Traffic Flow Prediction Using Deep Learning

Kunal Ranjan
IIT Kanpur
Mentor:
Rahul Jain
Anirudha Kurhade

KPIT Technologies Ltd.

Abstract

This paper investigates the multi-agent dynamics of sensors which will help in streamlining traffic to avoid jamitons. These sensors records traffic density and are located at various points in the roadways. Predicting traffic density is a challenging problem in automotive industry. It is being stimulated using a lot of statistical models but they are more mathematical rather than applied form. SVM and regressors are also tried for solving the issue but they are not much accurate. In recent years, various deep learning techniques are used to solve this problem. Sequence to sequence LSTM has solved the density problem to a certain extent but it is computationally expensive.

In this paper, we have modelled relation between different sensors so that we can reduce computation without compromising prediction accuracy. Pooling module has solved this problem and has opened up huge scope of improvement in traffic flow existing models. As this report shows, GANs with pooling module could definitely be of use for the efficiently modeling traffic flow, and we recommend investigating these methods more thoroughly.

1 Problem Description

Our problem consists of traffic density counts at different points of the roadways which are actually recorded by various sensors. Since a lot of sensors can be involved and our aim is to jointly predict traffic density, it can be computationally expensive in terms of cost and time both. So, It raises the possibility of predicting traffic density considering dependence on relationship between sensors. Only related sensors are then need for modeling density in contrast to all sensors.

2 Motivation

Traffic Jams are frequent in metropolitan cities. In order to resolve this somehow one need to know about flow of traffic in the surrounding roadways which helps automobile in regulating its speed. Embedding dynamic traffic density prediction module in all automobile can help drivers to manage his/her speed and leads to smooth flow of traffic. These modules can also be embedded with traffic signals located at crowded roadways.

3 Investigation/Research

3.1 Generative Adversarial Nets (`https://arxiv.org/pdf/1406.2661.pdf`)

This is the first generative adversarial network paper. This network consists of 2 architectures: Generator and Discriminator. Generator tries to generate fake images from random noise and Discriminator tries to distinguish real vs fake images.

3.2 Social GANs (`https://arxiv.org/pdf/1803.10892`)

Through this paper, we have explored Pooling module which generates a major breakthrough in our traffic flow dataset.

3.3 DCGAN(`https://arxiv.org/pdf/1511.06434`)

Architecture for GANs network improved drastically through fine tuning used in this paper especially in image processing part but it helped inour problem statement also.

3.4 Improve Tech. for Training GANS (`https://arxiv.org/pdf/1606.03498`)

It further improved most typical GANs problem one faces while training generator.

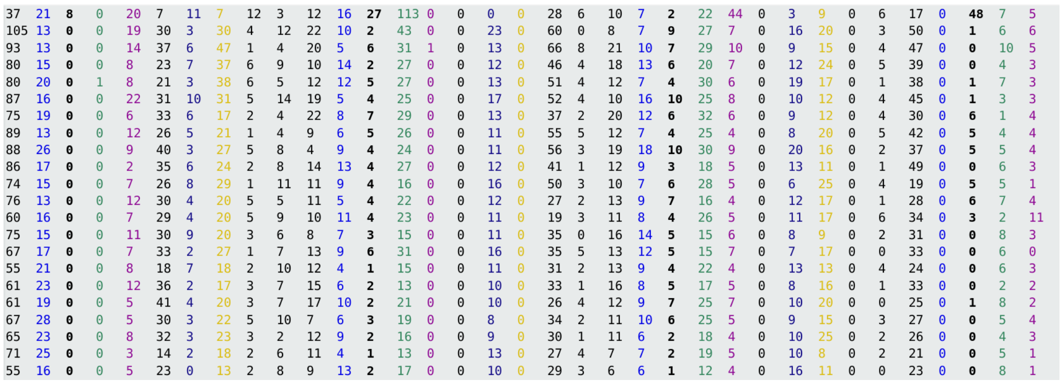
3.5 Seq2Seq LSTM (`https://arxiv.org/pdf/1503.04069`)

For benchmark setting we have gone through the Guillaume Genthial blog on Seq2Seq and LSTM paper which introduced us to Seq2Seq models like LSTMs and then Seq2Seq models with Attention.

