Al-Khwarizmi Engineering Journal, Vol. 18, No. 4, December, (2022)
P. P. 45- 59

Al-Khwarizmi Engineering Journal

Comparative Transfer Learning Models for End-to-End Self-Driving Car

Yahya Ghufran Khidhir* **Ameer Hussein Morad****

**Department of Mechatronics Engineering/ Al-Khwarizmi College of Engineering/ University of Baghdad*
*** Department of Information and Communication Engineering / Al-Khwarizmi College of Engineering/ University of Baghdad*
*Email: yahia.ghofran1202a@kecbu.uobaghdad.edu.iq
**Email: ameer@kecbu.uobaghdad.edu.iq

(Received 1 August 2022; Accepted 19 September 2022)
https://doi.org/10.22153/kej.2022.09.003

Abstract

Self-driving automobiles are prominent in science and technology, which affect social and economic development. Deep learning (DL) is the most common area of study in artificial intelligence (AI). In recent years, deep learning-based solutions have been presented in the field of self-driving cars and have achieved outstanding results. Different studies investigated a variety of significant technologies for autonomous vehicles, including car navigation systems, path planning, environmental perception, as well as car control. End-to-end learning control directly converts sensory data into control commands in autonomous driving. This research aims to identify the most accurate pre-trained Deep Neural Network (DNN) for predicting the steering angle of a self-driving vehicle that is suitable to be applied to embedded automotive technologies with limited performance. Three well-known pre-trained models were compared in this study: AlexNet, ResNet18, and DenseNet121.

Transfer learning was utilized by modifying the final layer of pre-trained models in order to predict the steering angle of the vehicle. Experiments were conducted on the dataset collected from two different tracks. According to the study's findings, ResNet18 and DenseNet121 have the lowest error percentage for steering angle values. Furthermore, the performance of the modified models was evaluated on predetermined tracks. ResNet18 outperformed DenseNet121 in terms of accuracy, with less deviation from the center of the track, while DenseNet121 demonstrated greater adaptability across multiple tracks, resulting in better performance consistency.

Keywords: *Autonomous driving, Deep learning, End-to-end learning, Self-driving car, Embedded system.*

1. Introduction

Self-driving cars have emerged as one of the most actively contested and researched topics in recent years. Despite their association with the automotive industry, these systems belong to the field of robotics as part of the third robotic revolution. [1].

From a philosophical and scientific standpoint, there are various predictions regarding how self-driving automobiles will impact our lives. Automobiles play an important part in modern civilization, which is defined by a high level of mobility. It is estimated that approximately one billion automobiles are on the roads worldwide. Autonomous driving, or intelligent driver assistance, is expected to result in a 90% reduction in road accidents; a 60% reduction in carbon dioxide emissions (due to smart trajectory planning); and savings of more than 1 billion hours every day for commuters around the world [2][3]. Autonomous vehicles will not only improve the

mobility of those who are elderly or disabled; but will also reduce transportation energy consumption by up to 90%. [4].

Self-driving cars are autonomous decision-making systems that analyze streams of observations obtained from multiple onboard sources, such as cameras, Radio Detection and Ranging sensors, Light Detection and Ranging sensors, ultrasonic sensors, Global Positioning System units, and/or inertial sensors, in which there is no human driver [5]. The camera is the most essential component of a vehicle's suite of sensors. The vast amount of data collected by a vehicle's cameras necessitates significant processing power in order to extract useful information. The only method for extracting information from incoming images is computer vision.

Sequential pipelines (perception-planning-action) or end-to-end learning are used to make driving decisions. Pipelines that follow the classic approach, in which first objects are recognized in the input image, then a path is planned, and lastly, the computed control values are performed. In contrast, the end-to-end method employs a unique theory to steer a vehicle, Figure 1. This strategy focuses on developing an AI model that mimics human driving. A human driver constantly makes steering predictions based on an immediate analysis of observed visuals. [6].

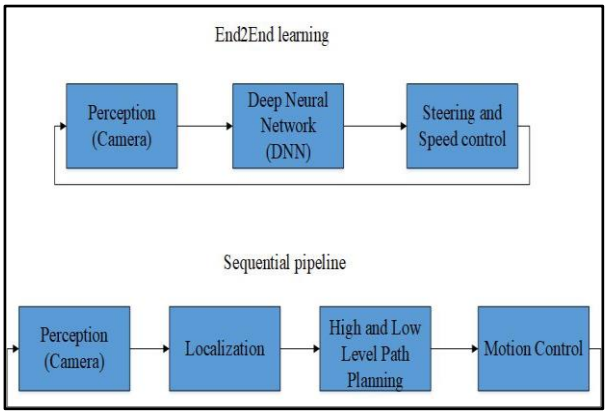


Fig. 1. Deep Learning-based self-driving car.

End-to-end Learning Control, as defined in the implementation of autonomous driving, where an image provided by a front-facing camera is fed into a neural network that then generates the car's control signals such as steering angle, throttle, and braking [5]. The primary advantage of this technique is that it learns the entire function automatically, from raw pixels to steering angles, without the need for any human intervention in the form of feature design and selection, geometry, camera calibration, or manual adjustment of parameter values. To be more specific, the goal of end-to-end learning is to diminish or eliminate the need for human expertise in terms of feature extraction and to solve issues using just data [7].

In this paper, three well-known DL models have been modified to perform the task of end-to-end driving autonomously. The main objective of this study is to analyze and evaluate the modified models and determine the best predictive model among them for steering angle control using a dataset collected from two different tracks. In addition, the study decides which model is most suitable to be applied to low-performing hardware with high qualitative results. The DL models have been selected according to their widespread use.

Results from the offered end-to-end learning network may find practical usage in applications like warehousing and delivery vehicle robot cars. It is important for practical industrial applications to be able to deploy well-known DNN solutions, such as those offered in this study, on integrated vehicle applications that have low-power technologies at low costs and compact sizes.

The relevant work will be presented in the next section. Section III describes the research method. Section IV: Experiments and Results. The last portion contains the conclusion.

2. Related Work

This section summarizes some of the related works proposed to perform the task of autonomous end-to-end driving. End-to-end learning models that are trained on sensor inputs and supervised by humans are one of the most promising approaches to achieving fully autonomous performance. A limited number of published works have used DNNs as part of a complete model to command a real-world autonomous system. Researchers in the field of autonomous vehicles rely heavily on artificial datasets to evaluate the efficacy of DL.

In the paper [8], the authors describe a Convolutional Neural Network (CNN) that can solve the problem of autonomous lateral control. The input image size to the network is (70x160x3). A proper steering angle was generated using CNN, enabling the automobile to complete laps. On simulated unknown tracks, CNN's steering control was tested. Unfamiliar single-lane routes were successfully navigated 89.02 % of the time. However, the training dataset was small and did not cover a wide range of scenarios.

In [9], the researchers designed DL models for the input image size of (70x160x3) in order to carry out longitudinal and lateral vehicle control. Predictions of both speed and steering angle were made by developing two separate models. Max-pooling was employed in each convolution in DNN. The dual-action model completed all loops of the simulation track without leaving the designated lanes, demonstrating full autonomy. However, there was insufficient memory for training due to the excessive use of threads.

