CS230

Probabilistic and Multimodal Trajectory Predictions for Autonomous Driving

Philippe Weingertner
Department of Computer Science
Stanford University
pweinger@stanford.edu

Abstract

We study the problem of Trajectory Predictions for Autonomous Driving which is of paramount importance to support Decision Making and Planning. We investigate different architectures: RNN based sequence to sequence and attention models but also and for the first time to the best of our knowledge, we study the customization and applicability of Transfomer models to trajectory predictions. We enhance with Spatial Attention the Convolutional Social Poling layer of a state-of-the-art architecture. We improve over the state-of-the-art baseline by 10%.

1 Introduction

We investigate the problem of Trajectory Predictions for Autonomous Driving [8] with the ultimate goal of supporting and improving Decision Making and Planning. We are interested by trajectory predictions that are: **probabilistic** to account for uncertainty so we can later on plan accordingly, **multimodal** to account for different possible maneuvers, as we want to better anticipate a set of possible trajectories to enable safer human-like anticipation, and **over 0s - 5s time horizon**.

Most of the studies focus on some specific parts of the problem. Some studies focus on short-term predictions and physical features mainly ignoring Behavioral features [1]. Other studies focus on capturing one or a few realistic human like driving models for later use in simulation [6]. We want to study and benchmark different candidate architectures and investigate how to best combine physical, contextual and behavioral features in a single architecture suitable for online 0s - 5s predictions.

2 Related work

We explore how to improve trajectory predictions over existing publications with state of the art results [7] and especially [4]. We use the later reference as a baseline and starting point for experiments. This reference has lots of features we are interested in: multimodality, probabilistic, reference implementation available, state of the art results published.

We investigate sequence to sequence models [13][10], attention mechanisms [3][9] to let the RNN decoder learn to focus over a specific range of the input sequence and transformer models [14]. While these techniques have been mainly developed to improve machine translation we believe they could be useful for trajectory predictions as in [12][10][1].

CS230: Deep Learning, Spring 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Dataset and Features

We use from the NGSIM dataset the US Highway 101 dataset (US-101) and Interstate 80 Freeway dataset (I-80).

The datasets made up of 6 recordings of 15 minutes each is captured from a bird's-eye view of the highway with a static camera at 10 Hz. In total we have 8.3 millions samples split into 70% for the training set, 10% for the development set and 20% for the test set. This split was used in [4] and [5] which makes it our reference. To predict the trajectory over next 5 seconds, we use past 3 seconds (x, y) information about a vehicle and past 3 seconds of a $(13, 3)$ occupancy grid around that vehicle; where each grid cell is of the size of a standard vehicle.

4 Methods

4.1 Loss function

We account for the probabilistic and multimodal aspects of trajectory predictions by combining one regression loss with two classification losses.

We predict a 2D trajectory over a 5 seconds horizon with a probabilistic model: at each time step, a 5D vector corresponding to the parameters of a bivariate Gaussian distribution is derived, giving the distribution of the future locations of the predicted vehicle at that time step. We are looking for the parameters of the following model:

$$f\left(x,y\right)=\frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}}\exp\left(-\frac{1}{2\left(1-\rho^2\right)}\left[\frac{\left(x-\mu_x\right)^2}{\sigma_X^2}+\frac{\left(y-\mu_Y\right)^2}{\sigma_Y^2}-\frac{2\rho\left(x-\mu_X\right)\left(y-\mu_Y\right)}{\sigma_X\sigma_Y}\right]\right)$$

i.e. the 5 parameters defining the mean vector and the covariance matrix:

$$\mu=\begin{bmatrix}\mu_X\\ \mu_Y\end{bmatrix},\mathbf{\Sigma}=\begin{bmatrix}\sigma_X^2 & \rho\,\sigma_X\sigma_Y\\ \rho\,\sigma_X\sigma_Y & \sigma_Y^2\end{bmatrix}\text{ with -1}\leq\rho\leq\text{1, }\sigma_X\text{>0, }\sigma_Y\text{>0}$$

On top of the RNN LSTM decoder (cf figure 2), at every time step, we use a linear layer converting 128 features out of the RNN LSTM output to 5 values y_1, y_2, y_3, y_4, y_5 followed by a non linear layer such that $\mu_X = y_1, \mu_Y = y_2, \sigma_X = \exp\left(y_3\right), \sigma_Y = \exp\left(y_4\right), \rho = \tanh\left(y_5\right)$.

