Autonomous Vehicle Control System Using Convolutional Neural Network

Ravi Kumar Sah[1], Abhijit Howal[2], Prashant Sharma[3], Omkar Poshatwar[4]
[1,2,3,4] *Department of Information Technology, Sinhgad Institute of Technology, Lonavala*

***Abstract*- Convolutional neural networks (CNNs) are a type of layered deep neural network comprised of artificial neurons. These neurons are initially taught a set of rules and conditions, through training, which dictate whether they will fire when given varying inputs. CNNs learn as they are used and make future decisions based on both the taught and learned information. A common application of CNNs is object and feature recognition in images. The CNN identifies features in an image by analyzing data pixels through layers of neurons. This is particularly useful in the field of autonomous vehicles where CNNs can be used to process driving footage and identify possible obstacles. CNNs will often classify sections of the preset image grid that potentially contain an obstacle. Errors that occur are fed back into the network for reclassification and further learning. After the analysis is complete and a final conclusion has been reached, the CNN outputs a signal for the vehicle to perform an action: keep driving, stop, turn, etc.**

***Index Terms*- Convolutional Neural Networks, Deep learning, Autonomous Vehicle, Image Recognition, Obstacle Detection, Depth Estimation, Deep Learning, Machine Vision, Autonomous/Self-Driving Vehicles.**

1. INTRODUCTION

Rushing around, trying to get errands done, thinking about the things to be bought from the nearest grocery store has become a part of our daily schedule. Driver error is one of the most common cause of traffic accidents, and with cell phones, in-car entertainment systems, more traffic and more complicated road systems, it isn't likely to go away. With the number of accidents increasing day by day, it has become important to take over the human errors and help the mankind. All of this could come to an end with self-driving cars which just need to know the destination and then let the passengers continue with their work. This will avoid not only accidents but also bring a self-relief for minor day to day driving activities for small items. Using Deep Neural Network, self-driving car can be achieved. Deep neural networks are computerized decision-making networks that mimic the mammalian visual cortex. The structures of deep neural networks consist of multiple layers of neuron-like components. Their layers allow them to be extremely versatile because they can process inputs through multiple parameters. Subtypes of these networks include convolutional neural networks (CNNs). Convolutional neural networks are traditionally used for image analysis and object recognition. In the past decade, there has been an effort to expand the applications of CNNs, specifically, their use in autonomous vehicles for object detection and depth estimation. In 2016, NVIDIA created an autonomous car using CNN technology. Their car exemplifies and demonstrates the validity of using CNNs in autonomous transportation. Employing CNNs in vehicles has the potential to improve road safety, but this comes with questions about the ethics and liabilities of a computer making decisions in a crash situation. It has not yet been found that a computer driven vehicle could actually keep passengers safer than a human driver. However, the implementation of decision making in autonomous vehicles will create a more sustainable driving environment. In this case, sustainability means there will be decreased negative impacts on the environment and preservation of the vehicle and its parts so as to give them a longer lifespan. Overall, CNNs are still a new technology, but have promising applications in autonomous driving.

II.STRUCTURING A CONVOLUTIONAL NEURAL NETWORK

A neural network is essentially a collection of artificial neurons arranged into layers with inputs from one layer being passed on to the next layer as outputs. One type of artificial neuron is a perceptron.

Inputs are weighted based on their importance to the final outcome and sent to the perceptron's which fire according to a binary system. Each perceptron has a given threshold that decides if it will fire. If the sum of the weighted inputs is greater than the
Perceptron's threshold, the output is one and the perceptron fires. When the threshold is moved to the other side of the inequality, it is renamed the bias where bias ≡ -threshold. This is represented by the equation in figure 1 where the sum of all the inputs and their respective weights is written as w ⋅ x and b is the bias.

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

FIGURE 1. Equation of a perceptron's firing potential

This equation shows that with a large positive bias, it is easier for the perceptron to fire and harder for it to fire with a large negative bias. As layers of perceptrons are added, the network is able to make more complex decisions. However, the outputs are binary, so changing the weight of one perceptron can greatly change the network's final output.

Convolutional neural networks contain multiple types of layers through which all data is fed. These layers are arranged in a hierarchical manner and can include convolution layers, pooling layers, fully connected layers, and a loss layer. Each layer has its own focus and purpose in the process of analyzing data. With every successive layer, the analysis becomes more abstract. In image recognition, this means the first layers react to stimuli such as oriented fields or changes in light intensity, while later layers determine the identification of an object and make intelligent decisions about its importance. This is a large generalization of what the layers "look for" in an image. The layers actually process each pixel of the image based on the mathematical functions in their neurons. While all layers are made of neurons, not all layers serve the same purpose.