4 Dataset Description

Data consists of traffic flow that were collected from October, 2015 to January, 2016, from spring to winter, including sunny, rainy and snowy days.Actually These data were collected by sensors which are located at various points in the roadways. Each sensor records the traffic count for every 5 minute time span and there are in total 35 sensors.
Dataset can be found at `https://kaggle.com/coplin/traffic/data`

Preprocessing is done to in order to ease the training procedure. Thus, final data consists of traffic count recorded by 35 sensors(columns) for every 5 min time span (rows)



37	21	8	0	20	7	11	7	12	3	12	16	27	113	0	0	0	0	28	6	10	7	2	22	44	0	3	9	0	6	17	0	48	7	5
105	13	0	0	19	30	3	30	4	12	22	10	2	43	0	0	23	0	60	0	8	7	9	27	7	0	16	20	0	3	50	0	1	6	6
93	13	0	0	14	37	6	47	1	4	20	5	6	31	1	0	13	0	66	8	21	10	7	29	10	0	9	15	0	4	47	0	0	10	5
80	15	0	0	8	23	7	37	6	9	10	14	2	27	0	0	12	0	46	4	18	13	6	20	7	0	12	24	0	5	39	0	0	4	3
80	20	0	1	8	21	3	38	6	5	12	12	5	27	0	0	13	0	51	4	12	7	4	30	6	0	19	17	0	1	38	0	1	7	3
87	16	0	0	22	31	10	31	5	14	19	5	4	25	0	0	17	0	52	4	10	16	10	25	8	0	10	12	0	4	45	0	1	3	3
75	19	0	0	6	33	6	17	2	4	22	8	7	29	0	0	13	0	37	2	20	12	6	32	6	0	9	12	0	4	30	0	6	1	4
89	13	0	0	12	26	5	21	1	4	9	6	5	26	0	0	11	0	55	5	12	7	4	25	4	0	8	20	0	5	42	0	5	4	4
88	26	0	0	9	40	3	27	5	8	4	9	4	24	0	0	11	0	56	3	19	18	10	30	9	0	20	16	0	2	37	0	5	5	4
86	17	0	0	2	35	6	24	2	8	14	13	4	27	0	0	12	0	41	1	12	9	3	18	5	0	13	11	0	1	49	0	0	6	3
74	15	0	0	7	26	8	29	1	11	11	9	4	16	0	0	16	0	50	3	10	7	6	28	5	0	6	25	0	4	19	0	5	5	1
76	13	0	0	12	30	4	20	5	5	11	5	4	22	0	0	12	0	27	2	13	9	7	16	4	0	12	17	0	1	28	0	6	7	4
60	16	0	0	7	29	4	20	5	9	10	11	4	23	0	0	11	0	19	3	11	8	4	26	5	0	11	17	0	6	34	0	3	2	11
75	15	0	0	11	30	9	20	3	6	8	7	3	15	0	0	11	0	35	0	16	14	5	15	6	0	8	9	0	2	31	0	0	8	3
67	17	0	0	7	33	2	27	1	7	13	9	6	31	0	0	16	0	35	5	13	12	5	15	7	0	7	17	0	0	33	0	0	6	0
55	21	0	0	8	18	7	18	2	10	12	4	1	15	0	0	11	0	31	2	13	9	4	22	4	0	13	13	0	4	24	0	0	6	3
61	23	0	0	12	36	2	17	3	7	15	6	2	13	0	0	10	0	33	1	16	8	5	17	5	0	8	16	0	1	33	0	0	2	2
61	19	0	0	5	41	4	20	3	7	17	10	2	21	0	0	10	0	26	4	12	9	7	25	7	0	10	20	0	0	25	0	1	8	2
67	28	0	0	5	30	3	22	5	10	7	6	3	19	0	0	8	0	34	2	11	10	6	25	5	0	9	15	0	3	27	0	0	5	4
65	23	0	0	8	32	3	23	3	2	12	9	2	16	0	0	9	0	30	1	11	6	2	18	4	0	10	25	0	2	26	0	0	4	3
71	25	0	0	3	14	2	18	2	6	11	4	1	13	0	0	13	0	27	4	7	7	2	19	5	0	10	8	0	2	21	0	0	5	1
55	16	0	0	5	23	0	13	2	8	9	13	2	17	0	0	10	0	29	3	6	6	1	12	4	0	16	11	0	0	23	0	0	8	1

