Driving Using End-to-End Deep Learning

Farzain Majeed
University of Central Florida
farza@knights.ucf.edu

Kishan Athrey
University of Central Florida
kishan.athrey@knights.ucf.edu

Dr. Mubarak Shah
University of Central Florida
shah@crcv.ucf.edu

Abstract

This work explores the problem of autonomously driving a vehicle given live videos obtained from cameras mounted around it. The current approach to this problem is to utilize hand-coded rules that explicitly tell a vehicle how to react given various stimuli (ex. stop signs, lane markings, etc). We instead seek to solve it using a single, end-to-end deep convolutional neural net trained on pairs of images and vehicle control parameters (ex. steering angle, throttle, etc) recorded from an actual driving sequence by a human. When visualizing the class activation maps for end-to-end models that currently exist for this task, it can be seen that the convolutional layers do not activate upon many regions of the image other than just the road. This is a major issue because driving requires understanding of the entire image to properly react to objects like pedestrians, traffic lights, and even other vehicles. We propose a new model that takes advantage of object detectors and multi-task learning to create a network that helps pays more attention to significant areas of an image in an unsupervised manner.

1. Introduction

The self-driving car will define the modern era and change billions of lives in that it will save time, save lives, and lead to more efficient use of resources. Self-driving cars have made impressive progress in the last decade due to advances in hardware and a growing interest in the field by large companies.

But perhaps one of the most important advances for self-driving cars came with recent breakthroughs in the field of neural networks and deep learning. These deep models have been shown to perform impressively on all sorts of tasks from predicting words in a sentence to playing a video game. While the problem of autonomously driving a vehicle can be can be from approached many different angles, we frame it as a pure vision problem (Figure 1). Because of this, it is only fitting that we take advantage of a type of neural network that has taken the computer vision community by storm: the deep convolutional neural network [4]

Figure 1. The driving scenes display the type of images the neural network receives. It also exhibits the complexity of driving especially in terms of a regression problem where the algorithm must be able to react to millions of different scenarios.

Companies like Google and Tesla, who are heavily pushing self-driving cars forward, have trained their own CNNs for tasks live traffic light detection, lane detection, action recognition, and many more. But, both of these companies (and others) utilize several different models for each of these important tasks and combine the outputs of all the models and car sensors in a central computer to make a decision.

While this method has been shown to perform well in most scenarios, it still utilizes handcrafted rules [2] that decide how to react given the output of the models and various car sensors. While simple rules, like stopping at a stop sign, can easily be accounted through handcrafted rules it is nearly impossible to account for every situation due to complexity of driving and the millions of scenarios that may occur.

We are more interested in developing a network that learns these handcrafted rules end-to-end by inputting frames captured from various cameras positioned around the vehicle and regressing upon control parameters, such as steering angles. We took some public end-to-end deep net model designs from NVIDIA [1] , who is a big believer in the concept, and trained it from scratch on our own smaller dataset. This report later goes into more detail about the specifics of the model and the dataset. While NVIDIAs model was found to easily able to keep our vehicle in lane, which was its main goal, it was found that it couldnt quite pick up on other complexities.

Taking these issues into account, we propose a new network that predicts control parameters in an end-to-end fashion and also incorporates a side task which predicts saliency in an unsupervised manner. The model learns the hand-

Figure 2. We utilize object detectors as part of our new "attention" based model

crafted rules that current systems incorporate end-to-end by inputting frames captured from various cameras positioned around the vehicle and regressing upon control parameters, such as steering angles. The model's goal was to create a deep net that would pay more attention to other aspects of driving while staying end-to-end.

We evaluated our model by comparing it to NVIDIAs model in terms of MSE and also visually inspect its performance in a variety of significant situations. Future work includes better evaluation methods in a virtual environment where we can catch the finer details and see how our model would truly drive in the real world.

2. Related Work

End-to-end networks for SDCs have popped up in recent months mainly because of NVIDIA who in April 2016 released a work that explained how they were able to steer a vehicle on a highway using a CNN trained end-to-end. Their network learned the various rules involved in driving by observing a human driver. The vehicle running this network was called PilotNet and was able to drive in several complex environments, from highways to dirt roads, without any explicit information.

NVIDIA released their models design, as shown in Figure 3, which was surprisingly simple and featured five convolutional layers and five fully connected layers with an input image size of 66x200 leading to about 1.6 million parameters. NVIDIAs model was optimized for an input image that looks just at the road. This is immediately a problem because to truly be end-to-end a network should be able to understand all parts of the input image in order to come to some final decision. Our model helps alleviate this issue by inputting the entire image and forcing it to pay attention to things other than just the road.