In addition, another group of researchers [10] developed a lightweight DNN model (J-Net) for car steering angle control and compared this model to two other models, AlexNet and Nvidia's model. Input images to the first convolutional layer of the proposed models have a size of (65x320x3). Successful autonomous driving was measured, inferred, and analyzed using a simulator environment. In comparison to the other two approaches, J-Net had the best latency and frame rate. The J-Net lite model, however, has not been implemented on an embedded platform with a limited number of processing cores.

Applying DL techniques including Transfer Learning, [11] study compared two different models for accurate steering angle prediction. The models were a 3D convolutional model with Long Short-Term Memory layers and Residual Neural Network (ResNet50) with the input image sizes (120x320x3) and (224x224x3), respectively. The models were trained using the Udacity self-driving car dataset. The (ResNet50) model outperformed the other model with an RMSE of 0.0709. Nonetheless, the paper's models benefit from limited data augmentation, which is a significant limitation. The topic of steering angle regression is framed as a classification challenge with an imposed spatial connection between neurons in the output layer.

Another study [12] presented a comparison between the three transfer learning models: Visual Geometry Group (VGG16), ResNet-152, Densely Connected Convolutional Networks (DenseNet-201, and Nvidia’s model for steering angle control. The input image size for all models was (66x200x3). The models were trained with 25 minutes of driving time at 30 frames per second using a Raspberry Pi camera. Nvidia’s model outperformed the other models with a Mean Square Error (MSE) of 0.3521. The authors used random videos from YouTube to visualize the performance of Nvidia’s model.

Using a monocular self-driving car prototype, the authors of [13] proposed a DNN applied to a Raspberry Pi with an input image size of (120x160x3) for steering angle prediction. As a vehicle platform, a 1/10 scale RC car was employed. Data was acquired from an oval and an 8-shaped track equipped with traffic signs. Experiments demonstrate that the autopilot product is capable of maintaining a lane with a Cross-Entropy Cost Function accuracy of 89.04%. Regardless of lane markers, the vehicle's top speed is 5–6 km/h at 10 frames per second. Nevertheless, the camera used in their car prototype lagged by 300-350 milliseconds.

Multichannel Convolutional Neural Networks (M-CNNs) were utilized for vehicle speed and steering angle calculations in [14]. The model uses (227x227x3) image size from front-view camera recordings and feedback speed sequences as its input. The public Udacity dataset and the gathered SAIC dataset are used to evaluate the suggested technique. The proposed concept was contrasted with the updated versions of the Cg Network and PilotNet from Nvidia. With an MAE of 1.26 degrees for steering angle and 0.19 meters per second for speed values, the M-CNNs were better than the implemented Nvidia PilotNet and the Cg Network.

Numerous studies have focused on constructing DNNs from low-resolution images for deployment throughout low-performance embedded systems. This research aims to develop an end-to-end method for forecasting the steering angle using well-known DNNs that can be implemented on low-performance embedded systems. In addition, provide an overview of well-known DL models, such as AlexNet, ResNet18, and DenseNet-121, that accurately predict autonomous driving commands but have not previously been studied. To train the selected models, data were collected from two distinct tracks with image sizes of (224x224x3), which is the most frequent resolution. Furthermore, the presented work's accuracy and consistency were examined and compared. The following are the paper's main contributions:

1. Image classification convolutional neural networks AlexNet, ResNet18, and DenseNet121 were modified and deployed for end-to-end autonomous driving.
2. The Jetson Nano 2GB, a low-performance embedded system, was applied to evaluate the modified models. Consequently, it proved that ResNet18 and DenseNet121 are well-suited to be employed in low-performance embedded devices.
3. The majority of data was collected from the first track, while some information was obtained

from the second. The results demonstrate that the robot can drive 89% of the time on an untrained track.

4. The performance of ResNet18 and DenseNet121 was evaluated on two different tracks to demonstrate their suitability for autonomous driving. ResNet18 outperformed DenseNet121 in terms of precision, with less deviation from the center of the track. DenseNet121 was more adaptable across tracks, resulting in better consistency.

3. Method

This section explains the algorithms used in this work in considerable detail followed by the robot system architecture.

3.1. Convolutional Neural Networks (CNN)

CNN is utilized to classify the labels using a supervised learning technique [15]. The first phase in the CNN architecture is the convolution layer, where a 2D convolution operation is applied to the image. In convolution, the three-channel RGB with a 2D size image is convolved with kernels. The output of convolution known as Feature Map focuses on particular features in the input image regardless of their location in the image. The convolutions with kernels are used to extract low-level particular features (like edges and corners of the road) from the previous layers and combine them with higher-level extracted features (full objects of the road) in the next hidden layers. which reduce the size of the image and increase the number of parameters (weights and biases) [16]. As the number of parameters increases, more training time is required. The activation function is applied after the convolution.

The activation function is responsible for determining whether or not a neuron is fired. In other words, it will use simpler mathematical operations to decide whether or not the neuron's input is essential to the network. Calculating a weighted sum and then adding bias to it is how the activation function determines whether or not a neuron should be activated or not[17]. The activation function adds nonlinearity to a neuron's output [18]. It transforms the input, allowing it to learn and execute highly complicated tasks. In this work, the (rectified linear unit) [19] activation function is applied to the hidden layers because it requires fewer mathematical operations and thus is faster to compute compared to tanh and sigmoid and neglects the effect of gradient vanishing. If x is positive, it returns x and 0 if x is negative or equal to zero.

The operation called pooling was next performed, which decreases the feature maps' dimensionality (resolution) and hence their output (the output of the convolutional layer) resolution [20][21]. This method helps to reduce spatial variance by removing unnecessary information from the neuron's receptive field by taking the maximum (max pooling) or average (average pooling) value of the neuron (filter window). Pooling not only helps in object recognition in images independent of their position but also reduces the number of parameters to train by reducing the number of pixels, thereby contributing to the prevention of overfitting.

The two-dimensional feature vector of the final convolution layer is flattened into a long one-dimensional array, which is known as a fully connected layer FC, or dense layer. A subset of FC layers maps the spatial features of convolution layers and pooling layers to the final outputs of the network, which directly predicts the steering angle value.

The steering angle of each image is represented by a single point on the image. In this study, the dataset consisting of road images and steering angle labels is used for training purposes. The trained model predicts the steering angle in real-time while detecting unlabeled images, as regression is the process of forecasting a value based on previous data. Adam is the optimizer implemented by CNNs in the current study. Optimizers modify neural network parameters including weights and learning rates.

The accuracy of the model is computed employing loss function. As described in Section 2.5, this study suggests utilizing MSE loss function to train a regression network for the task of predicting continuous steering angles from models. The (MSE) illustrates how far away from reality predictions were.

