Social Context Agent Based Network Trajectory Prediction

Manoj Bhat
Carnegie Mellon University
mbhat@andrew.cmu.edu

Suyash Narain
Carnegie Mellon University
snarain@andrew.cmu.edu

Umut Soysal
Carnegie Mellon University
usoysal@andrew.cmu.edu

Abstract

The aim of this project was to develop an agent based tracking and prediction model for subsequent use in autonomous vehicles. The moving agent tries to learn the motion behavior of external disturbances and predict their trajectory and then plan its own path accordingly to avoid them. We analyze how the human behaves in external circumstances and make the agent learn to mimic these behaviors in given social context. We look at different RNN models such as Vanilla RNN, and then proceed with improved prediction models such as LSTM and subsequently will use social grid based agent LSTM model for path planning of the agent with disturbance behavior as the variables. Stanford drone dataset is used for mapping and training the disturbance motion behavior. The developed model firstly runs on the test dataset given and generates the predicted trajectory. The social context tensor is used by the agent lstm with velocity and position vector of the agent, to generate the agent path.

1. Introduction

Humans have this innate ability to read other humans and predict their next course of actions. For example, a person walking in a crowded area like shopping malls, or subway stations, will respect other persons space and yield right of way. Similarly, a person driving a vehicle, on seeing a pedestrian in his path will make him predict his next course of action which will ultimately determine his driving path to avoid the pedestrian. These are certain rules or social conventions a human is already pre-programmed with. The ability to model these rules and use them to understand and predict human movement in complex real world environments is highly valuable for wide range of applications, like socially aware robot, or self driving cars to name a few. [] The ability to model human interactions and their motion prediction while considering their common sense behavior is of high importance and a lot of work is being done in this field. But majority of prediction models predicting human trajectory completely ignore the scene information, and hence dont explain as to how the agent should behave in given social context.

Consider our previous example of a person driving a vehicle. On seeing a pedestrian on the road, the driver can predict the path the pedestrian can take and on that basis chart his driving course to avoid the pedestrian. The ongoing research and development in the field of autonomous vehicles gives rise to further development of agent based tracking and prediction model, modelled upon a human drivers interaction behavior and subsequent driving path prediction based upon the predicted path of the pedestrians. This is important because, at ground level, (eg. If LiDAR fails etc.) the vehicle can use continuous (GPS) real time data to avoid undesired motion. For this, the agent should be able to take precise calculated steps and velocities. In other words, the vehicle would sense obstacles (pedestrians) in it's path, and on the basis of the training received by it, will predict their behavior and trajectory, and taking that predicted trajectory as input, apart from its current velocity and position, will chalk out it's path so as to avoid these disturbances.

To put this to action, we start with reviewing Recurrent Neural Network (RNN). They are a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit temporal dynamic behavior for a time sequence. All RNN's have feedback loops in the recurrent layer, which allows it to maintain information in memmory over time. But they suffer form vanishing gradient problem, and hence cannot be used to train models that require learning long term temporal dependencies. To overcome shortcomings of RNN, we proceed to LSTM, which are again a type of RNN, but consists of additional special units called gates, which regulate the flow of information. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. We use different manifestations of LSTM in our model, firstly using Vanilla LSTM, and then using GRU (gated recurrent units), LSTM and Social LSTM and comparing the obtained results using the following metrics:

1. average displacement error
2. final displacement error
the results were tabulated and visualizations obtained.

2. Related Work

Research upon human trajectory planning has been going on for quite a while and different approaches have been used for the same. In the existing pedestrian trajectory prediction research, the existing methods can be classified into two types: model-based prediction methods and recurrent neural network (RNN)-based methods.

In **model based prediction methods**, the mathematical functions were usually designed and specific pedestrian properties were defined. Pioneering work presented in [3] which modelled pedestrian motion based upon attractive and repulsive forces. This model was referred to a social force model, as it was of the view that humans were attracted or repelled by various social forces, and that defined their trajectories in motion. This model has given quite impressive results on various datasets and has even been extended to robotics [4], particularly for use in mobile robots in populated environments, where human trajectory tracking is important for safe functioning and deployment of these robots. But model based prediction methods were only able to predict future pedestrian trajectories for a short time period and in complex scenes, trajectories predicted were usually inaccurate, and even for models requiring learning long term temporal dependencies.

