Learning to Predict Driving States using Generative Adversarial Networks and Dashboard Mounted Multiple Sensors towards Autonomous Driving Assistance

Thesis Report Submitted in Partial Fulfillment
of the Requirements for the Award of the Degree of

Bachelor of Technology

by

Rahul Kumar Singh
13EE10065

Under the supervision of

Dr. Debdoot Sheet

Department of Electrical Engineering
Indian Institute of Technology Kharagpur
December 2016

Abstract

Autonomous driving in cars are gaining a centre-stage, as prospects of the future is triggering significant research in the machine learning and vision community. Research areas for autonomous car driving range from predicting driving trajectory to predictive simulation of driving conditions and generation of large multi-sensor datasets along with preparing benchmarks of autonomous drivability tests for special hardware-software suite development to facilitate such research. In order to make a conventional fuel based car with an internal combustion engine autonomous, we generally would like to control the gas pedal (accelerator), clutch, gear lever, brake pedal and the steering wheel based on driving condition inputs from car mounted cameras, light detection and ranging (LIDAR), radio detection and ranging (RADAR), accelerometer, gyroscope, magnetometer and GPS sensors. In this project, we are currently extending on the prior art for predictive modeling of the other driving states like gas pedal (accelerator), clutch, gear lever, brake pedal press using multi-sensor integration for modeling of these states. The publicly available driving dataset consisting of 7.25 hours of highway rich quality video feed with other sensor data like speed, acceleration, 3D LIDAR along with car driving control variable data for the training. Currently we have implemented the steering angle predicting by training on 9 videos set and predicting on 2 videos sets to obtain a MSE of $\mathbf{1 \times 10^{-2}}$

Keywords: Autonomous driving, autoencoder, convolutional neural networks, driving state estimation, long-short term memory, transfer learning, temporal modelling.

Contents

1 Introduction

AUTONOMOUS cars are future of mobility and field of deep learning (LeCun and Hinton, 2015) is helping in paving a way for solving core problems associated with it. Many automobile companies as well as scientific groups around the world are actively working in this field. Companies like NVidia[1] (Bojarski, 2016), Ford, Tesla (Kessler, 2015), BMW [2] are competing on global scale while companies like comma.ai[3], appleare developing software tools for autonomous cars. Car driving conditions are generally grouped as dense urban journey and initiate/terminate conditions when speed ranges in 0-10 kmph and is good for real-time processing in any environment, but with high dynamics in the environment where travel speed varies from 10 to 50 kmph with lots of unpredictability in the driving environment makes it quite challenging to model; however,cruising speeds of 50-100 kmph on the highways where less unpredictability of driving conditions assists in less challenging situations to model.

Car driving is a set of skills which requires controlling steering, transmission (gear, gas) and break. An autonomous car requires prediction of these driving states given the input from the sensors. Traditionally, approach towards self-driving cars was driven more in complexly coded structure which accounted enormous number of possibilities which can occur during driving. Vision has been an important sensory data which has been employed in self-driving cars (Stavens, 2011). But, it becomes nearly difficult to code the highly dynamic and complex vehicular environment from visual data. The recent growth of research in deep learning has led to a paradigm shift in the approach for self-driving cars. Earlier, more time was consumed in analysis of different situations (implementation of multiple algorithms) which prevented real-time implementation. Now, the trend has shifted, i.e. more time is spent during training while minimizing the time at the time of testing. With powerful GPUs coming in (NVidia e.g.), parallelism and computation hungry task are dealt more efficiently.

Deep learning uses the property that many natural images are constituted in a hierarchial order and the neighbouring pixels are highly correlated. The hierarchincal order consists of high level features like square consisting of sub low level features like egdes, lines. New deep learning techniques like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Auto encoders and their variants have helped in modeling complex sensory rich data and generate useful commands. Lot of research is being done is this field where variety of techniques using combination of these networks is being used. Donahue (2015) have used long-term recurrent convolutional network for visual recognition and description task while (A. Diaz Alvarez and Jimenez, 2014) used neural networks for modelling driver behaviour and (Wang and Lukic, 2011) has used driving style recognition for improving driving efficiency by increasing the battery performance of a hybrid vehicle. These techniques are data centric i.e. convergence is very much dependent on input data. Many open source datasets are available kitty (A. Geiger, 2012), comma.ai and udacity. These datasets are taken in urban environment on highways.

Application of this techniques include path planning for semi-autonomous vehicles, efficient

[1] http://www.nvidia.com/object/drive-automotive-technology.html
[2] http://www.bmw. com/com/en/insights/technology/connecteddrive/ 2010/future
[3] http://research.comma.ai/

fuel usage in hybrid cars by minimizing the switching between fuel and battery. This will be useful in design of Self driving cars as vast dataset from human driving can be used to train the system to operate in different environmental and road conditions.