We minimize the Negative Log Likelihood of $f\left(x,y\right)$:

$$L_{\text{nll}}(\text{targ=}\begin{bmatrix}x\\ y\end{bmatrix},\text{pred=}\begin{bmatrix}\mu_X\\ \mu_Y\\ \sigma_X\\ \sigma_Y\\ \rho\end{bmatrix})=\log\left(\sigma_X\sigma_Y\sqrt{1-\rho^2}\right)+\frac{1}{1-\rho^2}\left[\frac{\left(x-\mu_x\right)^2}{\sigma_X^2}+\frac{\left(y-\mu_Y\right)^2}{\sigma_Y^2}-\frac{2\rho(x-\mu_X)(y-\mu_Y)}{\sigma_X\sigma_Y}\right]$$

For maneuver predictions we use 2 cross-entropy loss functions, one for lateral maneuver classification (we are using 3 classes: Left-Change-Line, Right-Change-Line, No-Change-Line) and one for longitudinal maneuver classification (we are using 2 classes corresponding to brake lights indication: braking or not braking) that are added to above NLL loss function. $L_{\text{Crossent-lat}} = -\sum_{i=1}^{3} y_i \log \hat{y}_i$ and $L_{\text{Crossent-lon}} = -\sum_{i=1}^{2} y_i \log \hat{y}_i$. Ultimately the loss function is a combination of a regression loss with 2 classification losses: $\text{Loss} = L_{\text{nll}} + L_{\text{Crossent-lat}} + L_{\text{Crossent-lon}}$.

4.2 Neural network architecture

4.2.1 Neural network architectures

The Baseline is a state-of-the-art architecture described in [4]. The models we propose to investigate and benchmark against the above baseline are: a sequence to sequence encoder-decoder (seq2seq) with or without Attention mechanisms as described in figure 1 and a Transformer Model, initially introduced in [14], providing state of the art results for machine translation.

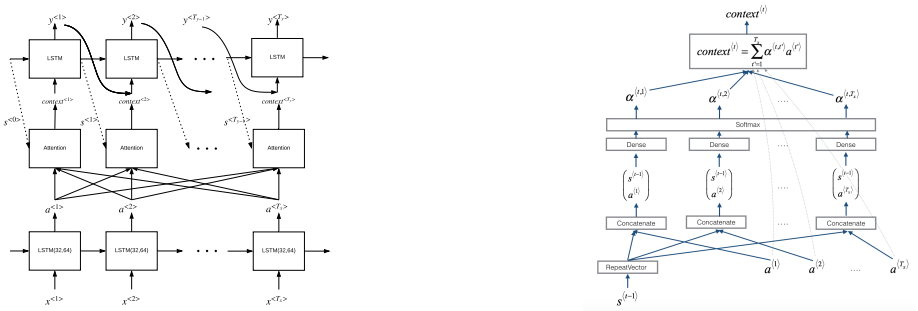

Figure 1: Seq2seq with attention

4.2.2 Convolutional social pooling with spatial attention

We enhance the CS-LSTM(M) architecture with Spatial Attention mechanisms so we can deal with weighted interactions in the convolutional social pooling layer. Like a human driver we do not focus equally on every neighbours and we would like to learn the best attention weights depending on the spatio-temporal relationships of the objects. We leverage on an extended set of features to capture the weights. We refer to this pipeline as CSSA-LSTM(M) for Convolutional Social pooling with Spatial Attention.