The convolution layers are the most involved layers. To understand how these layers work, it is important to first understand the concept of convolution. Convolution is officially defined as the integral that shows the amount of overlap of one function as it is moved across another function. In our case, the convolution part of the network refers to a representative kernel being moved across a given image. A kernel is a matrix of values created to detect different features. For example, you may have a 3x3 kernel which is assigned nine separate values. These values are assigned so that the kernel has a specific function. This kernel is then successively centered on each pixel of the image. As the kernel is iterated across the image, the inner product of the kernel and the overlapping image is calculated and that value is assigned to the corresponding pixel. This value is a measure of similarity between the input image and the kernel. The application of a kernel to an image is demonstrated in figure 3 where the pixel being analyzed is referred to as the source pixel and the pixel in the output image is the destination pixel.

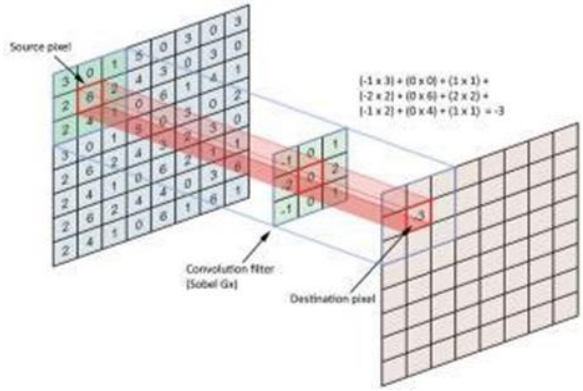

FIGURE 3. A kernel analyzing an arbitrary picture

In autonomous vehicles, the kernels are designed to detect specific features such as edges. When the convolution reaches a local maximum, that position is classified as an edge. Even within edge detection, different kernels are needed. Depending on their values, kernels can be used to detect differently oriented edges. Figure 4 shows an initial image alongside examples of different edge detection kernels and how they affect that image.

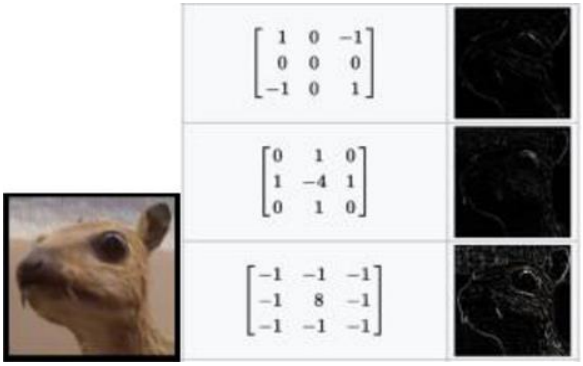

FIGURE 4. Original image (left) filtered by example edge detection kernels

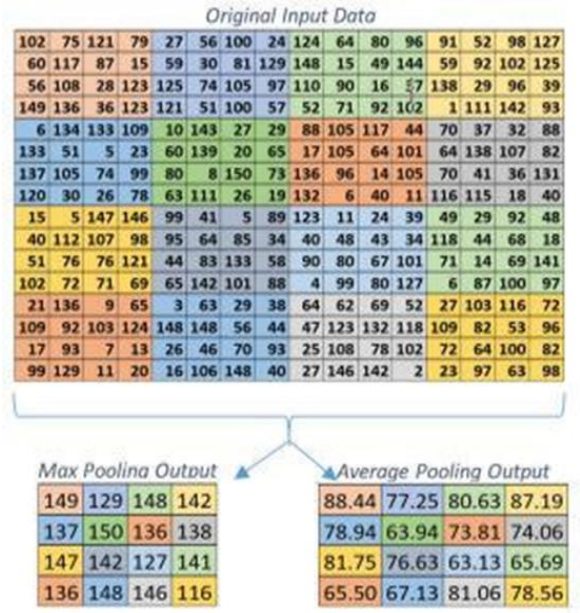

FIGURE 5. Max pooling versus average pooling of input data

As shown in figure 5, each 4x4 sub region is condensed to one value and the outcome is a 4x4 matrix which represents the original data. In a CNN, this smaller matrix allows for faster processing as it is passed to future layers. In the applications discussed in this paper, the most commonly used pooling method is max pooling. By using max pooling, insignificant data points such as those not containing an edge (lower likelihood of containing an obstacle) are eliminated and the future layers can easily analyze regions with higher risk. A pooling layer is thus a way to focus the CNN on regions which contain potential obstacles.