2 Background

2.1 Autoencoder

An autoencoder module consists of encoder and a decoder module. Encoder module converts a high dimensional data, for example, an image to a low dimensional latent space. Then the decoder module tries to reconstructs back the input data from the latent space. Hinton and Salakhutdinov (2006) show any effective way of weight initialization for autoencoder training. The encoder and the decoder can be defined as transitions ϕ and φ respectively such that,

$$\phi : \chi \rightarrow \Theta \tag{1}$$

$$\varphi : \Theta \rightarrow \chi \tag{2}$$

$$\arg\min_{\phi,\varphi} \parallel X - (\phi \circ \varphi)X \parallel^2 \tag{3}$$

Eq. (3) determines mean squared error between input and reconstructed data is often used as a cost function assuming probability distribution of the error signal is Gaussian. In (1), χ represents input space where as Θ represents the latent space. There are many variants of autoencoders for example sparse autoencoder (Ng, 2011), denoising autoencoder (Vincent, 2010) and variational autoencoders (D. P. Kingma, 2013).

2.2 Convolutional Neural Networks

Convolutional networks are inspired by biological processes and are very similar to ordinary neural networks and been employed in many image classification tasks (Krizhevsky and Hinton, 2012) . They are made up of neurons that have learnable weights and biases. ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. CNNs are difficult to train as they typically contain a high number of unknowns.

2.3 Long-Short Term Memory

In a traditional recurrent neural network, during the gradient back-propagation phase, the gradient signal can end up being multiplied a large number of times (as many as the number of time-steps) by the weight matrix associated with the connections between the neurons of the recurrent hidden layer. If the weights in this matrix are small (or, more formally, if the leading

eigenvalue of the weight matrix is smaller than 1.0), it can lead to a situation called vanishing gradients where the gradient signal gets so small that learning either becomes very slow or stops working altogether. It can also make more difficult the task of learning long-term dependencies in the data. Conversely, if the weights in this matrix are large (or, again, more formally, if the leading eigenvalue of the weight matrix is larger than 1.0), it can lead to a situation where the gradient signal is so large that it can cause learning to diverge. This is often referred to as exploding gradients. Traditional LSTM module with forget gates can be written as,

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{4}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{6}$$

$$c_t = f_t o c_{t-1} + i_t o \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{7}$$

$$h_t = o_t o \sigma_h(c_t) \tag{8}$$

These issues are the main motivation behind the LSTM model (F. A. Gers, 2000) which introduces a new structure called a memory cell. A memory cell is composed of four main elements: an input gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. The self-recurrent connection has a weight of 1.0 and ensures that, barring any outside interference, the state of a memory cell can remain constant from one time step to another. The gates serve to modulate the interactions between the memory cell itself and its environment. The input gate can allow incoming signal to alter the state of the memory cell or block it. On the other hand, the output gate can allow the state of the memory cell to have an effect on other neurons or prevent it. Finally, the forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state, as needed.

2.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) (I. Goodfellow, 2014) are a recent development in the field of deep learning where generative model is trained along with discriminative model on an adversary. It follows an analogy that generator module acts as a currency counterfeiter where it try to generate fake currency based on a given distribution (say Gaussian) and then the discriminator acts as the police which tries to detect it as fake or legit. The discriminator module optimizes to differentiate while generator optimizes to generate realistically image to fool the discriminator. Generative models pose the training process as a game between two separate networks: a generator network and a second discriminative network that tries to classify samples as either coming from the true distribution or the model distribution. Every time the discriminator notices a difference between the two distributions the generator adjusts its parameters slightly to make it go away, until at the end (in theory) the generator exactly reproduces the true data distribution and the discriminator is guessing at random, unable to find a difference. It leads to minimax optimization where both discriminator and generator reach to a wining probability of 0.5.

2.5 Prior Work

In a recent work by E. Santana (2016), proposed a model which uses GANs, CNNs and Variational Autoencoders (VAEs) to train an agent which learns to clone driver behaviour and plans manoeuvre by simulating future events on the road. The work aims at embedding the frames in the latent space and learning a compact intermediate representation of the data. The work is first of it type of predicting frames on which takes into account of the action conditioned transitions. Although the network was having problem with curvy roads (turns). Their results show that the model can predict realistic looking videos for several frames even when the model is not optimized using a cost function on pixel space. Their proposed model is divided into 2 sub-models i.e. an autoencoder to embed the video frames into a Gaussian latent space of 2048 and a transition model using RNN to represent the transitions.