The use of 3D CNNs also became part of our experiments because driving is made up of actions such as: left turn, slight right merge, brake at red light. These actions cant be represented by a 2D CNN, but a 3D CNN can take advantage of the temporal region. To do this, we specifically utilized techniques from C3D [9]. While the C3D

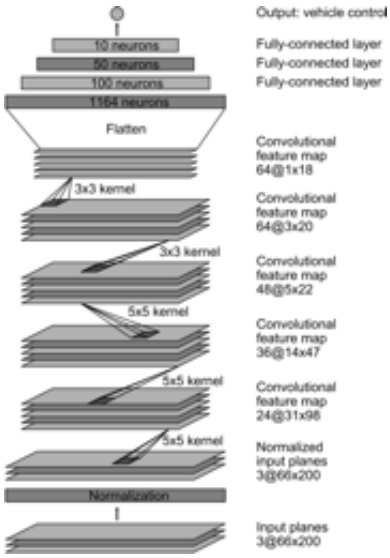

Figure 3. NVIDIA released this illustration to explain PilotNet's design, though many key items such as strides, activation functions, and pre-processing had to be tested

model itself was not used for our experiments, we required some grounds to choose certain parameters, such as kernel size. The 3x3x3 kernel, as explained in the C3D paper, was shown to perform well on a variety of tasks specifically when used for the C3D model. In addition, the max pooling layers it specified were shown to provide sufficient translational invariance for the task of action recognition. While our task is very different, certain similarities are still there.

Multi-task learning has been used a good deal where learning different tasks simultaneously and hard sharing parameters is advantageous. Our motivation for using multi-task learning is to help the model focus on more relevant features of a scene. Regressing on a steering angle doesn't seem to be a strong enough candidate on its own to properly teach the network.

Due to the lack of image level labels, we decided to take advantage of the YOLO9000 object detector [7], as show in Figure 2, to create saliency maps in an unsupervised manner. We did this by specifically labeling objects related to driving such as cars and pedestrians and setting these areas as white and the rest of the image to black. [5] does segmentation as a side task while their main task was saliency prediction. They arrived at much more defined saliency maps since the network better understood the contours of different objects.

[10] does something very similar, except now the main task is predicting steering angles from input frames while the side task is still segmentation. Their results improved due to the two very related tasks. Our model takes inspiration from this but instead of classifying each pixel as a side task, it specializes in saliency prediction. Understanding

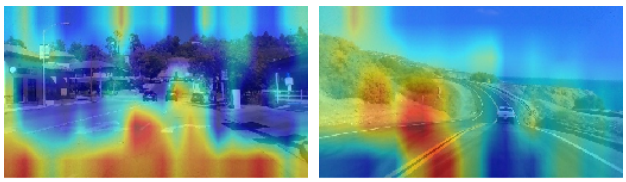

Figure 4. The first image (left) shows the pixels that activate most when the network predicts to drive straight. The other image (right) displays the pixels that decide to take a right. Most pixels are looking just at the road rather than other important parts of the image

every single thing in a driving scene is not necessary even for a human, which is our intuition for instead creating an attention based model.

3. Dataset

There are a variety of datasets that would be suitable to our needs and initially we began with the Udacity dataset which featured around 7 hours of data but the frames were found to be very noisy/shaky. While this wasnt a massive issue since it mimics a somewhat real world environment, our model was not obtaining the best of results. It seemed that the model was having a hard time paying attention to the road, and this problem may have been alleviated through data augmentation but instead we simply attempted to use a different dataset.

The CommaAI dataset is what we officially decided to use for our experiments for a variety of reasons. It gives us over 7 hours of data in a variety of scenarios such as daytime driving, nighttime driving, city driving, and highway driving. It also gives a good deal of other information associated with every frame such as GPS coordinates and steering torque that may be useful later on.

4. Experiments

For our initial experiments we utilized 100,000 160x320 frames from the CommaAI dataset which comes out to about an hour and went minutes driving since the data is captured at 20 frames per second. All experiments were done using Keras with a TensorFlow backend. The models were trained over a course of 50 epochs with a batch size of 32 while utilizing a generator to load frame data into memory. The model used stochastic gradient descent with a learning rate of 10-4 and a built in callback function was utilized that saved the best weights in regards to MSE on the validation set.

4.1. PilotNet

It was very necessary to recreate NVIDIAs results to understand where they went wrong and where they succeeded. In addition, it would aid in creating some sort of baseline to compare new methods to. We trained their network from scratch on the CommaAI dataset first on just a 66x200 image where just the road was cropped. The network performed very well in the task of just staying a lane but did not understand the scene in a global context since everything other than the road was removed due to cropping. We then fed the full image (160x320) and received very similar results.

In order to understand why, we visualized the class activation maps and found that most of the pixels that contributed to the model's final decision came from bottom of the image, the road (Figure 4). This is somewhat expected because: 1) most of the driving data is on a highway where the road is always in view and is the most common thing in every frame, 2) the network is simply regressing upon a steering angle which may not be powerful enough on its own for a deep net to pick up on certain nuances, and 3) the training data does not feature any image level labels and naturally features mostly straight driving and lacks significant situations (ex. stopping at a pedestrian crossing) 4) NVIDIAs network was optimized for a 66x200 image while we were using a 160x320 image.