3.2. CNN Models

The three CNN architectures used in this work are ResNet18, DensNet121, and AlexNet. All three models take a 224* 224 image size as input. Residual learning was used in ResNet18 to overcome the loss of precision with increasing network depth. It has 18 layers, including 17 convolutional layers and one FC layer with 3x3 filters. This network also has approximately 11 million parameters. Down sampling is performed

by convolutional layers with a stride of two, as shown in Figure 2[22]. The final FC layer and Softmax were removed from the original network, and an additional FC layer was added to convert 512 features to one output feature (steering angle). DenseNet was developed to address the problem of vanishing gradients in high-level neural networks [23], which causes accuracy to decrease. The network has 121 layers, which are divided into four dense blocks made up of convolutional layers and transition layers in between to facilitate information flow between layers, as shown in Figure 3. DensNet121 is made up of 7.2 million parameters. The original network's final FC layer and Softmax were removed, and an additional FC layer was created to convert 1024 features to one output feature (steering angle).

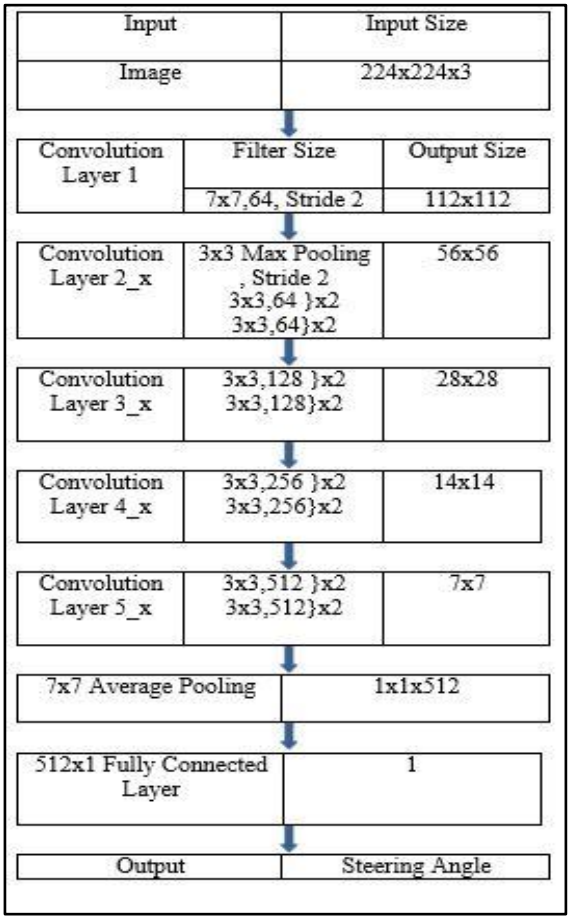


Fig. 2. ResNet18 architecture.

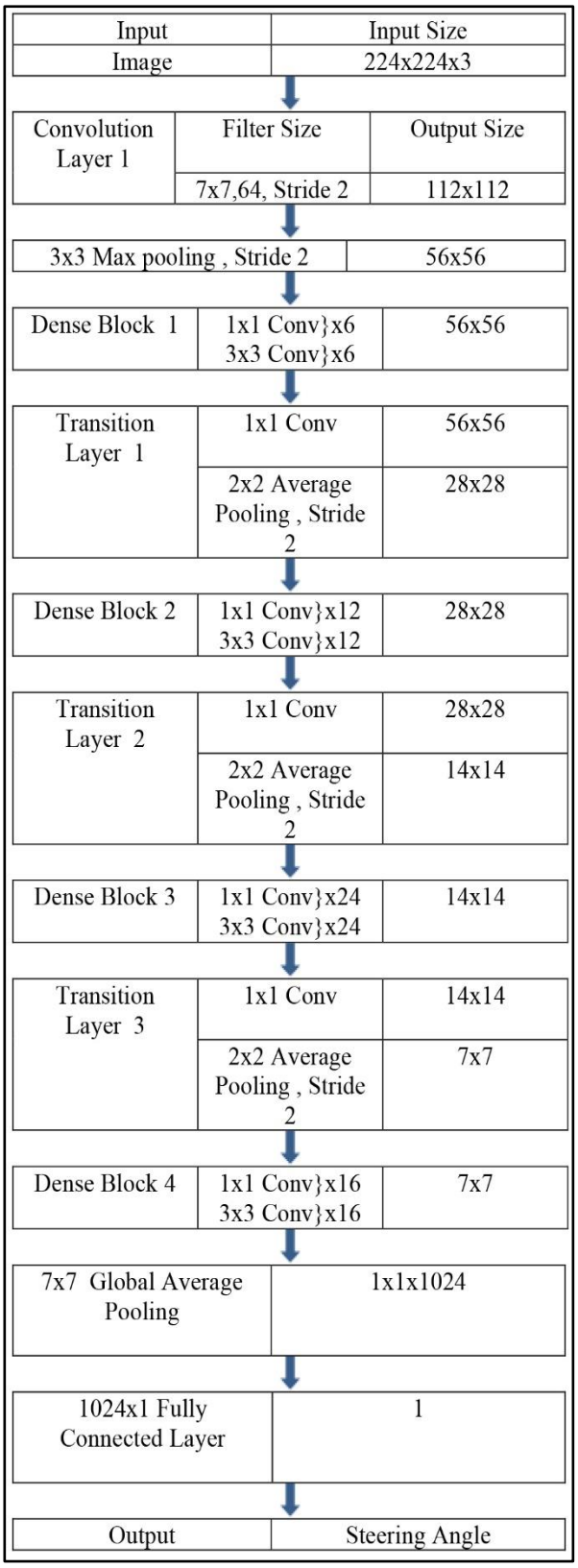


Fig. 3. DenseNet121 architecture.

3.3. Robotic System Architecture

The block diagram of training DNN is shown in Figure 4. Firstly, the required data was collected manually by driving the car. Secondly, data pre-processing and data augmentation were applied before feeding the images to the CNN models for training. Finally, the trained model was used in the inference track for the test. The successful model was one that drove independently on both tracks while avoiding the orange bolder lines and staying on the centreline.