Nvidia is actively working in the field of self-driving cars and is developing powerful GPUs for high level computing. In a recent paper published by Nvidia (Bojarski, 2016), a CNN model consisting of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers was used to model steering angle of a 2016 Lincoln or a 2013 Ford Focus using mean squared error minimization. It aims at developing a full end-to-end model for self-driving cars using deep CNNs. The model was trained on for 72 hours of driving dataset having clear, cloudy, foggy, snowy and rainy weather, both day and night. Their efforts show teaching a car how to drive using only cameras and deep learning. Their DAVE-2 deep learning system is capable of driving in many different weather conditions, avoiding obstacles, and even going off-road. For training, data from 3 cameras were used namely left, right and centre and for bias removal towards straight roads, higher proportion of data having turning was selected. For testing, the paper used simulation driven approach towards actual implementation. Their results show that the model was able to learn feature like lane marking, outline of roads, difference between unpaved road and forest. In some cases they were able to achieve 98% of autonomous time (on highways).

3 Methodology

3.1 Problem Statement

Driving a car is composed of set of states $\mathbf{W_t}$, where $\mathbf{W_t}$ represent set on states at time t. The state $\mathbf{W_t}$ can be written as $\mathbf{W_t} = \{\mathbf{s_t}, \mathbf{b_t}, \mathbf{g_t}, \mathbf{h_t}\}$ where S denotes steering wheel position, B denotes brake position, G denotes Gas pedal position and H denotes Gear position states at time t. Now, we will formulate a function for the prediction of steering wheel position at time t=t+1. Let $\mathbf{X_{t+1}} = \{\mathbf{x_{t-n+1}}, \mathbf{x_{t-n+2}}, .., \mathbf{x_t}\}$ denote the n-frame long video where $\mathbf{x_t}$ denotes the frame at t-th instant. Similarly, $\mathbf{A_{t+1}} = \{\mathbf{a_{t-n+1}}, \mathbf{a_{t-n+2}}, .., \mathbf{a_t}\}$ denotes n-frame long control signal which is the steering angle position, whereas $\mathbf{S_{t+1}} = \{\mathbf{s_{t-n+1}}, \mathbf{s_{t-n+2}}, .., \mathbf{s_t}\}$ denotes the n-frame long measured speed. Therefore, we can state a function, F which can predict the steering angle position at time t+1 such that

$$a_{t+1} = F_1(S_t, X_t, A_t) \tag{9}$$

It can be noted that this forms a sequential network whose current output depends on the previous inputs and output. Similarly, we will define $\mathbf{G_{t+1}} = \{\mathbf{g_{t-n+1}}, \mathbf{g_{t-n+2}}, .., \mathbf{g_t}\}$ for n-frame gas pedal command, $\mathbf{B_{t+1}} = \{\mathbf{b_{t-n+1}}, \mathbf{b_{t-n+2}}, .., \mathbf{b_t}\}$ for n-frame brake command and $\mathbf{H_{t+1}} = \{\mathbf{h_{t-n+1}}, \mathbf{h_{t-n+2}}, .., \mathbf{h_t}\}$ for n-frame gear position. The predicted states $\mathbf{b_{t+1}}, \mathbf{g_{t+1}}$ and $\mathbf{h_{t+1}}$ can be written as

$$b_{t+1} = F_2(B_t, G_t, S_t, X_t, A_t, H_t) \tag{10}$$

$$g_{t+1} = F_3(B_t, G_t, S_t, X_t, A_t, H_t) \tag{11}$$

$$h_{t+1} = F_4(B_t, G_t, S_t, X_t, A_t, H_t) \tag{12}$$

3.2 Proposed Model

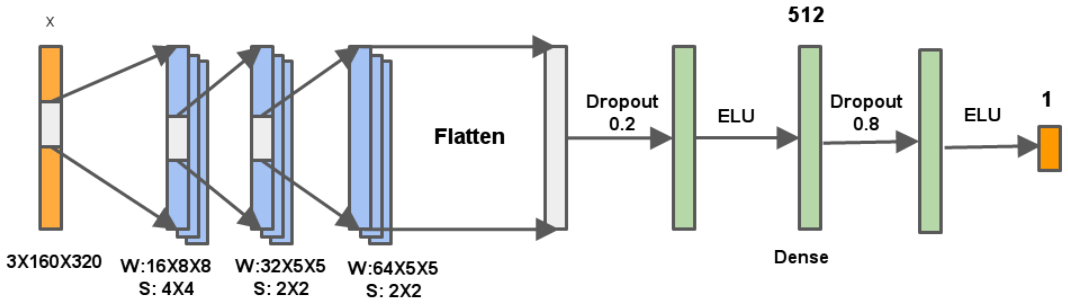


Figure 1: The Convolutional Neural Network architecture for feature extraction used in our work. The CNN model is based on the work by comma.ai