With advent of deep learning, **RNN based methods** have been used to solve sequence learning and prediction tasks. RNN and their derivatives like LSTM (Long Short Term Memory) and GRU(Gated Recurrent Units) have proven to be very successful for sequence prediction tasks. RNN can also be used for scene parsing, semantic segmentation and are also capable of learning the dependencies between spatially correlated data such as image pixels. Social LSTM [1], a novel approach by Alahi et al, which can jointly predict the paths of all the people in a scene by taking into account the common sense rules and social conventions that humans typically utilize as they navigate in shared environments.

Scene LSTM [5] a recently developed RNN based prediction model, is a human movement trajectory prediction model that incorporates the scene information (Scene-LSTM) as well as human movement trajectories (Pedestrian movement LSTM) in the prediction process within static crowded scenes. It superimposes a two-level grid structure (scene is divided into grid cells each modeled by a scene-LSTM, which are further divided into smaller sub-grids for finer spatial granularity) and explore common human trajectories occurring in the grid cell (e.g., making a right or left turn onto sidewalks coming out of an alley; or standing still at bus/train stops). Two coupled LSTM networks, Pedestrian movement LSTMs (one per target) and the corresponding Scene-LSTMs (one per grid-cell) are trained simultaneously to predict the next movements. We show that such common path information greatly influences prediction of future movement. They designed a scene data filter that holds important non-linear movement information. The scene data filter allows to select the relevant parts of the information from the grid cell's memory relative to a target's state. Another method called Social-Grid LSTM method [2] based upon RNN network predicts human trajectory. The method combines the human to human interaction model called social pooling and the Grid LSTM network model.

For our model, we tried to combine social and social-grid LSTM techniques and try to predict, apart from human trajectory, the agent's trajectory on the basis of earlier predicted pedestrian trajectory.

3. Data

Figure 1. Sample image from stanford drone dataset

In this work, Stanford Drone Dataset is used [6]. This dataset comprises of 8 unique scenes in crowded places such as university campus or sidewalks etc. and a total of 20,000 targets engaged in various types of interactions. Target trajectories along with their target IDs are annotated. Different type of targets are annotated as (car,pedestrian,bike and cart). For the purposes of our experiment, W mainly concentrated upon pedestrians and bikers, where pedestrians were the disturbances and bikers were the agents. Based upon the predicted trajectory of the pedestrians, the path of bikers was be determined. The dataset contains annotations and videos where annotations data are pre-tracked video frame wise along with width and height if tracker box for each disturbance or agent. certain parameters like occluded or generated or out of view are

provided which are just 1 and 0 identifying if the pedestrian or the agent is occluded or interpolation generated or is within frame or not.

Stanford drone dataset contains of dataset of videos out of

Figure 2. Tracked HYANG data in stanford dataset

which we used hyang dataset consisting of 14 videos and their annotations. In all the videos, maximum number of bikers was found to be 156, and pedestrians to be 212. the number of frames for each pedestrian trajectory is analysed and the whole sequence is divided into batches for training and testing. the same was dine for the agent as well. As the frame sizes are in pixels, we find the centroid of the frames and normalize the x and y positon of the same with respect to the width and height pixels of the video. At the end, the annotations data were converted to the given [Time-frames, max people/agents, (PedID/AgentID, x, y)]. average frames in videos is 16327 (sequences), and there are a maximum of 32 people/agents data in a frame. The average frame size of the video is 1440 x 1235px.

4. Methods

4.1. Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit temporal dynamic behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. All RNNs have feedback loop in the recurrent layer. This lets them maintain information in 'memory' over time. But it can be difficult to train standard RNNs to solve problems that require long term temporal dependencies, as in our model. This is because the gradient of the loss function decays exponentially with time, also called the vanishing gradient problem. Hence variations of RNNs are used to solve complex problems like LSTM, GRU etc. Hence, simple RNN was not pursued here, and we started with vanilla LSTM.

4.2. LSTM

LSTM or Long Short Term Memory is a variation of RNN, which uses special units in addition to standard units. LSTM units consist of a 'memory cell' that can maintain information in memory for long periods of time. A set of gates is used to control the information flow, the output and when its forgotten.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (1)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (2)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \quad (4)$$
$$h_t = o_t \circ \sigma_h(c_t) \quad (5)$$

the above equations correspond to the gates in LSTM cell.
$x_t \in \mathbb{R}^d$ - input vector to the LSTM unit
$f_t \in \mathbb{R}^h$ - forget gate's activation vector
$i_t \in \mathbb{R}^h$ - input gate's activation vector
$o_t \in \mathbb{R}^h$ - output gate's activation vector
$h_t \in \mathbb{R}^h$ - hidden state vector also known as output vector of the LSTM unit
$c_t \in \mathbb{R}^h$ - cell state vector
$W \in \mathbb{R}^{h \times d}, U \in \mathbb{R}^{h \times h} U \in \mathbb{R}^{h \times h} and b \in \mathbb{R}^h b \in \mathbb{R}^h$- weight matrices and bias vector parameters which need to be learned during training where the superscripts d d and h h refer to the number of input features and number of hidden units, respectively.