Figure 1: Final Dataset

5 Methodology

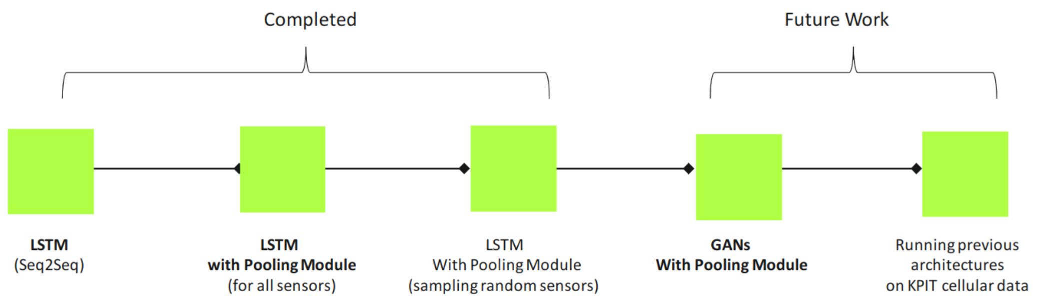


Figure 2: Project Realization

The project consists of 3 architecture:

1. **Sequence to Sequence LSTM**
2. **Sequence to Sequence LSTM with pooling module (for all sensors)**
3. **Sequence to Sequence LSTM with pooling module (sampling random sensors)**

5.1 Sequence to Sequence LSTM (Seq2Seq LSTM)

The given dataset has no deep learning solution yet. So, this architecture is developed as a benchmark which can be used further for comparison while developing advanced architectures.

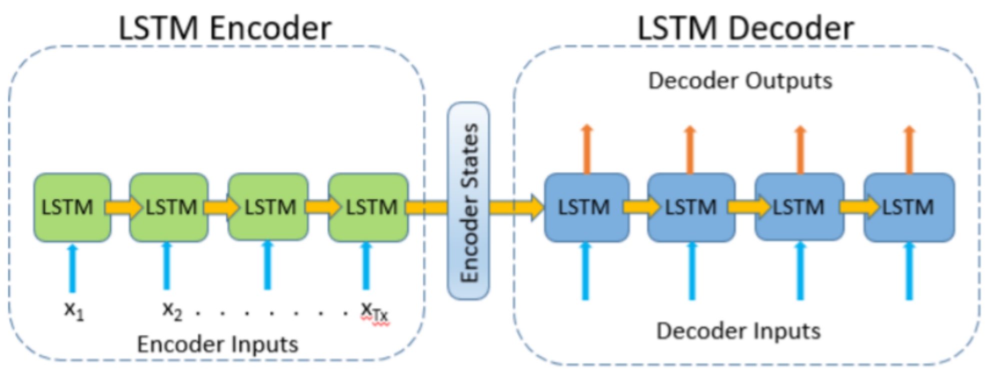


Figure 3: Seq2Seq LSTM Architecture

The architecture consists of 2 parts:

- **LSTM Encoder**
- **LSTM Decoder**

5.1.1 LSTM Encoder

This sub-architecture learns relevant information while training i.e. encodes the necessary correlations in the form of hidden states at every time step of training and finally last time step hidden states (Encoder States in figure 3) are passed to next sub-architecture(LSTM Decoder).

• Training Procedure

Encoder input x_i is a 35 dimensional vector i.e. dim(x_i) is [1×35] where each column represent traffic count of specific sensor S_i. So, encoder part learns the correlation between all the 35 sensors at each time step and stores it in hidden states. And finally, Encoder States store cumulative information while training upto Tx time step (see figure 3)

5.1.2 LSTM Decoder

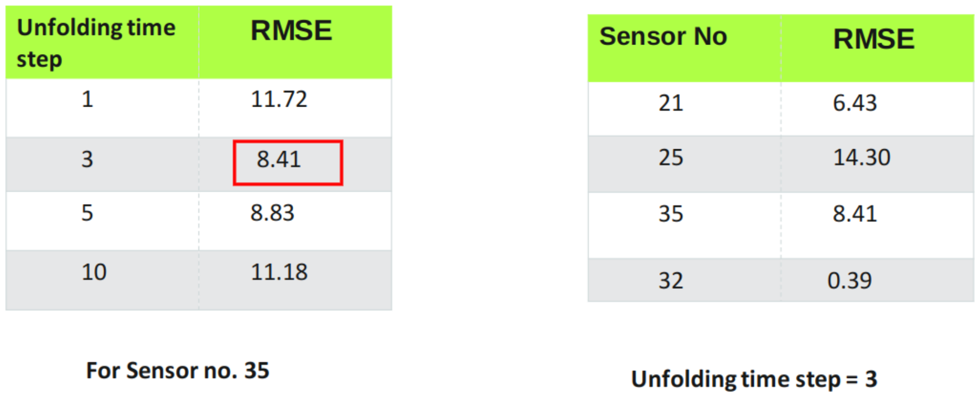
This sub-architecture gets Encoder States information from Encoder LSTM and starts decoding and generating output for each input feeded to decoder and also pass on relevant information further.

