LeGuess – Predicting or guessing the next scene using image sequences for accelerating autonomous capabilities.

Ranjan Akarsh
Sapiens Artificial Intelligence
Ranchi, Jharkhand, IN
akarsh@gloick.com

Abstract

This paper depicts a network called LeGuess (LeG) which, using computer vision, is able to precisely predict the future scenes given sequence of images. The network is able to automatically learn the features and representations of the objects present in the sequence of images fed as input. Furthermore, this network learns the movements of the objects and predicts very well.

The network is mainly designed for the domain of Autonomous Vehicles, which contains plenty of applications alone. Taking this in note, LeG can be applied to predict the steering angles, predicting the future positions of cars, trucks, cyclists, etc., ready a generative model to generate images along with steering angles, which could be used to train vehicles to drive, as well as generate images of road with/without lane to train segmentation for better autonomous driving.

The network is designed to make predictions (local) of up to given number of time-steps ahead.

1 Introduction

Autonomous driving is a difficult domain, but for India it is immensely complex, and has extreme potential for impacting lives of millions and billions. The challenge that Indian roads and traffic conditions offer is vastly different from what the likes of other big companies are up against in the developed countries. Here in India, lane discipline and pedestrian movement are wildly erratic, and the large numbers of two-wheelers and three-wheeler rickshaws that zig-zag between larger vehicles will likely provide the autonomous vehicles' sensors with far more data than they might pick up on roads of America and Europe.

The country has estimated 400 fatal road accidents per day in 2015, which is one death every 3.6 minutes. With intelligence technology behind the wheels, we might be able to reduce the number dramatically over-time.

In order to develop autonomous capabilities to perform the driving task, we need appropriate capabilities to sense and predict the traffic and road obstacles as well as planning coordination and control of the vehicle. Research into sensing and perception technologies has been progressing considerably, and current vehicles sensors seem to have the capabilities to detect relevant obstacles, vehicles and other traffic pertaining objects. However, predicting what the next could bring is a subject of ongoing research in the very domain, having extreme potential.

2 Overview

Consistent with the idea that prediction requires insight about object structure, it turns out that LeG can learn the internal and external representation of objects automatically that is well-suited to the subsequent recognition and decoding parameters of the objects. There are wide varieties of applications on which LeG could be applied to.

2.1 Data collection

Training data was collected from KITTI dataset which were captured by roof-mounted camera on car driving around urban environment in Germany. The dataset is roughly around 100 GB. Our processed data contains 8 sequence sampled from each category of KITTI dataset with 45 recording sessions used for training and 10 used for validation. Rest of the data was kept for future use. The training set contained roughly 35,000 sequence frames. For validation, we used CalTech Pedestrians dataset and matched the frame rate with KITTI dataset.

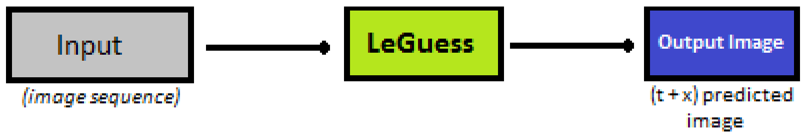

Figure 1: LeGuess minimal architecture. Network receives sequence of images, which is then compared to the predicted image outputted by LeG to estimate difference rate. Finally, the network outputs the image for given number time-steps $(t + x)$.

3 Architecture

LeGuess network has series of modules that is compiled together, that is used to make local prediction which is then compared to the original input to produce difference rate between the predicted and the original image, and the difference rate is then passed along to the higher layer.

That said, each of the module is divided into four parts. i) the convolution layer which acts as input controller (C_l) ii) generative convolution layer $('G_l)$ iii) prediction layer $('C_l)$ and iv) the difference rate layer (D_l). The generative convolution layer (G_l) generates local prediction, which then becomes input to the higher layer. The network then estimates the difference between C_l and $'C_l$ outputting difference rate which is split into separate positive (+) and negative (-) populations. The difference then is passed forward to the convolution layer and becomes the input to the next higher layer (C_{l+1}). The generative layer receives copy of difference rate generated along with generative layer of the higher layer. Activations used in the network are usually *rectified linear unit*. The loss function is embedded internally as the firing rates of the difference neurons.

The generative layer and difference layer is always initialized to zero, which means that the initial predictions are always spatially uniform, due to the convolution nature of the network.

