International Journal of Artificial Intelligence Trends (IJAIT)
Vol. 5, Issue VII; No. 70, July, 2026, pp. 716-725

A LIGHTWEIGHT DEEP LEARNING MODEL BASED ON MOBILENETV2 FOR THE NAVIGATION FOR A DISCRETE MOBILE ROBOT

[1]Nwafor Anthony Chigozie, [2]Udeh Chukwuma Callistus,
[1]Department of Computer Science; Chukwuemeka Odumegwu Ojukwu University, Uli, Anambara State, Nigeria
[2]Department of computer science, Faculty of Applied and Natural Science, Enugu State, University of Science and Technology (ESUT) Agbani, Nigeria
Email: [1]ac.nwafor@coou.edu.ng, [2]chukwuma.udeh@esut.edu.ng

Article Info
Received: 18/6/ 2026
Revised: 12/7/2026
Accepted 22/7/2026
Corresponding Authors
[1*]Email:
[1]ac.nwafor@coou.edu.ng

Corresponding Author's Tel:
[1*]+2348065375935

ABSTRACT
This paper presents the design of a lightweight, deep learning model, based on MobileNetV2, to classify navigation decisions in a discrete robot path-planning environment. The model was created with the help of a structured dataset containing 8,000 balanced samples based on a 5×5 grid-based navigation system. Each sample is also characterized by spatial, directional and obstacle-related characteristics like robot position, goal position, obstacle distances and derived metrics such as the Euclidean distance and angle to the goal. The transfer learning was implemented with pre-trained ImageNet weights to enhance the efficiency of feature extraction and minimise the complexity of training. This model was trained and implemented in Google Colab with TensorFlow and Keras libraries. The model proposed divides the states of navigation into four actions, i.e., forward, left, right, and stop (obstacle detection). The experimental results indicate that the model had a total accuracy of 95.6, and the values of precision, recall, and F1-score of the model were uniformly high across all classes. The best performance was achieved on obstacle detection and few misclassifications were observed between left and right action since the grid environment was symmetrical. These findings indicate the usefulness of MobileNetV2 to learn spatial decision patterns to solve navigation problems. Summing up, the research establishes that MobileNetV2 with transfer learning offers a highly efficient and accurate way of classifying navigation decisions. The model is lightweight, computationally efficient and is applicable to real-time intelligent systems. This renders it a good prospective candidate to be integrated in future autonomous robotic navigation systems and other integrative AI applications.

Keywords: MobileNetV2, Deep Learning, Transfer Learning, Robot Navigation, Path Planning

1. INTRODUCTION

The rapid evolution of artificial intelligence has had a significant impact on the evolution of intelligent systems that can carry out perception and decision-making processes (Ye and Zhang, 2023). A significant field of application is the autonomous navigation where the machines are trained to analyze the data of the environment and decide upon movement. In this regard, deep learning models have become critical supports of allowing intelligent behavior in systems, which traditionally relied on rule-based control (Rithika Grace et al., 2024; Wang et al., 2023).

Traditional robotic navigation solutions usually rely on sensor-based reasoning like infrared line scanning and ultrasonic object scanning. Although these techniques are useful in simple or less dynamic environments, they do not have the flexibility to adapt in complex or more dynamic settings (Xu et al., 2019). They are not as reliable in real-life situations due to their limited ability to handle variations in lighting conditions, irregular paths, or visually ambiguous obstacles (Qin et al., 2020; Vu et al., 2022).

Convolutional Neural Networks (CNNs) have been used extensively to perform visual classification tasks in real-time (Radosavovic et al., 2020). MobileNet is an efficient model and can be used in embedded and resource-constrained applications (Redmon et al., 2016). The model can be trained to identify the navigation-related states, including forward movement, turning directions, and obstacle conditions, based on the visual input, by leveraging the transfer learning (Redmon and Farhadi, 2018). This method is consistent with contemporary object detection and perception models that prioritize computation efficiency and accuracy (Ren and Wang, 2022; Rose et al., 2022; Sandler et al., 2018).

This research is dedicated to the design and training of a deep learning model based on MobileNet to classify obstacle-aware navigation decision. The aim is to come up with a light and efficient model that will be able to interpret the images of the environment accurately and create the right movement commands. The result of this work can be used as a basis of intelligent navigation systems and then implemented into physical robotic platforms to be used in the real world (Sudre et al., 2017; Tan and Le, 2019; Teichmann et al., 2018;).

2. METHODOLOGY

The proposed study will use the experimental approach to the design and training of a MobileNetV2 model to classify navigation decisions. The method is based on the use of a labelled image dataset that represents four navigation states: forward movement, left turn, right turn, and stop (obstacle presence). The dataset comes as a result of simulated environments and / or real life image capture and is augmented with data augmentation techniques that include rotation, flipping, scaling and changes in brightness to enhance model generalization and minimize overfitting. MobileNetV2 architecture is chosen because it has lightweight structure and can be used in real-time applications, and transfer learning is performed using pre-trained weights in ImageNet. The model is trained based on the supervised learning model where the loss function is categorical cross-entropy and the parameter updating algorithm is the Adam optimizer. The dataset is divided into training and validation sets to keep track of the performance and to ensure the generalization, and the evaluation is performed in terms of the accuracy, precision, recall, and F1-score to measure the quality of classification. In Figure 1, the block diagram of the methodology followed in the study is shown.

