Neural Networks for Vehicle Navigation

Matthew Linder
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
linde593@morris.umn.edu

ABSTRACT

Autonomous vehicle driving systems use many tools to drive safely and efficiently. Many of these tools use neural networks. In this paper, we describe the inner-workings of basic neural networks. We describe some advanced neural network concepts including: deep neural networks, the sigmoid function, reccurent neural networks, and convolutional neural networks. We present how these neural networks are implemented in computer vision tools and how they are used to avoid obstacles. Finally, we discuss how all these tools are integrated to drive vehicles effectively.

1. INTRODUCTION

Over the last few decades, there have been an increasing number of vehicles on the road. This has led to an increased number of accidents and deaths. There were 32,675 automobile deaths in the US in 2014 [2]. Most of these deaths can be contributed to human driving errors. To address this, there has been much research in the area of autonomous vehicle driving or computer driving. Currently, there are vehicles available with semi-autonomous driving features, such as lane keeping and automatic braking. In the next few years we expect to see fully autonomous driving vehicles becoming available to the public. *Neural networks* are a core technology used in vehicles to navigate safely.

We will present details on neural networks and how they are implemented in vehicles. Section 2 will present what neural networks are and how they work. Section 3 will explain some more advanced and complex ideas and types of neural networks that are commonly used in vehicle navigation. Section 4 will present many applications of neural networks that are being used in the world today. Section 5 will present details on a variety of computer vision sensors that are used with autonomous vehicle driving systems. Section 6 will present approaches to vehicle navigation using computer vision sensors and neural networks and will provide details on self-driving cars.

UMM CSci Senior Seminar Conference, May 2016 Morris, MN.

2. SIMPLE NEURAL NETWORKS

In this section, we present background on machine learning. Then we describe what an artificial neural network is and why it gets its name from biology. Then we present background on artificial neural networks and information on training artificial neural networks.

2.1 Machine Learning

Machine learning is an area of computer science that designs models that can learn from data to be able to make predictions on other data. This allows computers to learn patterns in data without being explicitly programmed by a human. Machine learning requires lots of data, usually thousands of data points, and often millions or billions. Many types of machine learning require the data to be labeled, which means the data must have input data and expected output data, where the expected output data is a known answer to what the model should predict. For example, we might train a model to predict if an email is spam. The input data would represent information about thousands of emails, and the expected output data would determine if the email is spam. To train a model that would recognize what letter a handwritten character is, the input data would represent an image of the character, and the output data would determine which letter it is. The data gets passed through the model, which makes a prediction about the data. Then the model updates or modifies itself, depending on how accurate the prediction was. As more data is passed through the model, the model learns; this is called training the model. Once the model is trained, it can take unknown data and make an accurate prediction about it. For example, a well-trained model can receive a new email and accurately predict whether it is spam. There are many complex approaches to machine learning. We will focus on approaches that use neural networks.

2.2 Biological Neural Network

Artificial neural networks (ANN) are complex models used for predicting data; they are inspired by *biological neural networks* inside of human brains. The human brain has 100 billion nerve cells called neurons. Inside the brain a group of many neurons connect to each other to form a biological neural network. Neurons send electrical signals to, and receive electrical signals from, other neurons. The connections between neurons have varied strengths. As a human brain learns, it forms new connections between neurons and drops old connections; the connections also strengthen and weaken. As a human performs an action repeatedly,

electrical signals are repeatedly sent from neuron to neuron. This causes connections to strengthen, making them more efficient. This is how the brain learns.

