International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 41, No. 1, February, 2024, pp. 469-476
Journal homepage: http://iieta.org/journals/ts

Optimization of Deep Neural Networks for Enhanced Efficiency in Small Scale Autonomous Vehicles

Shabana Urooj[1], Mudasir Ahmad Dar[2], Shabana Mehfuz[2*], Wafaa Shoukry Saleh[1]

[1] Department of Electrical Engineering, College of Engineering, Princess Nourah bint Abdulrahman University, PO Box 84428, Riyadh 11671, Saudi Arabia
[2] Department of Electrical Engineering, Jamia Millia Islamia (A Central University), Delhi 110025, India

Corresponding Author Email: smehfuz@jmi.ac.in

https://doi.org/10.18280/ts.410141

Received: 20 March 2023
Revised: 1 July 2023
Accepted: 10 August 2023
Available online: 29 February 2024

Keywords:
A.I. autopilot, autonomous vehicles, deep neural networks

ABSTRACT

Autonomous vehicles of the contemporary era constitute a sophisticated blend of artificial intelligence and electronic components. These vehicles operate autonomously by employing neural networks trained to interpret visual input from multiple onboard cameras and subsequently produce corresponding steering angles. However, the existing neural networks are characterized by their substantial scale, necessitating substantial GPU resources, and are prone to latency issues and complex architectural requirements. These factors render these networks unsuitable for small-scale applications where latency, complex architecture, and expensive hardware are prohibitive. This paper proposes a methodology for optimizing these neural networks for small-scale operations while preserving their accuracy and precision. This is achieved through a fine-tuning process that customizes the architecture and modifies various functional values and their parameters, resulting in a deep neural network tailored for small-scale applications. This optimized network boasts a simpler architecture, lower storage requirements, and reduced demand for GPU resources. The network is developed, trained, and evaluated using TensorFlow, a widely employed API for machine learning applications. The optimized network offers several advantages, including reduced latency, a customizable architecture, minimized memory requirements, and decreased GPU demand, making it a viable solution for various applications. The paper provides a detailed exploration of the development of this bespoke deep neural network and its potential implications for the future of small-scale autonomous vehicles.

1. INTRODUCTION

Automation, a technological advancement that minimizes human intervention, has been instrumental in various sectors, enabling machines to adapt to human needs. Notably, the scientific realm has demonstrated an intense focus on autonomous vehicles over the past few decades. The importance of automation in research cannot be overstated; it enhances efficiency, productivity, and reduces potential human error [1, 2]. By automating repetitive tasks, researchers can dedicate more time to significant aspects such as data analysis and hypothesis testing. Furthermore, automation allows large-scale data analysis and experimentation that would be challenging or impossible to achieve manually. The end results are faster outcomes, comprehensive datasets, and uncovered insights that might have remained hidden, accelerating the pace of scientific discovery [1, 2].

The breakthrough in autonomous vehicles was made possible with the advent of deep learning, a revolutionary technique that replaced traditional programming. Deep learning's ability to learn and adapt based on data has made it applicable in a wide range of fields, including the automation of vehicles [1, 3]. The advent of Artificial Neural Networks, inspired by the human brain, has been incredibly beneficial in creating truly autonomous vehicles. These networks can be trained on different roads and drive on them post successful training [1, 3].

Several companies have promised autonomous vehicles, but Tesla Inc. has been the most successful in delivering them. This American company has developed, produced, and sold autonomous vehicles with its patented autopilot, widely considered the finest in the market [1]. Despite the development of numerous convolutional networks, the focus of this paper is on the customization of AlexNet and PilotNet, two well-known deep neural networks used in autonomous vehicles. These networks require large GPUs and extensive memory spaces, leading to high costs. Therefore, there is a pressing need for engineers to optimize these networks to simplify them and reduce memory and GPU requirements, especially for small-scale applications like warehouses, laboratories, factories, or industries [2].

The process of optimization involves customizing the neural network, using various techniques such as fine-tuning, dropouts, batch normalization, regularization techniques, and hyperparameter fine-tuning. This ensures that the network is simpler, has fewer latency issues, and maintains accuracy in line with traditionally used networks (AlexNet & PilotNet) or within acceptable ranges [2, 4].

The motivation behind this work is to address the limitations of current deep neural networks used in autonomous vehicles and propose a need for optimizing these networks for small-scale operations. The objective is to create customized deep neural networks that are specifically tailored for small-scale operations, offering reduced latency, simplified architecture, lower memory requirements, and decreased GPU usage [2, 4].