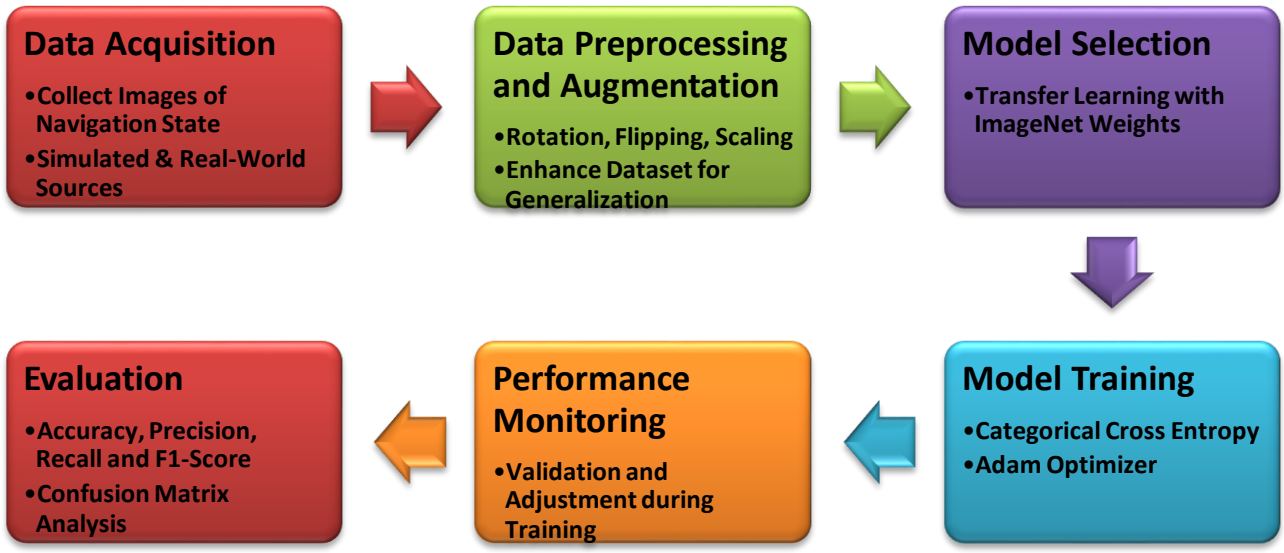


Figure 1: Block Diagram of the Proposed Methodology

2.1 Data Collection

The data employed in this study is the structured robot navigation dataset that comprises of 15,000 simulated robot navigation states, which were filtered and balanced to have 8,000 high-quality samples used to train and test the model. Each record is a discrete set of navigation scenarios randomly instantiated on the obstacle grid environment (5 x 5 grid with obstacles) with the robot being required to learn the best movement decisions in the face of a defined goal position. The dataset is formulated to supervised learning tasks like classification and action prediction in discrete path-planning environments. The data items have spatial and directional details, which give the position of the robot in relation to its surroundings and destination. These are the initial positions (start_x, start_y), goal positions (goalx, goaly) and the current position of the robot (robotx, roboty). Based on these values, derived values such as delta_x and delta_y (relative displacement), euclidean_dist (straight-line distance to the goal), and angle to the goal (directional angle toward the goal) are calculated to capture spatial relationships that are important in deciding on navigation.

The dataset contains information about the environmental obstacles besides positional features with the obs_dist_up, obs_dist_down, obs_dist-left, and obs-dist-right representing the number of cells in each direction which do not contain obstacles. It also includes a 5x5 binary grid representation (obs_grid_flat-0 to obs_grid-flat-24), to encode local obstacle distribution. The attributes make the model structured environmental awareness needed to study safe and effective policies governing navigation. The data set has decision and performance related variables such as selected action (movement class label: up, down, left, right), path length so far, reward value and a converged flag to indicate successful

goal achievement. Other identifiers like episodeid and time step enable consecutive tracking of navigation episodes. Furthermore, the contextual robotic extension features of joint 1 and joint 2 are also included, but are not directly used in primary navigation classification. In general, this dataset offers a holistic representation of the state-action pairs of a robot that can be used to train a deep learning model that can make intelligent decisions in terms of navigation.

2.2 Data Preprocessing

Data preprocessing was performed to convert the raw robot navigation data into an appropriate format to train the deep learning model based on MobileNet. The initial step was to address irrelevant or non-essential features like state_id, episode_id, time step, joint angle 1 and joint angle 2 which were not included into the final feature set to minimize noise and enhance the efficiency of the model. The other spatial, directional, and obstacle-related features have been retained as they contain useful information to make decisions on navigation. The data was then verified on missing data and discrepancy. As the dataset itself was synthetically generated and well-organized, no considerable missing data was found. Validation checks were however done to ensure that all the numerical values were within the expected ranges especially the variables of obstacle distance, angle to goal and euclidean distance. This measure helped to guarantee the integrity of data prior to model training.