2.3 Artificial Neurons

An artificial neural network consists of many interconnected nodes called *artificial neurons*. Each neuron contains an algorithm that receives several inputs and returns a single output. Each of the inputs to a neuron is assigned a certain adjustable multiplier called a *weight*. In Figure 1 each x represents an input and each w represents the weight associated with it. The first artificial neurons used algorithms

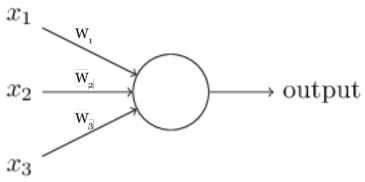

Figure 1: A Visual Representation of an artificial neuron

called perceptrons, inspired by neurons in the brain. This algorithm totals the product of the binary inputs, i, multiplied by their weights, and then returns a 0 or 1 depending on whether the total has reached a certain threshold value.

$$Output = \begin{cases} 0 \text{ if } & \sum_i x_i w_i <= thresholdValue \\ 1 \text{ if } & \sum_i x_i w_i > thresholdValue \end{cases}$$

This threshold value is sometimes called a *bias* and must also be adjustable. In section 2.5 we explain how ANNs are trained. As an ANN is being trained, or learns, the threshold values and weights are increased and decreased, similar to the brain learning by strengthening and weakening connections between neurons. Although ANNs are inspired by biological neural networks, the technical details and mathematics are quite different.

2.4 Sigmoid Function

In the previous section, we explained the perceptron algorithm that uses a single binary bit for the inputs and the output. In modern neurons, the inputs and outputs are continuous values, ranging from -1 to 1 or 0 to 1. Furthermore, neurons will use a wide variety of functions and algorithms to determine the output. One common algorithm is to total the product of the inputs multiplied by their weights, then add the bias value that is unique and adjustable for each neuron, then pass that number into a *sigmoid function* that returns a value between 0 and 1.

$$\text{Total} = \left(\sum_{i=0}^{inputs} w_i x_i \right) + biasValue \quad (1)$$

$$\text{Output} = \text{SigmoidFunction(Total)} = \frac{1}{1 + e^{-Total}} \quad (2)$$

The output of a sigmoid function is s-shaped, so the result is usually very close to 0 or 1 and there is only a small range where the output fluctuates. The bias value modifies the point of where that fluctuation occurs, as shown in Figure 2. This is how the bias relates to the threshold value in the perceptron algorithm. Common variations to this include using different types of sigmoid functions and functions with different shapes.

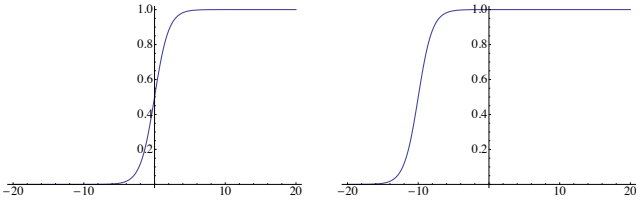

Figure 2: Left: Sigmoid with a bias of 0. Right: Sigmoid with a bias of 10.

2.5 Artificial Neural Networks Overview and Training

An ANN consists of input, output, and hidden neurons. All of the input neurons are called the *input layer* and all of the output neurons the *output layer*. The hidden neurons are in 1 or more *hidden layers*, as shown in Figure 3. A trained ANN makes predictions on data by passing the data into the input neurons. Then the algorithms at the neuron pass new data to the neurons in the hidden layers, which pass data to the output layer, which outputs a prediction about the data. This is called *forward propagation*.

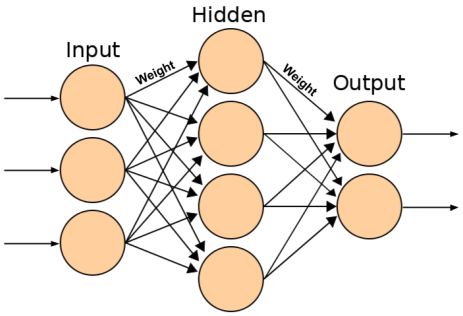

Figure 3: Neurons of a simple artificial neural network. Each connection has an adjustable weight value assigned to it.

The input and output data in an ANN can represent many things. We will present some examples. An ANN that predicts tomorrow's weather could have inputs that represent today's temperature, humidity, and wind speed. Other inputs could represent information about today's weather in surrounding areas. The output of this ANN could have a single output neuron that outputs a value that represents tomorrow's temperature, or there could be multiple output neurons that output different values representing tomorrow's high temperature, low temperature, humidity, and wind speed. An ANN that determines whether a picture is of a dog, cat, or bear could have 3 output neurons each representing one of those animals. The network classifies the image based on whichever output value is greatest.