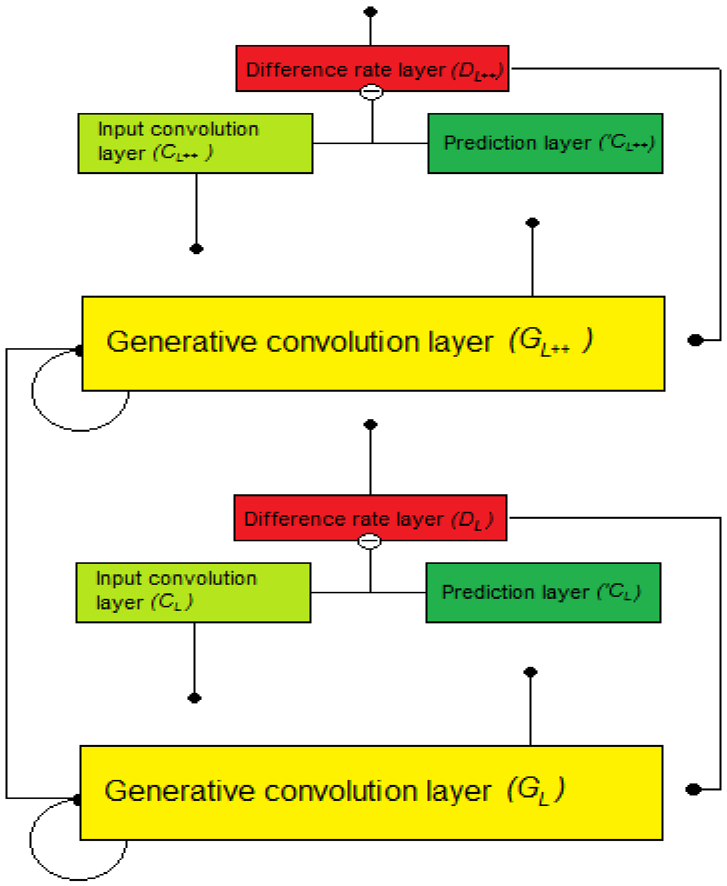
Difference rate layer (D_{L++})
Input convolution layer (C_{L++})
Prediction layer $('C_{L++})$
Generative convolution layer (G_{L++})
Difference rate layer (D_L)
Input convolution layer (C_L)
Prediction layer $('C_L)$
Generative convolution layer (G_L)

Figure 2: Image depicting the architecture of LeGuess. The generative convolution layer (G_l) generates prediction which is passed to prediction layer $('C_l)$. The predicted image is then compared to actual image present in input convolution layer (C_l) to estimate difference rate (D_l) between them. The generative convolution layer (G_l) then receives a copy of difference rate, and then the difference rate is passed laterally and vertically in the network.

3.1 Vivid understanding

This section gives the reader profound insight about the network.

Consider sequence of images x_i. The input to the lowest layer is the actual sequence of images (x_i) itself. The inputs to the higher layers are computed by a convolution over the difference units from the lower layer (D_{l-1}^{i}) followed by activation and max-pooling (non-linear down-sampling), Long-Short Term Memory is used for representing the neurons. The hidden states of the neurons here are update according to – generative layer (G_l^{i-1}), difference layer (D_{l-1}^{i}), as well as the higher generative layers (G_{l+1}^{i}) which is firstly partially nearest-neighbored, as pooling is present in feedforward path. The predictions $('C_l^i)$ are made through convolution of generative layer (G_l^i) stack followed by rectified linear unit non-linearity. For the lowest layer, the prediction is also passed through saturating non-linearity which is set at maximum pixel value:

$$saturate(x, p_{max}) := minimum(p_{max}, x)$$

At final, the difference is calculated by comparing the actual by predicted, and then split the values into positive (+) and negative (-) prediction difference concatenated along the feature dimension.

The model is trained for minimizing the weighted sum of the activity of the difference units. With difference units consisting of deduction followed by rectified linear unit activation, the loss at each layer is equivalent to L1 error. Other difference unit implementations could also be used in place. The update of the hidden states are first passed top-down where the states of generative layer is computed (G_l^i) and forward passed for evaluation of prediction, difference and inputs to the higher level of the layers.