Simple LSTM is even referred to as Vanilla LSTM, which was one of the methods used by us in our model. Herein, a separate LSTM is used for each trajectory in the scene. Hence, the predicted trajectory is independent of any social influence, and independent of the external environment. It takes in the previous x and y coordinates of the pedestrian in the frame, and predicts the future x and y coordinates. the predicted coordinates then become the input of the LSTM. As this model ignores the social context if the environment, results as expected would be highly flawed.

4.3. Social Grid Based LSTM

Social Grid Based LSTM is a variance of LSTM, wherein, apart from using separate LSTM network for each trajectory in the scene, we use a social pooling layer as well.This is important because of shortcomings of Vanilla LSTM model, which is skeptic to behavior of other sequences, that is, though it learns state of person and predicts their trajectory, but that is independent of social context of the environment, and is not influenced by external factors like other person's trajectory, and hence not applicable in real life.
The social pooling layer mentioned here connects LSTMs to each other, allowing spatially proximal LSTMs to share

information with each other. The hidden states of all LSTMs within a certain radius are pooled together and used as input at the next time step. The hidden state h_i^t of the LSTM at time t captures the latent representation of the i^{th} person in the scene at that instant. We share this representation with the neighbours by building a social hidden state tensor H_t^i. Given a hidden state dimension D, and neighborhood size N, we construct a NxNxD tensor H_t^i for the i^{th} trajectory:

$$H_t^i(m,n,:) = \sum_{j \epsilon N_i} 1_{mn}[x_t^i - x_t^i, y_t^i - y_t^i] h_{t-1}^j \quad (6)$$

Here, H_{t-1}^j is the hidden state of LSTM corresponding to j^{th} person at state (t-1). N is the set of neighbors corresponding to person i. and $1_{mn}[x,y]$ is indicator function to check if (x,y) is in (m,n) cell of grid. We embed the the pooled Social hidden-state tensor into a vector a_t^i and the co-ordinates into e_t^i.These embeddings are concatenated and used as the input to the LSTM cell of the corresponding trajectory at time t. The following recurrences are obtained:

$$r_t^i = \phi(x_t^i, y_t^i, W_r) \quad (7)$$
$$e_t^i = \phi(a_t^i, H_t^i, W_e) \quad (8)$$
$$h_t^i = LSTM(h_i^{t-1}, e_i^t, W_l) \quad (9)$$

where $\phi(.)$ is an embedding function with ReLU nonlinearity and W_r and W_e are embedding weights. W_l is LSTM weights. parameters of LSTM model are learned by negative log likelihood loss (L^i for the i^{th} trajectory):
$L^i(We; Wl; Wp) = -\sum_{t=Tobs+1}^{Tpred} log(P(x_t^i, y_t^i | \sigma_t^i, \mu_t^i, \rho_t^i))$
stacked LSTM architecture was implemented. A Stacked LSTM architecture can be defined as an LSTM model comprised of multiple LSTM layers. An LSTM layer above provides a sequence output rather than a single value output to the LSTM layer below. Specifically, one output per input time step, rather than one output time step for all input time steps. This is beneficial as depth of the network is more important than the number of memory cells in a given layer to model a skill.

4.4. GRU

GRU or Gated Recurrent Units is another variation of recurrent neural network which aims at solving the vanishing gradient problem. GRU are similar to LSTM with the exception of not having a cell state and having only two gates (update and reset) instead of (input, forget and output). A gated recurrent unit uses an update gate and a reset gate. The update gate decides on how much of information from the past should be let through and the reset gate decides on how much of information from the past should be discarded.

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \quad (10)$$
$$r_t = \sigma(W_r[h_{t-1}, x_t]) \quad (11)$$
$$h_t^{\sim} = tanh(W[r_t * h_{t-1}, x_t) \quad (12)$$
$$h_t = (1-z) * h_{t-1} + z_t * h_t^{\sim} \quad (13)$$

here, z_t represents update gate operation,whereby using a sigmoid function, we decide what values to let in from the past. h_t represents reset gate operation,, where we multiply the concatenated values from the previous time step and current time step with r_t. This produces the values that we would like to discard from the previous time steps. we tested our model with GRU as well and noted the results as discussed below.

