Available online at http://scik.org

Commun. Math. Biol. Neurosci. 2025, 2025:47

https://doi.org/10.28919/cmbn/9165

ISSN: 2052-2541

FINE-TUNING VGG16 MODEL FOR DRIVER BEHAVIOR CLASSIFICATION

SEBASTIANUS ADI SANTOSO MOLA[1,*], TAMAR D. I. DAPPA OLE[1], ANDREA STEVENS KARNYOTO[2,3], NALDO F. TADJO UDJU[1], ANNIKA L. HIPIR[1], BENS PARDAMEAN[2,3]

[1]Department of Computer Science, University of Nusa Cendana, Kupang 85148, Indonesia

[2]Bioinformatics and Data Science Research Center, Bina Nusantara University, Jakarta 11480, Indonesia

[3]Computer Science Department, BINUS Graduate Program - Master of Computer Science Bina Nusantara University, Jakarta 11480, Indonesia

Abstract: This study aims to improve the performance of the VGG16 model with fine tuning. The transfer learning method is adopted to accelerate the learning process. The fine tuning carried out includes adding layers to the VGG16 model, freezing the initial layer during the training process, and using an adaptive learning rate and using dropout. The VGG16 model with fine-tuning is used for multi-class classification of five categories of driver behavior. The learning results show an increase in accuracy of up to 99.54%, with a decrease in the validation loss value to 0.0278, compared to the VGGNet baseline without fine-tuning which only achieved an accuracy of 96%. Layer freezing also reduces the number of trainable parameters by more than 50%. Fine-tuning has proven effective in improving model performance by adjusting parameters in the final layer so that the model is able to capture specific features of the target dataset and maintain basic features with the initial freeze layer. Testing using a confusion matrix shows a high level of prediction accuracy. This study underlines the importance of applying fine-tuning to CNN models to improve driver behavior detection capabilities.

Keywords: driver behavior detection; CNN; VGG16; fine-tuning.

2020 AMS Subject Classification: 68T45, 68T01.

*Corresponding author

E-mail address: adimola@staf.undana.ac.id

Received January 29, 2025

1. INTRODUCTION

Distraction driving is a form of driver inattention and is one of the main causes of traffic accidents in the world [1-2]. Distraction driving is defined as 'the diversion of attention away from activities for safe driving toward a competing activity' [3]. There are three categories of distraction driving, namely visual, manual and cognitive [4-5]. Visual distractions such as taking your eyes off the road, changing your focus of vision such as looking at maps, videos and other objects on the road or in the car. Manual distractions such as taking your hands off the wheel for activities such as using a mobile phone [5-8], eating/drinking [9-10], smoking [11-12], and operating the radio [13-14]. Cognitive distractions include loss of focus, talking to other people in the car, listening to the audio system, daydreaming, fatigue and stress [15]. However, distraction driving can be a combination of these three categories [1].

There has been a lot of research related to distraction driving. A machine learning approach using Random Forest (RF) has been carried out by [16-17]. Support Vector Machine (SVM) methods have been used in research [18-19]. The main problem in the machine learning approach is that the classification process involves features that must be extracted and selected using a separate method. This brings its own complexity to the study.

Deep machine learning models offer more adaptive automatic feature extraction capabilities compared to manual methods. The Naturalistic Driving Study (NDS) approach is used to observe activities such as mobile phone usage, operating dashboard devices, and talking to passengers. Studies on NDS with mobile phone usage activities have been conducted by [20-21] using the Bi-LSTM architecture. Research [20] adds an attention mechanism to improve accuracy, and [21] enhances the feature vector from the image to improve accuracy. The RNN architecture has been used in research [22-23]. Research [22] improves the performance of RNN by adding hierarchical layers while [23] uses Forward Weight Adjusted (FWA).

CNN architecture has also been commonly used in distracted driving detection, especially by utilizing pre-trained models. The advantages of each model are claimed in the following studies: ResNet [24-26], Xception [27], AlexNet [28], EfficientNet [29], MobileNet [30], and VGG16 [31]. With transfer learning and fine-tuning methods, the performance of pre-trained models can be improved. Various fine-tuning methods have been carried out in research such as dropout by turning off some neurons during the learning process [32], optimizer to reduce errors [33-34], early stopping to stop the learning process before overfit occurs [35], weight regularization as a form of penalty on weights to increase generalization [36] and changes in learning rate to produce faster

learning convergence [37-38].

This study aims to maximize the performance of the VGG16 model which is famous for its architectural simplicity and image recognition capabilities [39] by performing transfer learning and empirical fine tuning on its architecture and learning process in detecting driving distractions using raw data without modeling specific characteristics. The contribution of this research is that model-based transfer learning is proposed for the case of distracted driving detection, modification of VGG16 architecture by adding layers at the end to recognize specific patterns of distraction such as mobile phone usage or distracted driving, freezing of early layers of VGG16 architecture to retain common features of distracted patterns, enhancement of model generalization ability by dropout, and changing learning rate to accelerate learning.