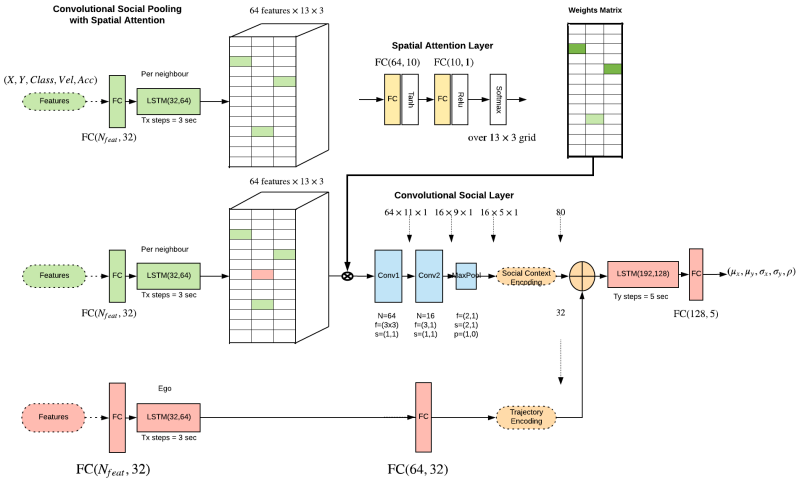

Figure 2: CSSA-LSTM(M) Model

Every Fully Connected, LSTM and Convolutional layers, with the exception of the Spatial Attention layer, are followed by a leaky relu non-linearity of coefficient 0.1 . For the convolutional layers, N stands for the number of filters, f for the filter size, s for stridding and p for padding.

5 Experiments/Results/Discussion

The gihub repo is cs230-project.

5.1 Transformer experiments

The Transformer has been extensively optimized for Machine Translation tasks. One of the author, Lukasz Kaiser refers to a list of tricks that are required to make it work. The Learning Rate (LR) strategy is very unusual, it is recommended to increase regularly the LR before decreasing it with a rate proportional to step$^{-0.5}$, label smoothing is used, an auto-regressive decoding is required and used with beam search and length penalties. Checkpoints averaging is also used to improve

results. Dropout during training at every layer just before residual connections is required and teacher forcing is used to speed up convergence. In our case, we are dealing with a regression problem so we have a customized generator layer and we have integrated convolutional processing in the embeddings layer to process 2D features. Teacher forcing overfits a lot; similar effect is observed with RNN-LSTM. We get the training working with regular Adam setting but the convergence is slower with Transformer compared to RNN-LSTM. With the Transformer it is recommended [11] to use a batch size as big possible. We use a size of 1024, limited by the 16GB memory of a GPU V100 card. With RNN-LSTM we use a batch size of 128.

In terms of network parameters, in the Transformer publication [14] the recommended settings are: $N_{layers} = 6, d_{model} = 512, d_{feed-forward} = 2048, h_{heads} = 8$ corresponding to a very big network trained with 8 GPUs in parallel. With such a network, the training on a single GPU was too slow. Training with $N_{layers} = 2$ provided good results on the Training Set but on the Validation Set results are better with $N_{layers} = 1$. With a much smaller dataset than [14], we tend to overfit even if we are using dropouts. Finally we use a configuration with $N_{layers} = 1, d_{model} = 256, d_{feed-forward} = 256, h_{heads} = 4$.

5.2 RNN-LSTM experiments

For the RNN-LSTM architecture we tried several modifications of the state-of-the-art baseline. CS-LSTM(M) is not a proper sequence to sequence architecture as the decoder output $y^{<i>}$ is not used to fed the next input $x^{<i+1>}$: the same encoder output is used for every input $x^{<i>}$ of the decoder, at every time step. But using a seq2seq architecture, a bidirectional encoder, additional layers, increasing the decoder size and varying the default settings of CS-LSTM(M) was not useful. The modifications were tested in isolation and the results were almost always similar or worst. The architecture was most probably already very well optimized. The pure temporal attention mechanisms in our context are less useful than for machine translation: the input length is relatively small with $T_x = 16$, as a down sampling factor of 2 is used for the 3 seconds of history data recorded at 10 Hz. Also by construction a trajectory has much weaker long term dependencies than sentences. On the other hand spatial attention is more useful. Like a human driver we do not focus equally on every neighbors and we learn the best attention weights depending on the relationships of the objects and additional features related to behavior and shapes.

5.3 Qualitative Results

We visualize the decoding results on NGSIM test set. An example is provided below. A trajectory with an associated probability is predicted for every possible maneuver. For every possible maneuver a set of 25 tuples, one tuple every 200 ms, is generated: $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)$. Every tuple corresponds to the parameters of a bivariate Gaussian distribution. On the figure below we visualize such a distribution for a point at a 3 seconds time horizon. The red points correspond to the predicted trajectory while the green points correspond to the ground truth. We can check that the level of uncertainty is bigger for the longitudinal prediction than for the lateral position: which makes sense given the road geometry and traffic flow. This point is important to keep in mind when looking at RMSE results.