In order to get an ANN to predict information accurately, the ANN must be trained. ANNs are usually trained by an algorithm called *backpropagation*. Backpropagation requires an ANN with starting values for the weights and biases; they

can be arbitrarily assigned. It also requires a large amount of labeled training data with inputs and expected outputs, often billions or more data points. Forward propagation passes known data through the network, and then backpropagation compares the generated data with the expected data. The difference between the generated and expected data is called the *error* or cost. Then the backpropagation algorithm updates the weights and biases in the network using a function of the error. There are many different methods and formulas used to determine the amount to change each weight and bias. A common method is to increment or decrement each weight by the value equal to the partial derivative of the error with respect to that specific weight, multiplied by a constant learning rate.

$$\triangle Weight_i = \frac{\partial error}{\partial weight_i} * learningRate$$

As training data is passed through the network, the network continues adjusting its weights and biases so the error gets smaller for the entire training set. This process is repeated until the errors are reduced to minimal values.

3. COMPLEX NEURAL NETWORKS

In the previous section, we presented the very basics of ANNs. However, neural networks function in a variety of ways. Many networks have their own specific modifications that the creators implemented to make them work best for their specific applications. Now we present the basics on some more complex neural networks including: deep neural networks, recurrent neural networks, and deep convolutional networks. These networks are some of the core technologies used in vehicle navigation.

3.1 Deep Neural Networks

For many years, most neural networks consisted of only a few hidden layers, commonly 1-3 layers. ANNs with many layers of hidden neurons are called *deep neural networks* (DNNs), whereas, ANNs with 1 or few layers are commonly called *shallow neural networks*. DNNs are theoretically and practically more accurate and powerful at solving complex problems than shallow neural networks [3]. However, the backpropagation training algorithm exponentially decreases in effectiveness with each additional hidden layer. The specifics on why backpropagation becomes ineffective are complicated and not relevant to this paper. Training DNNs used to be so ineffective that it would take years to train some DNNs with many layers. Therefore, DNNs were impractical and not very commonly used. Fortunately, in 2006, breakthroughs were made by Geoff Hinton that made major advancements to training DNNs [6]. Due to these advancements and powerful modern graphics processing units, DNNs became much more popular in recent years. The process of training a DNN is known as *deep learning*.

3.2 Recurrent Neural Networks

DNNs are often used to process data that is changing over time. This means that the input data is being periodically changed or updated. To handle this, a special kind of DNN is used called a *recurrent neural network* (RNN). A RNN can receive a sequence of values as inputs, and it can output a sequence of values. Each time the input data is updated, the new input data is sent into a new instance or copy of the network. Each instance of the network is called a *sequence*. During forward propagation, data is also passed from neurons in the previous sequence into neurons of the current sequence. This enables a RNN to process information across time. Each sequence in the RNN outputs unique values. This means the network will continuously output new data at the same rate that it receives new data. RNNs are commonly used in speech recognition applications.

3.3 Deep Convolutional Neural Networks

A *deep convolutional neural network* (DCNN) is a specialized kind of DNN designed for image processing. DCNNs are inspired by the visual cortex in the human brain. To process an image with a neural network, each pixel in the image is connected to an input neuron. Regular DNN architecture does not work as well for images because they treat two pixels that are adjacent the exact same way as they treat two pixels that are far apart. DCNNs use a special architecture that is specifically well-adapted to classify images.

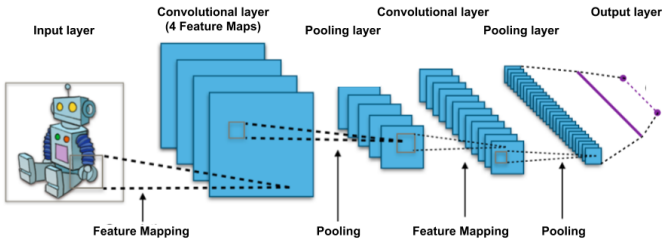

Figure 4: Typical DCNN architecture

The hidden layers in a DCNN alternate between *convolutional layers* and *pooling layers*, as shown in Figure 4. The convolutional layers detect specific features in the image. The early convolutional layers detect primitive shapes like edges and blobs; the later layers detect complex features like car tires, tree trunks, or the limbs of a pedestrian. Each convolutional layer is divided into groups of neurons called *feature maps*, where each feature map detects a specific feature. Each neuron in the feature map only connects to a small subset of neurons in the previous layer so that the neuron detects a certain feature in a small region or window of the image.

