CS 726 Project Summary

Sushil Khyalia 160050035
Kartik Khandelwal 160070025
Debanjan Mondal 160050071
Sriram Y 16D070017

February 16, 2024

1 Project Goal

The problem of traffic forecasting deals with predicting Traffic related metrics like future vehicle speeds, traffic Volume, Taxi pick-up and drop-off rates, traffic flow etc all of which are of great importance to traffic management and public safety.

In this project, we are trying to predict the future traffic speed given previously observed traffic flow from N correlated sensors on the road network.

2 Related literature

The task of traffic prediction is very challenging since it is affected by many complex factors, such as spatial dependency of complicated road networks and temporal dynamics, and many more. The factors make traffic prediction a challenging task due to the uncertainty and complexity of traffic states.

2.1 Non Deep learning Approaches

Non-deep learning based models mainly include knowledge driven methods, time series models and machine learning models. Two popular time series methods include ARIMA [**arima**] and Kalman filter which focus on discovering the patterns of the temporal variation of traffic data for prediction. However, these only depend on the traffic sequential data and ignore the dynamic spatial dependency. Other popular methods include KNN, Linear Regession, RF etc.

2.2 Deep Learning Models

In recent papers Convolutional Neural Networks(CNN s) are being used to capture the spatial dependency of road networks. To model the non-linear temporal dependency Recurrent Neural Networks(RNN's) are used. Variations of RNN

like LSTM, GRU etc have been adopted for traffic prediction because they are inherently suitable for processing time series data.

Some notable deep learning Models include DCRNN, FC-LSTM, STANN, STGCN, MRes-RGNN etc. STANN uses an encoder decoder model and attention for capturing both spatial and temporal dependency. STGCN used GATED CNNs to extract spatial and temporal features and then used a conolutional block to fuse those features. MRes-RGNN used Residual Recurrent Graph Neural Networks to jointly extract spatial and temporal features from the data.

3 Tried Approaches

3.1 First Approach

Initially we build upon the code (in tensorflow) of the DCRNN (Diffusion Convolutional Recurrent Neural Network) model available at `https://github.com/liyaguang/DCRNN`. In this model, the authors used an Encoder-decoder model to predict future speed given past speed of vehicles. The authors considered the sensor network as a graph with nodes as sensors and edges between these nodes weighted by their proximity. The special contribution of their work is the *Diffusion Convolution* which they used in place of the normal matrix multiplication in the GRU cells of the RNN layers. The author's claimed that the diffusion convolution is a good way to model traffic flow because of its stochastic nature.

Our Approach was to use the Message Passing Neural Network (MPNN) Framework[Gil+17] which have been shown to work on well on graphical structured data. We tried replacing the graphical convolution operation with the MPNN framework. Unfortunately after implementing this approach, we realized that the model was taking too much time to run.

3.2 Second Approach

Looking at our first approach, we realized that using MPNN along with 4 RNN layers in the DCRNN model was not feasible so we wrote a new model (in PyTorch[ADD CITATION]) just using the MPNN framework for traffic prediction.

In MPNN, we have node features x_v and message features e_{vw} associated with each node and message in the graph. Furthermore we define additional hidden state vector h_v^t a message vector m_v^t between each pair of nodes which are updated as per the following two rules :

$$m_v^{t+1} = \Sigma_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$
$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

Finally after using these update rules for T time steps, we use the final hidden h_v^T to get the output vector for each node as per the following rules:

$$\hat{y} = R(\{h_v^T | v \in G\})$$

Here M, U and R are called Message function, vertex update function and Readout function respectively respectively. They can be defined as per the needs of the user and can range from simple concatenation to complex NN.

For our model we have defined both M and R to be a single layer Fully Connected NN and U as a Gated Recurrent Cell units(GRU) cell.

4 Experiments

4.1 Code Description

As mentioned before, for our first approach we used Tensor Flow library for creating our model. We built upon the code from the GitHub repository mentioned in the paper 'Diffusion Convolutional Recurrent Neural Network'. We then tried to implement graphical convolution function for the DCGRUCell class using a message passing neural network consisting of feed forward networks.

For the second approach we defined the model from scratch in PyTorch. The training script was also rewritten taking inspiration from the one in previous approach. We also had to redefined the loss functions in PyTorch. The script describing our model consists of 57 lines, the training script has 123 lines and the metrics scripts consisting of 90 lines.

4.2 Experimental Platform

We initially performed debugging of the code on our personal computers and then trained our final version of code using cloud services offered by Google Colab. We trained our model for approximately 5-6 hours on Python3 runtime with a GPU back-end.

4.3 Results and Commentary

Due to large value of training time required by our first approach we weren't able to produce any kind of results from it. However with our second approach we were able to generate a model which able to learn some relation between the input and the output. But the results from our model were not impressive. This might be due to one or more of the following reasons : Network Architecture (i.e. the hypothesis space) is not capable of modelling the true relation between the input space and the output space, unoptimal choice of hyper parameters, poor initialization of weights leading to convergence to a local optimum.

5 Effort

5.1 Time Distribution

First couple of weeks were spent doing the literature review and looking at the state of the art models for the problem. Also we met with Prathamesh Deshpande on recommendation of our project Supervisor who guided us towards different approaches to tackle the problem for which we are grateful.

The next couple of weeks was spent on finalizing our course of action for the first approach and writing our code.

During the endsem week, we realized that our first approach was practically unfeasible and started thinking about alternative approach's.

The period post endsem(3-4 days) was spend implementing and debugging our second approach.

5.2 Challenges

The most challenges part about the project was coming up with new approach to solve the problem.

5.3 Work Distribution

The brainstorming and distribution

References

[Gil+17] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *CoRR* abs/1704.01212 (2017). arXiv: 1704.01212. URL: http://arxiv.org/abs/1704.01212.