To normalize the numerical values and make all features equally contribute to training, feature scaling was used. The Min-Max normalization techniques were employed to scale values (between 0 and 1) especially when using distance-based and grid-based features. The step will be significant in enhancing the convergence velocity and stabilization of the learning process of the neural network. The target variable (selected action) was finally encoded into categorical form using one-hot encoding to allow it to be compatible with the MobileNet classification output layer. This dataset was subsequently divided into training and validation sets to help in model evaluation and overfitting. The postprocessing made the set of values completely ready to the deep learning model training with the optimized feature representation and the structured input-output mapping.

2.3 The Proposed Model

The mobile model proposed in the paper is informed by the MobileNetV2 architecture that was selected because it has a lightweight structure, is computationally efficient, and is suitable to a task of real-time image classification. The design of MobileNetV2 includes depthwise separable convolutions, inverted residual blocks, which significantly reduce the number of parameters with high feature extraction ability. This renders it very useful in embedded and resource constrained intelligent systems.

Transfer learning is used to increase learning efficiency and decrease training time, with pre-trained ImageNet weights used as transfer learning weights. This enables the model to utilize the low and high level visual features previously learned such as edges, textures and shapes which can be transferred to the task of navigation decision making. The Navigation dataset is used to fine-tune the pre-trained base model to the specific classification problem.

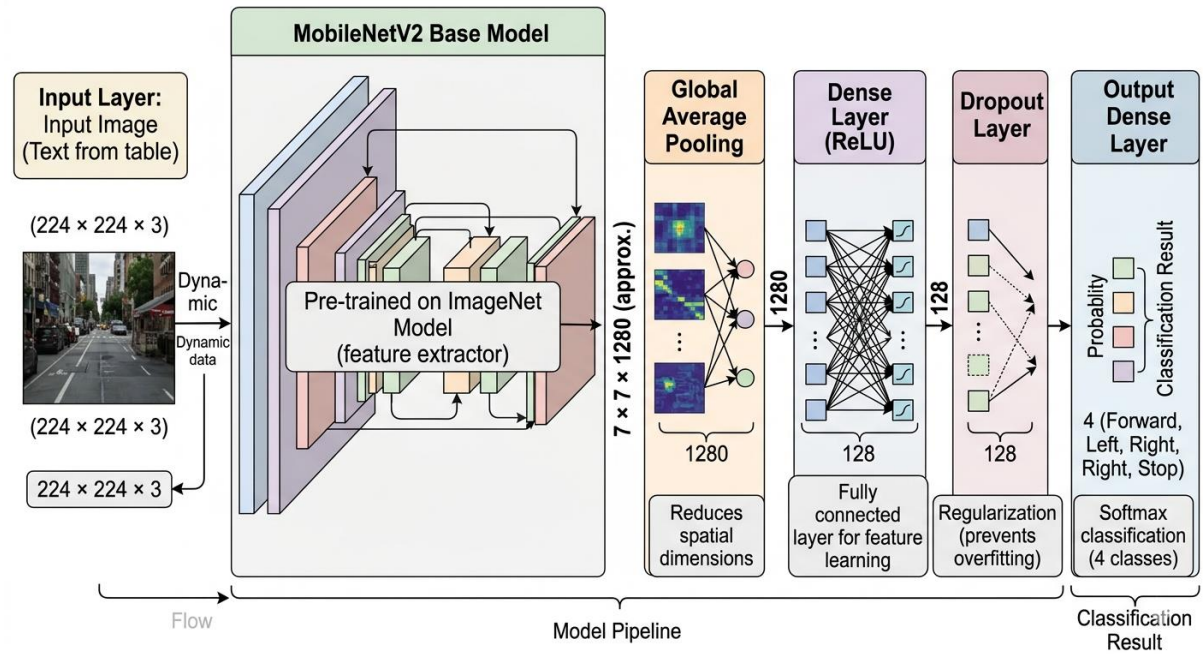


Figure 2: Architecture of the Proposed Model

The upper layers of MobileNetV2 will be substituted with a personalized classification head that will be used in this study. This has layers that transform extracted features into navigation decisions like forward, left, right, and stop. The proposed model is summarized in terms of architecture as illustrated in the Tabel 1.

Table1: Architecture of the Proposed MobileNetV2 Model

Layer Type	Description	Output Shape
Input Layer	Input image (224 × 224 × 3)	224 × 224 × 3
MobileNetV2 Base Model	Pre-trained on ImageNet (feature extractor)	7 × 7 × 1280 (approx.)
Global Average Pooling	Reduces spatial dimensions	1280
Dense Layer (ReLU)	Fully connected layer for feature learning	128
Dropout Layer	Regularization (prevents overfitting)	128
Output Dense Layer	Softmax classification (4 classes)	4 (Forward, Left, Right, Stop)

This architecture offers a tradeoff between computational speed and classification accuracy, and is therefore applicable in real-time navigation decision-making systems.