The pooling layers condense the information from the previous convolutional layer and send the condensed information to the next convolutional layer. This process is called *pooling*. The purpose of pooling is to reduce the amount of neurons and connections in the network, which allows the the network to be trained more quickly. The architecture of a DCNN and an RNN can be combined to form a *reccurent convolutional neural network* (RCNN).

4. CURRENT APPLICATIONS

In addition to vehicle navigating, neural networks are doing many other innovative things in the various industries. Microsoft, Google, and other companies use RNNs for speech recognition applications, such as voice-to-text, voice search, and real-time translation [1]. DCNNs are being used by many companies including Facebook and Google to extract information from photos. Some of these DCNNs have proven to be more accurate than humans [13]. DNNs have also been used to beat humans in complex board games and

video games. The Korean world champion Go player was defeated by AlphaGo, a DNN made by DeepMind. DNNs are also being used to make video game characters behave more like humans by watching them play. This enables a more realistic and life-like video game. Another major industry that uses neural networks is healthcare. Neural networks have been demonstrated to be a useful tool for predicting and diagnosing diseases like diabetes and cancer. They are also used for electronic signal analysis, medical image analysis, radiology, and cardiology.

5. COMPUTER VISION TECHNOLOGIES

Autonomous vehicle driving systems must be situationally aware at all times. To be situationally aware, the system must be able to detect and classify objects around it. Important objects that must be detected include: cars, trucks, road signs, pedestrians, roads, and road lanes. Systems use many different tools and sensors to do this including: infrared, RADAR, SONAR, LIDAR, GPS, and digital video. Systems have many sensors for redundancy and to increase performance and safety. In this section we will explain detecting objects with both digital video and LIDAR.

5.1 Digital Video

Regular cameras are a primary sensing tool used by autonomous driving vehicles. Cameras are mounted around the vehicle in pairs with overlapping fields of view. This provides stereo vision which is similar to eyes on a human.

The main objective for detecting objects in digital video is to place accurate bounding boxes around the objects in the video frames. A *bounding box* is the smallest possible rectangle that encloses the object, as shown in Figure 5. When

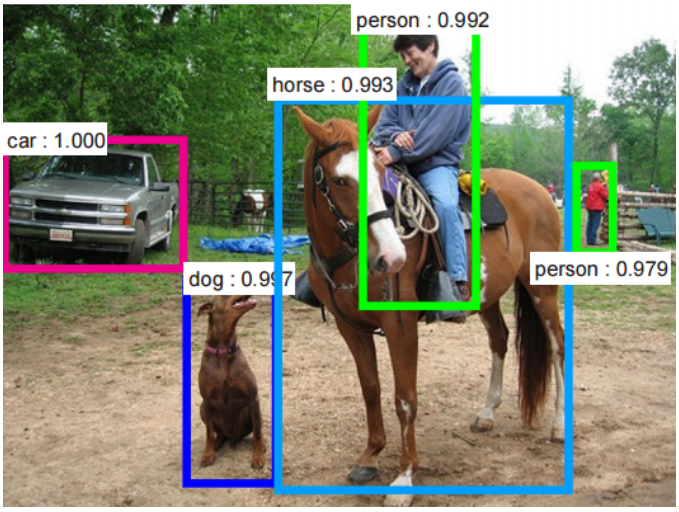

Figure 5: An example of bounding boxes

a system has two different viewpoints and has determined accurate bounding boxes, the system can accurately determine the size and location of this object in 3D space. There are many algorithms and tools that can detect objects in an image. However, trained DCNNs have been shown many times to be the most accurate and efficient [10] [7]. It is important for the system to find an accurate bounding box because if the box is too large or too small the system will think the object is larger or smaller than it is in 3D space. If the bounding box is displaced, then the system will think the object is in an incorrect location in space. It is also important for the system to be continuously aware of the location of objects. To achieve this, a system will find new bounding boxes several times every second [11]. This means that the process of finding the bounding boxes must be very efficient, and it also means systems must be equipped with powerful hardware to handle the computations.