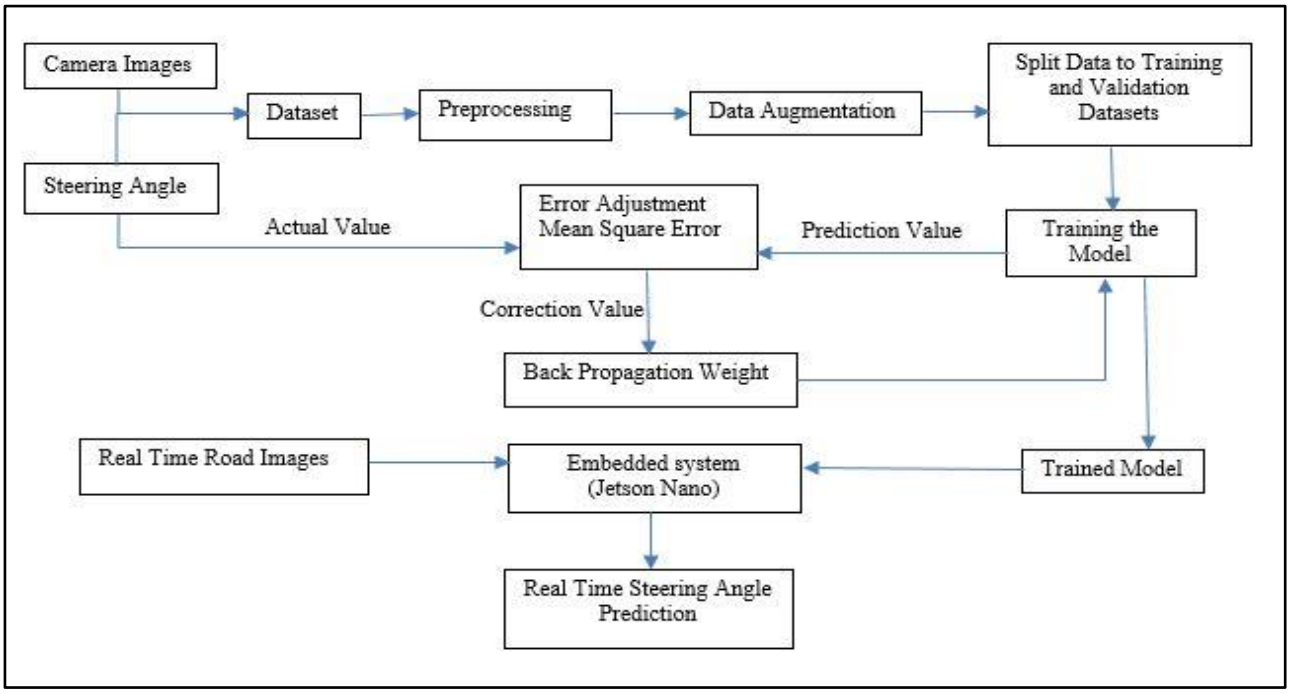


Fig. 4. Block diagram of the training DNN.

3.4. Database Acquisition

Collecting, inferring, and evaluating data for autonomous driving was done on two tracks. Data were collected along the first, 3x2 meter path, while tests and evaluations were conducted along the second, 4x2 meter path with curves in the middle, Figure 5(B). The inference track was brighter than the collection track.

The dataset's features are represented by images, with the corresponding steering angles serving as labels. The Jetracer robot was placed on the representative track to collect data for the training process. Continuous orange lines on the borders and dashed white lines in the center define the track. A single camera was installed instead of three cameras to capture frames in three different positions (center, left, and right). The objective was to keep the car in the center of the road. As a result, the images from the camera at the same level in the middle, left, and right were taken with the same steering measurement value. The measurements are manually entered from a remote PC while the images are being captured by clicking the desired XY coordinates on each frame as shown in Figure 6. The X-axis latitude represents the amount of angle from left to right and is controlled by the servo motor with a range of -1.0 to 1.0. As a result, a unique dataset is created for the training process. To obtain high-quality results, the images and measurements should be collected manually while driving in the manner we expect during testing. Data augmentation has been used to improve training results as well as drive the robot clockwise and counterclockwise[24][25]. The term "augmentation" refers to the technique of improving the training data without affecting the dataset's essential characteristics. The amount of data in the training directory is increased by adding slightly modified copies of pre-existing data. The augmentation procedure in our work involves only flipping the images around vertical access. The steering angle should be inverted while flipping each frame. Therefore, the new steering angle equals:

$$Steering\ angle\ after\ flipping = (-1) * original\ steering\ angle \quad \ldots (1)$$

So as to adapt the robot's throttle to its high speed in a straightforward path and slower speed in turns, the throttle is equal to:

$$Y = 1 - abs(X) \quad \ldots (2)$$

As a result, when the car is driving in the center of the path, x = 0, the throttle is at its maximum value, and in sharp turns, the throttle is at its minimum value.

(A) **(B)**

Fig. 5. Database Acquisition tracks
(A) Data collection track (B). Inference track

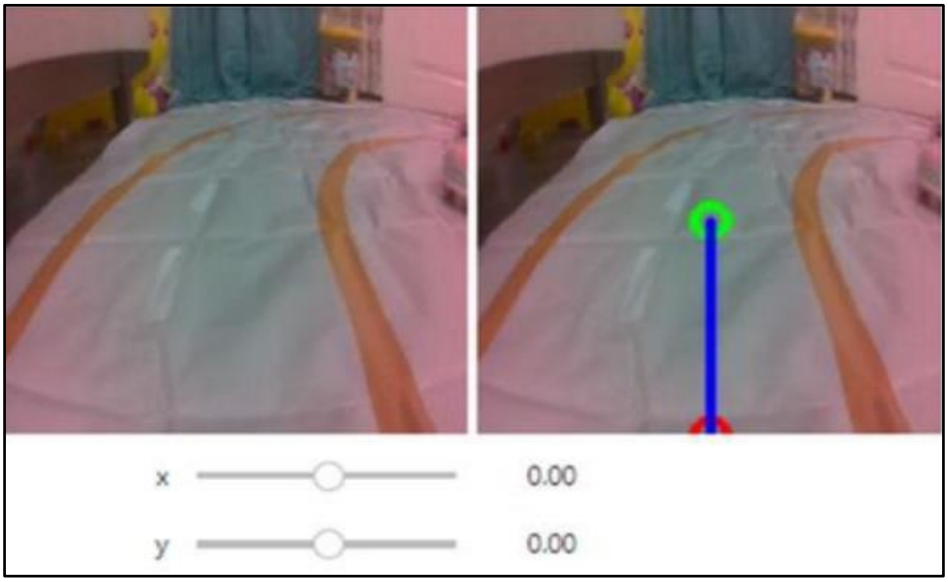


Fig. 6. Database Acquisition.

The number of images collected from the first track in Figure 5(A) was 681 in total. 83 samples from track b's inside turns were added to the dataset to enable the robot to drive on both tracks, Figure 5(B). The images' resolution was (224x224x3). (224 pixels width, 224 pixels height, and 3 channels of RGB colors). The number of images increased to 1528 samples after data augmentation. The dataset was divided into two portions: 80% for training and 20% for validation. Figure 7 depicts the normalized steering angles associated with each frame.

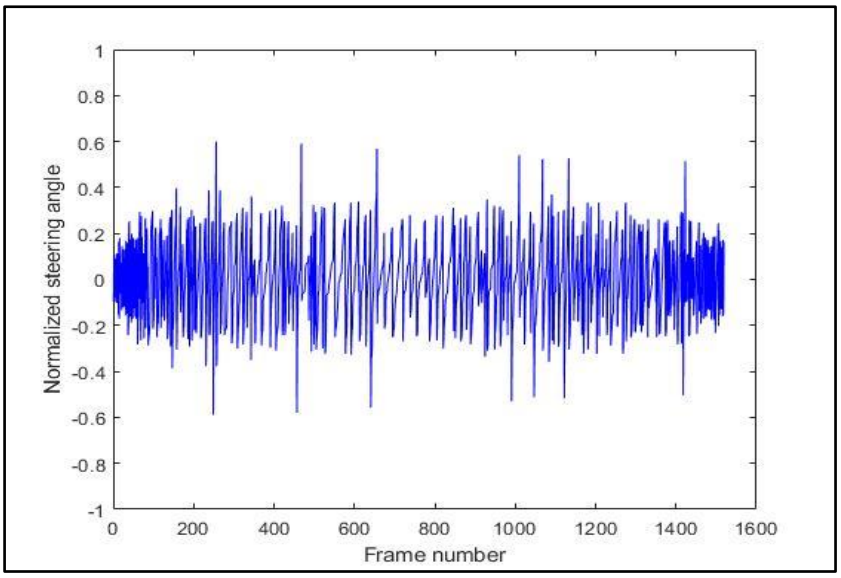


Fig. 7. Dataset of Normalized Steering Angles.