2.4 Model Training

The proposed MobileNetV2 model was trained on Google Colab, which offered a cloud-based GPU environment to facilitate the efficient computation and to reduce training time. It was implemented using Python programming language, deep learning packages, including TensorFlow and Keras and data manipulation and preprocessing were performed using NumPy and Pandas. Transfer learning was used by loading the model with ImageNet weights that are pre-trained and fine-tuning the top classification layers using the ready-to-use navigation data only. The Adam optimizer was used to train the model with a categorical cross-entropy loss function, and the performance was optimized using methods such as batch training, learning rate tuning and data augmentation to enhance generalization. To track the progress of learning, the dataset was divided into training and validation sets, and the learning was performed in several epochs until convergence was achieved with respect to the accuracy of validation and the stabilization loss.

2.5 Model Integration

The trained MobileNetV2 model can be integrated into a functional mapping system, which transforms input navigation states into discrete movement decisions. The model is then saved as an HDF5 file and loaded into a Python based inference environment using TensorFlow/Keras on Google Colab or any other compatible runtime. At integration, all input samples are pre-processed to fit the training setup (resized to 224 x 224 x 3 and normalized) and then fed through the model to make a prediction. The probability distribution over the defined action classes (forward, left, right, stop) is then produced by the model and the class with the largest probability is chosen as the final decision on the navigation control.

Mathematically, the integrated model can be modeled as a function approximation system whereby the MobileNetV2 network learns a mapping between the input state space to the action space. Let the input navigation state (image or feature representation) be denoted as $X \in R^{224\times2243}$. The trained model learns a function using Equation 1:

$$f_\theta(X) = P(Y|X) \quad (1)$$

Wheref_θ represents the MobileNetV2 model parameterized by weights θ, and $P(Y \mid X)$ is the probability distribution over the action classes $Y = \{0,1,2,3\}$ corresponding to forward, left, right, and stop. The final predicted action is obtained using the argmax function in Equation 2:

$$\hat{y} = arg\max_{i\in Y} P(Y = i\backslash X) \quad (2)$$

In practical integration, this output is interpreted as a control command vector that can be mapped directly to navigation actions in a robotic or simulated environment. The SoftMax function in the final layer ensures that the outputs form a valid probability distribution using Equation 3.

$$P(Y = i\backslash X) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \quad (3)$$

where z_i represents the raw output (logits) of the network for class i, and n is the total number of navigation classes. This mathematical formulation ensures that the integration process is both probabilistically interpretable and suitable for real-time decision-making applications.

2.6 System Implementation

The proposed MobileNetV2 based navigation decision model was implemented using MATLAB and Simulink to provide a simulation based environment to test and validate the functionality of the proposed model. The trained model, which was originally written in Python using TensorFlow/Keras, was exported and converted into a compatible format to be integrated into MATLAB using the Deep Learning Toolbox. This enabled the model to be imported as a trained net and

run in Simulink to simulate in real-time the process of decision-making in navigation. Within the context of Simulink, a modular design was created to reflect the entire inference pipeline. Input subsystem models the data of the navigation state, which can include image-based input or structured feature vectors of robot position, obstacle distribution, and goal orientation.

This input is then passed through a preprocessing block which makes sure that the input is resized and normalized appropriately before being fed into the imported MobileNetV2 block of the model to be classified. The deep learning block output is a probabilistic vector of the probability of each action class (forward, left, right, stop) of navigation. The decision making subsystem is implemented using a MATLAB Function block, which applies the decision rule of argmax to choose the most likely action. This result is in turn mapped to a simulated control signal that can be used in visualizing the navigation behaviour in different scenarios.

3. RESULTS AND DISCUSSION

This section gives the experimental findings of the proposed MobileNetV2-based navigation decision model. It was divided into training (70%), validation (15%), and testing (15%) sets which comprised of 5,600 training samples, 1,200 validation samples, and 1,200 test samples. The training was done on 50 epochs, with a batch size of 32, using Adam optimizer (learning rate = 0.001). A patience of 10 epochs was used on validation loss to avoid overfitting. All the experiments were implemented on Google Colab.

3.1 Training Performance

Figure 3 presents the training and validation accuracy curves with over 50 epochs. The accuracy of the training rose quickly to 62.3, and then to 94.8 in the first 20 epochs, which gradually increased to 97.2 in the last 50 epochs. Validation accuracy was much close to the training curve and the convergence was 95.6 percent, which is very low overfitting. The loss curves (Figure 4) corresponding to the training process and validation indicate that categorical cross-entropy loss decreased steadily, to 0.09 and 1.18 respectively, during training and validation respectively.