CNN Model: As formulated in the above section, we can observe that the inputs to our model have varying dimensions. The image frame is $\mathbb{R}^{160\times320\times3}$ dimensional vector while the speed, brake, gas and gear position are $\mathbb{R}^{1\times1}$ dimensional vector. Therefore, we use a CNN model which take image frame as an input and give out a feature 512 dimensional feature vector. As, this will help in reducing the very high dimensional information into a comparatively low dimension. It also help in removing the redundant information and focus only on the useful features. Therefore, for the CNN model we used a 8 layer model including 3 convolutional layers each having 3X3 dimensional convolution kernel with a stride of 4 for cnn layer2, 2 for both cnn layer 4 and layer 6. The activation function used was ReLu while a dropout of 0.2 was used before the 512 dense feature vectors.

LRCN Model: It consists of CNN model which we have described earlier upto the 512 feature vector layer along with a LSTM model. As, driving states are sequential in nature and

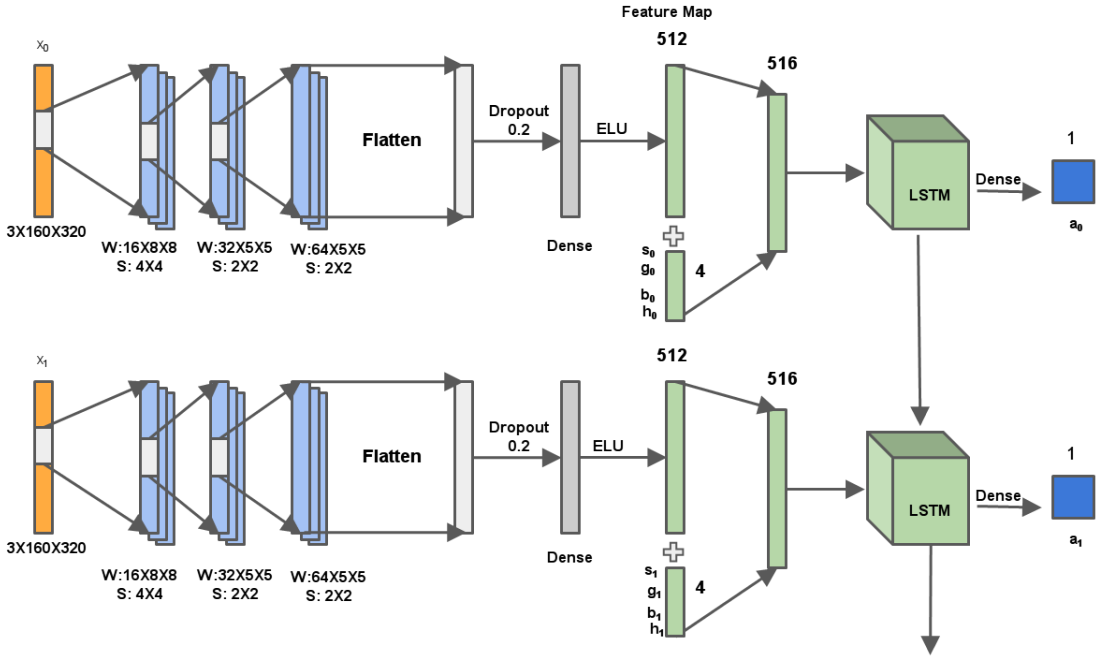


Figure 2: The deep LRCN architecture used in our work. The CNN model is adopted from the above mentioned CNN model

each state has temporal information. In order to extract the temporal relation, we added the values of brake, gas, speed and gear to the feature 512 feature vectors. Therefore, we now get 516 vectored element compressed information about the current state which will be then used for predicting the next generated command. In order to preserve the sequential information in the input, we use a LSTM model having 256 hidden layers which give a 1 dimensional dense output layer. The optimizer used for both the models is *'adam'* while the loss is minimized on mean square error.

Generative Model: The generative model architecture is based on the work by E. Santana (2016) paper on Learning a Driver Simulator. We have used the main underlying idea and model proposed in the paper. The Encoder module use 4 convolution layer with each using batch norm followed by LeakyReLu activation function. The Generator model use 3 Deconvolution layers along with batch normalization after each layer with ReLu being used as an activation function. The 4th deconvolution layer uses tanh as activation function. Lastly, the discriminator model is pretty much similar to encoder model. The learning rate used is 0.0002.

