Multivariate Time Series Forecasting with Graph Neural Networks

Natalie Koh, Zachary Laswick, Daiwei Shen

March 15, 2023

0.0.1 Introduction

For this project, we explored the use of graph neural network (GNN) architectures for multivariate time series forecasting. Specifically, we wanted to use state of the art time series forecasting GNNs to predict the future sensoring readings for a movement given the historical readings. GNNs, which have only recently gained traction within the deep learning field, are uniquely suited for solving time series forecasting problems that require the consideration of spatial dependencies between nodes because they can capture complex spatial and temporal relationships between objects.

We had initially set out to compare two state-of-the-art GNNs — Recurrent Graph Evolution Neural Network (ReGENN) and ReGENN combined with a pre-training model known as STEP, as outlined in our project proposal. STEP (STGNN is Enhanced by a scalable time series Pre-training model) is a recently developed pre-processing framework that involves pre-training a model to learn temporal patterns from long-term history time series. This unsupervised pre-training model generates segment-level representations using a transformer-like architecture, which can be fed as inputs into a downstream GNN to boost performance. In the original paper that proposed STEP, the authors used their pre-training model with Graph WaveNet, a GNN that uses graph convolution to learn hidden spatial dependencies automatically from data. Both STEP and Graph WaveNet were originally formulated to predict traffic flow on complex road networks.

As we soon found out after starting on the project, the pairing of STEP with a downstream GNN requires some pretty significant changes to the downstream GNN in order to fuse the two models (see section 3.2 of STEP paper). Due to time constraints and the complexity of implementation, we decided to do away with ReGENN and focus on the implementation of STEP in combination with Graph WaveNet instead. We thus aimed to assess the performance of STEP with Graph WaveNet against using Graph WaveNet alone on two novel datasets that these architectures were not benchmarked on.

Given the eventual difficulties encountered in getting Graph WaveNet to work with our datasets (explained below), we have also included an implemention of a simple graph convolutional network (GCN) with a LSTM layer in Keras to also compare against the performance of STEP with Graph WaveNet.

Ultimately, we quantified the performance of these architectures by examining the ability of these GNNs to **forecast a single time series given the history of a multivariate time series dataset**.

All code for this project can be found at our GitHub repository.

0.0.2 Datasets

We tested the above GNNs on two physiological multivariate time series datasets.

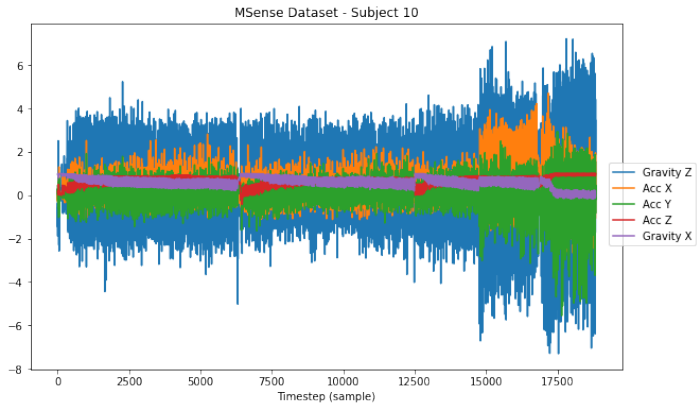
The first is the MHEALTH (Mobile HEALTH) dataset from UC Irvine, which comprises body motion and vital signs recordings for 10 volunteers of diverse profile while performing several physical activities. Sensors placed on the subject's chest, right wrist and left ankle were used to measure the motion experienced by diverse body parts – namely, acceleration, rate of turn and magnetic field orientation.

The second is the MotionSense (MSense) dataset from Kaggle, which includes time series data generated by accelerometer and gyroscope sensors (attitude, gravity, userAcceleration, and rotationRate) collected from an iPhone 6s kept in the participant's front pocket. This data was collected from 24 participants who performed various activities such as walking, jogging, sitting, and standing.

In the traffic flow forecasting datasets that STEP and Graph WaveNet were tested on, multiple sensors on different roads captured the same information, providing a [*time, num_sensors, num_features*] dataset.