4.5. Optimizer Selection

we used RMSprop optimizer for our model. RMSprop is an unpublished, adaptive learning rate method proposed by Geoff Hinton. It is an optimizer that utilizes the magnitude of recent gradients to normalize the gradients. We always keep a moving average over the root mean squared (hence Rms) gradients, by which we divide the current gradient. RmsProp has several advantages; for one, it is a very robust optimizer which has pseudo curvature information. Additionally, it can deal with stochastic objectives very nicely, making it applicable to mini batch learning. Mathematical expression of RMSprop is:

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \quad (14)$$

and

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \quad (15)$$

where default value for η is 0.001.

we even tried to run our model with AdaGrad and found out that the model runs faster with RMSprop than with AdaGrad.

5. Experiments

To solve the problem of agent prediction modelling based on social environment, we used RNN based architectures for time frame based sequential prediction. We start experimenting by comparing with pre existing prediction models. These prediction models are applied to only agent model but in case of social model, we use the social grid pooling for the agent trajectories. Pure TensorFlow library was used for modelling. Below are the description of models compared and observations;

Vanilla LSTM: we use the LSTM cell from tensorflow first using single cell based and then stacked LSTM based. For every agent in subsequent frames, the positions (x,y)

were given as gate inputs and for every five frames, the prediction of next three frames were visualized. The training of the model was done using the learning rate of 0.05 which showed faster loss drops than the learning rates below it. The model was run for 5 consecutive frame and loss for next subsequent frame was optimized, using RMSprop. The output of LSTM was passed through a single linear layer to downsample those outputs into dimension of [1x5], with dimension 5 representing two means, two standard deviations and one correlation factor.

The dataset for training was using a dataloader which used annotations of 6 videos out of 14, and was trained for 40 epochs. The loss function used was a 2D distribution over x and y and target data of observed x and y points. The negative of log was maximized for numerical stability. Loss functions was calculated for each time step, summing log probabilities for each datapoint. The gradient clipping was done based on given hyperparameter threshold before training the operator using the optimizer.

GRU: A simple tensorflow GRU cell was used in the grid of the LSTM to compare the predictive efficiency of the models with the same learning rate of 0.05 and run for 40 epochs. All other parameters including loss function and optimizer were kept the same as in vanilla LSTM.

Social Grid LSTM: In Social Grid LSTM, the inputs are passed through a linear layer, and at the same time, a social grid tensor is also passed through its own linear later who's weights are concatenated and given to LSTM cell of particular LSTM. The final output state of the LSTM is passed through the linear layer which is downsampled to five data points as mentioned in the case of vanilla LSTM. The loss function is maximizing the negative of log likelihood which is optimized by RMSprop as well as AdaGrad and a comparison is done between the two.