4 Experimental Result

4.1 Dataset

The dataset used in the project is taken from the the online release dataset from comma.ai. The dataset consists of 7.5 hours of driving on highways. It is divided into 11 sub-datasets where currently we are using 9 for training while other 2 for testing. The dataset is taken on Acura

Figure 3: Sample image frame taken from dataset

ILX 2016 using a Point Grey camera. All the datasets are interpolated to fit the sampling rate at 100 Hz. The sensor information include steering wheel, gear, gas pedal position, clutch, brake, radar, IMU, GPS, etc. The dataset image frames are classified as *good* and *not good* frames. Frames were steering angle is within 20 degrees are termed as *good* while rest of other frames having steering angle greater than 20 degrees are treated as *not good* frames. These data will be used for the ground truth comparison. The dataset is stored in HDF5 format which make it easy for python usage.

Apart from commai.ai dataset, we are also using the datasets provided by Udacity . The dataset provided by Udacity is also taken on 2016 Lincoln MKZ. It uses 2 velodyne VLP-16 LiDARs, 1 Delpi radar, 3 Point Grey Blackfly cameras, an Xsens IMU, an ECU for sensor data collection. As it can be observed that both the datasets i.e. by comma.ai and Udacity were taken from same car, therefore, the trained model can easily be tested on the cross dataset. Apart from that we are also planning to use Kitty dataset in training and testing of the proposed model.

4.2 Training

CNN Model Training: Following the training methodology used by comma.ai, we tried to follow a similar approach towards training. The comma.ai's 11 video datasets were divided such that 9 were used for training while 2 were used for testing. The model was trained for 200 epochs with each epoch consisting of 10000 normalized frame of dimension $3 \times 160 \times 320$. Each epoch consisted of a batch size of 64 pertaining to the resource constraints. The whole model was trained on a Nvidia GT 740 GPU with 2Gbs of memory and 12 Gbs of RAM. The whole CNN model takes around 11 hours for training.

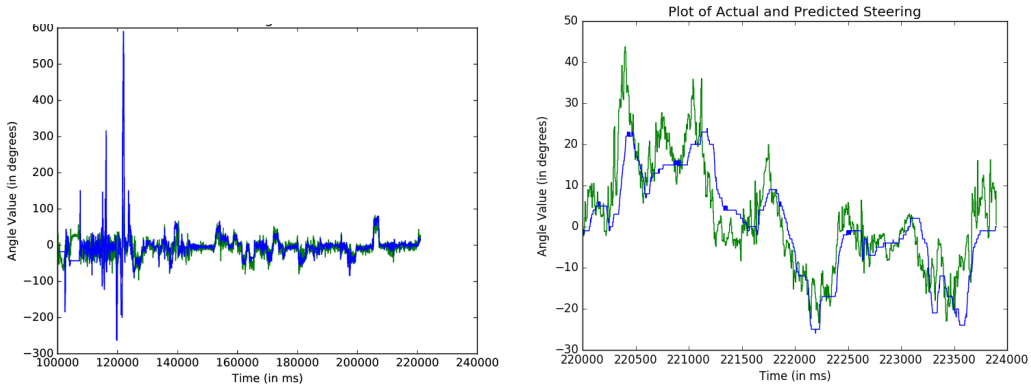
LRCN Model Training: For our LRCN model, we have used time distribution of CNN model where we used 4 CNN models in our time distributed model. Training of LRCN model is done in a divide and conquer principle with an iterative approach. As mentioned earlier, we

already have our CNN model trained. Therefore, for training LSTM model we keep the CNN model weight unchanged and train the LSTM model on top of that for 20 epochs. Each epoch consists of 10000 frames with each epoch having batch size of 32. So, each batch consists of 32 sets of 4 consecutive frames sent to time distributed CNN model. Our output of the CNN model will be $\mathbb{R}^{4\times512\times1}$ vectored feature map which is concatenated with corresponding gas, gear, speed and brake position values to give a vector of dimension $\mathbb{R}^{4\times516\times1}$ which is sent to our LSTM model. The output of our model will be a single value scalar which is our objective for optimization.

Generative Model Training: Generative model training is consists of two phases of training. Initially, autoencoder module is trained at first for 200 epochs with each epoch having 10000 images in a batch of 32. During training, three consecutive training metadata is stored. Once the autoencoder module is trained, we then start training our transition model using RNN model. Each epoch takes approximately 5 minutes to train on GT 740.