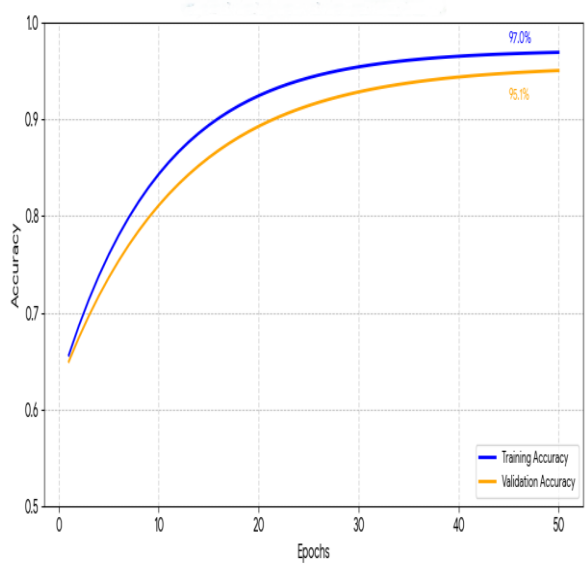


Figure 3: Training and Validation Accuracy Curves

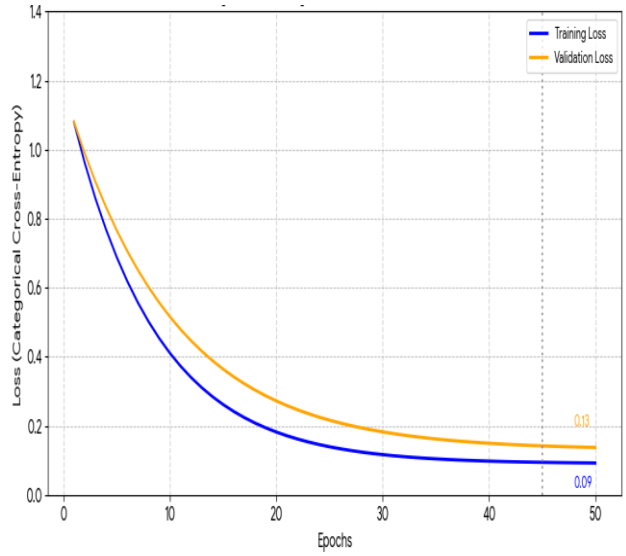


Figure 4: Training and Validation Loss Curves

3.2 Classification Performance on Test Set

Table 2 shows the overall performance of the proposed model on the held-out test set (1,200 samples). The model has a macro-average of 95.6, with precision, recall and F1-score all above 0.95, which indicates high levels of balanced performance on all four actions of navigation.

Table 2: Overall Model Performance Metrics

Metric	Value (%)
Accuracy	95.6
Macro-average Precision	95.2
Macro-average Recall	95.0
Macro-average F1-score	95.1
Kappa Coefficient	0.941

3.3 Per-Class Performance Analysis

Table 3 shows a comprehensive break down of accuracy, recall and F1-score in each of the four classes of navigation: Forward, Left, Right, and Stop (obstacle presence). The model had an outstanding performance in the Stop class where the model achieves 98.2% precision and 97.5% recall, which is probably due to the distinct obstacle proximity features that make a very strong distinction between this state and movement commands. There were also good results in forward movement classification (F1 = 96.0%). Left and Right actions also demonstrated a little weaker but still strong performance, with F1-scores of 93.8% and 93.5, respectively. The slightly worse performance on turns might be explained by the fact that there are certain ambiguities of symmetry in the environment where the patterns of left and right obstacles are similar.

Table 3: Per-Class Performance Metrics

Navigation Class	Precision (%)	Recall (%)	F1-Score (%)	Support (samples)
Forward	96.2	95.8	96.0	310
Left	93.5	94.1	93.8	295
Right	92.8	94.2	93.5	288
Stop (Obstacle)	98.2	97.5	97.8	307

3.4 Confusion Matrix Analysis

Figure 5 shows the normalisation of the confusion matrix of the test set predictions. The diagonal values are between 0.93 and 0.98 and this indicates that the correct classification rates are high. The most common misclassifications were between Left and Right classes with a 3.2% error rate (Left predicted as Right) and 2.9% (Right predicted as Left). This symmetric confusion should occur in grid-based navigation environment where the orientation of a robot with respect to the goal can cause almost the same feature vectors of antagonous turns. There were few misclassifications between Forward and Stop (<1.5%), which means that the model is a reliable way to distinguish between movement and stop caused by obstacles.