For our datasets, however, each sensor on a participant collected separate information from other sensors. We thus decided to work with data from a single randomly chosen subject from each dataset. For each of these subjects, we selected five features (giving us five time series). Hence, for both the MHealth and MSense datasets, we used as input into our GNNs a [*time, 1, num_features*] tensor, and data was split into training, validation, and test datasets using the ratios 0.6, 0.2, and 0.2 respectively. We fed into our models 50 history timesteps to forecast 50 future timesteps, and used a [*num_features, num_features*] identity matrix as the adjacency matrix because actual distances between sensors were not provided for either dataset.

Below we see the time series traces for a randomly chosen subject from the MHealth dataset, and the time series traces for another randomly chosen subject from the MSense dataset.

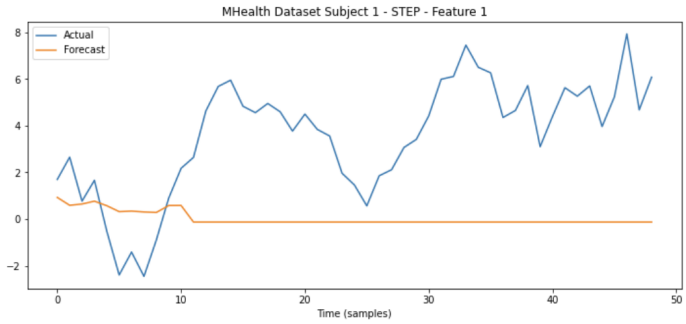


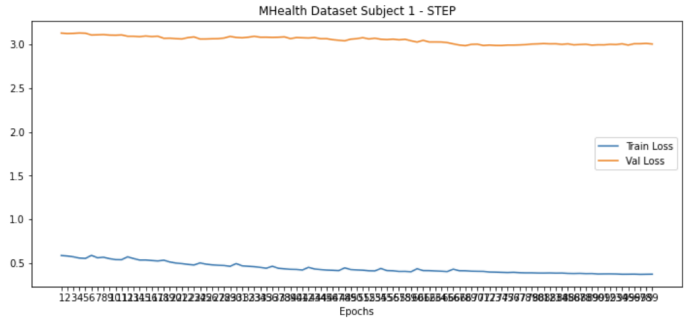
0.0.3 STEP Framework

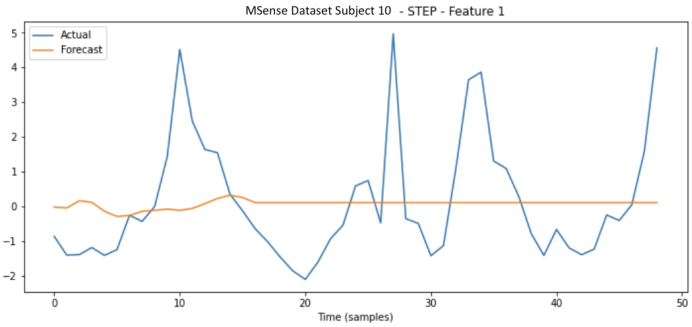
As introduced above, STEP is a recently developed framework that combines model pre-training with a GNN for time-series forecasting. For the STEP Framework, significant portions of the code that the authors made available on GitHub had to be adjusted to handle the data used for inputs – namely the number of nodes within the GNN, reshaping of the model's final output, and the input parameters to handle the historical and future sequence length of 50. Due to STEP's structure, the number of nodes within the model should be equal to the number of features, which for our data was 5. Additionally, the output of the STEP model contained a bug that resulted in the addition of a dimension with a shape of 1 into the output when used with new data, but this bug was able to be resolved by reshaping the output to reduce this dimension down. Finally, the initial parameters, including the input length, steps, patch size, and out channel, had to be adjusted to enable the model to take an input with time length of 50 and predict a time length of 50.