(a) Frame with label "Predicted vs Actual on a Straight road"

(b) Frame with label "Predicted vs Actual on a Curvy road"



(c) Frame with label "Plot of Predicted vs Ground Truth on whole video dataset 10"

(d) Frame with label "Plot of Predicted vs Ground Truth on frame number 220000-223000 of video dataset 10"

Figure 4: Comparison between predicted steering angle (green) vs ground truth (blue) for the video dataset 10 from comma.ai by our proposed model

4.3 Evaluation

Steering Angle

To start with, we trained our model to predict steering angle. We tested our model on video dataset 10 and 11 given by comma.ai. Fig. 2 (a) and (b) show the predicted (green line) vs the ground truth (blue line) trajectory, and it can be inferred that both are quite close while the model has some problem predicting for curve lane as it shows more error. The predicted vs ground truth angle values for all frames of video dataset 10 is plotted as shown in Fig. 2(c) while Fig. 2(d) show the time zoomed version from the frames 224000-223000. It can be seen that the model is able to estimate the steering angle. Another inference can be drawn is that the model output has high frequency jittering.

Convolutional Layers Output

Convolutional layers bring a hierarchical abstraction in the data and are output of each layers gives very critical information about the what the model is learning. Fig 3 (a), (b) and (c) show the output of convolutional layers 1, 2 and 3 respectively. It can be seen that the layer 1 learns highest level of abstraction where in some of output frames both right and left sides of lane are learnt. While, in Fig. 3(b) it can be seen that model learns lane centre, edges, boundary and other information which is hard to comprehend.

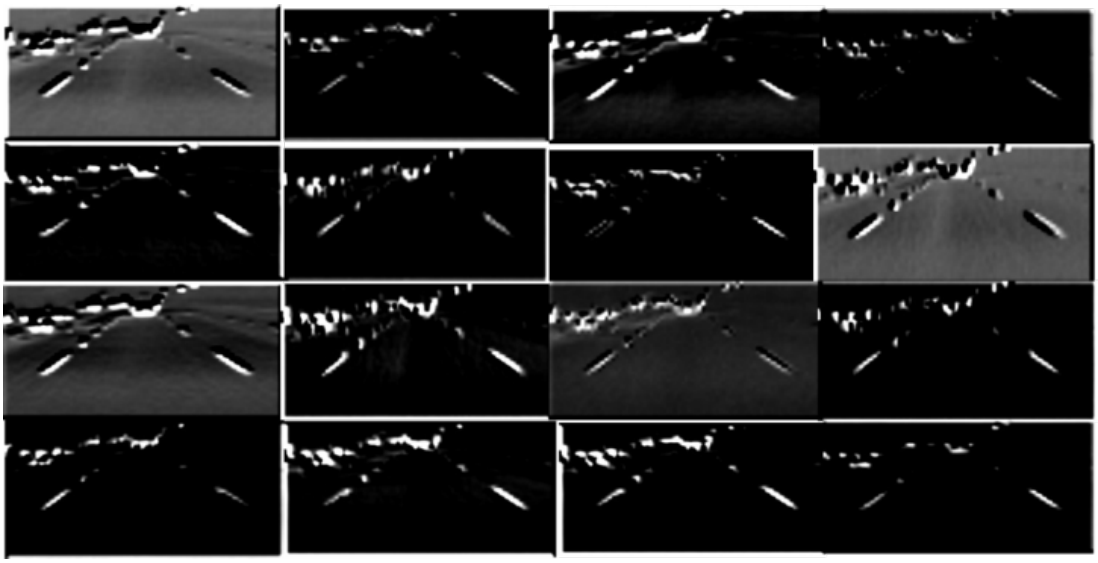
Generative Model Output

Based on the work of comma.ai on using generative model for learning the transitions in the image frame, we used their designed model for training. We observed that the model is able to learn upto 100 frames. It works well for straight roads but has a problem modelling curvy roads and after few frames, the model always transitions from curvy road to straight one. Fig. 4 shows output of our generative model show learnt transitions. It can be seen that the learnt transition is blurred.