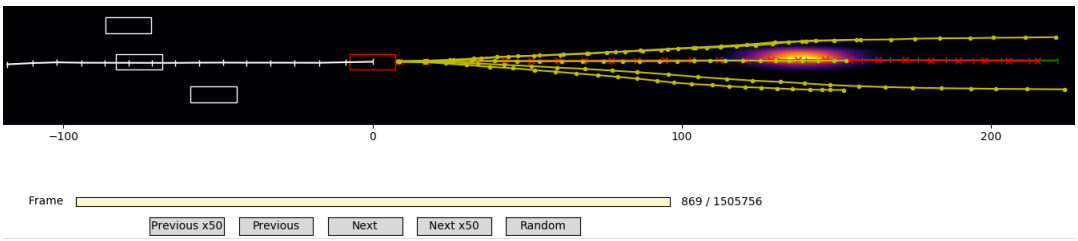

Figure 3: Probabilistic and Multimodal Model

5.4 Quantitative Results

The different architectures without enhanced features pipeline provide similar results. But the RNN-LSTM variants are 10 times smaller in size, faster too train (time per epoch) and faster to converge than the Transformer (fewer epochs). We nevertheless have to be careful before drawing firm conclusions: the NGSIM dataset is relatively small (less than 10 million samples) and some big network architectures shine with lots of data. Finally the results are improved by injecting additional features processing capabilities. The Spatial Attention is a natural extension to achieve human like driving capabilities. We use additional features related to the shape and class of objects to better weight the interactions.

Table 1: RMSE (m) Benchmark results

Prediction horizon	CV	Baseline CS-LSTM(M) [4]	Seq2seq	Transformer	**CSSA-LSTM(M)**
1 sec	0.73	0.58	0.59	0.52	**0.42**
2 sec	1.78	1.27	1.28	1.23	**1.06**
3 sec	3.13	2.12	2.14	2.17	**1.85**
4 sec	4.78	3.19	3.25	3.23	**2.85**
5 sec	6.68	4.51	4.59	4.70	**4.11**

We also quantify the benefits of using a grid of past 3 seconds trajectories encoded by a LSTM network over a raw occupancy grid 4:

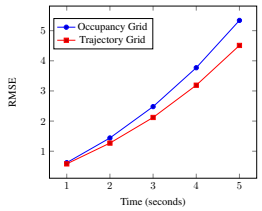

Figure 4: RMSE per grid type

We keep on reasoning with unquantized (X, Y) coordinates with a 13×3 grid of trajectories: this may be the main benefit over an history of a raw 13×3 occupancy grid.

A remaining problem with the existing CS-LSTM(M) or CSSA-LSTM(M) trajectory grid mechanism is that due to the grid quantification, some vehicle trajectories may be ignored: 2 cars may spread over a same grid cell but a single trajectory is accounted for by grid cell. This problem could be solved by using either a lower granularity grid or 3D convolutions, by stacking several trajectories per grid cell.

We present below the inference time results obtained on different hardware: making predictions for 16 or 6 targets in parallel by using the full pipeline: RNN-LSTM encoder-decoder + social pooling + spatial attention mechanisms. The input consists of 3 seconds past trajectories and the output corresponds to 5 seconds predictions. Every prediction is multimodal and probabilistic: with a $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)$ vector predicted for every 25 time steps (1 every 200 ms) over 5 seconds future trajectories, for each one of the 6 possible maneuvers, for each of one of the 16 targets. The metric obtained with the Nvidia GPU GTX1050 card is of particular interest as the performance of this GPU is very close to what is achievable with 1 of the GPUs of a Drive PX2 board: based on hardware specifications and based on previous benchmark results done for semantic segmentation networks.

Table 2: CSSA-LSTM(M) Inference time results

Predictions	GTX1080TI	**GTX1050**	**Drive PX2**	iCore7@4GHz	iCore5@2.5GHz
16 targets	7 ms	**9 ms**		20 ms	25 ms
6 targets	5.5 ms	**7 ms**		16 ms	19 ms

The RNN-LSTM Network Model used is of size 480 KBytes in ONNX format: using 115018 Float32 parameters. A typical memory and speed (lower bandwidth) optimization consists in doing Int8 weights quantization with usually a marginal performance degradation as reported in TensorRT optimizations.