2. PRELIMINARIES

2.1 Research methodology

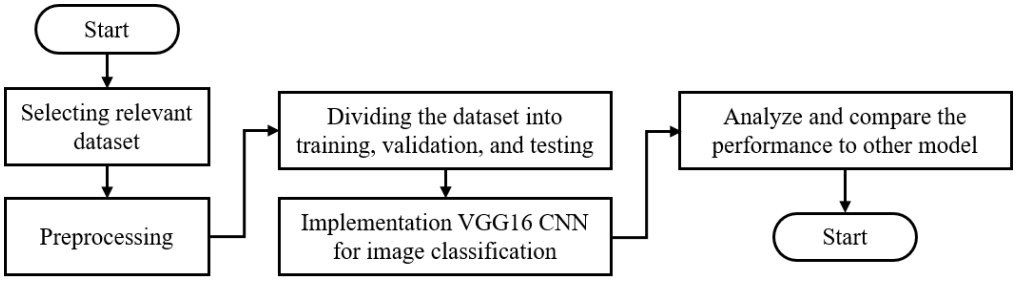

Figure 1. Research process

Figure 1 illustrates our research process; it involves several key steps, starting with selecting relevant datasets. This step is crucial because we can determine the quality and relevance of the research findings. Also, researchers can ensure accurate and reflective analyses by choosing appropriate datasets. Pre-processing data is to provide quality and consistency. This stage involves removing any outliers or errors from the dataset. The final step is to analyze the data and draw meaningful conclusions. We divide the dataset into training, validation, and testing datasets. The training dataset is used to train our model, while the validation dataset is utilized to evaluate the trained model. Finally, the testing dataset tests the model's accuracy and performance. Implementing a CNN network architecture for image classification. After training, model performance evaluation is based on accuracy, precision, recall, and F1-score metrics. Experimental results are analyzed to compare the performance of CNN models, including AlexNet, VGGNet, and ResNet34, in multi-class classification on a dataset consisting of five classes.

2.2 VGG16 CNN Architecture

This study uses the VGG16 model architecture in image multiclass classification. 'VGG' refers to the Visual Geometry Group of the University of Oxford. The VGG16 model has simple and effective hierarchical structure [40]. It using 16 layers consisting of the convolutional layer to extract essential features from the image [41], ReLU activation to add non-linearity, pooling layer to reduce the dimensionality of features while retaining key information, and fully connected layer (as shown in Figure 2) [39]. Evaluation in several experiment shows that VGG16 with fine-tuning provides accuracy. Also, the VGG16 involves adjusting a model that has been trained on a large dataset such as ImageNet for a specific task using a smaller dataset [42].

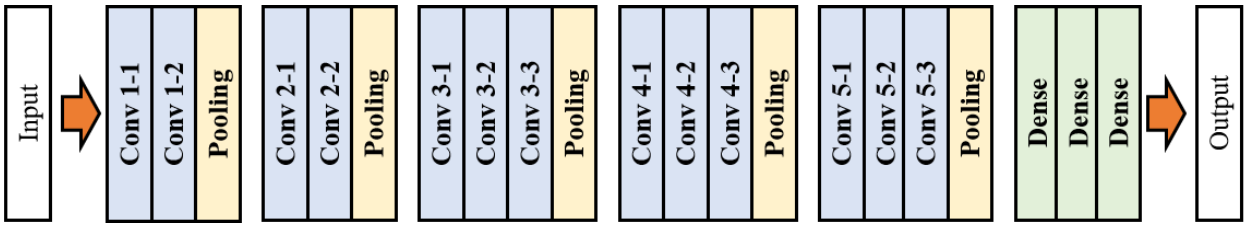

Figure 2. Original VGG16 architecture

Input tensor size in VGG16 is 224×224 with 3 RGB channels, so it takes a 224×224×3 input tensor-size input. In contrast to many hyper-parameters, VGG16 has a convolution layer of a 3×3 filter with stride one and always uses the same padding and maxpool layer of a 2×2 filter with stride 2.

In this experiment, the initial layers that learn basic features are kept frozen, while the last layers are retrained so that the model can adapt to the new dataset. This condition allows the model to learn more complex features without being affected by the initial layers, resulting in a more accurate model. The retraining process also ensures that the model remains up-to-date and can be applied to new data.

2.3 Empirical Fine-Tuning

Fine-tuning is the process of adjusting a pre-trained model to be used on a specific dataset [43]. In this case, the VGG16 model that has been trained using ImageNet is adjusted for the task of detecting driver behavior. The empirical fine-tuning steps in this study include:

1. Adjustment of the Final Layer

   Additional layers are added to the base model to support driver behavior classification. Adding layers to the final layer aims to improve the model's ability to recognize specific features from different domains. As a result, the model can learn to identify nuanced

features that indicate different driving styles or conditions. Also, this approach enhances the classification system's accuracy and robustness.

2. Early Freeze

   The initial layers of the VGG16 model are maintained without retraining. This is done because the initial layers are already able to recognize common features such as basic patterns, edges, or textures.