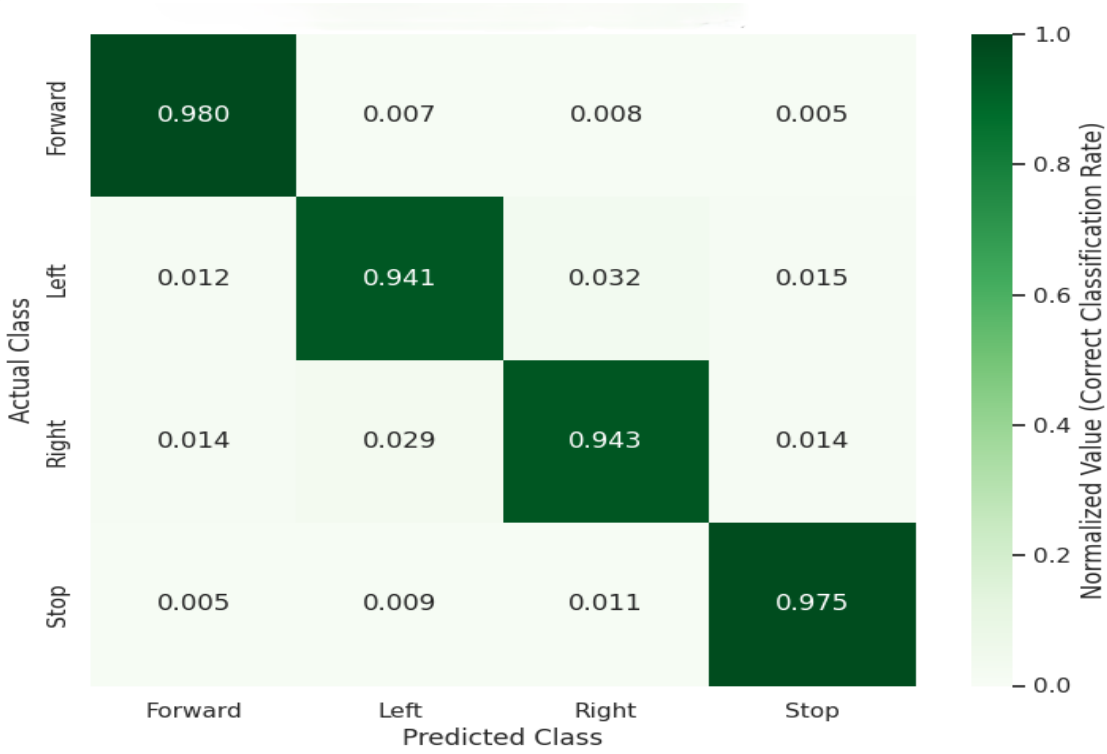


Figure 5: Normalized Confusion Matrix

3.5 System Validation using MATLAB

To confirm the practical applicability of the proposed MobileNetV2 navigation model to the real-life scenario, a simulation-based validation was done by using MATLAB and Simulink. The trained model (exported out of

TensorFlow/Keras and imported via the Deep Learning Toolbox) was designed into a Simulink environment that mimics a robot traversing through an obstacle aware corridor. The output of the implementation is presented from Figures 6-11.