3.5. Training and Hyper-Parameter

On the Google Colab platform, the three models were trained in the Python programming language. Color jitter data pre-processing was used in addition to data normalization because of the difference in color brightness between data collection and inference tracks. Each model was trained separately with the same hyperparameters. The Adam [26] optimizer utilized with an initial learning rate of (0.0005). To avoid missing local optima, it's recommended to keep the learning rate low because it's more stable. In spite of the above fact, it is possible that a minimal learning rate could lead to overfitting. The number of epochs was 70, with early stopping patience of 10 epochs. The batch size was set to 8. As the batch size is reduced, the end losses are reduced too. As a suitable loss function for regression networks, (MSE) [27] was selected to minimize the error between the steering forecast and the steering measurements. MSE is calculated by taking the average of the squared difference between the expected value and the actual value.

$$MeanSquaredError = \frac{1}{N}\sum_{j=1}^{N}\left(y_j - \hat{y}_j\right)^2 \quad \ldots (3)$$

The vector $\hat{y}_j$ represents the values of N forecasts and y_j is a vector that contains N true values.

In order to get fast inference performance with a simple workflow, the models were converted and optimized using the PyTorch to TensorRT converter (torch2trt) to be applied to the Jetson Nano embedded system. All three models were trained with the same hyper-parameters and datasets, differing only in the number of epochs. These differences arose from early stopping to reduce the overfitting. The model was saved automatically if there was no progress in validation loss for 10 epochs.

The performance of the models was evaluated by their ability to navigate a robot through predetermined tracks. The models were determined to be successful based on their ability to drive autonomously on both predetermined tracks. As such, the model was determined to be a failure if the car deviated from the path of the predetermined tracks. The robots were driven through each track three times. Their paths were recorded by different colored markers that were affixed to the back of the robot.

Images were captured from the paths and the trajectories were converted into XY coordinates in pixels. The Frechet distance theorem was used to measure how different the drawn paths were from each other in relation to the centre of the track.[28][29]

The Frechet distance between two curves can be defined mathematically as:

$$\delta_F(P,Q) = \min_{\substack{\alpha[0,1]\to[0,\mathrm{N}]\; t\in[0,1] \\ \beta[0,1]\to[0,\mathrm{M}]}} \left\{\max \; d\left(P(\alpha(t)), Q(\beta(t))\right)\right\} \quad \ldots(4)$$

Where P and Q are polygonal curves of length N and M, $\alpha(t)$ and $\beta(t)$ are two continuous and increasing functions, and $d\left(P(\alpha(t)), Q(\beta(t))\right)$ is the Euclidian distance between $P(\alpha(t))$ and $Q(\beta(t))$.

According to equation (4), the Frechet distance algorithm first determines the maximum distance that can exist between curves P and Q as they progress along their respective paths for each value of the functions $\alpha(t)$ and $\beta(t)$, then maintains the minimum distance that can exist between them.

4. Experiments and Results

When the findings of this research were compared to those of similar studies, it was revealed that the metrics and datasets utilized were not standardized, and there was a lack of transparency in the presentation of the results. For instance, the percentage of switching lanes over time on multi-line tracks is quantified as a "degree of autonomy" in [8], [9]. The sample sizes in the training, validation, and testing sets in the simulated data presented in [8], [9], and [10] are significantly different. In studies, the following metrics were employed: [10], [12] MSE, [13] Cross-Entropy, [14] MAE, and [11] RMSE. MSE was implemented to evaluate the performance of the proposed architectures on the validation dataset in this study. As shown in Table 1, DesneNet121 has the best overall performance, with a score of 0.0057. However, ResNet18's value of 0.0058 is quite close to DenseNet121's value. Among the two remaining models, ResNet18 and DenseNet121, ResNet18 had the larger file size. The size of the model indicates the complexity of the model. After training three CNN architectures on Google Colab, testing for the AlexNet model was suspended since its 217 MB size exceeded the storage capacity of the Jetson Nano 2G. As a result, this model was not considered for further examination. Only ResNet18 and DenseNet121 were tested on the aforementioned tracks using the Jetson Nano 2G.

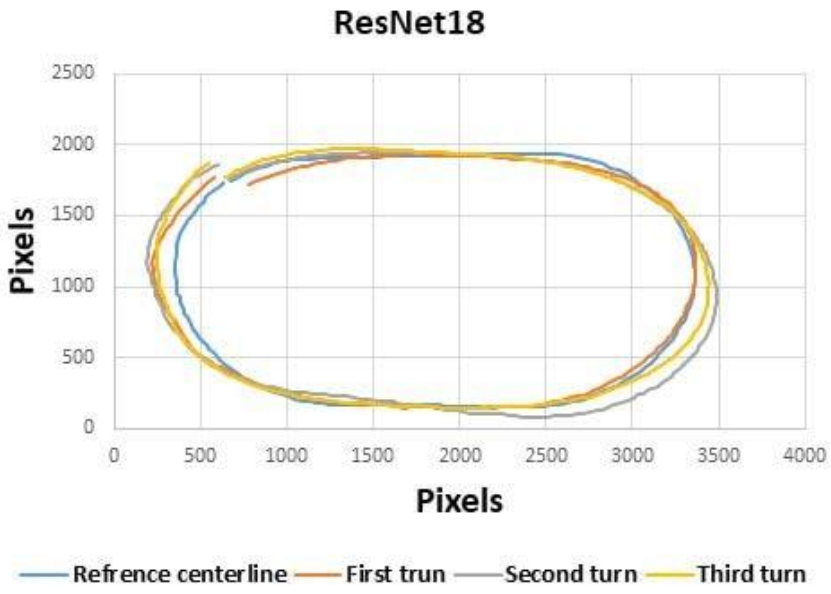
Table 1,
MSE and Model Size

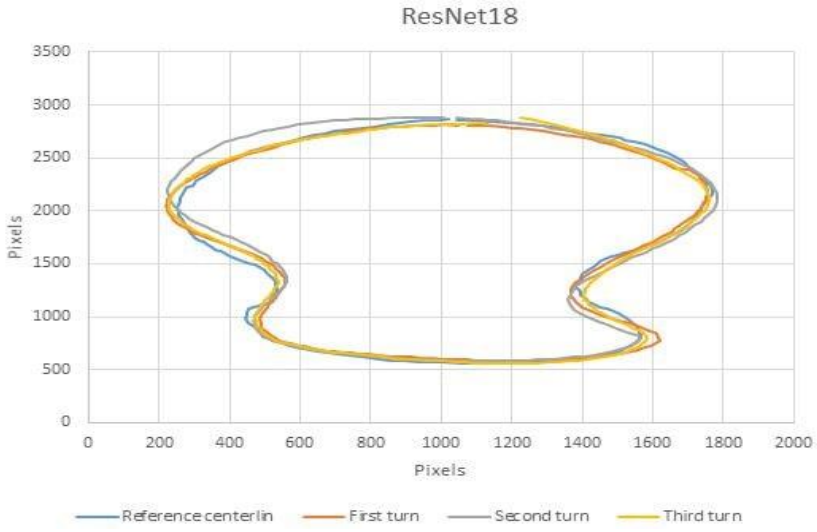
Network	MSE	Model Size
ResNet18	0.0058	42.7 MB
DenseNet121	0.0057	27.1 MB
AlexNet	0.1333	217 MB