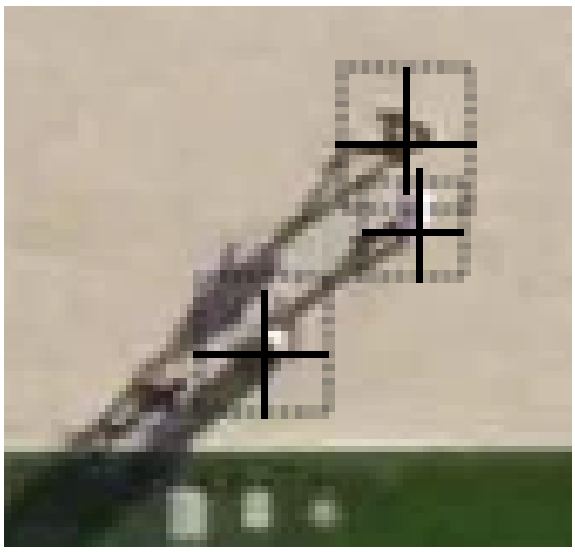

Figure 3. Social Grid Occupancy representation for social grids

Agent Based Social Grid LSTM (ASGM): here, we present a model which tries to predict the trajectory for agents based on the social context. The proposed network architecture has a coordinate tensor which is based on coordinates for a given time frame and a social context tensor which is shared with the similar agent network. This tensor is an embedding which follows social attention LSTM model for social grids. We want the model to learn a sense of relative distances for which we present gradients which are representative of velocity and hence a relative pooling layer. The concatenation of all three embeddings helps the agent LSTM network to learn as much detailed steps as possible. The final prediction tensor from the whole model consists of sequence of position data of both the disturbance and the agent, which is fed to the next frames as the next state of the RNN. We train the model using six of the videos in Hyang video set in stanford drone dataset. two possible combinations were explored. One with common embedding, and the other with independent embedding. The common embedding is a tensor of weights representative of social context shared by both, whereas independent embeddings are the learned weights by the both RNN networks of disturbance and agent. In both cases, the social context tensor would be the same. Along with that, two configurations were explored, one with the gradient embeddings as concatenation, and second without relative gradient embedding concatenation.

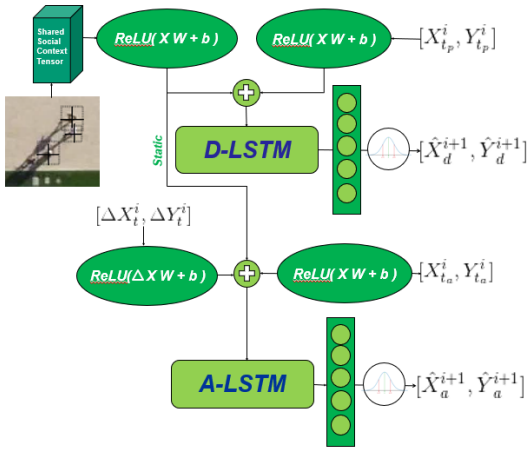

Figure 4. Model architecture for ASGM

The model was grained for 20 epochs with a learning rate of 0.03 for both the networks and with a decay rate of 0.9 and 0.95 with no dropouts. The spatial coordinate and the gradient embedding weight size is 64 and the grid around the pedestrians is of neighborhood size 32 px. The grid size for social grid is 2. The observed average time per batch of size 10 for the whole dataset of 113560 frames for training is 7.5 seconds. The social tensor is created by tak-

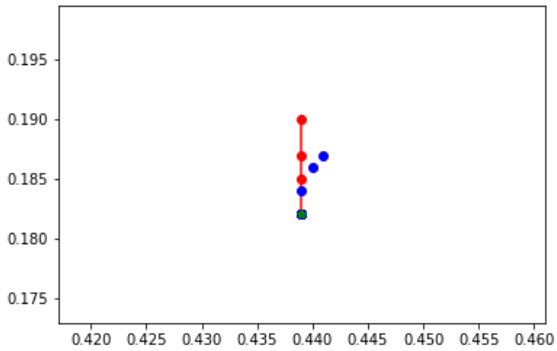

Figure 5. Vanilla LSTM 3 data point prediction for sequence length of 8

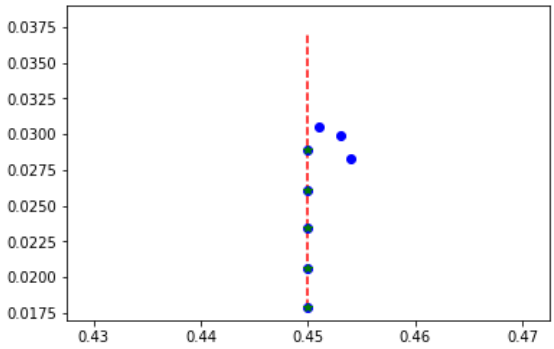

Figure 6. Gru Cell based 3 data point prediction for sequence length of 4

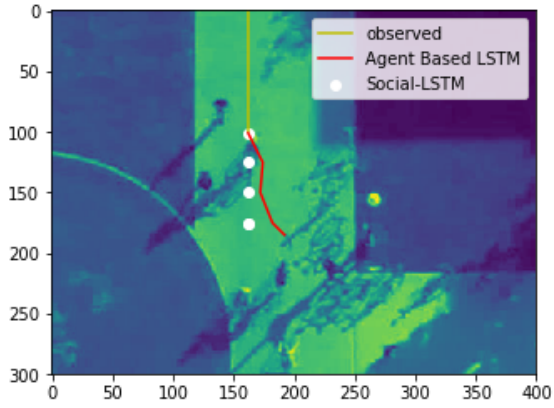

Figure 7. Gru Cell based 3 data point prediction for sequence length of 4

ing one reference pedestrian and then checking for number of neighboring pedestrians in and outside of the grid, identified by an indicator function and hence creating a social grid of size [mnp,mnp,$grid_size^2$], where mnp is maximum number of pedestrians. The outputs of both the networks are passed through a linear layer to downsample 5 gaussian parameters which is compared by the observed trajectory