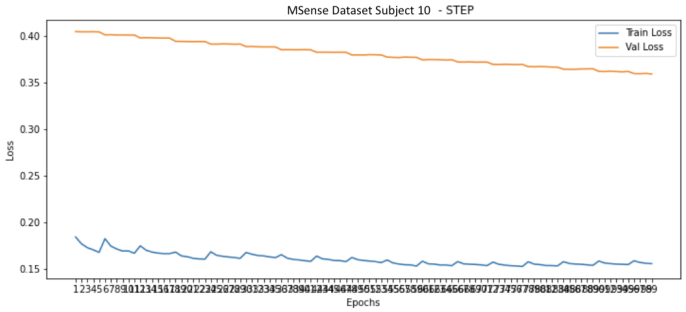
To train our modified STEP model, we used an ADAM optimizer with a learning rate of 0.005 and weight decay of 1e-5, with a dropout layer of 0.1 frequency and an encoder depth of 4 with decoder depth of 1. Using the model on our novel datasets, we were able to obtain a test loss of roughly 0.5 and 0.15 and a validation loss of roughly 3.0 and 0.4 for MHealth and MSense respectively, as seen in the figures below. This loss steadily decreased, but only very slowly, with increasing epochs.

However, since the prediction for the best loss barely resembles the true future values for a feature as seen in the images, the STEP model for this data did not perform as well as expected. Some of the challenges in using this model apart from the decreased performance than expected, include the inability of the model to handle single subject datasets, hardcoding by the authors, and the limited nuber of nodes used, since the node number equaled the feature number, which was only 5. The biggest challenge was the hardcoding done by the authors, where they had embedded numerous parameters within various files, each of which needed to be adjusted to handle new data and needed to be specified for each dataset. This hardcoding resulted in eight files that needed to be significantly modified for the model to work with our data.







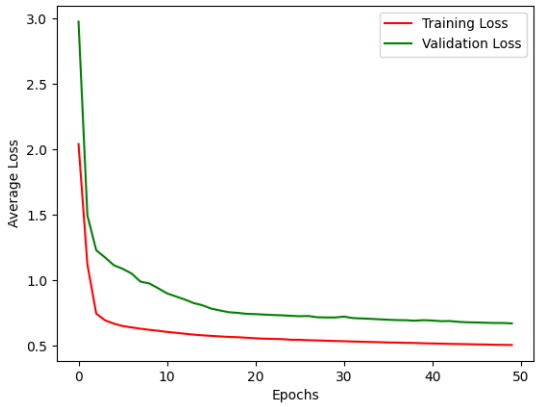


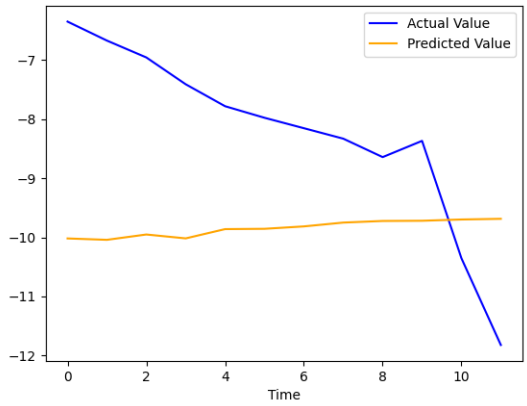
0.0.4 Graph WaveNet

Graph WaveNet is a GNN designed to address some of the shortcomings of traditional architectures, such as CNNs and RNNs, when it comes to time series forecasting. In particular, the architecture emphasizes the usage of an adjacency matrix, as well as convolution based on exponentially growing its kernel dimensions with respect to the number of layers. The authors of this neural network found that these components allowed it to learn long time-series sequences, as well as spatial-temporal dependencies, on a set of traffic data. Below, we test the model's performance on two medical datasets and visualize the results.

Below we illustrate the model's performance on the MHealth dataset. This model was trained using the following parameters: Learning rate = .0001; Number of nodes = 1; Input dim = 5; Seq length = 12; Batch size = 64; Epochs = 50; Drop out probability = .2. We used a sequence length 12 because we observed during our testing that this model produces particularly poor results when using a sequence length other than 12. Graph WaveNet was also not able to handed unequal history or forecast sequence lengths. We suspect that this is because the backend of the model was heavily hard-coded by the authors.