1.1 Objectives

The recent surge in interest and contribution to artificial intelligence has highlighted the potential of deep neural networks in solving automation problems of varying scale and complexity. However, building a deep neural network from scratch for an autonomous vehicle, considering the application, scale, and complexity, becomes a time-consuming process. Hence, the objective of this work is to fine-tune existing neural networks for automation tasks of varying scale and complexity, particularly for small-scale and low-complexity tasks, while maintaining relative accuracy [2, 4].

1.2 Contributions

This work aims to optimize existing neural networks for small-scale operations while maintaining similar accuracy and precision levels. The main contributions include customizing, fine-tuning, and optimizing neural networks for small-scale and low-complex applications. Various methods are employed, such as augmentation, addition of layers in the architecture, modification of architecture parameters, data preprocessing and normalization, and batch normalization. Lastly, these models are trained and tested, and a comparison of various performance metrics (such as accuracy loss) is conducted [2, 4].

The remaining sections of the paper are organized as follows: Section 2 reviews related work, Section 3 provides an overview of machine learning and deep learning, Section 4 offers a brief introduction to autonomous vehicles, Section 5 presents the proposed optimized deep neural network, Section 6 explains the proposed DNN architecture, and Section 7 provides training and testing results. Finally, Section 8 elaborates on the findings, followed by the conclusion.

2. RELATED WORKS

Artificial Intelligence (AI) has seen tremendous popularity both in the media and also in the engineering community. The technique of deep learning has been implemented by engineers to solve various problems in the field of automation, speech recognition, image processing, data analysis, etc. Naturally, when the idea of autonomous cars was conceived, the engineers looked into AI for help and used deep learning. They developed deep neural networks to do the transformation on data which is gathered by various sensors and methods [5, 6]. There are many types of neural nets such as convolutional neural networks, recurrent neural networks, auto encoders, etc. which can be used in the application of autonomous vehicles. Out of these convolutional neural networks is most famous because it is efficient at performing the required task and consists of various layers which perform operations and transformations on images. Naturally, the architecture of such neural net has many parameters and variables which in turn accept various values and holds different functions. Therefore, we can fine tune them according to the application and keeping various parameters in mind. Techniques such as augmenting of architecture such as layers, using of normalization methods such as batch, L2 and use of dropouts, etc can help in optimizing and customizing a neural network. The effect these techniques can be found in the studies [7-10].

The aim of the work is to combine all these ideas and methods while building neural nets and then implement these techniques to customize, optimize and make a net efficient for the required task with keeping the complexity, purpose and application in mind.

3. MACHINE LEARNING AND DEEP LEARNING

Traditional coding or programming had always run into the problem of real time adaptation. Usually, engineers used to write codes for specific applications but since the code wasn't flexible to adapt or change according to any change or parameter variation this made traditional coding run into a wall. So, machine learning was developed and basically meant that we could take help of various algorithms to analyze the data and then after that a learning process would happen. Therefore, in machine learning we could learn from the data and therefore we would be comfortable enough to make generalized predictions about the data.

Hence, machine learning had this feature of being flexible and better than traditional programming as we would not be required to write explicit codes due to learning taking place. Therefore, machine learning saw tremendous application in numerous fields such as data analysis, stock market predictions, image recognition, medical diagnosis, speech recognition etc.

Machine learning can be considered somewhat of a blanket or general term and has many subfields such as deep learning, A.I., etc.

Deep learning is mostly considered a subfield of machine learning and is also sometimes referred to as a technique which can be used to implement machine learning. Currently deep learning is most widely used in many fields such as medical diagnostic field to figure out cancer diagnosis from blocks of data which are usually patient test scans. Deep learning has also found application in autonomous vehicle industry too for developing 'autopilots' for the vehicles.

The algorithms used in deep learning are inspired by human neural networks and therefore are termed artificial neural networks (ANN) because being artificial in nature. The algorithms used must learn from the data just like in machine learning and after learning they are capable of giving general predictions about the data which is useful to us. Their structure is built by units called 'neurons' which are similar to a biological neuron, and these are organized into vast layers. These layers are termed to be input layers, output layers and hidden layers. The hidden layers always exist between input and output layer and usually are numerous in nature which give a model its complexity. Deep learning explicitly uses these artificial neural networks (ANN) with multiple hidden layers. These networks are alsocalled deep neural networks (DNN) and have many hidden layers between their input and output layers and hence are termed to be deep in nature.

3.1 Artificial neural networks

Since we have understood how artificial networks are formed by organizing neurons into various layers, we use these layers to form a complex structure which is able to do mathematical operations on any data or we could say we are creating mathematical computing structures which can assess data and then learn from that particular data and then provide general predictions on a new set of the similar data [3].

Every layer of the model or structure will serve a different task or function and can-do mathematical operations. The data enters the structure or model through its input layers and then passes through the hidden layers and then the output layer which is termed as a 'forward pass'. We must do these forward passes through the model when we are training the model.

