Supplementary for Structural-RNN: Deep Learning on Spatio-Temporal Graphs

Ashesh Jain[1,2], Amir R. Zamir[1], Silvio Savarese[1], and Ashutosh Saxena[3]

Stanford University[1], Cornell University[2], Brain Of Things Inc.[3]
{ashesh,zamir,ssilvio,asaxena}@cs.stanford.edu

4. Experiments

4.1. Human motion modeling and forecasting

User study. We randomly sampled three seed motions from each of the four activities (walking, eating, smoking, and discussion), giving a total of 12 seed motions. We forecasted human motion from the seeds using S-RNN, LSTM-3LR and ERD, resulting in total of 36 forecasted motions – equally divided across algorithms and activities. We asked five users to rate the forecasted motions on a Likert scale of 1 – 3, where a score of 1 is bad, 2 is neutral, and 3 is good. The users were instructed to rate based on how human like the forecasted motion appeared. In order to calibrate, the users were first shown many examples of ground truth motion capture videos.

Figure 1 shows the number of examples that obtained bad, neutral, and good scores for each algorithm. Majority of the motions generated by S-RNN were of high-quality and resembled human like motion. On the other hand, LSTM-3LR generated reasonable motions most of the times, however they were not as good as the ground truth. Finally, the motions forecasted by ERD were not human like for most of the aperiodic activities (eating, smoking, and discussion). On the walking activity, all algorithms were competitive and users mostly gave a score of 3 (good). Hence, through the user study we validate that S-RNN generates most realistic human motions majority of the times. Look at the supplementary video for more details.

Training S-RNN for motion forecasting. We closely follow the training procedure by Fragkiadaki et al. [1]. We cross-validate over the hyperparameters on the validation set and set them to the following values:

- Back propagation through 100 time steps.
- Mini-batch size of 100 sequences.
- We use SGD and start with the step-size of 10^{-3}. We decay the step-size by 0.1 when the training error plateaus.
- We clip the L2-norm of gradient to 25.0, and clip each dimension to [-5.0, 5.0]
- We gradually add Gaussian noise to the training data following the schedule: at iterations {250, 500, 1000, 1300, 2000, 2500, 3300} we add noise with standard deviation {0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7}. As also noted by Fragkiadaki et al. [1], adding noise is very important during training in order to forecast motions that lie on the manifold of human-like motions.

Figure 2 examines the test and train error with iterations. Both S-RNN and ERD converge to similar training error, however S-RNN generalizes better with a smaller test error for next step prediction. The number of parameters in S-RNN are marginally more than ERD. S-RNN have more LSTMs than ERD, but each LSTM in S-RNN is half the size of the LSTMs used in ERD. For ERD we used the best set of parameters described in [1]. There, the authors cross-validated over model parameters. In the plot, the jump in error around iteration 1500 corresponds to the decay in step size. Due to addition of noise the test error of S-RNN exhibits a small positive slope, but it always stays below ERD.

4.4. Driving maneuver anticipation

We now present S-RNN for another application which involves anticipating maneuvers several seconds before they happen. For example, anticipating a future lane change maneuver several seconds before the wheel touches the lane markings. This problem requires spatial and temporal reasoning of the driver, and the sensory observations from inside and outside of the car. Jain et al. [2] represent this problem with the st-graph shown in Figure 3c. They model the st-graph as a probabilistic Bayesian network called AIO-HMM. The st-graph represents the interactions between the observations outside the vehicle (eg. the road features), the driver's maneuvers, and the observations inside the vehicle (eg. the driver's facial features). We model the same st-graph with S-RNN architecture using the node and edge features from Jain et al. [2].

Table 1: **Maneuver Anticipation on 1100 miles of real-world driving data**. S-RNN is derived from the st-graph shown in Figure 3c. Jain et al. [2] use the same st-graph but models it in a probabilistic frame with AIO-HMM. The table shows average *precision*, *recall* and *time-to-maneuver*. Time-to-maneuver is the interval between the time of algorithm's prediction and the start of the maneuver. Algorithms are compared on the features from [2].