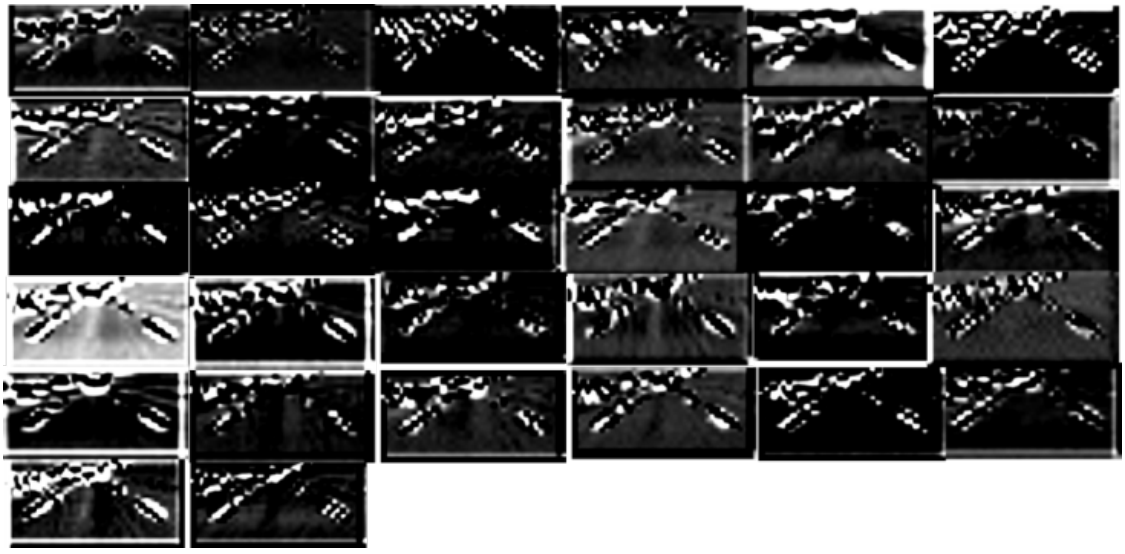
5 Future Work

The model is trained for prediction of steering wheel till now and our next objective is to train the other driving parameters by the use of LRCN model. In our initial training of LRCN model, the model is having difficulty of learning the objective function. Therefore, we would like to see the effect of change in the number of feature vectors on the model prediction and relevant optimizations required in making the system robust towards modelling turns on road. Also, our initial work is based on estimation of steering wheel angle and plan to extend prediction of other driving parameters.

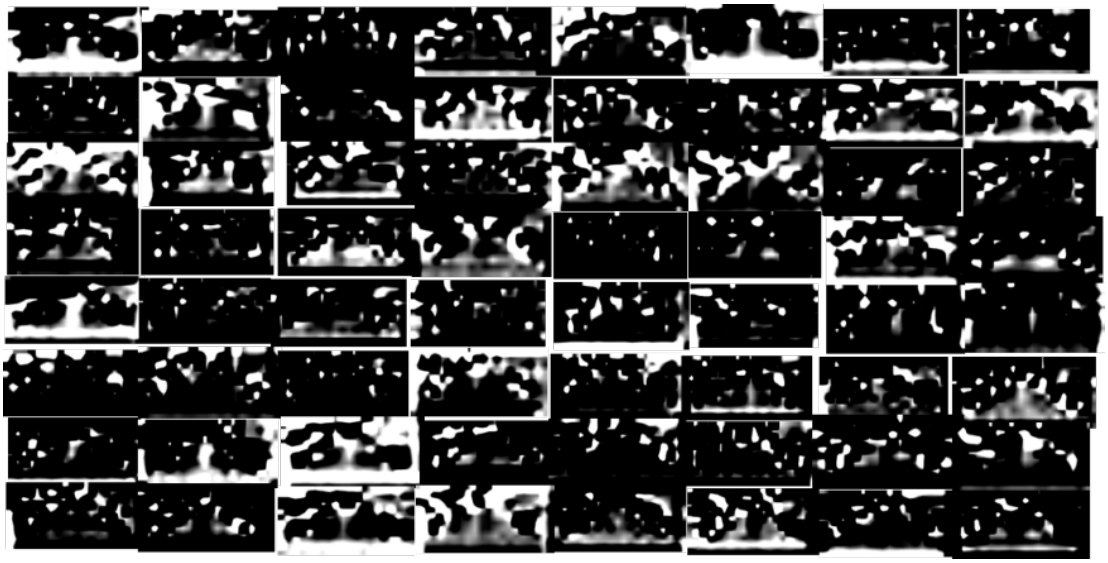
Our future work also includes improving on the generative model developed by comma.ai to make it work on curvy roads too. As discussed in earlier section that the developed model by comma.ai has difficulty in learning the transitions on a curvy road and after few frames it transits back to straight road. This problem arises due to unbalanced dataset i.e. majority of frames are for straight roads as the dataset was collected on highways. Therefore, we aim to improve the



(a) Frame with label "Output from Convolutional layer 1 "



(b) Frame with label "Output from Convolutional layer 2"



(c) Frame with label "Output from Convolutional layer 3"

Figure 5: Ouput of convolutional layers of the proposed model when a frame from video *dataset* 10 was given to model as an input

