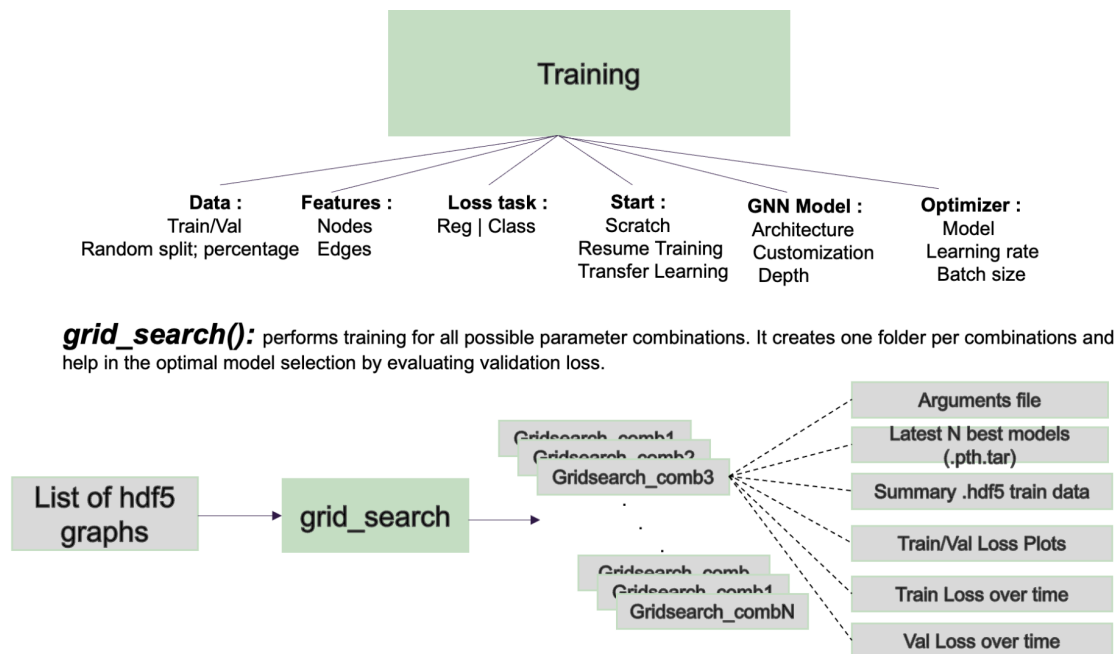
Dataset balanced around 10 A : (you will get multiple copies of the graph so that you can use them for gridsearch without interference)

```
cp -r /bgfs01/insite02/yanis.tazi/data/graph_project_data/hdf5_replicates/  
<YOUR_HDF5_target_10_replicates>/
```

Dataset balanced around 5 A : (you will get multiple copies of the graph so that you can use them for gridsearch without interference)

```
cp -r /bgfs01/insite02/yanis.tazi/data/graph_project_data/hdf5_target_5_replicates/  
<YOUR_HDF5_target_5_replicates>/
```