The data which is fed to the network has many components or dimensions and these components of the data determine how many nodes in the input layer we require. The hidden layers have generally random number of nodes which we choose randomly. The node in the output layer is dependent on what desired prediction we are looking for from the model. Hence, artificial neural networks can be constructed with specific applications in mind and then trained on our data sets and after the training is complete, they can be used to perform predictions on new data sets.

3.2 Basics of artificial neural networks

To succinctly understand ANN lets define the following terms which are associated with artificial neural networks.

a) Layers: These are specifically organized neurons or nodes which form what we call layers and are of different types and perform numerous mathematical transformations on a data. They can be dense layers, convolutional layers, pooling layers, recurrent layers etc. and each has their own operation and are used according to the purpose of the model.

b) Layer Weights: The connection between each neuron in a network has assigned numbers which are called weights. These weights are either randomly generated or can also be specifically chosen. The always get multiplied with the input data as the data does a forward pass through the model.

c) Activation Function: These are usually nonlinear functions used to map a nodes or neurons input to a corresponding output by doing a specific mathematical operation or transformation. The output is a number between an upper and lower limit value which depends on the function used. Examples are ReLu function, threshold function etc.

d) Data Sets in ANN: There are three data sets used in ANN to remove over fitting & under fitting and these are explicitly used for training and testing the model. These are listed below.

i) Training Set: In this set we have the data and the corresponding label of the data which is used to train the model repeatedly.

ii) Validation Set: It has data which is used to validate the model while its training to make sure it isn't over fitting the training data (good at generalizing training data only).

iii) Test Set: This is an unlabeled and different unseen data which is fed to model after training is done to check if model can do successful prediction.

e) Training: It refers to passing data through a network so that it learns the input and output mapping by using different algorithms. It's essentially an optimization problem where the model is changing the values of weights and moving them towards an optimal value. It makes use of a loss function which helps the model in making successful prediction on the given data. Therefore, we repeatedly send the training data through the model and the model starts the learning process.

f) Prediction: After training of the model is complete and we are satisfied that the model has learned well according to different metrics then we do predictions with the model on the test sets. As, the name suggest the prediction means the network is trying to map a given input to an output and if the training is done successfully this mapping will also be a correct output and hence, we will say the model predicted successfully.

g) Batch Size: It's defined as number of the training data samples which are passed to the network at once and usually data is broken down into batches.

h) Epoch: It's defined as the single pas of the entire data through the network.

i) Over fitting: It is a term used to define the phenomenon when a neural network can only classify data it is trained on and not the data it hasn't seen. It is reduced by using data augmentation techniques to add diversity to the training set and also by using drop out to reduce complexity of the model by making some layers not take part in the process.

j) Under fitting: It is a term used to define the phenomenon when the model cannot even classify the data which is present in the training set and is reduced by making the model more complex and adding more features in the input data.

Relation between batch size & epoch:

$$\text{Batches In Epoch}=\frac{\text{Training Set Size}}{\text{Batch Size}}$$

E.g., if training set=100 images

Then 1 epoch is completed when 100 images are passed through the network.

If batch size=10 i.e., only 10 images are passed to the model at one time.

4. AUTONOMOUS VEHICLES

Autonomous vehicles are vehicles which can steer themselves autonomously without human intervention. They are made capable to do this by installing them with a trained deep neural network & a steering control system. The DNN gives appropriate steering angle input to the steering control system & the vehicle is steered autonomously. The deep neural networks (DNN) are backbone and essence of an autonomous vehicle.

4.1 Basic working

An autonomous vehicle has a 'brain' which process the information which is passed to it by its sensors and then it gives a corresponding response to that information which is basically a steering angle command given to the steering control system of the vehicle [3]. The basic block diagram of the working of an autonomous vehicle is shown in Figure 1.

As we can see in the figure, the vehicle is loaded with various cameras and other sensors. Usually, cameras are used which take pictures of the surroundings of the vehicle which is usually the road in front of the vehicle on which it must be driven. These images are taken from various positions but usually three positions set up is used which are referred to as the center, right & left position [3].

The center position denotes the image which is taken from a camera placed at the center of the car usually at the center of the front of the car. The other two positions are relative to this center position i.e., the right position means the camera is placed to the right of the center and similarly the left position is relative to the center where the camera is placed to the left of the center. These three positions are used to get the maximum information about the road and the surrounding which in turn gives the data set diversity and enough components and dimensions which makes the model learn better from the data and predict more accurately on test sets.

Images from these cameras are then taken and passed to the trained deep neural network (DNN) as inputs. The DNN serves as the brain of the operations as it now must process these images and perform mathematical operations on them to give the necessary output [11]. This output is going to be the steering angle which the DNN will give to a steering control system of the car and the car will be steered autonomously by the DNN [9].