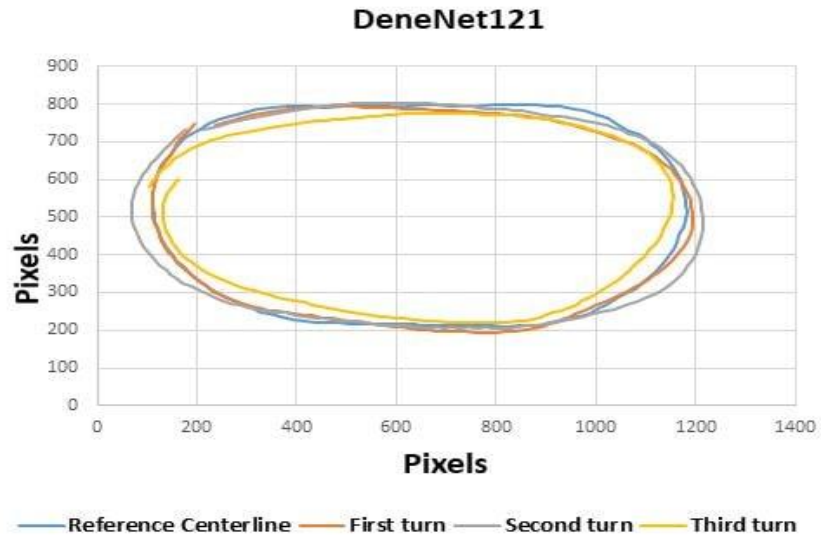
In order to compare the models' accuracy and repeatability. Due to the simplicity, three full laps of driving were recorded at a constant speed. Figure 8 shows the drawn paths of the robot as it navigates each predetermined path, whether using the ResNet18 model or the DenseNet121. Recorded trajectories suggest that both ResNet18 and DenseNet121 guided the robot through both of the predetermined paths. However, there were some differences between the two models. The Frechet distance was applied to show how both models' curves deviated from the centerline. In the initial step of the Frechet distance algorithm, it calculates the maximum distance between two pairing points of each model's drawn path and the centerline path. In addition to this, it selects the smallest distance from all maximum pairwise distances to demonstrate the dissimilarity between the two paths. Figures 8 (a) and (b) show the paths drawn by the ResNet18 model for three full laps on both trajectories. Figures 8 (c) and (d) show the paths drawn by DenseNet121 for three full laps on both tracks.

Table 2 shows the calculated dissimilarity distances between each curve and the centerline for three turns. Table 2 shows that when comparing ResNet18 and DenseNet121, the former has relatively small dissimilarity distances.

ResNet18 demonstrated an average distance of 3.26 cm from the track's centerline on the first track. On the same track, DenseNet121 achieved an average distance of 4.39 cm. ResNet18 achieved an average distance of 4.12 cm on the second track. DenseNet121 achieved an average distance of 4.38 cm on the same track. ResNet18 appeared to be less accurate on the second track when compared to the first. DenseNet121, on the other hand, produced comparable results on both tracks. This means that ResNet18 is more accurate than DenseNet121, but DenseNet121 can adapt to different tracks and perform more consistently.


ResNet18
Pixels
Pixels
Refrence centerline
First trun
Second turn
Third turn


ResNet18
Pixels
Pixels
Reference centerlin
First turn
Second turn
Third turn


DeneNet121
Pixels
Pixels
Reference Centerline
First turn
Second turn
Third turn

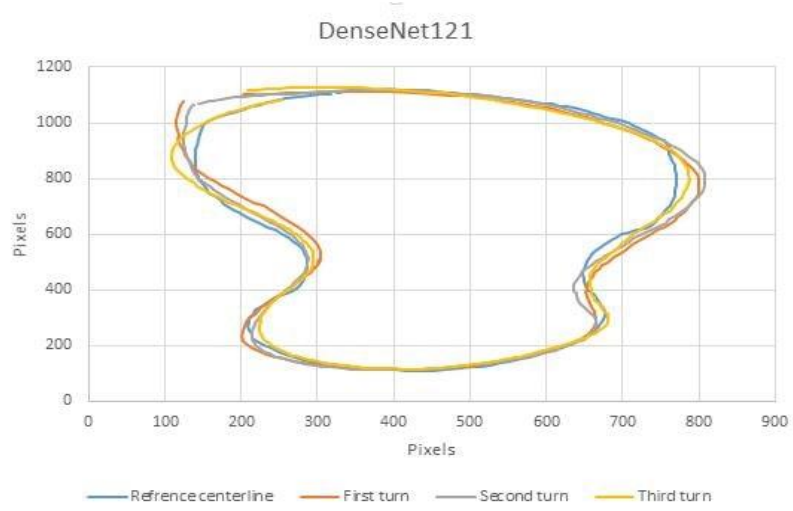


Fig. 8. The drawn paths for three full laps on both tracks using ResNet18 and DenseNet121. (a) The drawn paths of ResNet18 on the first track. (b) The drawn paths ResNet18 model on the second track. (c) The drawn paths of the DenseNet121 Model on the first track. (d) The drawn paths DenseNet121 model on the second track.

Table 2,
The dissimilarity between curves and centerline using discrete Frechet distance.

	First track		**Second track**	
Num of turns	ResNet18	DenseNet121	ResNet18	DenseNet121
First turn	1.81 cm	2.69 cm	3.57 cm	3.12 cm
Second turn	3.21 cm	4.76 cm	4.39 cm	4.89 cm
Third turn	4.77 cm	5.731 cm	4.41 cm	5.13 cm

Figure 9 shows the predicted steering with frames for each model as a performance measure. Figure 9 also depicts three types of driving paths: (a) internal turns, (b) straightforward paths, and (c) long-end turns. The two large oscillations depict the long end turns with high deviation from the track center as the car drives counterclockwise. With less oscillation, the straightforward path is depicted. The small upside-down oscillations represent short internal turns in the middle of the path. Furthermore, as shown in Figure 9, both DenseNet121 and ResNet18 have higher steering angle deviation compared to the collected dataset in Figure 7. Based on the trained dataset, Resent18 outperforms than DenseNet121, as evidenced by its higher steering angle of 0.8 (i.e., closer to the center of the track). The steering angle of DenseNet121 was approximately 0.98, indicating that it was quite distant from the track's center.