III - TRAINING OF THE GRAPH NEURAL NETWORK



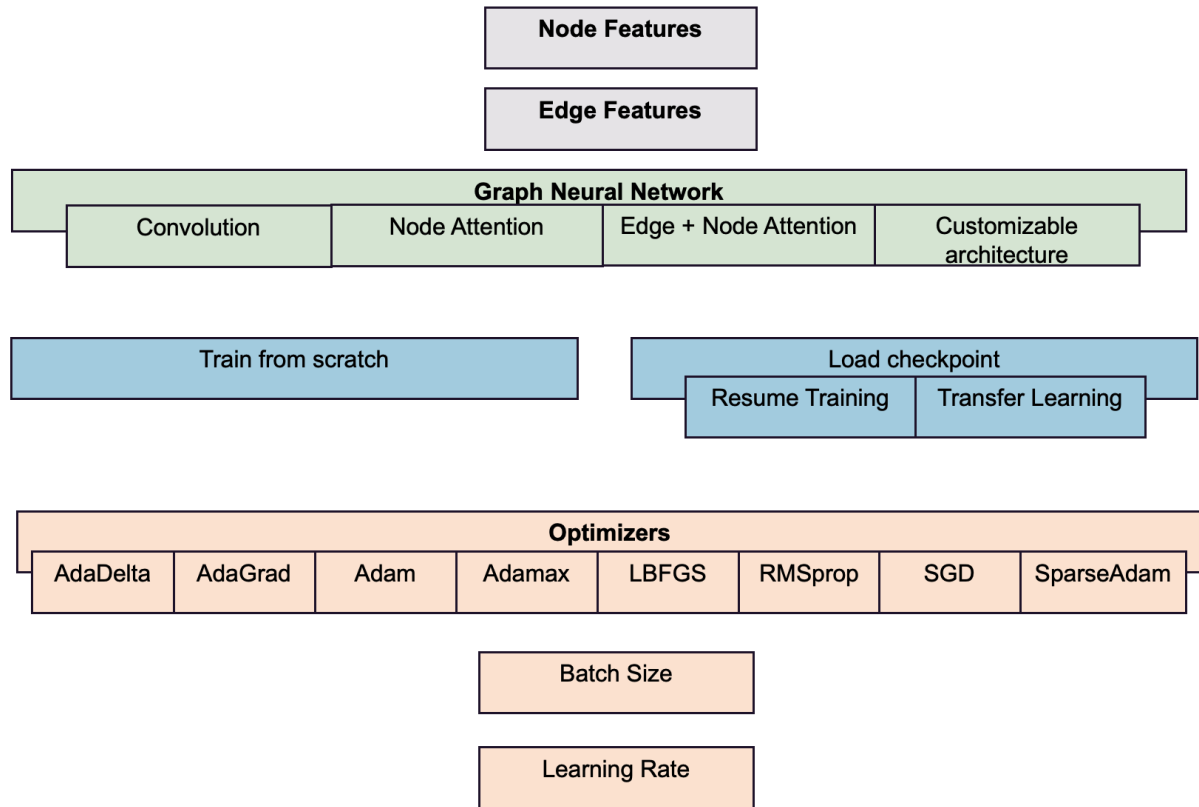
Tutorial :

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/blob/main/Graph_Project/Tutorials/Model_Training_Tutorial.ipynb

OR

Also available in : `Graph_Project/Tutorial/Model_Training_Tutorial.ipynb`

Follow Training tutorial that performs gridsearch and optimal parameter combination search . Users can train the model from scratch , load a preexisting trained model to resume training or perform transfer learning, they can choose among different existing/new customized model (details below), chose the edges/nodes features, optimizers, learning rates, ... The gridsearch will also save the train/val loss plots over time , the N best saved models, the train/val loss values over time, provide a file with all the arguments to keep track of the parameters used and a summary of the training data over time with the predicted values every e epochs where e is also a hyperparameter.



*For details about all the hyperparameters and the gridsearch, please consult the documentation:
(you will find all the information and detailed explanation there, so please read it)*

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/blob/main/Graph_Project/Model_Functions/Model_Training.py

OR

Also available in : Graph_Project/Model_Functions/Model_Training.py

Customizable architecture :

Users can either use one of the preexisting architecture or create a new class with their own architecture that they can just import before training the model,

1. Chose from one of the existing architectures : GINet, EGAT_Net, FoutNet available in [{ginet.py,foutnet.py,EGAT_GNN.py}](https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/tree/main/DeepRank-GNN/deeprank_gnn)

OR

Also available in :

Graph_Project/DeepRank-GNN/deeprank_gnn/{ginet.py,foutnet.py,EGAT_GNN.py}

2. Create a customized architecture such as the 2 examples provided :

(DO NOT FORGET TO IMPORT THEM)