Method	Turns			Lane change			All maneuvers		
	Pr (%)	Re (%)	Time-to-maneuver (s)	Pr (%)	Re (%)	Time-to-maneuver (s)	Pr (%)	Re (%)	Time-to-maneuver (s)
SVM	64.7	47.2	2.40	73.7	57.8	2.40	43.7	37.7	1.20
AIO-HMM [2]	80.8	75.2	4.16	83.8	79.2	3.80	77.4	71.2	3.53
S-RNN w/o edgeRNN	75.2	75.3	3.68	85.4	**86.0**	3.53	78.0	71.1	3.15
(Ours) S-RNN	**81.2**	**78.6**	3.94	**92.7**	84.4	3.46	**82.2**	**75.9**	3.75

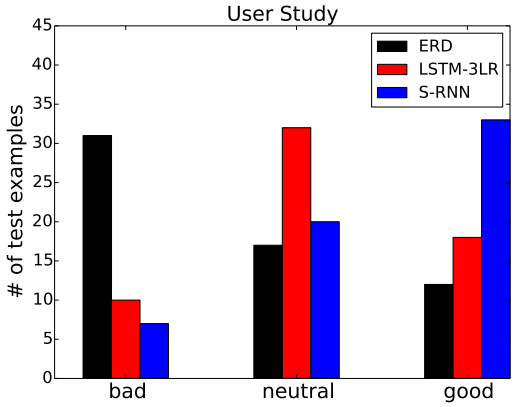


Figure 1: **User study** with five users. Each user was shown 36 forecasted motions equally divided across four activities (walking, eating, smoking, discussion) and three algorithms (S-RNN, ERD, LSTM-3LR). The plot shows the number of bad, neutral, and good motions forecasted by each algorithm.

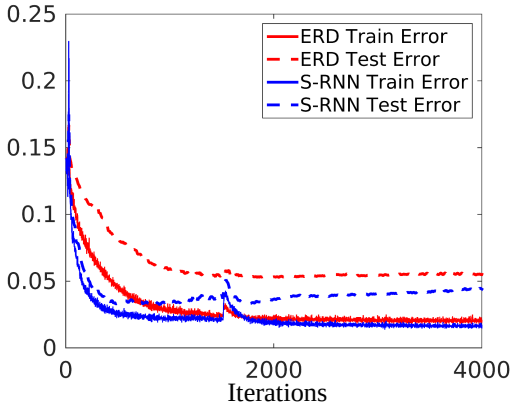
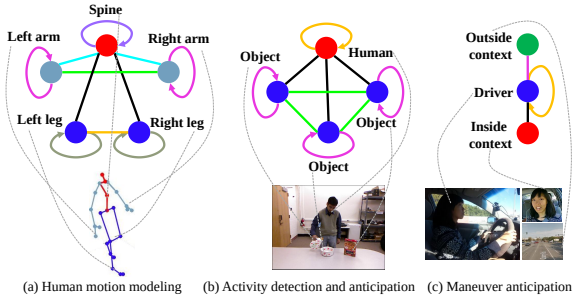


Figure 2: **Train and test error**. S-RNN generalizes better than ERD with a smaller test error.

The nodeRNN models the driver, and the two edgeRNNs model the interactions between the driver and the observations inside the vehicle, and the observations outside the vehicle. The driver node is labeled with the future maneuver and, the observation nodes do not carry any label. The output of the driver nodeRNN is softmax probabilities of the following five maneuvers: {*Left lane change, right lane change, left turn, right turn, straight driving*}. Our nodeRNN architecture is RNN(64)-softmax(5), and edgeRNN is LSTM(64).



(a) Human motion modeling (b) Activity detection and anticipation (c) Maneuver anticipation

Figure 3: **Diverse spatio-temporal tasks**. We apply S-RNN to the following three diverse spatio-temporal problems. (View in color)

We train S-RNN on the features provided by Jain et al. [2] on their 1100 miles of natural driving data set. The algorithms are evaluated on their precision and recall in anticipating maneuvers under the following three prediction settings: (i) Lane change: algorithms only anticipate lane changes. This setting is relevant for freeway driving; (ii) Turns: algorithms only anticipate turns; and (iii) All maneuvers: algorithms anticipate all five maneuvers. Table 1 shows the performance of different algorithms on this task. S-RNN performs better than the previous state-of-the-art AIO-HMM [2] in every setting. It improves the precision by 5% and recall by 4% with predicting all five maneuvers. Both AIO-HMM and S-RNN model the same st-graph but using different techniques. The table also shows that the performance decreases if we remove edgeRNNs and simply feed the concatenation of edge features into the nodeRNN. This emphasizes importance of the edgeRNNs, and the need for separately modeling different kinds of edge interactions.

References

[1] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *ICCV*, 2015. 1

[2] A. Jain, H. S. Koppula, B. Raghavan, S. Soh, and A. Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *ICCV*, 2015. 1, 2