Once the system has detected objects in the video frames, it must classify the objects. For example, it needs to determine whether each object is a car, pedestrian, traffic cone, or other object. Different objects have different behaviors; a pedestrian has different behavior than a traffic cone, which has different behavior from a semi-truck.

The process of detecting and classifying objects and placing bounding boxes around them involves many steps. The raw data coming from the cameras is often fuzzy or noisy. Therefore, the raw video frames are commonly first preprocessed to reduce noise, which increases the performance and accuracy of the later steps. Then a common approach is to send the processed data through a well-trained RCNN that will detect objects and place bounding boxes around them. There are various approaches to find the bounding boxes but most approaches break it down into smaller steps using a series of RCNNs at each step. One approach is to first use a special version of a DCNN that outputs a set of rectangular object proposals, called a *region proposal network* (RPN) [9]. Then, each region in the RPN is passed to a RCNN. This RCNN must be trained to classify important objects like cars, pedestrians, road signs, etc. Finally, once the object has been classified, it will be sent through a final RCNN to refine the dimensions of the bounding box. This process is known as *bounding box regression*. During this process, if an object is classified as a road sign, it will be sent to a separate algorithm or neural network to understand the information on the sign.

5.2 LIDAR

The performance and accuracy of object detection with digital video decreases in adverse weather conditions and when it is dark out. Another sensor used in autonomous vehicle driving systems is *light detection and ranging* (LIDAR), which performs well in all conditions, even when in complete darkness. LIDAR sensors are commonly placed on the top and center of vehicles. LIDAR scans an area by emitting laser pulses, and then measuring the time it takes for a return signal to be received. These laser pulses use a wavelength and intensity that makes them invisible and harmless to people and animals. The time measured from the return signal is used calculate distance from objects up to 200ft away [11]. This generates a precise 3D map of the vehicle's surroundings, as shown in Figure 6. The information from the 3D map is passed into the RCNNs discussed in section 5.1 as additional input data which forms additional feature maps.

LIDAR is also very good at detecting the edges of a road and road markings. If the edge of the road has a curb, then LIDAR detects there is a change in height. In addition to recording the time of a return pulse, LIDAR systems also record the intensity, or the magnitude, of the return pulse. Surfaces with high reflectivity return a higher intensity pulse. So the pavement on the road will return a different intensity value than dirt or grass that may be on the

edge of the road. Furthermore, road lanes and other road markings that are marked with paint are also easily read by LIDAR because the painted road will return a different intensity value [8].

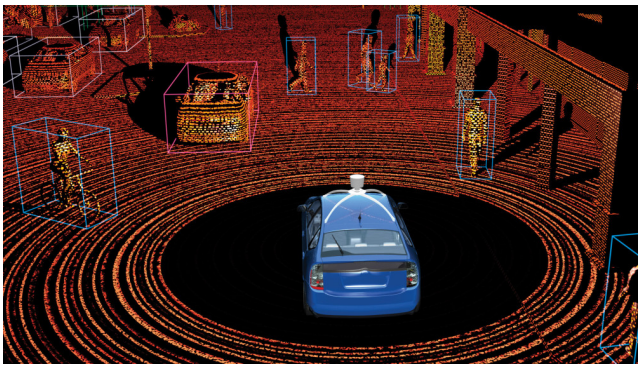

Figure 6: 3D map created by LIDAR

6. VEHICLE NAVIGATION

In the previous section we explained how autonomous vehicle driving systems use a variety of sensors to collect and process data from the outside world. Systems integrate all the data collected from the sensors to build a virtual world, or an internal 3D map, of the vehicle's surroundings. This is called *high definition mapping*. This virtual world is very similar to a virtual or digital world in a video game. Teaching a vehicle to navigate using this virtual world is very similar to teaching a video game character to navigate a video game world.