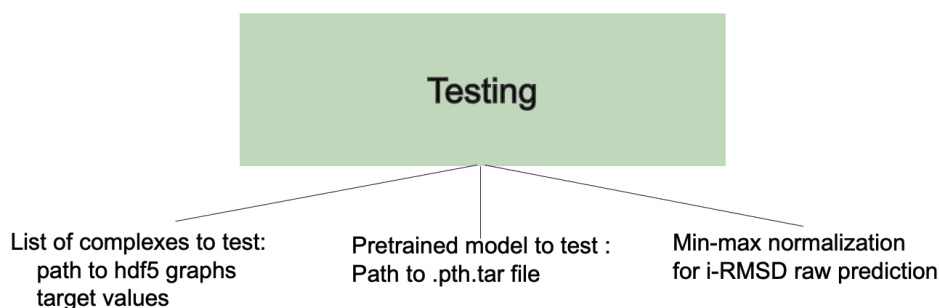
GINet_deeper and GINet_deeper_attention_fc available in

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/tree/main/DeepRank-GNN/deeprank_gnn/ginet.py

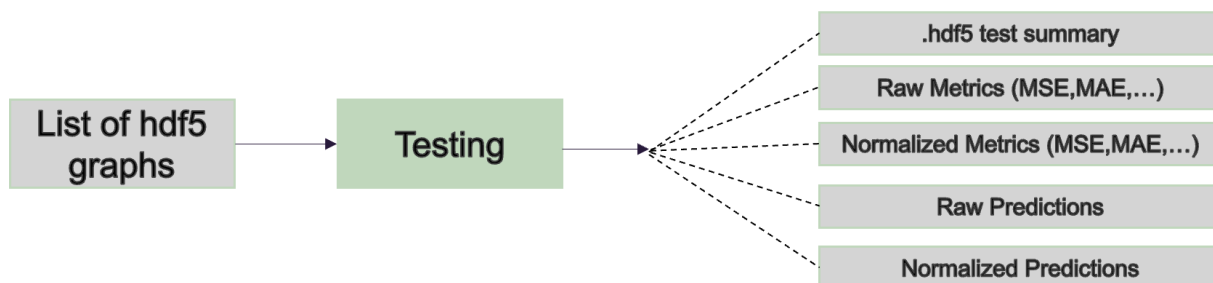
OR

Also available in : Graph_Project/DeepRank-GNN/deeprank_gnn/ginet.py

IV - TESTING



model_testing(): test any trained model and return summary , predictions and metrics



Tutorial :

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/blob/main/Graph_Project/Tutorials/Model_Testing_Tutorial.ipynb

OR

Also available in : Graph_Project/Tutorial/Model_Testing_Tutorial.ipynb

Follow Testing tutorial (part 1) that tests the model on new graphs data. You just need to provide the path to the checkpoint of the model you want to test , specify the architecture that was used , specify whether or not you want the raw predicted values by giving the min-max normalization values and provide the path to the test .hdf5 data. The model will then return a summary of the data (predictions and targets), lots of raw and normalized metric results (MAE, R^2 , MSE, RMSE,...) , the raw and normalized predicted values.

For details about all the arguments, please consult the documentation:

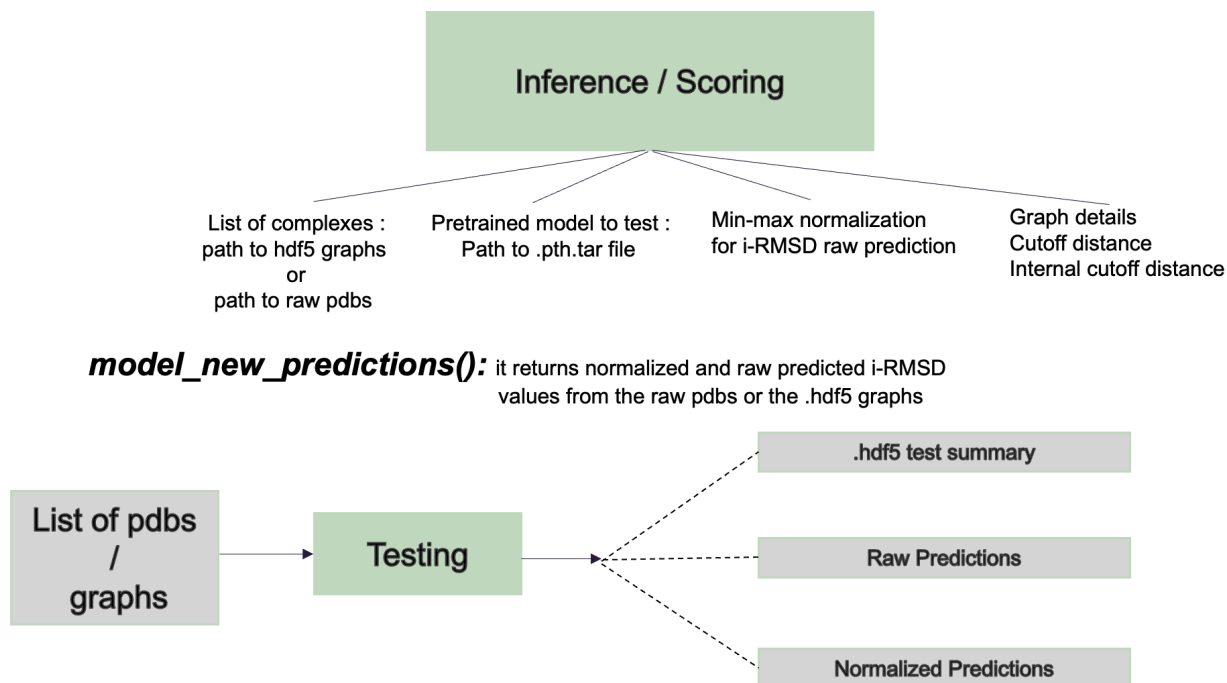
(you will find all the information and detailed explanation there, so please read it)

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/blob/main/Graph_Project/Model_Functions/Model_Testing_Predictions.py

OR

Also available in : Graph_Project/[Model_Functions/Model_Testing_Predictions.py](#)

V - INFERENCE ON NEW PDB STRUCTURE



Follow Testing tutorial (part 2) that takes raw pdb files , convert them into graphs (or can take graphs directly) and predict the i-RMSD. It returns the raw and normalized (if normalization is provided) predictions.

For details about all the arguments, please consult the documentation:

(you will find all the information and detailed explanation there, so please read it)

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/blob/main/Graph_Project/Model_Functions/Model_Testing_Predictions.py

OR

Also available in : Graph_Project/Model_Functions/Model_Testing_Predictions.py

VI - INITIAL RESULTS

The initial results were promising and we demonstrated that a model with edge and node information on a dataset that has been provided independently (11K pdb files) can reach a precision of 2Å.

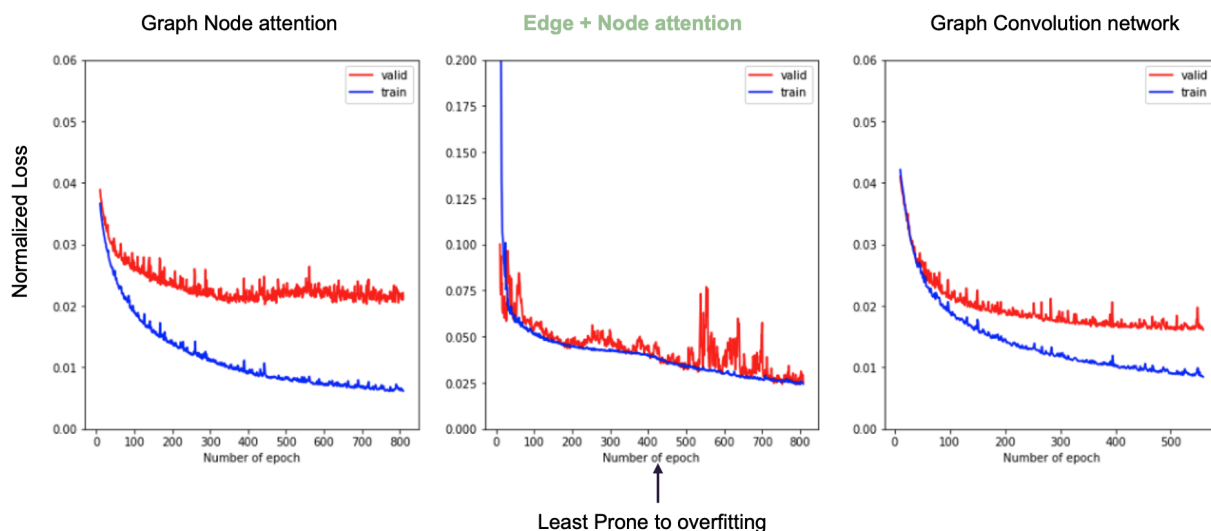
Results Analysis available :