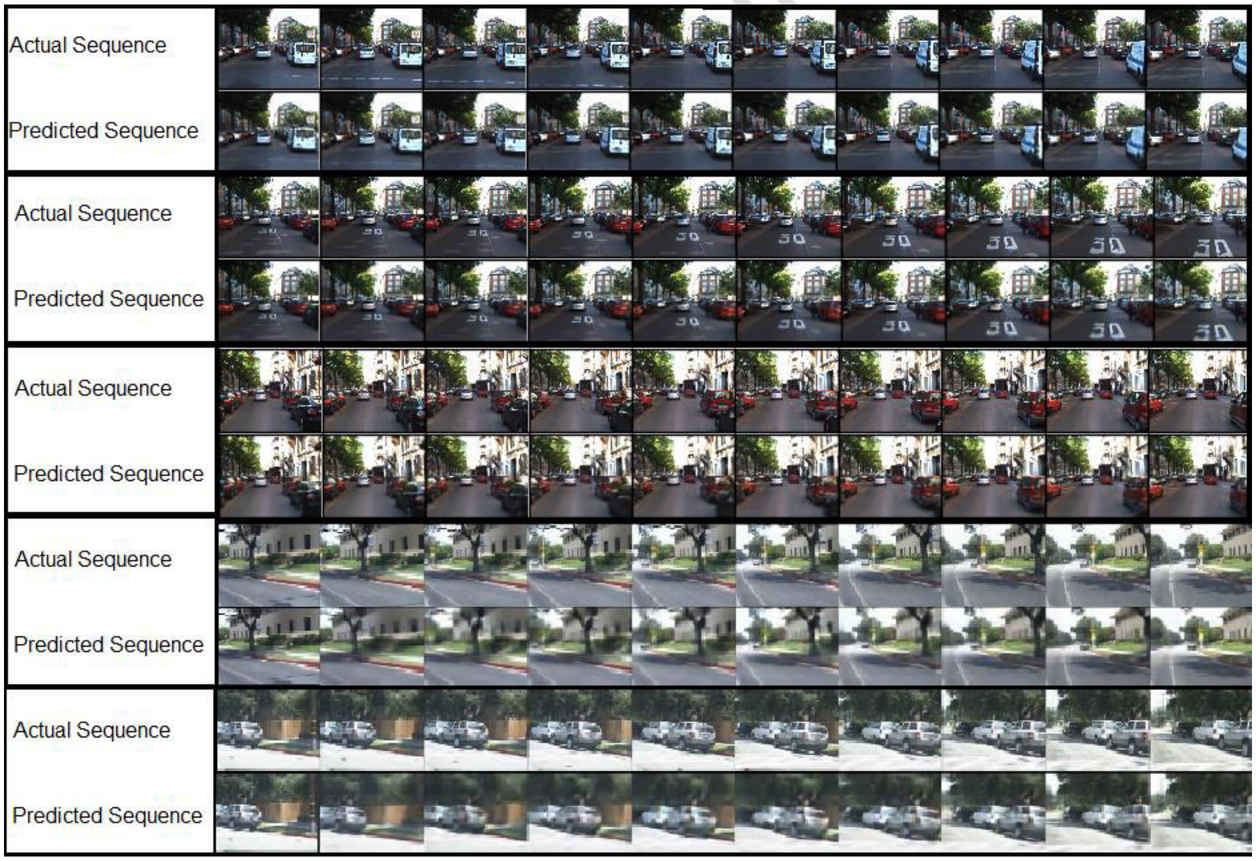

Figure 3: The network precisely predicts the next image sequences. The images above was not present during the training, they are never-seen images for the network.

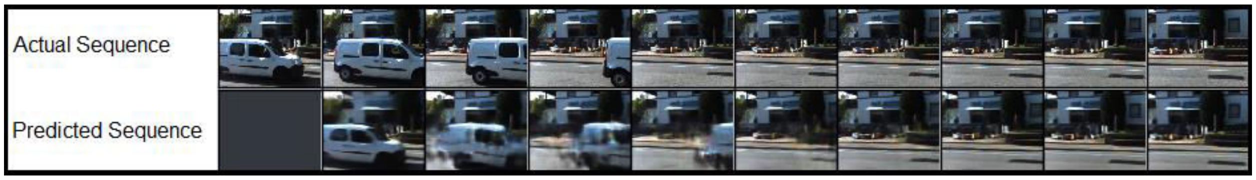

Figure 4: The network outputs uniform image at first, since the representation neurons are initialized to zero, so the first time-step prediction is always uniform. In the second time-step, the network learns the representation of the objects present in the sequence. From the layers after, it becomes stable and the blurriness behind the car is mostly because the network thinks car is moving.

We provided the network with images that it never saw. For the first time-step prediction, the model is unable to figure out what possibly next could be and is uniform. This is due to the representation neurons as initialized to zero. In second time-step, the model starts learning the internal and external representation of the objects and scene, and so it outputs blurry but precise image. In the third image, the network gets better insight but still the image predicted is blurry. After further iterations, the network adapts to the underlying dynamics to generate predictions that closely match the actual images.