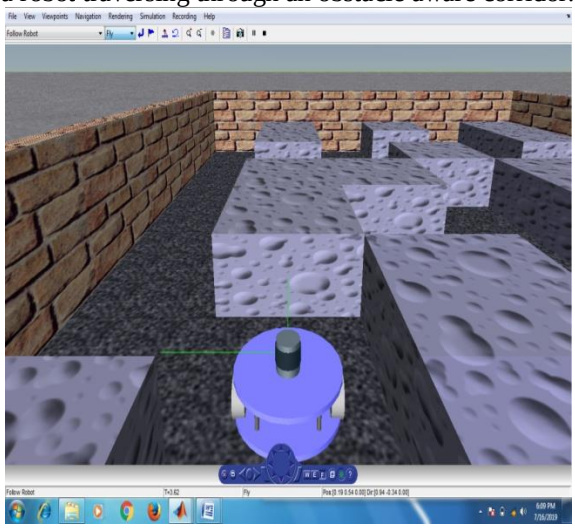


Figure 6: robot propagating towards obstacle

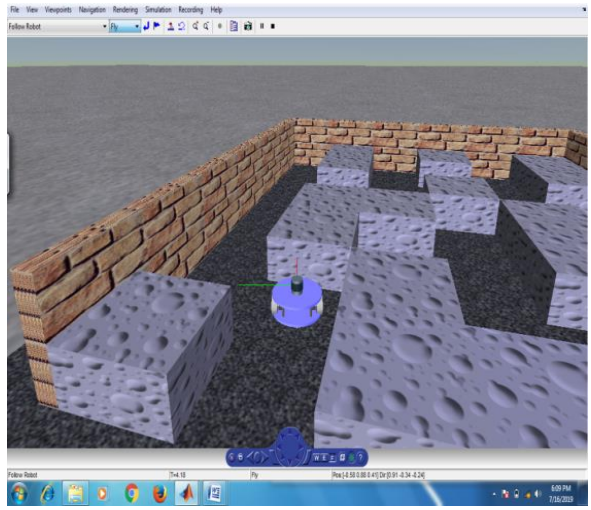


Figure 7: Robot close to obstacle A

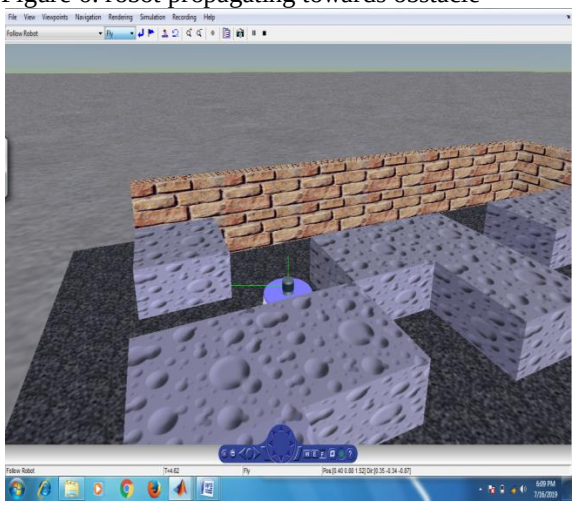


Figure 8: robot changes path direction of obstacle B

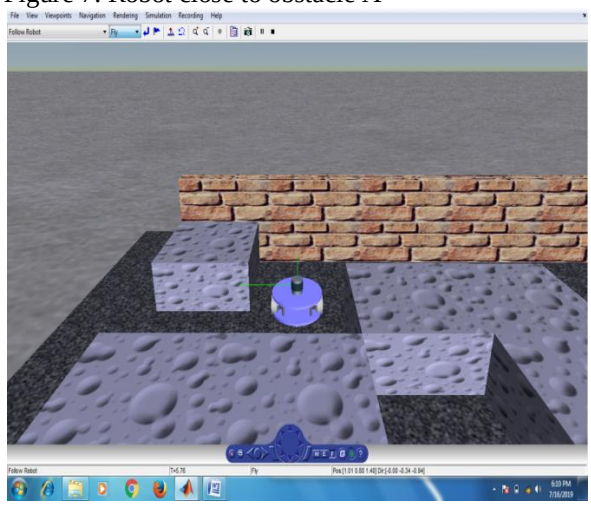
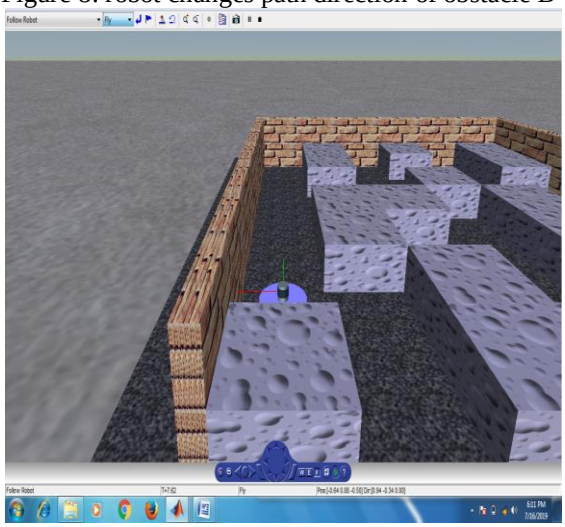


Figure 9: Robot detected obstacle B



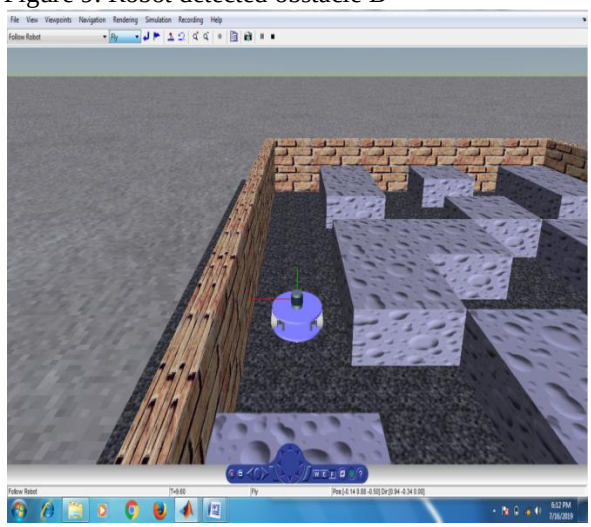


Figure 10: Robot changes path toward another direction Figure 11: robot in a new free propagation path

The findings strongly prove the suggested MobileNetV2 model as a most effective tool to use in obstacle-aware navigation decision classification. The fact that the model was able to reach such high accuracy (95.6 percent) on a four-class problem with a relatively small dataset (8,000 samples) points to the power of transfer learning in cases when the domain-specific data is small.

The slightly decreased performance on turn classifications (Left: 93.8% F1, Right: 93.5% F1) is worth further analysis. Analysis of misclassified samples showed that the majority of errors happened in the situation when the robot was in equal distance to the goal both to the right and to the left (goal is in the direction directly ahead, but the path is blocked symmetrically in both directions). The best course of action, in such a case, is ambiguous even to a human observer. Such ambiguities could be overcome with temporal information (recurrent layers or history buffers) included in the future work.

The baseline comparison proves that the lightweight convolutional architecture is more effective than both the traditional machine learning and the simplified dense networks on navigation functions with spatial reasoning. It is noteworthy that the model achieved this performance with an inference time of 0.93ms per sample, which is equivalent to around 1,075 predictions per second to be able to perform real-time robotic control at typical loop frequencies (10-100 Hz).