The trained DNN has trained on similar images of different roads with labeled information which is always corresponding steering angles. So, once the DNN was trained and tested on numerous road images and corresponding steering angles it was employed in the field and hence will be able to successfully predict the steering angle for the input image present on its nodes [10].

The other sensors which can also be used in combination with cameras are usually ultrasonic sensors (measure the distance of any object by emitting ultrasonic sound waves & then receives reflected wave & calculates the time taken between emission and reception to measure the distance), radar (radio detection & ranging) and Lidar (light detection & ranging) Sensors. These also help in making the autonomous vehicle safer and less accident prone [12-16].

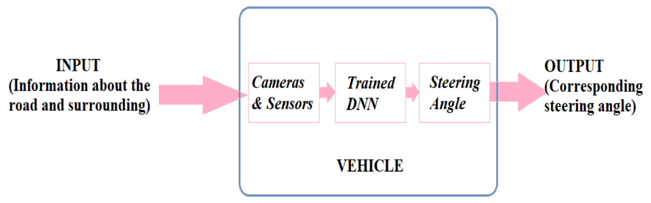
INPUT
(Information about the road and surrounding)
Cameras & Sensors
Trained DNN
Steering Angle
VEHICLE
OUTPUT
(Corresponding steering angle)

Figure 1. Basic block diagram of an autonomous vehicle

4.2 AlexNet&PilotNet

Currently engineers can build different DNNs with different features for different vehicles and accordingly change and update the model. AlexNet&PilotNet are two of the most famous neural networks used for image classification & autonomous application [1, 11]. These Neural nets were developed by researchers initially for image classification or object classification and were eventually modified and made suitable for autonomous driving [5].

AlexNet is a convolutional Neural Network which was made for the prime application of image classification but can be easily modified for autonomous application. It's trained on two graphic processing units (GPUs). It has five convolutional layers with max pooling operation taking place between each layer and three fully connected and it has about 63 million trainable parameters. It can take up to 500MB of memory [1].

PilotNet which is also referred to as Nvidia Convolutional Neural Network is a model specifically developed to improve autonomous vehicle system in DARPA (Defense Advance Research Project Agency). It is also a convolutional neural network and has five convolutional layers and then four fully connected layers and it has about 348,219 trainable parameters [1].

The deep neural networks which are 'traditionally' used for autonomous driving have complex architectures, bigger size, have multiple convolutional layers with max pooling occurring in between those layers and then multiple dense layers etc. These neural networks are also trained using multiple graphics processing units and which require updated specifications and huge processing powers. Due to this the customization and optimization of these networks is important for small scale operation where such big processing units should not be used, and such architectural complexity would be overkill and unnecessary [6].

Hence, when using these neural nets, we need to optimize and customize them so that we can make the architecture simple, memory space less, not dependent on huge GPUs and make latency less. Of course, when such kind of optimization is done tradeoffs occur such as scale and complexity of the application will decrease but as long as the accuracy of the optimized model is within acceptable ranges, we can say the optimization is successful [7, 8].

What we can propose is a customized deep neural network which will have a simpler and customized architecture with lesser layers and can perform within acceptable ranges.

5. PROPOSED OR OPTIMIZED DEEP NEURAL NETWORK

Since, the idea is to feed an image from a camera as input to a trained DNN and get steering angle as output. The network will have convolution layers and we can figure out how many of these are needed to keep the architecture simple by using trial and error method until we achieve a satisfactory result [1, 3].

This will help us to figure out the number of convolutional layers, its dimensions, number of filters etc. This fine tuning of other DNNs and then customizing each layer and adjusting hyper parameters to get desired results is essentially customization and optimization. Similarly, after that we require deeply connected layers the number of which will depend upon the complexity and scale of the task.

The customized model's architecture will have two Convolution layers which will each have max pooling to reduce the dimensions of the output image from these convolution layers and then we will have one flattened layer which will make the output from convolution layer suitable for the dense layers and finally two deeply connected layers with one output node. The architecture of the proposed DNN is shown below in Figure 2.