Fully connected layers usually come at the end of the network to condense the data to one final output. Fully connected layers function similarly to convolution layers in that they both produce inner products. However, every neuron in a fully connected layer is connected to every output of the previous layer whereas in the convolution layers, neurons are connected to groups of outputs. This means the fully connected layer analyzes all of the data points simultaneously without performing a convolution function. Figure 6 shows a typical CNN structure starting with convolution layers of increasing intricacy and ending in a fully connected layer to condense the data.

The filtered images as shown in figure 4 would then be fed to the next layer thus allowing the network to narrow down which areas of the image contain potential obstacles. Not all image areas are important and some can easily be eliminated with little cost to the final outcome. Pooling layers are one way to reduce the spatial size and number of parameters in the input data. They take a sub region of data and condense it to one representative value. To best explain this concept, we will take a 16x16 example matrix in figure 5 and perform pooling operations on it using two different methods.

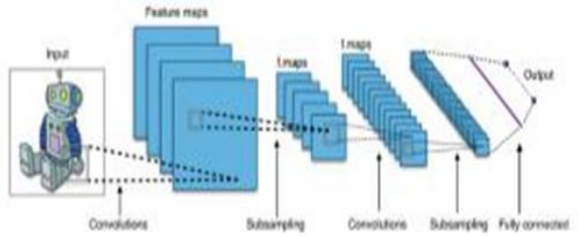

FIGURE 6. Example of the typical layered structure of a CNN

By placing the fully connected layer at the end, the network is able to collect in-depth data in many dimensions through its convolution layers, but condense this information into a readable output in the final fully connected layer

The loss layer comes at the end of the network and functions as an error feedback loop. This type of layer is only needed for problems involving learned parameters such as when using training examples. Convolutional neural networks learn using stochastic gradient descent and back propagation. The goal is to make the predictions of the CNN match the ground-truth (original input image) by minimizing a cost function. To train a CNN, it must be run in both a feed forward and feedback configuration. Over the forward run, small errors are collected and analyzed by the loss layer. The cost shows the margin between what the CNN recognizes and the ground-truth. This

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2.$$

FIGURE 7. Equation of a common cost function

Nielsen defines the variables as "w denotes the collection of all weights in the network, b all the biases, n is the total number of training inputs, a is the vector of outputs from the network when x is input, and the sum is over all training inputs". In this equation, y(x) represents the desired output and a is the output the network has provided. As you can see,

the closer a gets to y(x), the smaller the cost C. Gradient descent is the algorithm used to find the collection of weights and biases that will allow C to reach its minimum value. The main idea behind the gradient descent algorithm is to find the gradient of the function to be minimized and move the values of the variables in the opposite direction so as to guarantee the gradient will always be decreasing. This is the point where back propagation comes in. Back propagation occurs when the network is run in the backward direction to adjust the parameters of each layer. In the case of a CNN these parameters are the kernels. The loss layer collects the errors and analyzes them with the gradient descent algorithm. Then, adjustments for the parameters are propagated back through the layers of the network. Figure 8 illustrates errors being sent back through a sample network.

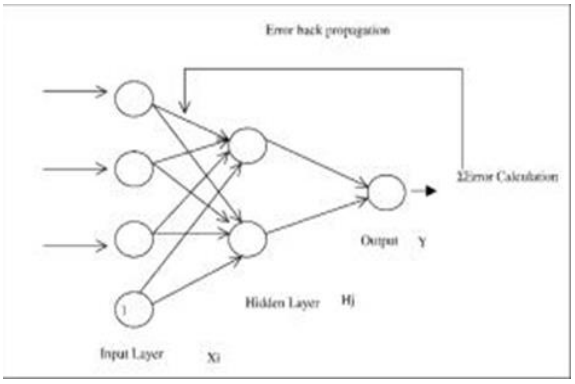

FIGURE 8 Example of an error feedback loop

It is important to realize that are calculations are not part of the permanent layer, but rather they will be removed once the network is finished being trained. These error calculations occur every time the network is fed a subset of training images. It is only after the network has been sufficiently trained that it can be put to use for image recognition or for use in a vehicle.