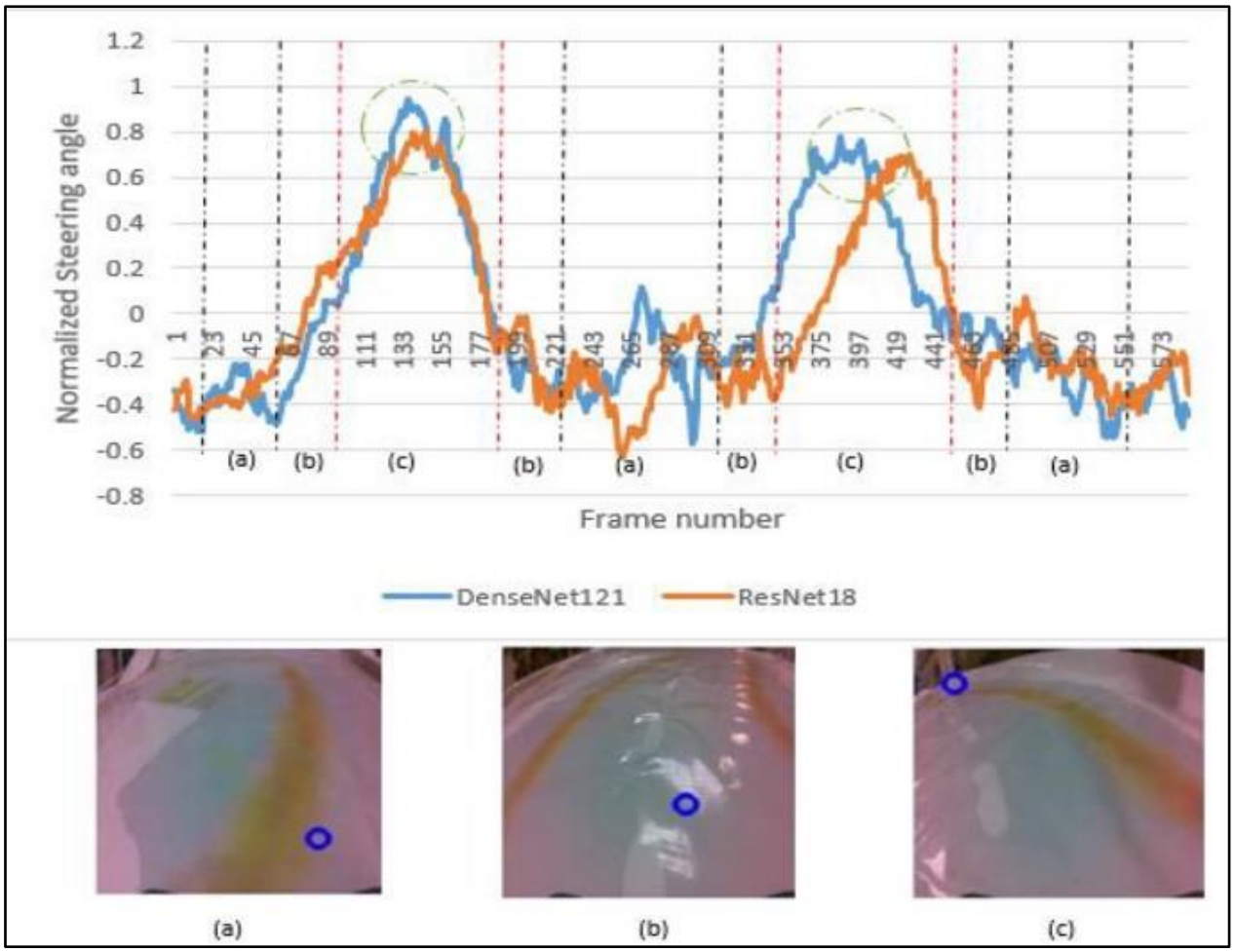


Fig. 9. Relative deviation from the center of the trajectory per one full lap of autonomous driving using ResNet18 and DenseNet121.

(a) Internal turns (b) Straightforward path (c) Long-end turns.

5. Conclusion

In this study, a comparison of three transfer learning models (AlexNet, ResNet18, and DenseNet121), which are capable of handling the task of autonomous driving, was conducted. In addition, the modifications of the pertained models' designs and the system architecture for the prediction of steering angle have been described. Three architectures were applied and evaluated on two featured paths. Through a comparative analysis of three well-known DL models. The main goal of this study is to determine the best DL model for predicting steering angle. The numerical results based on the loss function (MSE) and the Frechet distance were discussed. The main idea of this research is to determine whether the better result for the steering angle prediction problem can be obtained by employing less sophisticated algorithms that can be implemented on low-performance hardware. The results show that both ResNet18 and DenseNet121 are compatible with the Jeston Nano 2G embedded system. The results demonstrate that the ResNet18 is more accuracy than DenseNet121, whilst DenseNet121 are able to adapt new tracks and keep the performance stable over time. On the other hand, Resent18's is achieved a good fitted line with the trained dataset, as indicated by its higher steering angle of 0.8 (i.e., closer to the center of the track). DenseNet121's steering angle was about 0.98, which is an extremely wide divergence from the track's center. In addition, different experiments were carried out in order to investigate how the training dataset affected driving performance. The results demonstrate that the best driving performance was obtained when the validation loss was less than 1% and the vehicle was capable of handling abrupt curves while remaining on the centerline. The biggest obstacle in the end-to-end DL technique is its reliance on clean and extensive training data.

The future idea is to add an IMU sensor to measure the robot's actual orientations along the path in order to use them to find the difference between the neural network's expected steering angle and the IMU sensor's output.

6. References

[1] D. B. O. Boesl and B. Liepert, "4 Robotic revolutions - Proposing a holistic phase model describing future disruptions in the evolution of robotics and automation and the rise of a new Generation 'R' of Robotic Natives," IEEE Int. Conf. Intell. Robot. Syst., vol. 2016-Novem, pp. 1262–1267, 2016.

[2] Anderson, J., Kalra, N., Stanley, K., Sorensen, P., Samaras, C. and Oluwatola, T., 2016. Autonomous Vehicle Technology: A Guide for Policymakers. [online] Rand.org. Available at: <https://www.rand.org/pubs/research_reports/RR443-2.html> [Accessed 9 September 2022].

[3] S. Singh, "Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey," Natl. High. Traffic Saf. Adm., no. March, pp. 1–3, 2018, [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812506.

[4] A. Brown, B. Repac, and J. Gonder, "Autonomous Vehicles Have a Wide Range of Possible Energy Impacts," Work. Road Veh. Autom., no. 2012, p. 59210, 2013.

[5] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," J. F. Robot., vol. 37, no. 3, pp. 362–386, 2020,

[6] Y. Wang, D. Liu, H. Jeon, Z. Chu, and E. T. Matson, "End-to-end learning approach for autonomous driving: A convolutional neural network model," ICAART 2019 - Proc. 11th Int. Conf. Agents Artif. Intell., vol. 2, no. Icaart, pp. 833–839, 2019.

[7] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," Adv. Neural Inf. Process. Syst., pp. 739–746, 2005.

[8] S. Sharma, G. Tewolde, and J. Kwon, "Behavioral Cloning for Lateral Motion Control of Autonomous Vehicles Using Deep Learning," in 2018 IEEE International Conference on Electro/Information Technology (EIT), May 2018, pp. 0228–0233.