Vehicles use the data from this internal 3D map to navigate to their destination while avoiding obstacles. There have been several of approaches implemented to navigate vehicles using various tools, algorithms, and neural networks. We will present approaches that use neural networks. The approaches we present all have one thing in common: the vehicle driving system is not programmed to navigate by teaching it various rules. The system learns how to drive in the same way that a human learns how to drive. The system watches humans drive and learns from their patterns, and then it drives on its own, practices, and improves its performance.

6.1 Obstacle Avoidance

Here, we present a common approach used for navigating many kinds of robots and vehicles over the last decade. In this approach, there is a set destination the vehicle is trying to reach, and usually many static obstacles in the way. The destination is one of the inputs to the network, commonly in the form of coordinates or vector with the angle and distance. The rest of the inputs to the network form some kind of representation of all the obstacles. These obstacles could be defined in terms of their dimensions and location of the objects, or it could be the angle and distance of objects in the vehicle's view. Finally, the network outputs a direction or angle for the vehicle to steer and the amount the vehicle should accelerate or decelerate, as shown in Figure 7.

We know that in real world driving situations there are many obstacles that are not static. There are many obstacles that are moving and changing. This approach can be

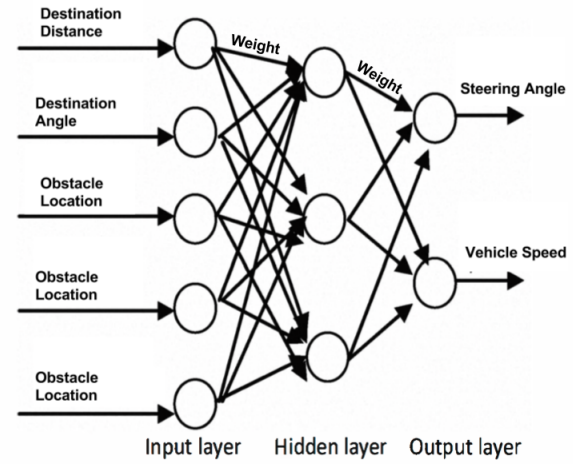

Figure 7: An example of a vehicle controller

modified so that the inputs to the network are not only the location of obstacles, but also the direction and speed of the obstacles in the form of a velocity vector [4].

6.2 Path Planning

In the approaches we have presented so far the neural network would only output a single steering instruction. In more complex approaches, the network will output a *collision-free path* or trajectory to the destination or to a intermediate destination. Once the path has been generated the vehicle can follow the path by continuously pointing the front wheels of the vehicle to be parallel with the path. As the vehicle drives, a new updated path will be generated every few seconds or milliseconds depending on the system. This approach alone does not react to obstacles moving onto the path. Many systems will use this mid-range path planning approach along side a short-range obstacle avoidance approach [12].

Many systems will generate multiple collision-free paths to the destination, choose one path as the optimal one, and save the other paths as backup paths. In case the optimal path becomes obstructed, back up paths are already prepared.

6.3 Modern Self-Driving Cars

Many semi-autonomous driving cars are available and on roads today. Many more will be available in the near future as well as fully autonomous driving cars or self-driving cars. In this section, we discuss details on how these cars will function. In all the vehicle navigation approaches we have presented so far used a feed-forward network. However, modern systems today use RNNs. A vehicle driving system loops several times every second depending on the system. In each loop, the system perceives the current state of the world using all of its sensors, processes the information, builds a 3D map of its surroundings, and sends that information into the the input layer of one sequence of the RNN, then uses the output from that sequence of the RNN to control the vehicle, as shown in Figure 8.

