Internship Report

Subject : Train graph neural network to predict residue conformation changes in a protein structure.

Nour Ladhari

May - July 2023

Grenoble INP Ensimag | UGA

Laboratoire Jean Kuntzmann
Mathématiques appliquées - Informatique

Introduction

Prepare protein structure-based data suitable for training graph neural networks to predict what residues in a protein structure are likely to change conformation when pocket opening happens (apo-to-holo transition). Here, 'apo' means a protein with at least one inactive and partially or completely closed pocket, 'holo' means a protein with at least one pocket where a ligand can fit.

Activities and tasks

This task demanded multiple steps, starting from data preparation till training the model and test evaluation.

Input Data Preparation :

It consists of the first Step which is CSV file preparation. We treated two different type of CSV files :
1) nodes CSV file :
The preparation of nodes file consists of the **normalization** of the nodes. We start by selecting the attributes nodes to normalize and then we apply the following formula :

$$z\_score = \frac{x - node\_mean}{node\_std} \tag{1}$$

for which the node_mean and std_mean represent respectively the mean and standard deviation of each selected row attributes of the CSV node file
2) links CSV file : For this type of file, the preparation consisted on :
a) Link-Normalization :

We select the we used the following formula :

$$z\_score = \frac{x - link\_mean}{link\_std} \tag{2}$$

b) Adding bidirectional connections :
if there is (i -> j) link, there should also be (j -> i) link with the same attributes

c) Adding self-connections : there should be (i -> i) link with appropriate attributes for every node i
d) Added **is_self** attribute : is_self is a self-link indicator

$$\begin{cases} is\_self = 0 \; for\, a\, normal\, link \\ is\_self = 1 \; for\, a\, self\_link \end{cases}$$

Graph representation :

To train the model, we need to convert the model input and output to Convert the graph data into a suitable format for GNNs which is the Tensor format. Thus, we created a **PocketDataset** class that satisfies this task.

Model Architecture selection :

We selected the GATv2 architecture for our model.

Define the loss function :

In consequence of having imbalanced data, we used a weighted loss_function. We applied weights to the absolute mean error using this function :

```python
def weighted_mae_loss(y_pred, y_true, weight_factor, gt_std, gt_mean):
    absolute_errors = torch.abs(y_pred - y_true)
    weights = 1 + weight_factor * (y_true*gt_std+gt_mean)
    weighted_errors = weights * absolute_errors
    loss = torch.sum(weighted_errors) / torch.sum(weights)
    return loss
```

Train process :

In this process, we iterate over the training data in mini-batches of size 10. For each batch, perform the following steps :
Forward pass : Pass the input data through the GNN model to obtain predictions.
Compute loss : Calculate the loss between the predictions and the ground

truth labels using the loss function explained before.
Backward pass : Compute gradients with respect to the model parameters using backpropagation.

for each model the loss is computed as :

$$\frac{weighted\_mae\_loss \cdot node\_size\_in\_batch}{total\_graphs\_size} \tag{3}$$

Each model and train_loss are then saved into a file.

Evaluation process :

Using validation set : For each output_trained_model we compute the validation_loss and save it in a file.

Selecting the best model :

We run some model experiments to change different hyper-parameters, in order to select the best trainable model.
1) Loss_weight values : we trained model using different weight_loss values : 2, 4 and 8.
2) Dropout_values : 0.25 and 0.5.
3) Learning_rate values : 0.0001 and 0.001.
4) Model configurations : we trained a model with fewer layers.
5) Model architecture : we used **TransformerConv**

results :

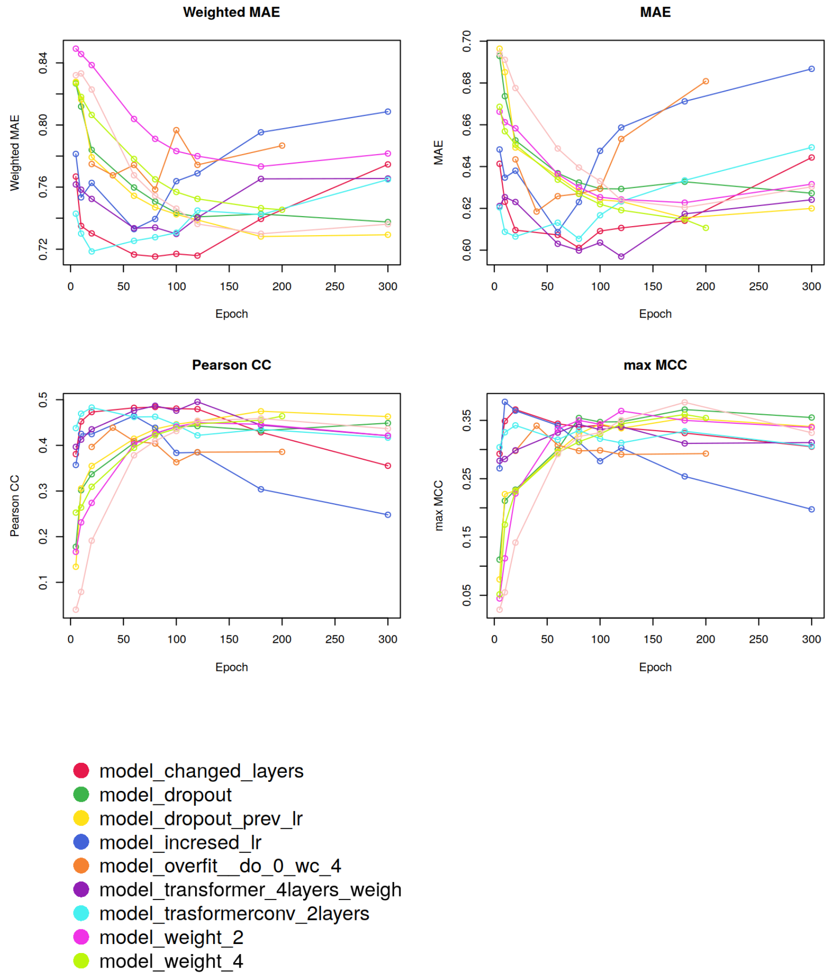

FIGURE 1 – result of modified models

Given the result curves given above, we selected the **model_changed_layers** which is the model which just 2 convolution layers.