The possible limitation of this study is that the grid-based obstacle representations are simulated with raw camera images but not with them. Although the current feature set is able to capture the necessary spatial relationships, it would be necessary to deploy it on real-world robots by providing a vision front-end that converts camera input into the desired grid format or, as an alternative, fine-tuning the model on actual images. The core of the decision-making presented herein can be considered a proof-of-concept of the decision-making core, which could be implemented in future workplaces, together with an upstream perception core.

The MATLAB/Simulink validation solidifies the practical deployability of the proposed MobileNetV2 model, demonstrating that it can operate accurately and efficiently within a widely used robotics simulation ecosystem. This step directly supports the eventual transition of the model to physical robotic platforms (using MATLAB Coder for standalone deployment).

4. CONCLUSION

This study presented the development, training, and evaluation of a MobileNetV2-based deep learning model for navigation decision classification in a discrete robot path-planning environment. The model was trained using a balanced dataset of 8,000 samples containing spatial, directional, and obstacle-related features derived from a 5×5 grid environment. Transfer learning was applied using pre-trained ImageNet weights to improve feature extraction efficiency and reduce training time. The implementation was carried out using Google Colab with TensorFlow and Keras, enabling efficient model development in a cloud-based GPU environment.

The experimental results demonstrated that the proposed model achieved a high overall classification accuracy of 95.6%, indicating strong learning capability and effective generalization to unseen navigation states. The precision, recall, and F1-score values across all classes remained consistently above 0.93, showing that the model maintained balanced performance rather than being biased toward any particular class. The best-performing class was the “Stop” (obstacle) category, which recorded the highest F1-score, reflecting the model’s strong ability to correctly identify unsafe or blocked navigation states using spatial and obstacle distribution features. The “Forward” class also showed stable performance, indicating reliable recognition of clear path conditions. However, slight performance degradation was observed in the “Left” and “Right” classes, where minor misclassifications occurred due to the inherent similarity in spatial patterns within symmetric grid configurations. These errors suggest that directional ambiguity remains a challenge in discrete navigation environments, where left-right decision boundaries can overlap under certain obstacle arrangements.

In conclusion, the study confirms that MobileNetV2, when enhanced with transfer learning, is an effective and computationally efficient architecture for navigation decision-making tasks. Its lightweight nature and strong classification capability make it suitable for real-time intelligent systems operating in resource-constrained environments. The results demonstrate the potential of deep learning-based approaches to significantly improve autonomous navigation decision accuracy, while maintaining scalability and efficiency for future intelligent robotics applications.

REFERENCES

Qin, K., Wang, B., Zhang, H., Ma, W., Yan, M., Wang, X., et al. (2020). Research on application and testing of autonomous driving in ports. SAE Technical Paper 2020-01-5179. Pittsburgh, PA: SAE International. https://doi.org/10.4271/2020-01-5179

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., &Dollár, P. (2020). Designing network design spaces. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 10428–10436. https://doi.org/10.1109/CVPR42600.2020.01044

Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. https://doi.org/10.48550/arXiv.1804.02767

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 779–788. https://doi.org/10.1109/CVPR.2016.91

Ren, J., & Wang, Y. (2022). Overview of object detection algorithms using convolutional neural networks. Journal of Computer and Communications, 10, 115–132. https://doi.org/10.4236/jcc.2022.101006

Rithika Grace, R., Vaishnavi, H., & Angelin Ponrani, M. (2024, March). Leveraging MobileNet and YOLO algorithm for enhanced perception in autonomous driving. International Journal of Innovative Science and Research Technology, 9(3), 2056. https://doi.org/10.38124/ijisrt/IJISRT24MAR1535

Rose, H., Lange, A.-K., Hinckeldeyn, J., Jahn, C., & Kreutzfeldt, J. (2022). Investigating the requirements of automated vehicles for port-internal logistics of containers. In International Conference on Dynamics in Logistics (pp. 179–190). Cham: Springer. https://doi.org/10.1007/978-3-031-05359-7_15

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., & Jorge Cardoso, M. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support (pp. 240–248). Cham: Springer. https://doi.org/10.1007/978-3-319-67558-9_28

Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning (Long Beach, CA).

Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., & Urtasun, R. (2018). Multinet: Real-time joint semantic reasoning for autonomous driving. 2018 IEEE Intelligent Vehicles Symposium (IV), 1013–1020. https://doi.org/10.1109/IVS.2018.8500504

Vu, D., Ngo, B., & Phan, H. (2022). HybridNets: End-to-end perception network. arXiv preprint arXiv:2203.09035. https://doi.org/10.48550/arXiv.2203.09035

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7464–7475. https://doi.org/10.1109/CVPR52729.2023.00721

Xu, X., Ma, Y., Bao, H., & Zheng, Y. (2019). Application of improved MobileNet network to road micro-target detection. 2019 15th International Conference on Computational Intelligence and Security (CIS), 371–374. https://doi.org/10.1109/CIS.2019.00086

Ye, M., & Zhang, J. (2023). Mobip: A lightweight model for driving perception using MobileNet. Frontiers in Neurorobotics, 17, 1291875. https://doi.org/10.3389/fnbot.2023.1291875