The specifications of the networks used in modern self-driving cars is not available, but it is reasonable to believe that the inputs to their RNNs will include the location of cars, pedestrians, and other obstacles and their velocities. The edges of the road and road lanes will also be inputs

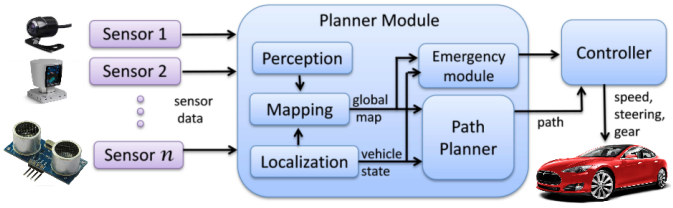

Figure 8: A flow chart of 1 loop in a driving system

that can be interpreted as obstacles. Other inputs would be information from road signs and traffic lights. It is likely that self-driving car systems will keep an updated list of collision-free paths as they are driving. They must also be equipped with a powerful computer on-board the car to process the data from the sensors and to do the forward propagation calculations on all the neural networks.

Like the other systems we presented, self-driving cars are trained by watching humans drive and learning from their behavior. This process will train the system to perform similar to the way the human driver performed. However, car manufacturers want their cars to have super-human performance. Google has created a simulator to test and train self-driving cars. In a few hours they can test thousands and thousands of complex and dangerous driving scenarios, which would otherwise take decades to test in the real-world [5]. In real world driving, when humans encounter a complicated dangerous situation, say a reckless driver, or a vehicle's tire pops, it is unlikely that they have ever encountered a similar situation, and if they had it is unlikely they would remember how to react. On the other hand, an autonomous driving system would have been trained how to react based on hundreds of similar situations.

Furthermore, once self-driving cars are on the market, the training does not end. As self-driving cars are purchased and driven on public roads, they will continue to collect new data and train the neural networks even more. The new data collected will be uploaded to that self-driving car's manufacture's server. The network on the server is then trained with that new data. The server will periodically push out updates to the self-driving cars. This implies that when a self-driving car encounters a new unique and potentially dangerous situation, every other self-driving car that uses that network will learn and improve from that situation.

7. CONCLUSION

To conclude, we have presented some advanced neural networks which are key technologies used in autonomous vehicle driving. Self-driving cars use DCNNs to process sensory information from video cameras and LIDAR to quickly and accurately to detect surrounding objects. RNNs are used to navigate self-driving cars safely and smoothly. Google has been testing fully autonomous driving cars on public roads for 5 years now. Several major automotive companies have announced they plan on selling vehicles with autonomous driving features in the near future. The US Department of Transportation plans on working closely with these companies to help them get past legal barriers. These changes in transportation infrastructure will lead to dramatically reduced motor-vehicle deaths in the future.

Acknowledgments

Thanks to KK Lamberty, Elena Machkasova, Nic McPhee, Skatje Myers, Joyous the Song, and Josh Chapman for their help and feedback.

8. REFERENCES

[1] *A Recursive Recurrent Neural Network for Statistical Machine Translation*. ACL, June 2014.

[2] N. H. T. S. Administration. Traffic fatalities fall in 2014, but early estimates show 2015 trending higher, 2015.

[3] M. Bianchini and F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, Aug 2014.

[4] I. Engedy and G. HorvÃąth. Artificial neural network based local motion planning of a wheeled mobile robot. In *Computational Intelligence and Informatics (CINTI), 2010 11th International Symposium on*, pages 213–218, Nov 2010.

[5] M. Harris. Google lobbies to test self-driving cars in matrix-style virtual world, Aug 2014.

[6] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[7] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1725–1732, Washington, DC, USA, 2014. IEEE Computer Society.

[8] T. Li and D. Zhidong. A new 3d lidar-based lane markings recognition approach. In *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pages 2197–2202, Dec 2013.

[9] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[10] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[11] M. Szarvas, U. Sakai, and J. Ogata. Real-time pedestrian detection using lidar and convolutional neural networks. In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 213–218, 2006.

[12] M. U. A. A. H. Umar Farooq, Muhammad Amar and S. O. Saleh. Design and implementation of neural network based controller for mobile robot navigation in unknown environments. *International Journal of Computer and Electrical Engineering*, 6(2), April 2014.

[13] T. Weyand, I. Kostrikov, and J. Philbin. Planet - photo geolocation with convolutional neural networks. *CoRR*, abs/1602.05314, 2016.