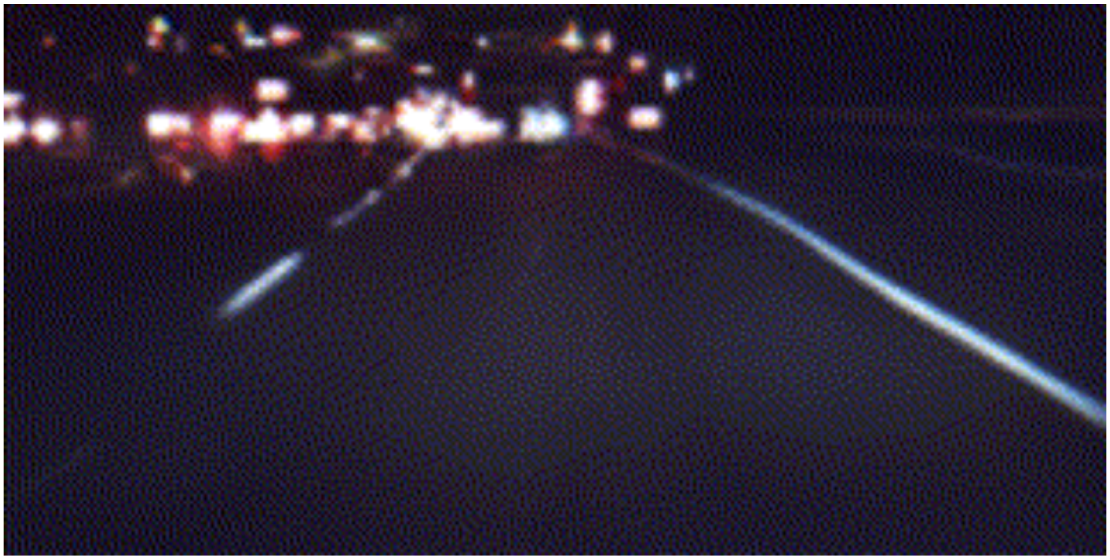


Figure 6: Transition output of Generative Adversarial Network model

situation by training it on balanced dataset having an equal mix of both. This methodology was employed by Nvidia during their training of CNN model.

6 Conclusion

The work proposes a way using recent developments in deep learning to model complex driving behaviours and patterns in order to predict driving states using the dashboard mounted sensors on a car. The developed model uses convolutional neural network for converting image frame to feature vectors which models the spatial information and then LSTM to model the temporal information in the data in order to predict the gear, gas and break position in order. We used open source publicly available datasets from comma.ai and udacity for training and testing of our model.

References

A. Diaz Alvarez, F. Serradilla Garcia, J. E. N. J. J. A. and Jimenez, F. (2014). Modeling the driving behavior of electric vehicles using smartphones and neural networks, *IEEE Intelligent Transportation Systems Magazine* **6**(3): 44–53.

A. Geiger, P. Lenz, R. U. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite, *Proc. IEEE Conf. Comp. Vis., Patt. Recog.* pp. 3354–3361.

Bojarski, Mariusz, e. a. (2016). End to end learning for self-driving cars, *arXiv preprint arXiv:1604.07316* .

D. P. Kingma, M. W. (2013). Auto-encoding variational bayes, *arXiv preprint arXiv: 1312.6114* .

Donahue, Jeffrey, e. a. (2015). Long-term recurrent convolutional networks for visual recognition and description, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* pp. 2625–2634.

E. Santana, G. H. (2016). Learning a driving simulator, *arXiv preprint arXiv:1608.01230* .

F. A. Gers, J. Schmidhuber, F. C. (2000). Learning to forget: Continual prediction with lstm, *Neur. Comp.* **12**(10): 2451–2471.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks, *Science 313.5786* pp. 504–507.

I. Goodfellow, e. a. (2014). Generative adversarial nets, *Proc. Adv. Neur. Inf. Process. Sys.* pp. 436–444.

Kessler, A. (2015). Tesla adds high-speed autonomous driving to its bag of tricks, *New York Times* .

Krizhevsky, Alex, I. S. and Hinton, G. E. (2012). Tesla adds high-speed autonomous driving to its bag of tricks, *Imagenet classification with deep convolutional neural networks* pp. 1097–1105.

LeCun, Yann, Y. B. and Hinton, G. (2015). Deep learning, *Nature* pp. 436–444.

Ng, A. (2011). Sparse autoencoder, *CS294A Lecture notes 72* pp. 1–19.

Stavens, David Michael, e. a. (2011). Learning to drive: perception for autonomous cars., *Stanford University* .

Vincent, Pascal, e. a. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, *Journal of Machine Learning Research* pp. 3371–3408.

Wang, R. and Lukic, S. M. (2011). Review of driving conditions prediction and driving style recognition based control algorithms for hybrid electric vehicles, *IEEE Vehicle Power and Propulsion Conference, Chicago* pp. 1–7.