Assignment #7 – Transformer

https://github.com/Lemorita95/1FA006/tree/main/7_transformer

Written summary

Also available in https://github.com/Lemorita95/1FA006/blob/main/7_transformer/README.md

approach

1. hyperparameters:

   batch_size = 64
   learning_rate = 0.8e-4
   num_epochs = 50
   D = 128
   n_heads = 2
   dim_feedforward = 64
   n_layers = 2

2. Load data with awkward and normalize features and labels with get_normaized_data();
3. Data is normalized through z-score normalization;
4. Train, validation and test dataset is created with PyTorch Dataloader and a custom collate function collate_fn_transformer();
5. Training loss function is defined as MSE;
6. Gives the user the possibility to load a model (if found at models/model.pth);
7. Default model is defined with (n_layers) TransformerEncoderLayer layers with (n_head) attention heads + residual connection + MLP + residual connectiona MLP output layer each a final Linear layer is used to get the output dimension;
8. Model TransformerEncoder.forward() pass: 1) input embedding with a Linear layer that outputs tensor at dimension (batch_size x N, D) where N is dependent on data, 2) padding and masking;
9. Train and validate model train_validade_model();
10. Test model test_model();
11. Model outputs the predicted labels TransformerEncoder.forward();
12. Denormalize predictions and true values denormalize();
13. Plot scatter of averages and residuals;

results

1. the training loop for the hyperparameters took around 12 second per epoch;
2. the training had convergence;
3. similar performance of GNN for this dataset;

challenges

1. the simplest transformer with 2 attention heads, 2 layers ("BASE MODEL") performed very similar to GNN as in results, in losses (Transformer: 0.0892 x GNN: 0.0883), normal distribution of residuals as in computation time;
2. changing attention head 2 -> 8 from base model did not improved much (training loss: 0.0892 -> 0.0819);
3. changing number of layers 2 -> 3 from base model did not improved much (training loss: 0.0892 -> 0.0823);
4. changing number of layers 2 -> 6 from base model did not improved much (training loss: 0.0892 -> 0.0839) but slower down (training time per epoch: 12s -> 18s) with early stop at 50% of num_epochs;
5. best result so far by changind dim_feedforward 32 -> 64 (train loss: 0.0892 -> 0.0806, training time per epoch: 11.8s -> 12s);
6. final model was with the result of 5.

Result plots

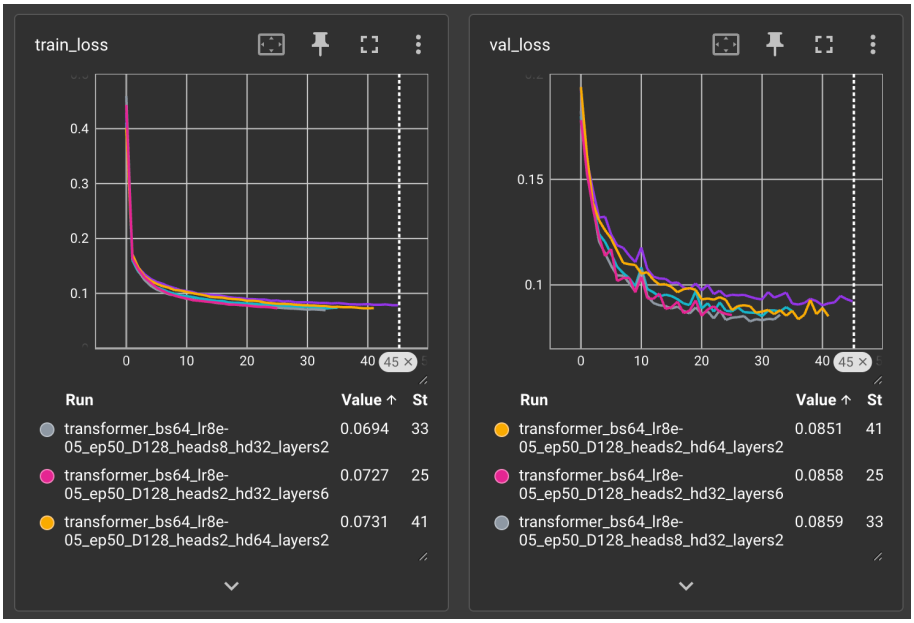

Figure 1: Train and validation loss for different models and hyperparameters.

From Figure 1 we see a decrease as the model goes through the epochs. This experiment used $Batch\ size = 50, Learning\ Rate = 8 \cdot 10^{-5}, Epochs = 50$, although all models had an early stop and did not complete the whole 50 epochs. The best models were the ones achieved when changing the number of layers and the feedforward dimension.

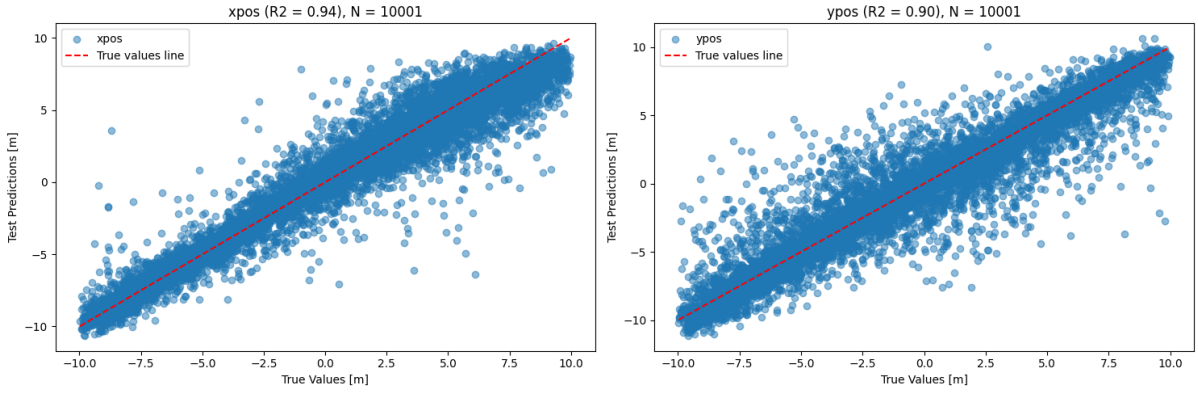

Figure 2: Scatter plot of predicted means.