Taking all these points into account, we began attempting to improve upon PilotNet so that it would perform even better at staying in a lane. One of the first things we did to improve results was simply skewing the dataset by giving angles closer to 0 a lower chance of being included in the training data. The main idea behind this was that the model would be able to learn more since it wouldnt just be barraged with frames/angle pairs close to zero. We also attempted to temporally downsample the data from 20hz to 10hz but this didnt show much improvement in results as compared to skewing the angles.

Another aspect that was explored was the use of 3D convolutions which is a task widely used in action recognition [8, 3]. The main intuition behind this is that even a human has a hard time learning from random image/angle pairs as the 2D CNN does. A human would learn much quicker from clips/angles associated with every frame. To test this, we took PilotNet and combined it with concepts from the well know C3D model, mainly the 3x3x3 kernel and the max pooling layers. While still utilizing the base PilotNet model, we added these ideas from C3D to the model and trained on 8 frame clips. This model performed better and actually understood the dynamics of driving much better. While the 2D PilotNet seemed to fail at things like sharp twists and turns the 3D PilotNet was able to complete them with nearly 100 percent accuracy.

While utilizing 3D convolutions definitely seems to be the best way to go, due to training time we first prototype in a 2D manner. We had also attempted to train C3D from scratch and didnt obtain very good results which was expected due to a lack of training data. Future work includes

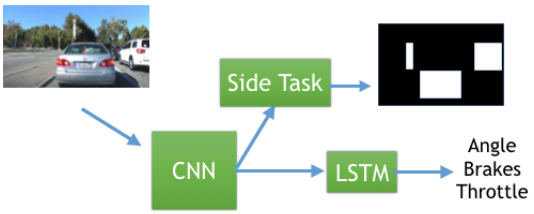

Figure 5. Our newly proposed model uses the YOLO object detector to generate ground truths for the saliency side task

fine-tuning C3D pretrained on Sports 1M as the primitive feature it in the early convolutional layers may be able to be extended to this task as well. Reinforcement learning [6] has also shown itself to be a candidate because of the ease of training a vehicle in a virtual environment.

4.2. Attention Based Model

The entire goal of an end-to-end model for SDCs is to learn the rules of driving by observing, but PilotNet failed to do this when it came to a variety of things namely: other vehicles. PilotNet often predicts steering angles that would cause collision. While most of our experiments revolved around regressing upon a steering angle, future work would also involve predicting things like brake and throttle values. In that case, our network definitely needs to take into account other parts of an image and pay proper attention to key pieces of a driving scene. In order to alleviate the problems PilotNet experiences, we sought to create an attention based model.

A big problem with end-to-end models and SDCs as a whole is the lack of significant situations in the training data. A SDC will not know how to react to every sort of car accident or traffic pattern, but it can try its best. Something for our future work is to somehow leverage dashcam videos from the internet which can provide thousands of hours of footage. Many of these videos features accidents and other driving mishaps which can be very valuable. But, all this data is unlabeled and all we can utilize are the pixels given to us. This leads us toward unsupervised approaches.

4.3. Saliency Side Task

Object detection was used as the core of this new model (Figure 5 because aside from staying in a lane, driving is heavily based around recognition of certain objects such as vehicles, traffic lights, and pedestrians. For our experiments the YOLO9000 object detector was used to detect these objects in the form of a bounding box. From these bounding boxes very rough saliency maps were created to indicate the position of these objects on the frame. This is the ground truth for our side task. We specifically chose saliency prediction because detection of salient objects directly correlates with control parameters. If the network doesnt pay attention to special objects, such as other vehicles, than it will have a hard time properly predicting parameters.

The side task takes place right before the fully connected layers and specializes in saliency prediction. The model itself features a decoder [5] after the last convolutional layer which was done because the fully connected layer will cause us to lose spatial information necessary for the side task. These side task layers take advantage of convolutional transpose layers. Though, It is also possible to use upsampling layers and convolutional layers with a stride of 1. At the last layer of the side task, there is a convolutional layer with a single filter and sigmoid activation that will output the final 160x320 image. The loss is calculated using binary cross entropy with the saliency map ground truth produced from the YOLO bounding boxes. Apart from the side task, we still use the fully connected layers to predict control parameters.

Early experiments with our new model actually resulted in a lower validation loss when observing the MSE for the main t ask of steering angle prediction. But, the side task curve never actually converged. It's very possible that the extra noise introduced in the model helped it avoid over fitting and led to a better result for the main task. Though, this is something we are still experimenting with.

5. Conclusion

In this report we proposed a new attention based model for self-driving cars to help push it toward understanding certain significant situations. While we are still investigating new methods to evaluate and test our model, initial results seem promising. Though, more work needs to go into actually more thoroughly visualizing the layers of PilotNet to improve upon it.

References

[1] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. 04 2017.

[2] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 2722–2730, Washington, DC, USA, 2015. IEEE Computer Society.

[3] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] X. Li, L. Zhao, L. Wei, M.-H. Yang, F. Wu, Y. Zhuang, H. Ling, and J. Wang. Deepsaliency: Multi-task deep neural network model for salient object detection. 10 2015.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. 12 2013.

[7] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. 12 2016.

[8] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. 06 2014.

[9] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. 12 2014.

[10] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. 12 2016.