The network architecture for CSSA-LSTM(M) is:

```
highwayNet(
  (ip_emb): Linear(in_features=5, out_features=32, bias=True)
  (ip_behav_emb): Linear(in_features=5, out_features=32, bias=True)
  (enc_behav_lstm): LSTM(32, 64)
  (op_att1): Linear(in_features=64, out_features=10, bias=True)
  (op_att2): Linear(in_features=10, out_features=1, bias=True)
  (enc_lstm): LSTM(32, 64)
  (dyn_emb): Linear(in_features=64, out_features=32, bias=True)
  (soc_conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
  (soc_conv_3x1): Conv2d(64, 16, kernel_size=(3, 1), stride=(1, 1))
  (soc_maxpool): MaxPool2d(kernel_size=(2, 1), stride=(2, 1), padding=(1, 0))
  (dec_lstm): LSTM(117, 128)
  (op): Linear(in_features=128, out_features=5, bias=True)
  (op_lat): Linear(in_features=112, out_features=3, bias=True)
  (op_lon): Linear(in_features=112, out_features=2, bias=True)
  (leaky_relu): LeakyReLU(negative_slope=0.1)
  (relu): ReLU()
  (softmax): Softmax()
)
```

6 Conclusion/Future Work

We came up with a detailed analysis of trajectory predictions for Autonomous Driving dealing with probabilistic and multimodal predictions. We explored and benchmarked different architectures. We demonstrated the applicability of transformer models to this problem with results that are close to state-of-the-art RNN-LSTM results. The transformer models are mainly used and heavily customized for Machine Translation: they are quite difficult and specific to tune. We can expect that there is still room for improvement. We enhance with Spatial Attention an existing state-of-the-art CS-LSTM(M) model. We call it CSSA-LSTM(M). We improve over the state-of-the-art baseline by 10%. The NGSIM dataset is relatively limited in terms of variety of interactions: it is limited to highway and freeway driving with cars, trucks and motorbikes. As a consequence future work would require experimenting in more heterogeneous environments like urban cities where Spatial Attention should be even more relevant.

Acknowledgements

We would like to thank our TA Ashwin Sreenivas and all the teaching staff of CS230 for their support and guidance via lectures, office hours and section hours.

References

[1] Florent Altché and Arnaud de La Fortelle. An LSTM network for highway trajectory prediction. *CoRR*, abs/1801.07962, 2018. 1, 2

[2] Ernest Cheung, Aniket Bera, Emily Kubin, Kurt Gray, and Dinesh Manocha. Identifying driver behaviors using trajectory features for vehicle navigation. *CoRR*, abs/1803.00881, 2018.

[3] K. Cho D. Bahdanau and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015. 2

[4] N. Deo and M.M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. *CVPR*, 2018. 2, 3, 4.2.1, 1

[5] Nachiket Deo and Mohan M. Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. *CoRR*, abs/1805.05499, 2018. 3

[6] Alex Kuefler, Jeremy Morton, Tim Allan Wheeler, and Mykel John Kochenderfer. Imitating driver behavior with generative adversarial networks. *CoRR*, abs/1701.06699, 2017. 1

[7] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher Bongsoo Choy, Philip H. S. Torr, and Manmohan Krishna Chandraker. DESIRE: distant future prediction in dynamic scenes with interacting agents. *CVPR*, 2017. 2

[8] S. Lefevre, C. Sun, R. Bajcsy, and C. Laugier. Comparison of parametric and non-parametric approaches for vehicle speed prediction. *ACC*, 2014. 1

[9] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. 2

[10] SeongHyeon Park, Byeongdo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. *CoRR*, abs/1802.06338, 2018. 2

[11] Martin Popel and Ondrej Bojar. Training tips for the transformer model. *CoRR*, abs/1804.00247, 2018. 5.1

[12] Oliver Scheel, Naveen Shankar Nagaraja, Loren Arthur Schwarz, Nassir Navab, and Federico Tombari. Attention-based lane change prediction. *CoRR*, abs/1903.01246, 2019. 2

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. 2

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 2, 4.2.1, 5.1