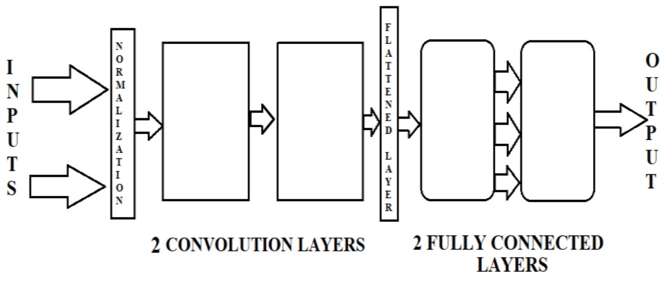

Figure 2. Model architecture

6. UNDERSTANDING PROPOSED DNN ARCHITECTURE

The basic architecture of the model is shown in Figure 3 below and we can see that it has two convolutional layers with max pooling happening between them. The first convolutional layer has 16 nodes and the input shape of 320×65×3 which means the dimension of the input image are 320×65 pixels with 3 representing it is a colored image (RGB) and the filter used for convolution operation or pattern detection has a size of 2×2. The activation function used during this layer is 'ReLu' or rectified linear unit which transforms the input data to the maximum of either 0 or the input data itself. The convolving filter which performs the dot product, or the convolution operation has a size of 2×2. There is also use of zero padding in the layers to make sure the image sizes aren't reduced excessively, and dimensions are retained while the images do a forward pass through the layers. The shape of input images is 320×65 and the output channel from the first convolutional layer reduces it to 160×33 which then is kept through the second convolutional layer.

The model has two dense layers and the first one has 10 nodes and the second one or the output one has one node. There is also a flattened layer between the convolution and the dense layer to make the out from convolutional layers suitable for the dense or fully connected layers.

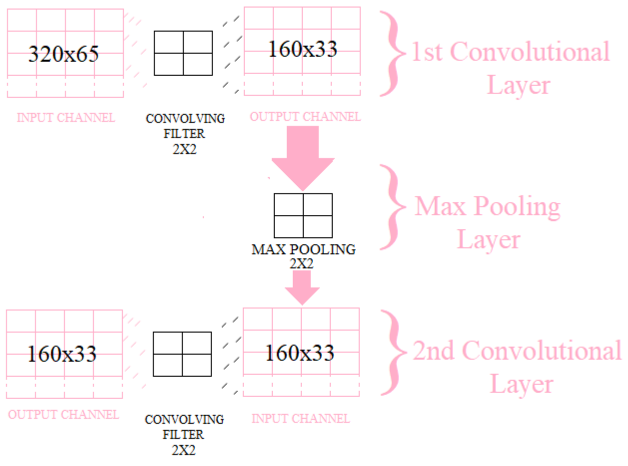

Figure 3. Layers of optimized model

The 1st Convolutional Layer is responsible for extracting features from input images using convolutional filters, which help in identifying patterns and visual characteristics relevant to the task. The max pooling layer then downsamples the output of the previous layer, reducing the spatial dimensions while retaining important information. This layer helps in reducing computation and extracting the most relevant features. Finally, the 2nd Convolutional Layer further analyzes the pooled features, capturing higher-level representations and spatial relationships. This layer provides a more abstract understanding of the input data, enabling better discrimination and classification. Together, these optimized layers enhance the model's ability to extract meaningful features, reduce the dimensionality of the data, and capture complex patterns, ultimately improving the model's overall performance.

6.1 Methods used for customization and optimization

The customization entails changing various layers of DNN according to the need or purpose or application it has to perform. The optimization of a model or DNN entails changing or tuning the parameters, functions and other variables and values which are present in these customized layers till desirable performance metrics are achieved [6-8].

The customization and optimization of the model was done in the following ways:

i. Addition and subtraction of various layers of the DNN. To perform a small-scale operation, it is often required to lessen the complexity of a model and such can be achieved by using dropout which makes some layers not take part in the training process and reduces the complexity of the model. We can also customize such layers by reducing or increasing their nodes, increasing, or decreasing the number of filters in the layers, changing the dimensions of the filters i.e., rows and columns and also initializing the values of the filters depending upon how successfully the filter is performing pattern detection.

ii. Reducing the dimensions of the input images of the feature map change i.e., reduce as it goes through the convolution operation, and it reduces the computational load. Also, the max pooling operation after each convolutional layers helps in reducing over fitting in the model or DNN by keeping values of the most activated pixel which help in making better pattern detection and help in successful feature extraction.

iii. Also, over fitting and variance are reduced by using L2 regularization which is a regularization technique, and it augments the loss function in the DNN by adding certain terms to it which incentivizes the optimizer to update the weights of the model to a value which will make the loss function have a minimum optimal value [6].

iv. Tuning of batch size according to how well the model was learning and an optimum batch size makes the model fit correctly on the data by keeping the computational power of the system in mind.

v. Batch normalization was also applied to the layer where it prevents the weights from updating to a high value which may cascade down to other layers and cause instability. It reduces the impact a high layer weight might influence the training of the model and batch normalization will happen per batch. Normalization will happen to the activation output of each layer [7].

vi. Xavier initialization is used to counter the problem of randomly initialized weights which lead to gradient instability and makes learning difficult for the model. In Xavier initialization the variance of the weights is shifted from 1 to a new value which reduces the gradient problems. For the 'ReLu' activation function the variance is shifted to $\frac{1}{n}$ (where n is the number of weights connected to the previous layers).

vii. Data pre-processing done to the training data set involves cropping the images from their original size to a size where there is less useless information in the image. Usually, the sky and other surrounding objects are cropped from the image. The images are also flipped and doubled to increase the data set diversity which helps in training and gives model the ability to successfully generalize on unseen data after training.

viii. Tuning of learning rate was also done. The value set for learning rate is a hyper parameter and setting it too low and too high will lead to problems of learning and computation and therefore this value is set with trial-and-error method.