[9] S. Sharma, G. Tewolde, and J. Kwon, "Lateral and Longitudinal Motion Control of Autonomous Vehicles using Deep Learning," in 2019 IEEE International Conference on Electro Information Technology (EIT), May 2019, pp. 1–5,

[10] J. Kocić, N. Jovičić, and V. Drndarević, "An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms," Sensors, vol. 19, no. 9, p. 2064, May 2019,

[11] S. Du, H. Guo, and A. Simpson, "Self-Driving Car Steering Angle Prediction Based on Image Recognition." 2019, [Online]. Available: http://arxiv.org/abs/1912.05440.

[12] M. K. Islam, M. N. Yeasmin, C. Kaushal, M. Al Amin, M. R. Islam, and M. I. Hossain Showrov, "Comparative Analysis of Steering Angle Prediction for Automated Object using Deep Neural Network," in 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Sep. 2021, pp. 1–7.

[13] T. D. Do, M. T. Duong, Q. V. Dang, and M. H. Le, "Real-Time Self-Driving Car Navigation Using Deep Neural Network," Proceedings 2018 4th International Conference on Green Technology and Sustainable Development, GTSD 2018. pp. 7–12, 2018.

[14] Z. Yang, Y. Zhang, J. Yu, J. Cai, and J. Luo, "End-to-end Multi-Modal Multi-Task Vehicle Control for Self-Driving Cars with Visual Perceptions," in 2018 24th International Conference on Pattern Recognition (ICPR), Aug. 2018, pp. 2289–2294, doi: 10.1109/ICPR.2018.8546189.

[15] M. Al-Mukhtar, A. H. Morad, M. Albadri, and M. S. Islam, "Weakly Supervised Sensitive Heatmap framework to classify and localize diabetic retinopathy lesions," Sci. Rep., vol. 11, no. 1, p. 23631, Dec. 2021.

[16] M. Bojarski et al., "End to End Learning for Self-Driving Cars," pp. 1–9, 2016.

[17] A. Sangeetha and T. Rajendran, "Levenberg-Marquart logistic deep neural learning based energy efficient and load balanced routing in MANET," Indones. J. Electr. Eng. Comput. Sci., vol. 23, no. 2, pp. 1002–1010, 2021.

[18] A. Hussien Mary, Z. Bilal Kadhim, and Z. Saad Sharqi, "Face Recognition and Emotion Recognition from Facial Expression Using Deep Learning Neural Network," IOP Conf. Ser. Mater. Sci. Eng., vol. 928, no. 3, p. 32061, 2020.

[19] B. Jabir and N. Falih, "Dropout, a basic and effective regularization method for a deep

learning model: A case study," Indones. J. Electr. Eng. Comput. Sci., vol. 24, no. 2, pp. 1009–1016, 2021.
[20] T. H. Abdullah, F. Alizadeh, and B. H. Abdullah, "COVID-19 Diagnosis System using SimpNet Deep Model," Baghdad Sci. J., vol. 19, no. 5 SE-article, p. 1078, Oct. 2022.
[21] A. M. Hamad Alhussainy and A. D. Jasim, "A novel pooling layer based on Gaussian function with wavelet transform," Indones. J. Electr. Eng. Comput. Sci., vol. 20, no. 3, pp. 1289–1298, 2020.
[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conf. Comput. Vis. Pattern Recognit., pp. 770–778, Jun. 2016.
[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul. 2017, pp. 2261–2269.
[24] A. Asroni, K. R. Ku-Mahamud, C. Damarjati, and H. B. Slamat, "Arabic speech classification method based on padding and deep learning neural network," Baghdad Science Journal, vol. 18. pp. 925–936, 2021, doi: 10.21123/bsj.2021.18.2(Suppl.).0925.
[25] A. Mikolajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in 2018 International Interdisciplinary PhD Workshop (IIPhDW), May 2018, pp. 117–122.
[26] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017 2nd IEEE Int. Conf. Cloud Comput. Big Data Anal. ICCCBDA 2017, pp. 156–160, Dec. 2014. 10.1109/ICCCBDA.2017.7951902.
[27] C. Sammut and G. I. Webb, Eds., "Mean Squared Error," in Encyclopedia of Machine Learning, Boston, MA: Springer US, 2010, p. 653.
[28] A. Driemel, S. Har-Peled, and C. Wenk, "Approximating the Fréchet distance for realistic curves in near linear time," Discret. Comput. Geom., vol. 48, no. 1, pp. 94–127, 2012.
[29] S. L. Seyler, A. Kumar, M. F. Thorpe, and O. Beckstein, "Path Similarity Analysis: A Method for Quantifying Macromolecular Pathways," PLoS Comput. Biol., vol. 11, no. 10, pp. 1–36, 2015.

مقارنة نماذج التعلم النقل من النهاية الى النهاية لسيارات الذاتية القيادة

يحيى غفران خضر* **أمير مراد حسين****
** قسم هندسة الميكاترونكس/كلية الهندسة الخوارزمي/ جامعة بغداد*
*** قسم هندسة المعلومات والاتصالات/كلية الهندسة الخوارزمي/ جامعة بغداد*
*البريد الالكتروني: yahia.ghofran1202a@kecbu.uobaghdad.edu.iq
**البريد الالكتروني: ameer@kecbu.uobaghdad.edu.iq

الخلاصة

السيارات ذاتية القيادة لها مكانة بارزة في العلوم والتكنولوجيا، وتؤثر على التنمية الاجتماعية والاقتصادية. يعد التعلم العميق (DL) حاليًا أحد أهم مجالات الدراسة في الذكاء الاصطناعي (AI). في السنوات الأخيرة، تم تقديم حلول قائمة على التعلم العميق في مجال السيارات ذاتية القيادة وحققت نتائج باهرة. تدرس هذه الدراسات مجموعة متنوعة من التقنيات المهمة للمركبات ذاتية القيادة، بما في ذلك أنظمة الملاحة في السيارات، وتخطيط المسار، والإدراك البيئي، والتحكم في السيارة. التحكم في التعلم الشامل هو التحويل المباشر للبيانات الحسية إلى أوامر تحكم في القيادة الذاتية. تهدف هذه الدراسة إلى تحديد الشبكة العصبية العميقة (DNN) الأكثر دقة التي تم تدريبها مسبقًا للتنبؤ بزاوية التوجيه لمركبة مستقلة. الهدف من هذا المشروع هو إنشاء قدرات قيادة ذاتية بالكامل يمكن استخدامها مع تقنيات السيارات المدمجة ذات الأداء المحدود. في اقتراحنا، قمنا بمقارنة ثلاثة نماذج معروفة جيدًا: AlexNet وResNet18 وDenseNet121. تم استخدام تعلم النقل من خلال تعديل الطبقة النهائية من النماذج المدربة مسبقًا من أجل التنبؤ بزاوية توجيه السيارة. أجريت التجارب على مجموعة البيانات التي تم جمعها من مسارين مختلفين. وفقًا لنتائج الدراسة، فإن ResNet18 وDenseNet121 لديهما أقل نسبة خطأ لقيمة زاوية التوجيه. علاوة على ذلك، تم فحص أداء النماذج المعدلة على مسارات محددة مسبقًا. كان أداء ResNet18 أفضل من DenseNet121 من حيث الدقة، مع انحراف أقل عن مركز المسار، بينما كان DenseNet121 أكثر مرونة عبر المسارات المختلفة، مما أدى إلى أداء أكثر اتساقًا.