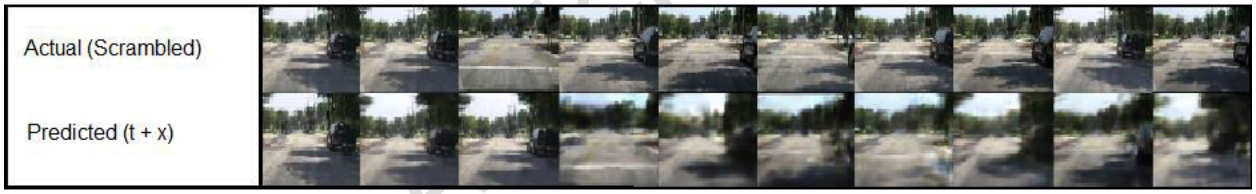

Figure 5: Image depicting the prediction by the model for 1 time-step ahead. (t + 1)

We performed random hyper-parameter search, with model selection based on the validation set, which resulted in 4 layer model with 3x3 sized convolution layer and channel size of (3, 48, 96, 192). The optimizer best suiting to our settings was Adam optimizer, using a loss either mainly computed on the lowest layer or weight of 1.0 on the lowest layer and 0.1 on the upper layers. Parameters of Adam optimizer were set to their default as mentioned in the paper, and the learning rate,∝, was set to decrease by a factor of 10 halfway through training. For robust representation, network was trained on CalTech Pedestrians Dataset. We made testing sequences that matched the frame rate of KITTI dataset and were cropped to 100x130 pixel size.

The model is able to make accurate predictions in a wide range of scenarios. In the top sequence of Figure 3, a car is passing in the direction opposite to the drivers', while not being perfect yet, the network is able to predict its trajectory, as well as it fills the road behind. In the sequence 4, the network predicts the position of tree as the car turns onto a road, and the very sequence also proves the ability of the model to fill out relevant spaces that are left behind, as it is able to fill out the sky and tree textures as unseen regions come into view.

In Figure 5, we show an input where is has been temporarily scrambled. In this case, the model generates blurry images, which mostly just resembles previous layer. At final, although the network was trained for one time-step ahead, it is totally possible to predict multiple time-steps ahead in the future by feeding back predictions as inputs and recursively iterating.

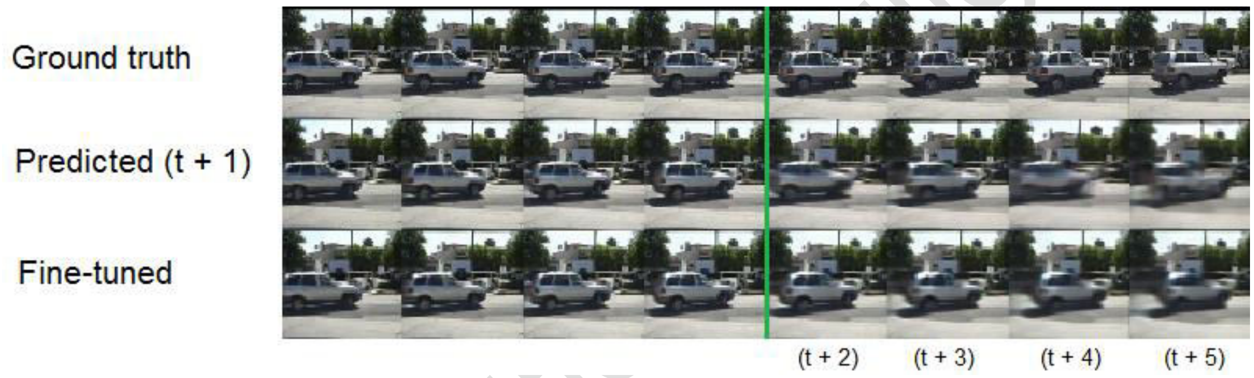

Figure 6: Sequence generated by the network by feeding predictions back into the network. Left segment represents the normal predictions $(t+1)$, while right segment represents sequences generated by recursively iterating using predictions as input to the next layers.

LeGuess is mainly designed to predict the behavior of traffics on the Indian roads but it turns out that it can be extended to vast varieties of applications. Although the next image prediction is accurate, the model breaks down when extrapolating further into future and this is nothing new as the predictions will have different statistics for actual image that the model was trained on. The network would be able to extrapolate better if we additionally train the network to process on its own.