6.2 Comparison

The comparison between the three DNNs is shown in Table 1.

Table 1. Comparison of DNNs

	Optimized Model	AlexNet	PilotNet
Convolution Layers	2	5	5
Dense Layers	2	3	4
Trainable Parameters	1,691,909	63,000,000	348,219
Memory Size	2 MB	508 MB	5 MB
Training Epochs	20	20	20

7. TRAINING AND TESTING RESULTS

The optimized model is trained on the data collected from various track of a gaming simulator called Unity game simulator and is built by Unity Technologies. So, we would require the training & test data set to have the images of the various tracks which are present in that gaming simulator. The game simulator is shown in the Figure 4.

The simulator allows us to record the various laps in different tracks as images and corresponding information is labeled to them. The information labeled to the images is the steering angle, throttle, and speed and brake values. These values are normalized for the DNN as normalization is an important preprocessing step. The training set for the model F will contain these images of various tracks with labeled information so that the model can learn accurately and sufficiently so that in the testing phase it is able to perform accurate generalizations on unlabeled and unseen data. The example of the data sets is shown in the Figure 5.

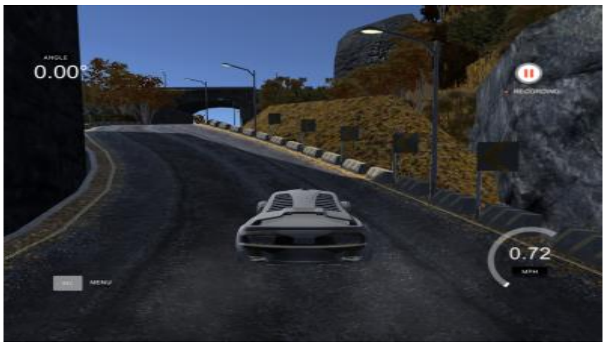

Figure 4. Gaming simulator

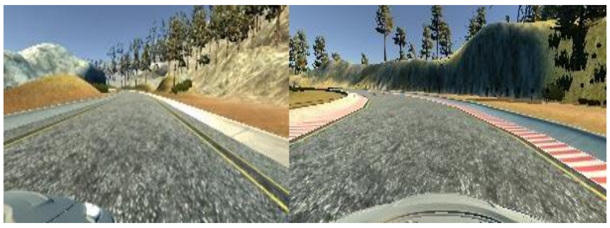

Figure 5. Example of data set

The images are from the various tracks which are in the game simulator. The training set will contain labeled images of the tracks which are obtained by manually driving the car on these tracks for multiple laps with the help of a gaming controller. The labels to the training images are their corresponding steering angles. These steering angles are normalized values. The total number of images collected is 30,000 and out of these 20% will be used for the validation set and 80% for the training set.

Since we require three data sets namely training set, validation set and test set. We can easily use 80% of the collected images from the simulator for the training set and the remaining 20% for the validation set. For the testing set we will use different unlabeled images taken from a separate lap. The test set can also have images from a different track which is also built in the same game simulator. The optimized model was trained on 24,000 images and initially the epochs were set to 15 and the training set images were not resized, and the results are shown below in Table 2.

In order to improve the training accuracy of the model, the training set images were augmented by resizing, cropping them and taking away the portions of the image which didn't have any useful information. Initial the images were 320×160 pixels and after resizing they were 318×102 pixels. After doing the resizing of the training set, the model was trained again with epochs set to 20 and the training results are shown below in the Table 3. These training results are considered the final training results of the model.

In the Figure 6 the model loss for training and validation is shown for 20 epochs for each model i.e., AlexNet, PilotNet and the optimized model. After training the optimized model was tested on 10,000 unlabeled images and the results are shown in Table 4 and Figure 7.