• Training Procedure

Decoder input is a 34 dimensional vector i.e. dim(x_i) is [1×34] where each column represent traffic count of specific sensor. Note that input only has 34 values i.e. only 34 sensors are used. Remaining 1 sensor is the object of interest for which we want to do traffic flow prediction analysis. Thus, at T_{pred} we feed traffic count value of 34 sensors and decoder part predicts the count at T_{pred} for the remaining sensor.

5.1.3 Experimentation and Result

- During training encoder, various unfolding time step is tried for better learning of correlation among sensors. We get best result for unfolding time step = 3 for sensor no. 35(see figure 4)
- We have then assumed that this unfolding step works best for all remaining sensors when they are object under consideration for prediction.
- we have trained the model only for sensor no. 21,25, 32 and 35 (see figure 4)



Unfolding time step	RMSE
1	11.72
3	8.41
5	8.83
10	11.18

For Sensor no. 35

Sensor No	RMSE
21	6.43
25	14.30
35	8.41
32	0.39

Unfolding time step = 3

Figure 4: Root Mean Square Error(RMSE) Results

5.2 Sequence to Sequence LSTM with pooling module (for all sensors)

The key feature of this architecture is that we train LSTM models for different sensors independently and use pooling module to learn the correlation between sensors in contrast to vanilla Seq2Seq LSTM where we train a single model by using all sensors values at once at every times step. This architecture has 3 major components:

- Encoder LSTM
- Pooling Module
- Decoder LSTM

5.2.1 Encoder LSTM

- We first transform the traffic count of each sensor into higher embedding dimension.

$$e_i{}^t = \phi\left(x_i{}^t; W_{ee}\right) \tag{1}$$

where ϕ is an embedding function with ReLU non-linearity, W_{ee} is the embedding weight.

- Before feeding Hidden state of one LSTM cell to next time step cell we transform it using pooling module.

$$H_i{}^t = LSTM\left(H_i{}^{t-1}, e_i{}^t; W_{encoder}\right) \tag{2}$$

$$P_i{}^t = PM\left(H_1^t, H_2^t ..., H_n^t\right) \tag{3}$$

where The LSTM weights are denoted by $W_{encoder}$

- Pooled feature are then concatenated with hidden states and passed to MLP to get final hidden states

$$H_i{}^t = \gamma\left(P_i{}^t, H_i^t; W_e\right) \tag{4}$$

where γ is an MLP.

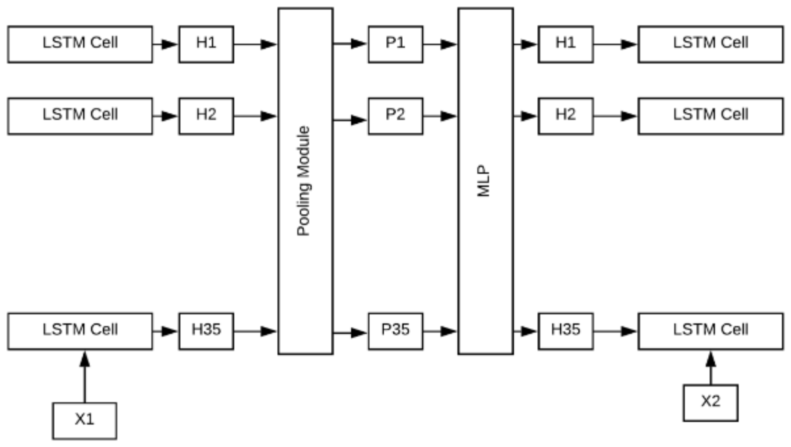
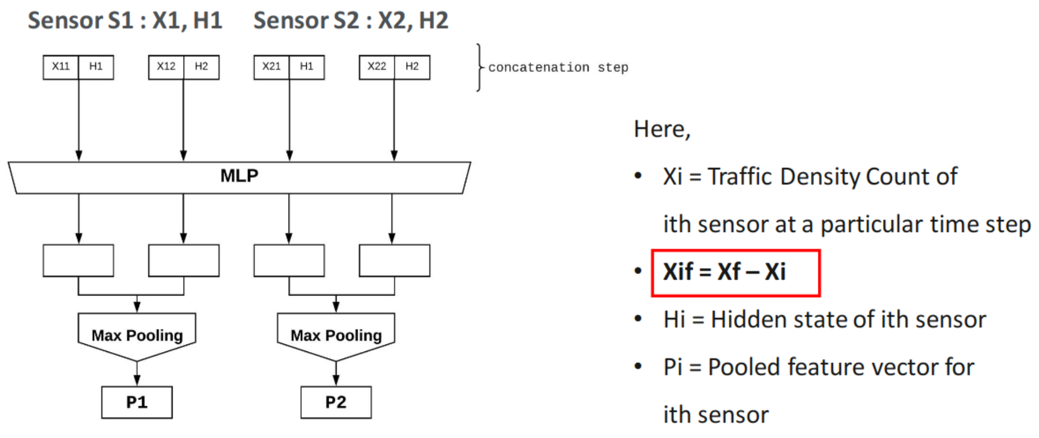