The model was trained with a loss over 20 time-steps, where the actual image was fed for first 10 and then network predictions were used input to the network for last 10. This was calculated with *mean absolute error* metric with respect to the ground truth image sequences. Despite the blurriness which is due to uncertainty, the fine-tuned model captures some key structure in the first sequence in Figure 6. The network estimates the shape of an upcoming car, despite minimal information in last seen frame.

In the second sequence the network is able to extrapolate the motion of a car moving to the right. In easier words, this is success.

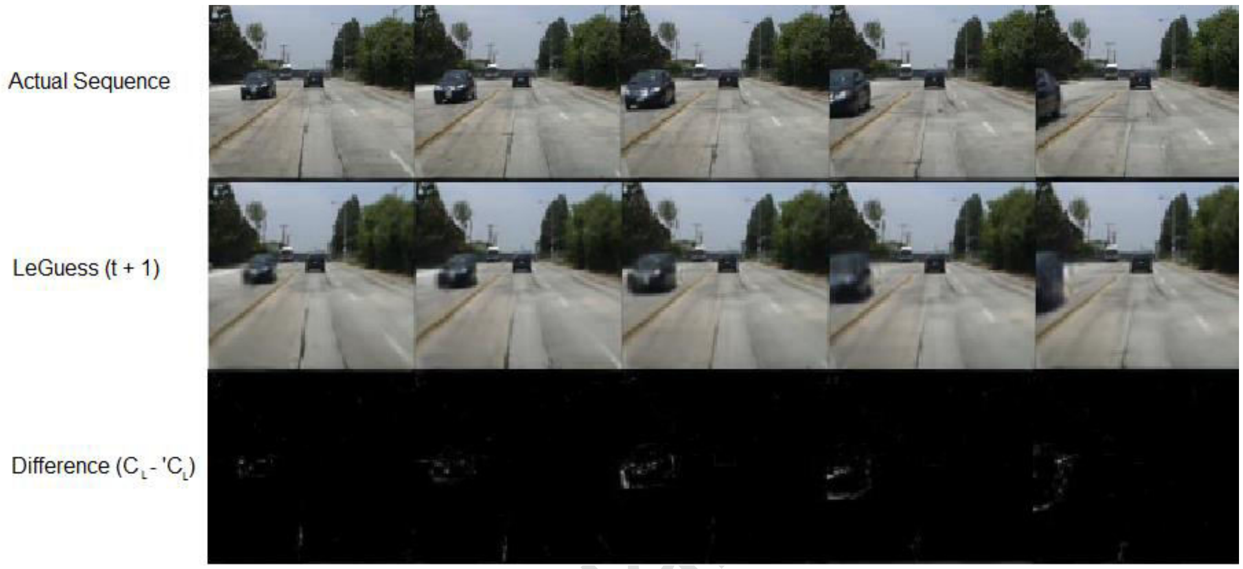

Figure 7: next-scene predictions of the network on CalTech Pedestrians Dataset and comparison to actual input. The difference $(C_l - 'C_l)$ visualization shows where the pixel error was smaller to the predicted than the actual image.

4 Conclusion

We showed that convolution layers could be used to no-limit extent, and we optimized convolution neural networks to predict for example where a specific car on the road would be next, or where the cyclist would go next. LeGuess takes its own steps, i.e. it makes a guess what possibly could happen in future given the sequence of images.

More work is needed to improve the robustness of the network, to find methods to verify the robustness, and to improve visualization of the network-internal processing steps.

We already are planning to apply LeGuess to the application for steering angle predictions for accelerating Autonomous capabilities.

5 References

[1] LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backprop-agation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551, Winter 1989.

[2] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory.Neural Computation, 1997.

[3] Pulkit Agrawal, Jõao Carreira, and Jitendra Malik. Learning to see by moving.CoRR, 2015.

[4] Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. CoRR, 2014.

[5] Ranjan Akarsh. Predicting or guessing the next scene using image sequences for accelerating autonomous capabilities. 2017.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. CoRR, 2016

[7] Hossein Mohabi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In ICML, 2009.

[8] Large scale visual recognition challenge (ILSVRC). http://www.image-net.org/challenges/LSVRC/

.

[9] Rajesh P. N. Rao and T. J. Sejnowski. Predictive sequence learning in recurrent neocortical circuits. NIPS, 2000.

[10] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014.

[11] Michäel Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. ICLR, 2016.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In NIPS. 2014.

[13] Raman Arora, Amitabh Basu, Poorya Miyanjy, Anirbit Mukherjee – Understanding deep learning with Rectified Linear Units. Nov, 2016.