Table 2. Initial training results

Initial Training Results					
Training Set	Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy
24,000	15	0.1219	0.8700	0.0427	0.8400

Table 3. Final training results

Final Training Results					
Training Set	Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy
24,000	20	0.0811	0.9100	0.1936	0.8900

Table 4. Testing results

Testing Results		
Testing Set	Correct Prediction	Wrong Prediction
10,000	8900	1100

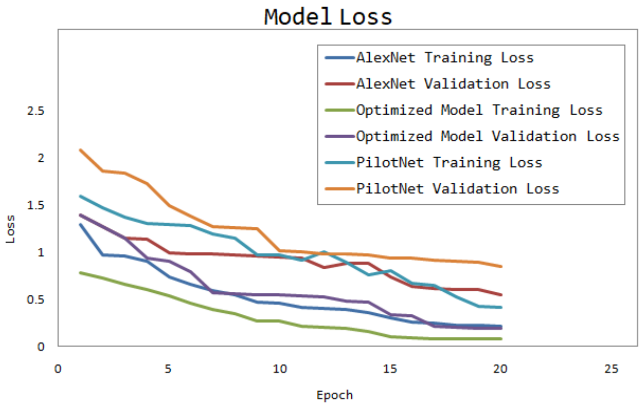

Figure 6. Training and validation loss for all models

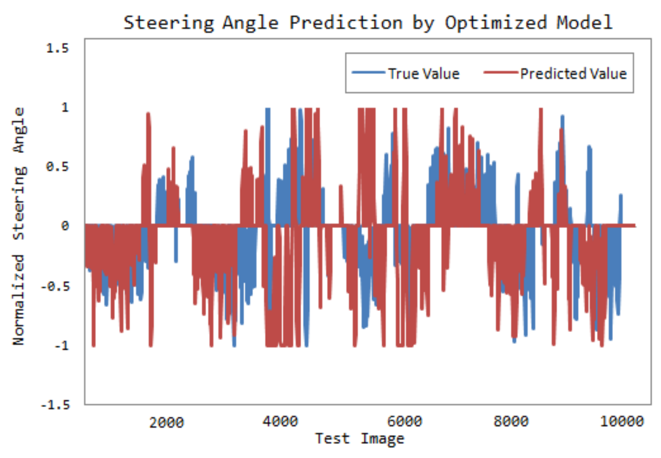

Figure 7. Normalized steering angle (º) prediction by optimized model

In the Figure 7, the steering angle (º) predictions by optimized model has been shown for 10,000 test images. This work holds significant managerial implications for the application of autonomous vehicles. By optimizing deep neural networks for small-scale operations, the research offers practical benefits to managers and decision-makers in the industry. Customized neural networks with reduced latency, simplified architecture, and lower memory and GPU requirements can enable the deployment of autonomous vehicles in contexts where resource constraints and cost considerations are critical factors. This opens up opportunities for implementing autonomous vehicle technology in various industries, such as logistics, transportation, and delivery services, even in scenarios with limited hardware resources. The findings of this study provide valuable insights for managers looking to leverage autonomous vehicle technology in a cost-effective and efficient manner, thus facilitating its adoption and implementation in real-world applications.

8. CONCLUSION

The optimization of any neural net is a continuous process as various hyper parameters have to be changed using trial and error methods or until the desired results are achieved. There will always be therefore, tradeoffs and tolerance for margin of errors. The net or model which has been optimized and customized for a small-scale operation shows that training metrics improve after customization and optimization. The customization of convolution layers, the selection of various parameters of these layers and the overall fine tuning is a decent jumping point or reference point to understand what more can be done to add more gains to the progress and success which will reflect positively on the training and testing parameters. Optimizing neural nets for various applications involves comprehensive and time-consuming work to make sure these hyper parameters are selected in such a way that yields positive results.

The optimization of the model for small scale application involved customizing the different layers of the model and then changing or setting the hyper parameters to make sure the metrics aligned with desired results. The work outlined here also involved training that particular model on a vast data set and yielded positive results in terms of training and testing parameters. However, it is still important to note that this work had certain caveats where the input variables were kept constant to keep the model simple. In addition, the training and testing of the model in virtual environment renders it not generalizable for complex road or paths.

The aim of the work has been to understand the need of customization of complex traditional deep neural networks and then customize them according to the scale and complexity of the task or operation and the need to continuously update the model's architecture to make it simple and scalable. Such type of customization, optimizations or fine tunings can be done exhaustively to make sure desired results are achieved. Out of those few customizations and optimizations were performed to yield acceptable performance metrics.

Limitations of data access: Due to restricted data availability, the study may have relied on a limited dataset, potentially impacting the generalizability of the findings. Expanding access to diverse and comprehensive datasets would enhance the robustness and applicability of future research in this domain.

Methodological limitations: The study acknowledges potential limitations in the methodology employed, such as assumptions made during the optimization process or constraints imposed on the customization of neural networks. Future research should explore alternative methodologies and approaches to validate and refine the findings presented in this study. Future research directions include: enhancing optimization techniques, real-world implementation and validation, cost efficiency analysis, and expanding application domain.