III. IMAGE RECOGNITION

Convolutional neural networks are applied in many image recognition problems. However, they are currently used mainly in the identification of specific objects. This means that, when used in a car, the network analyzes the input image and finds areas that have a specific feature such as another car or a pedestrian. In order to classify an object within an image the CNN must know what to look for. This is why network is given set of training images. The general hierarchy for the identification of an image is as follows: pixel → edge → texton → motif → part → object. Pixels and edges are just as one might expect. Textons are "fundamental micro-structures in generic natural images and the basic elements in early (pre-attentive) visual perception". They are small sections of texture or pattern which are then combined into motifs. Motifs are sections of repeating patterns that can later be combined into larger image parts.

These parts are then combined to form a whole image to be identified. Because the network has been trained, it can use its set of specific kernels to identify this image.

The first step in classifying an image is to break it up into sections, typically pixels, however these sections can be larger than pixels. The image is then analyzed by the CNN. Each layer's kernels are designed to extract specific features.

Each layer's kernels are designed to extract specific features. These kernels follow the previously mentioned hierarchy and become more intricate and specific as the layers progress.

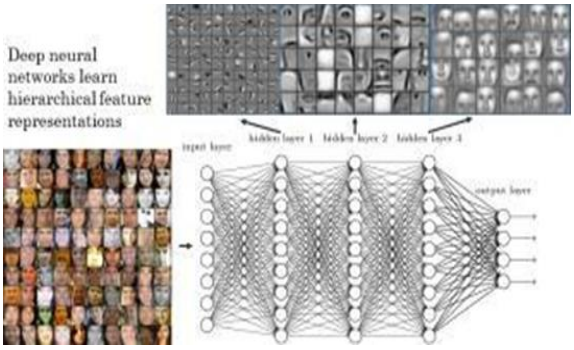

FIGURE 9 Positioning of kernels within a facial recognition CNN

Currently, most CNNs are designed to classify a specific object such as handwriting, faces, or animals. For CNN classification to be effective in autonomous vehicles, the network needs to be able to classify a diverse set of objects or be able to recognize simply what is a potential obstacle and what is not.

IV. CONVOLUTIONAL NEURAL NETWORKS IN AUTONOMOUS VEHICLES

Image identification in autonomous vehicles is not as simple as facial or handwriting recognition because vehicles need to process a full 360 degree dynamic environment. This creates the need for dual frame

processing because collected frames must be combined and considered in context with each other. A vehicle can be equipped with a rotating camera to collect all relevant driving data. The machine must be able to recognize metric, symbolic, and conceptual knowledge as demonstrated in figure 10.

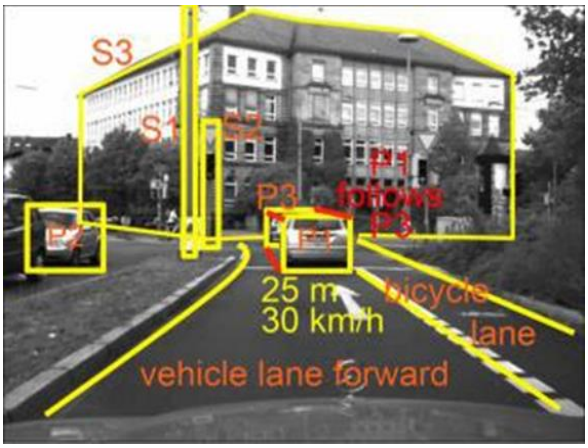

FIGURE 10 Metric Knowledge (yellow), Symbolic Knowledge (orange), and Conceptual Knowledge (red) applied to a driving scene

V. CONCLUSION

The capabilities of CNNs to distinguish both the presence and depth of obstacles makes them promising backbones for autonomous transportation. However, the ethics of collision decision making still provide a considerable obstacle for the use of autonomous vehicles in everyday life as they have not yet proven to be safer than human drivers. On the other hand, these vehicles will promote sustainability by decreasing environmentally harmful emissions and reducing wear on vehicle parts. The more research that is compiled relating to CNNs in autonomous vehicles, the closer we are to introducing these vehicles as a main form of transportation.

REFERENCES

[1] M. Nielsen. “Neural Networks and Deep Learning.”, in 2017

[2] J. Wu. “Introduction to Convolutional Neural Networks.”, in 2016

[3] A. Karpathy. “Convolutional Neural Networks for Visual Recognition.”, in 2016.

[4] B. Jia, M. Zhu, W. Feng. “Obstacle detection in single images with deep neural networks.”, in 2015.

[5] Souhail Guennouni, Ali Ahaitouf, Anass Mansouri. “Object detection using OpenCV on an embedded platform.”, in 2014.