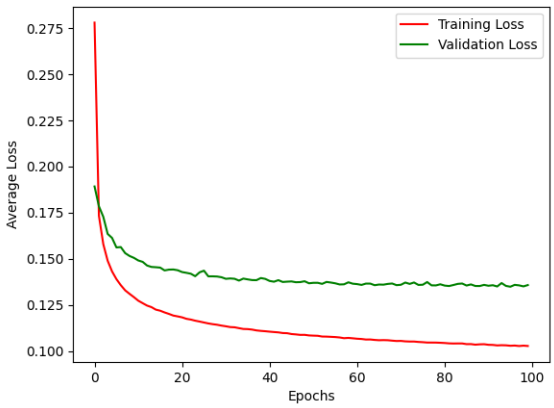
The time-series forecast performance on the features of subject 1 in our dataset is visualized as a single plot of 12 time steps.

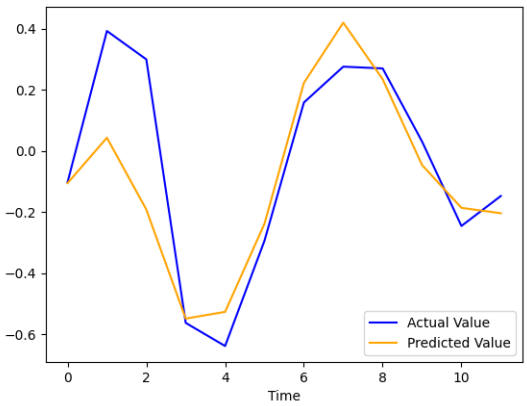




For the MSense dataset, we used the following parameters: Learning rate = .001; Number of nodes = 24; Input dim = 8; Seq length = 12; Batch size = 64; Epochs = 100; Drop out probability = .2.

The following plots illustrate the results of the model for the MSense dataset. The time-series forecast performance on the features of subject 10 in our dataset is visualized as a single plot of 12 time steps.





We find that the performance of Graph WaveNet on MSense is acceptable, but not spectacular. We do observe clear improvement over training epochs, but the gains are small, particularly when it comes to validation loss. We could not get the model to achieve loss lower than about .10. Despite these shortcomings, the results obtained in our testing do reflect untapped potential in this architecture. Based on the predictive performance shown, it is clear that the model is learning the spatial relations in the input data. We believe that given more time, additional modifications and optimizations could be made to the model to improve its performance, particularly when it comes to the MSense dataset.

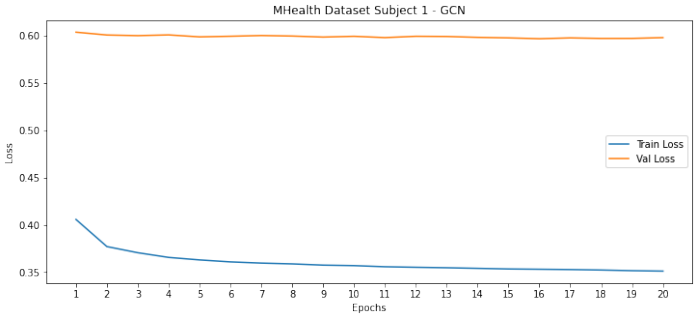
Although both the training and validation losses decrease sharply at the start, they both hit a wall at around .5 and .7, respectively, and do not improve further with additional epochs. For this reason, we set the number of epochs to just 50. Adjusting the parameters did not lead to significant improvement in performance. Needless to say, the performance of WaveNet on the MHealth data is extremely poor. We observe that the forecasted time-series curve does not remotely resemble the target curve. It is difficult to pinpoint the reason for the model's performance discrepancy between the MHealth and MSense datasets. Our results suggest that the model may achieve significantly higher performance certain datasets than others.