From Figure 2, we can observe that, overall, the data follows the true values line trend but with significant variance.

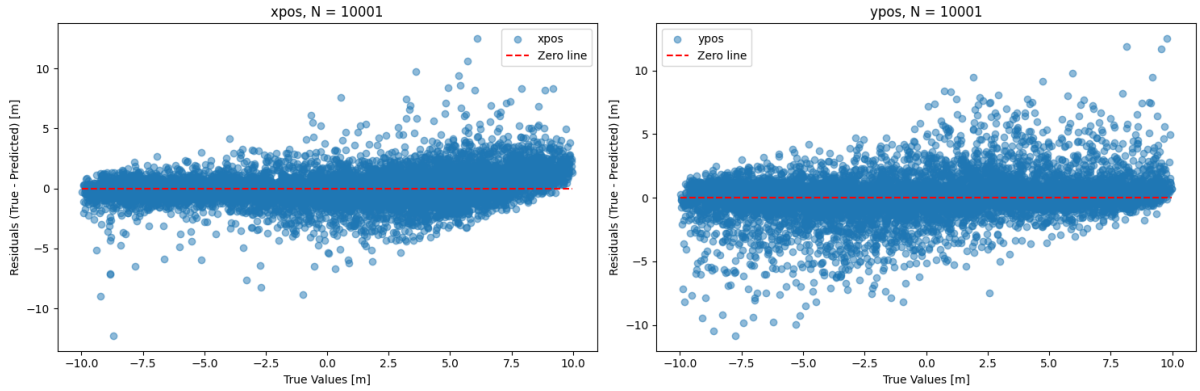

Figure 3: Scatter plot of predicted means residuals.

Figure 3 show a slight correlation of residuals and true values for `ypos` and some larger residuals are observed in the inferior and superior limit of the true values axis.

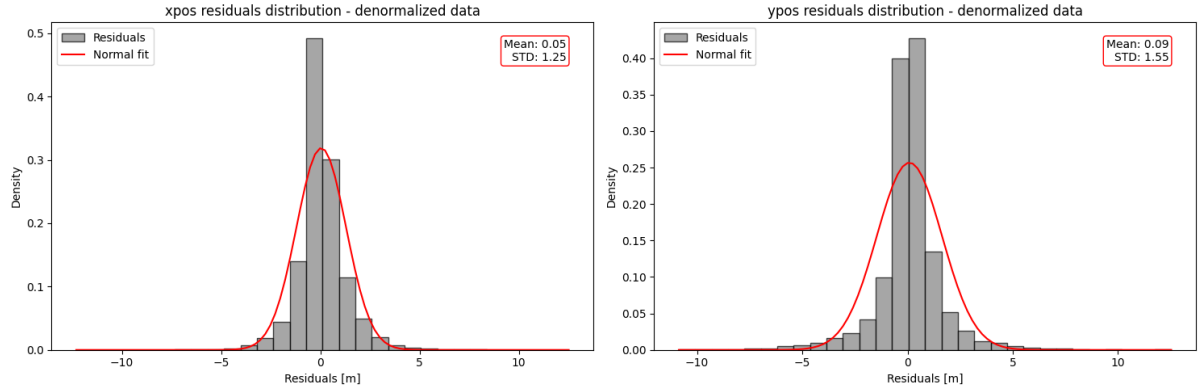

Figure 4: Distribution of predicted means residuals for Transformer model.

Figure 4 shows a good performance on the model's predictions regarding bias, when comparing the residual distribution to a normal distribution. When comparing to the GNN model on the same dataset (Figure 5), not much difference can be seen, except a slight reduction in the uncertainty for the transformer model.

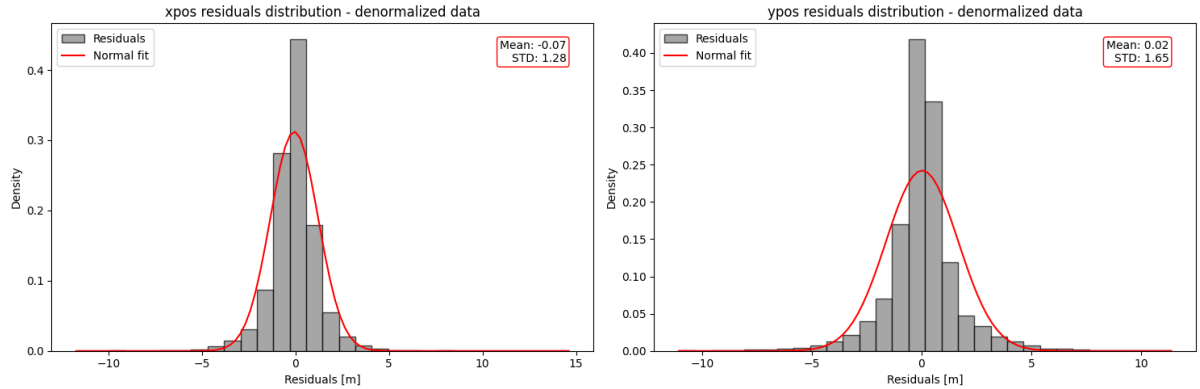

Figure 5: GNN predicted means residuals distribution