Figure 5: Encoder LSTM

5.2.2 Pooling Module

This Pooling Module is introduced in Social-GAN paper and has ability to learn various context among interacting objects.

- learns a “global” pooling vector
- stores the subtle information for all sensors interacting
- works as an better alternative for hidden states which store history in the vanilla Seq2Seq LSTM



Here,

- Xi = Traffic Density Count of ith sensor at a particular time step
- **Xif = Xf – Xi**
- Hi = Hidden state of ith sensor
- Pi = Pooled feature vector for ith sensor

Figure 6: Pooling Module for 2 sensors

In Pooling Module, hidden states of sensors are concatenated with relative traffic count (see figure 6) then passed to MLP and after that max pooling is to be done to get final pooled feature vector for each sensor.

5.2.3 Decoder LSTM

- Decoder architecture remains as it is in vanilla Seq2seq LSTM
- But a key difference is that we only use hidden state of the sensor under study for prediction. There is no need to consider all 35 hidden state i.e. final context came from encoder LSTM because pooling module learns and transfer relative information of other sensors at each time step.

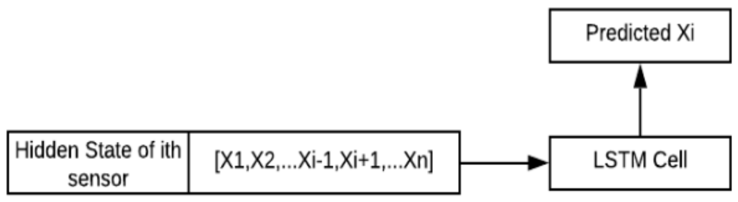
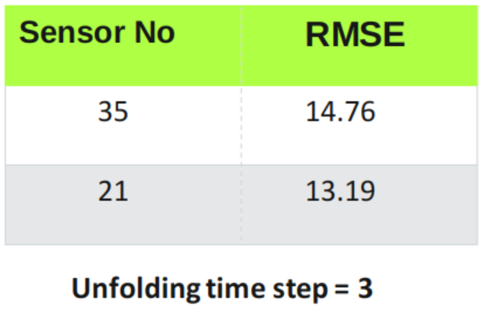


Figure 7: Decoder Cell Unit

5.2.4 Experimentation and Result

- During decoding step, we have also tried all hiiden states by them to make a large hidden state in constract to only 1 hidden state which we actually used. But it increased computation time and loss drastically
- We have then assumed that the unfolding step 3 works best for all sensors when they are object under consideration for prediction.
- we have trained the model only for sensor no. 21, and 35 (see figure 8)



Sensor No	RMSE
35	14.76
21	13.19

Unfolding time step = 3

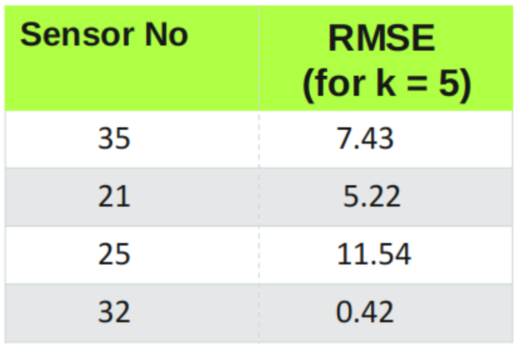
Figure 8: Result when all sensors are assumed to be correlated

5.3 Sequence to Sequence LSTM with pooling module (sampling random sensors)

This architecture is same as the previous architecture with the only difference is that instead of trained LSTM cells on all 35 sensors, we randomly sample particular number of sensors and run seq2seq LSTM with pooling module on these sensors only.