	average displacement error	final displacement error
LSTM RMSprop	0.61	1.5
LSTM AdaGrad	0.8	1.47
GRU AdaGrad	0.65	1.2
Social Grid LSTM	0.59	0.7
Agent Social Grid Model with common embedding with gradient input	**0.57**	0.49
Agent Social Grid Model with independent embedding with gradient input	0.73	**0.47**
Agent Social Grid Model with independent embedding without gradient input	3.8	2.3

Table 1. error loss as observed by different models

and hence giving the output as the next datapoint.

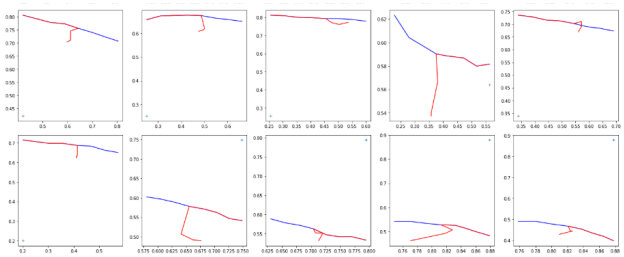

Figure 8. observed and predicted trajectory with VLSTM. Red path is ground truth, orange is predicted and Blue path is observed. As VLSTM doesn't take into account social context, hence there were inaccuracies observed.

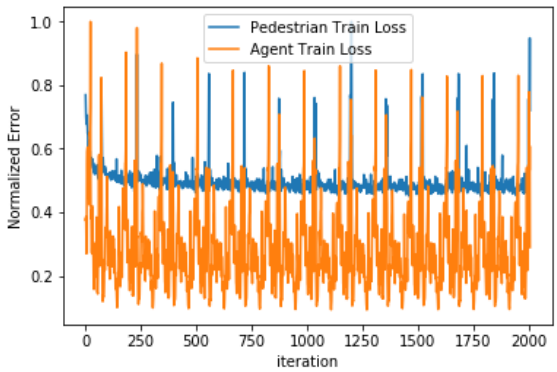

Figure 9. Training Loss for Bike and Pedestrian

6. Failure Modes

One of the main disadvantages of our proposed model is that due to the variation in dataset, the model is blind about whats around but is focused on the movement and behaviors of pedestrians. Hence for any scenario given to it with how disturbances move, the model can predict its own path. Not all scenarios lead to collision hence the predicted path needs to be a fit line. Like straight trajectories might not be handled easily by this model, but tuning and normalizing would make it do so.

7. Conclusion

The model has improved predictions as compared to vanilla LSTM models. The path observed and predicted are representative of human behaviors which shows potential for human characteristic determinations. While training Agent Based Social Grid LSTM model, a peculiar thing was observed. One every video dataset, while running the algorithm, the training loss for the disturbances (pedestrians) showed a downward trend and reached a minima. But for the agent (bikers), we observed that the training loss showed a sinusoidal curve. Hence, we reached a conclusion that since the training is occurring in individual grids surrounding the agent, and as the movement of pedestrians is not static, but dynamic, hence at every time instant, number of pedestrians entering and exiting the pedestrian grid would differ, and hence, with respect to every pedestrian trajectory, the pedestrian weights would be updated so as to chart a new path to avoid collision with the pedestrian.

In one of the scenarios it was observed that the predicted path for the agent bike showed curved trajectory of four points which says that the model was able to predict the movement of the pedestrian coming towards it. In future, a clustered analysis of predicted paths and actual trajectories can be used to analyze the human behavioral tendency. Different loss and accuracy metrics can be used to inspect and improve the efficiency of the model and implementation of multi layered bi-directional LSTM also with HMM for probabilistic estimate of direction change and collision statistics can be implemented.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[2] B. Cheng. Pedestrian trajectory prediction via the social-grid lstm model. *The Journal of Engineering*, 2018:1468–1474(6), November 2018.

[3] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, May 1995.

[4] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras. People tracking with human motion predictions from social forces. In *2010 IEEE International Conference on Robotics and Automation*, pages 464–469, May 2010.

[5] H. Manh and G. Alaghband. Scene-lstm: A model for human trajectory prediction. *CoRR*, abs/1808.04018, 2018.

[6] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016.