ACKNOWLEDGMENT

This research project was funded by the Deanship of Scientific Research, Princess Nourah bint Abdulrahman University, through the Program of Research Project Funding after Publication, Grant No. (43-PRFA-P-4).

REFERENCES

[1] Kocić, J., Jovičić, N., Drndarević, V. (2019). An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms. Sensors, 19(9): 2064. https://doi.org/10.3390/s19092064
[2] Timmis, I., Paul, N., Chung, C.J. (2021). Teaching vehicles to steer themselves with deep learning. In 2021 IEEE International Conference on Electro Information Technology (EIT), Mt. Pleasant, MI, USA, pp. 419-421. https://doi.org/10.1109/EIT51626.2021.9491894
[3] Raj, M., Narendra, V.G. (2021). Deep neural network approach for navigation of autonomous vehicles. In 2021 6th International Conference for Convergence in Technology (I2CT), Maharashtra, India, pp. 1-4. https://doi.org/10.1109/I2CT51068.2021.9418189
[4] Dangskul, W., Phattaravatin, K., Rattanaporn, K., Kidjaidure, Y. (2021). Real-time control using convolution neural network for self-driving cars. In 2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST), Pattaya, Thailand, pp. 125-128. https://doi.org/10.1109/ICEAST52143.2021.9426255
[5] Do, T.D., Duong, M.T., Dang, Q.V., Le, M.H. (2018).

Real-time self-driving car navigation using deep neural network. In 2018 4th International Conference on Green Technology and Sustainable Development (GTSD), Ho Chi Minh City, Vietnam, pp. 7-12. https://doi.org/10.1109/GTSD.2018.8595590

[6] Ni, J., Chen, Y., Chen, Y., Zhu, J., Ali, D., Cao, W. (2020). A survey on theories and applications for self-driving cars based on deep learning methods. Applied Sciences, 10(8): 2749. https://doi.org/10.3390/app10082749

[7] Liu, J. (2020). Survey of the image recognition based on deep learning network for autonomous driving car. In 2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT), Shenyang, China, pp. 1-6. https://doi.org/10.1109/ISCTT51595.2020.00007

[8] Badola, A., Nair, V.P., Lal, R.P. (2020). An analysis of regularization methods in deep neural networks. In 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, pp. 1-6. https://doi.org/10.1109/INDICON49873.2020.9342192

[9] Wu, S., Li, G., Deng, L., Liu, L., Wu, D., Xie, Y., Shi, L. (2018). L1-norm batch normalization for efficient training of deep neural networks. IEEE Transactions on Neural Networks and Learning Systems, 30(7): 2043-2051. https://doi.org/10.1109/TNNLS.2018.2876179

[10] Neary, P. (2018). Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning. In 2018 IEEE International Conference on Cognitive Computing (ICCC), San Francisco, CA, USA, pp. 73-77. https://doi.org/10.1109/ICCC.2018.00017

[11] Curiel-Ramirez, L.A., Ramirez-Mendoza, R.A., Bautista-Montesano, R., Bustamante-Bello, M.R., Gonzalez-Hernandez, H.G., Reyes-Avedaño, J.A., Gallardo-Medina, E.C. (2020). End-to-end automated guided modular vehicle. Applied Sciences, 10(12): 4400. https://doi.org/10.3390/app10124400

[12] Fridman, L., Ding, L., Jenik, B., Reimer, B. (2019). Arguing machines: Human supervision of black box AI systems that make life-critical decisions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 1335-1343. http://dx.doi.org/10.1109/CVPRW.2019.00173

[13] Fridman, L., Brown, D.E., Kindelsberger, J., Angell, L., Mehler, B., Reimer, B. (2019). Human side of tesla autopilot: Exploration of functional vigilance in real-world human-machine collaboration. https://teslamotorsclub.com/tmc/attachments/tesla-autopilot-human-side-pdf.394047/.

[14] Olayiwola, J.O., Badejo, J.A., Okokpujie, K., Awomoyi, M.E. (2023). Lung-related diseases classification using deep convolutional neural network. Mathematical Modelling of Engineering Problems, 10(4): 1097-1104. https://doi.org/10.18280/mmep.100401

[15] Youssef, A.M. (2018). Operations of electric vehicle traction system. Mathematical Modelling of Engineering Problems, 5(2): 51-57. https://doi.org/10.18280/mmep.050201

[16] Kotapati, G., Selvamani, P.K.D., Lella, K.K., Shaik, K.S., Katevarapu, V.R., Bommagani, N.J. (2023). Deep learning based optimization model for energy consumption of new electric vehicles. Revue d'Intelligence Artificielle, 37(4): 825-834. https://doi.org/10.18280/ria.370402