5.3.1 Experimentation and Result

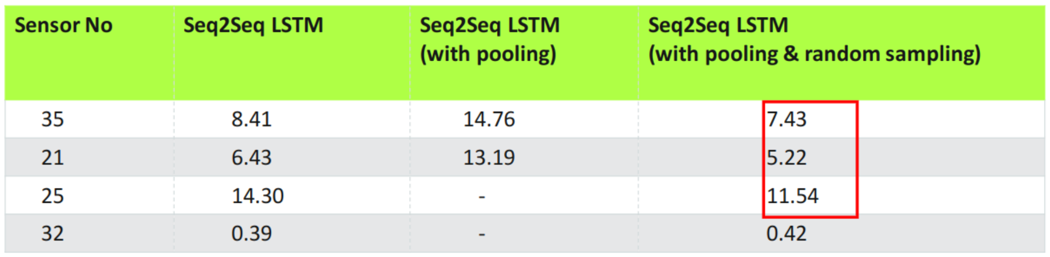
- During training encoder, various unfolding time step is tried for better learning of correlation among sensors. We get best result for unfolding time step = 10
- We have then assumed that this unfolding step works best for all remaining sensors when they are object under consideration for prediction.
- we have trained the model only for sensor no. 21,25, 32 and 35 (see figure 9)



Sensor No	RMSE (for k = 5)
35	7.43
21	5.22
25	11.54
32	0.42

Figure 9: Results for unfolding timestep = 10 where k is no of sensors sampled

6 Conclusion and Comparison



Sensor No	Seq2Seq LSTM	Seq2Seq LSTM (with pooling)	Seq2Seq LSTM (with pooling & random sampling)
35	8.41	14.76	7.43
21	6.43	13.19	5.22
25	14.30	-	11.54
32	0.39	-	0.42

Figure 10: Results (RMSE) for all 3 architectures

- when we sample sensors the resulting rmse error is low (last column)as compared to base model but when we sample all sensors then error got increased (2nd column).
- One of the reason for this is that in long run pooling module is not able to store correlation for all sensors which is also an issue with recurrent neural net hidden states.
- There is also a possibility that only few sensors are related actually. So, when we sampled all sensors then pooling module store extra irrelevant information which diverts the model output and increased the loss.
- Sensors traffic density standard deviation varies from 0.0 to 177.38
- Sensors with density sd of order 38 generates good result under random sampling pooling model. Although This is evident from training 10 sensor models. So, It might be true for higher orders but not sure !

7 Further Optimizations/Future Work

- Implementation of GANs whose generator has core buildup of seq2seq LSTM with pooling module.

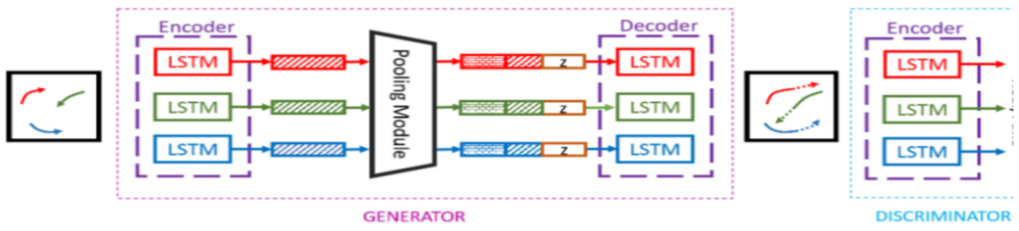


Figure 11: This architecture is from Social GAN paper

Note:we have also tried this architecture on traffic flow data mentioned above but it's limitation is that it takes discontinuous non-uniform time steps while unfolding LSTM cells. And for this type of time series also, we don't have any benchmark. So, we planned to use this architecture by modify latter it in such a way that it takes continuous time steps as such we can compare it with our benchmark.

- Implement all the architecture on KPIT cellular data