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/blob/main/Graph_Project/Results_Analysis.ipynb

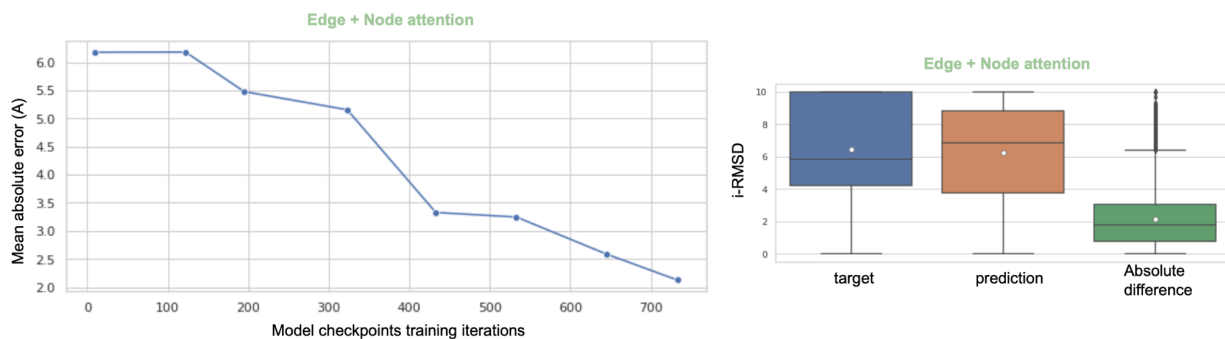
OR

Also available in : Graph_Project/Results_Analysis.ipynb

Comparison of three architectures



Results on Testing set ~ 11K pdb files (Training+validation on 15K pdb files from 52 complexes)



VII - SLURM JOBS ON CLUSTER

A slurm job for each part of the pipeline (preparation of the data, training/gridsearch and testing) with its corresponding python function for :

1. Data Preparation

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/tree/main/example_slurm_jobs_cluster/prepare_data

OR

example_slurm_jobs_cluster/prepare_data/

(Modify the jobs based on your data preparation and the customizable parameters you chose)

Launch the job :

```
sbatch prepare_hdf5_data-slurm.sh
```

2. Gridsearch training :

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/tree/main/example_slurm_jobs_cluster/training_gridsearch_irmsd_5/

OR

example_slurm_jobs_cluster/training_gridsearch_irmsd_5/

(Modify the jobs based on the gridsearch customizable parameters you chose)

Launch the job :

```
sbatch train_gridsearch-slurm.sh
```

3. Testing on new data:

https://github.com/yanistazi/Graph_Neural_Net_Protein-Protein-Complexes/blob/main/example_slurm_jobs_cluster/Testing_model_slurms/

OR

example_slurm_jobs_cluster/Testing_model_slurms/

(Modify the jobs based on the model and the data you want to test)

Launch the job:

```
sbatch Testing_models-slurm.sh
```