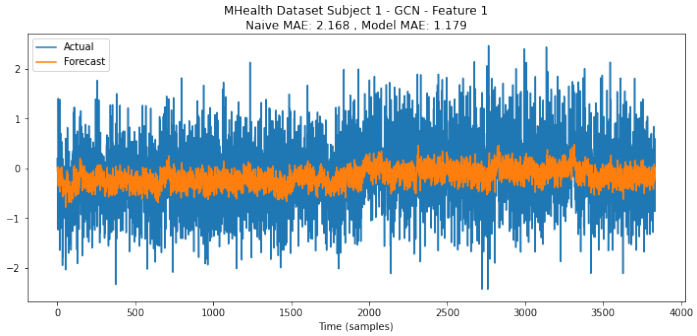
Indeed, we ran into multiple obstacles involving code adaptability. For instance, the model only accepts data structured as a 4 dimensional tensor in the following way: *(samples, seq_length, nodes, features)*. This data structure was not ideal for the datasets we used, but we made do by mapping analogical categories between our data and that the authors' used (e.g. instead of nodes on dimension 3, we had subjects in the MSense data). Another challenge we ran into was that we could not use the proposed adaptive adjacency matrix because the authors had hard coded it for their traffic data. Therefore, we opted for the identity matrix, which is one of the alternatives discussed in the research article.

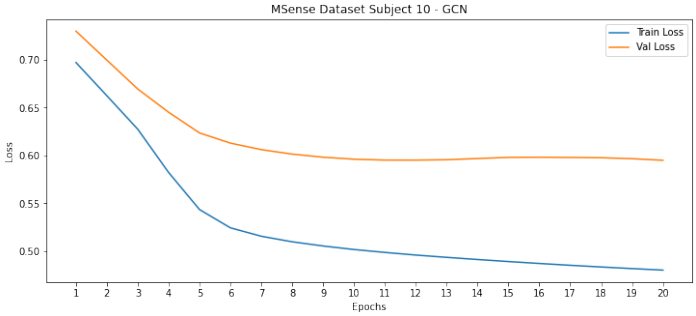
0.0.5 Simple GCN with LSTM Layer

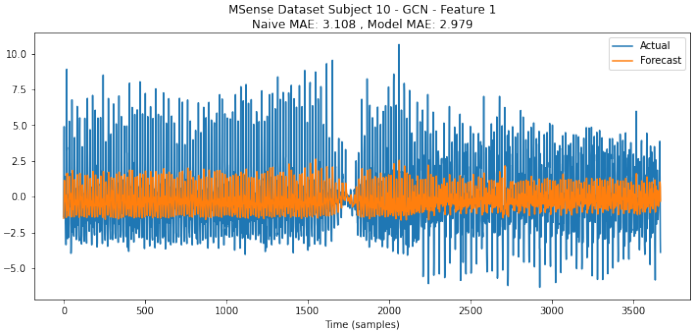
Given the challenges we faced in getting Graph WaveNet to work, we built a simple model in Keras comprising a graph convolution layer coupled with a LSTM layer that is capable of capturing both spatial and temporal information in time series data to compare our results to. For this model, we used a batch size of 64, with 64 LSTM units for the LSTM layer.

As we can see from the results, the model did a mediocre job of forecasting the 1st feautre for each dataset. For the MHealth dataset, validation loss remained high throughout training while training loss decreased slowly and only by a small amount, indicating that the model was not learning to predict the time series features very well. More significant reductions in validation and training loss were observed for the MSense dataset, but the loss error remained high at 20 epochs. In both cases, the model seemed to do a decent job of fitting the means of the time series, but failed to capture their variability.









0.0.6 Conclusions

Based on our results, the STEP framework performed the worst on both our datasets, while the simple GCN with LSTM layer did the best for the MHealth dataset and Graph WaveNet alone did the best for the MSense dataset.

This is surprising and disappointing in light of STEP's performance on the traffic datasets it was benchmarked on.

As explained above, we faced faced numerous challenges in implementing both STEP and Graph WaveNet, and wrangling with the code base that was provided by the authors of these models (for example, Graph WaveNet's codebase was unmaintained and had several issues flagged on their GitHub page that were not addressed by the authors). We also found that numerous parameters and design components were hardcoded into the architectures to deal specifically with traffic datasets.

Other factors that could have contributed to the poor performance of STEP and Graph WaveNet include the fact that we were testing these models on physiological data that are inherently noisy. In addition, we were not able to fully leverage the spatial modeling abilities of these GNNs because we were not able to obtain information regarding the actual or functional distances between sensors.

We note that STEP or Graph WaveNet may work better if it was tested on more than two subjects and longer historical sequences are used since this has been shown to improve the forecasting of other types of neural networks.