3. Hyperparameter Settings

   Several main parameters are determined during training:

   - Learning Rate to minimize large changes in weights
   - Batch Size to achieve a balance between speed and stability
   - Dropout Rate to prevent overfitting on small datasets
   - Epoch to ensure the model learns complex patterns without overfitting

4. Dataset Division

   The model is trained using a dataset that has been divided as follows:

   - 90% of data for training
   - 10% of data for testing

2.4 Confusion Matrix

Confusion matrix is used to evaluate the model's ability to classify categories in driver behavior. This matrix presents the distribution of model prediction results in detail, including the number of correct predictions (both positive and negative) and incorrect predictions (positive and negative) [44-45].

3. MAIN RESULTS

3.1 Dataset

Table 1. Driver Behavior Dataset

Class	Driver Behavior	Number of Images
0	Safe Driving	2,203
1	Talking on Phone	2,169
2	Texting on Phone	2,203
3	Other Activities	2,128
4	Turning	2,063
	Total Data	10,766

The dataset used in this study was taken from Driver Behavior Detection | CNN, available on Kaggle, accessed on September 23, 2024 [46]. It is an open dataset and is publicly available to ensure the transparency and reproducibility. This dataset contains images of car driver behavior while driving. It is divided into five classes with a total data of 10,766 images in Joint Photographic Group (JPG) format, as seen in Table 1. The dataset is divided into training data and test data. Training data consists of 9,689 data (90%), and test data consists of 1077 data (10%).

Examples of images representing the five driving activity classes can be seen in Figure 3. These images show the various poses and objects associated with each activity class. The photos were chosen to represent multiple scenarios within each driving activity. Manual selection and algorithmic filtering were used to ensure that the images accurately depict typical poses and associated objects. This selection process aimed to provide a comprehensive visual representation of each class for better understanding and analysis.

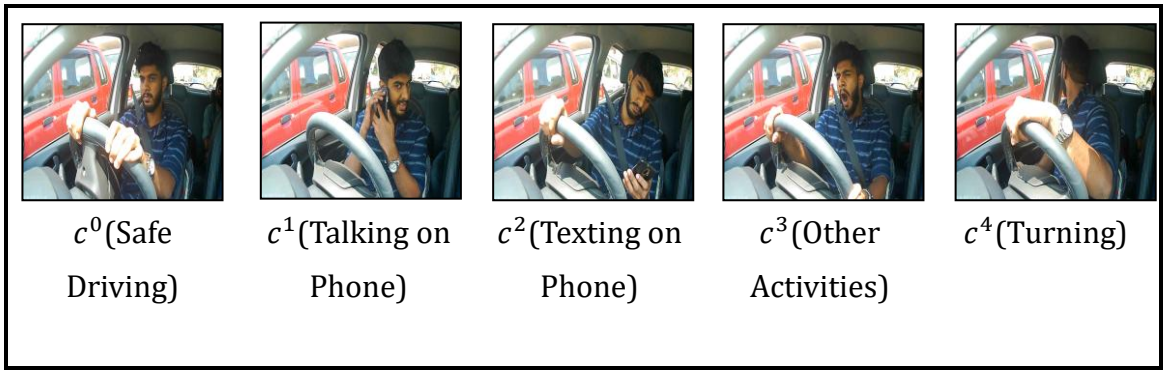

Figure 3. Examples of driver behavior activity classes

3.2 Final Layer Adjustment of VGG16

The VGG16 model architecture is modified by adding a final layer to increase the model's capacity in recognizing specific features by adding layers to VGG16. These adjustments include:

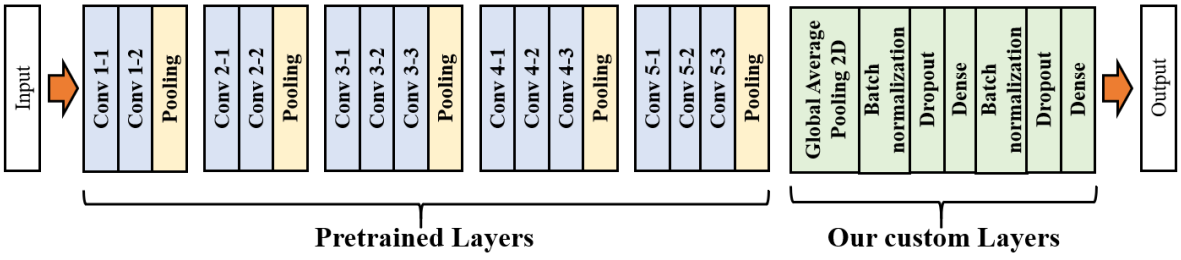

Figure 4. Our custom VGG16 architecture

As we can see in Figure 4, we replace the fully connected layer to custom layers that contain:

- Global Average Pooling 2D, to reduce the output into a more straightforward vector
- Dense layer with 512 units and ReLU activation to process deeper information
- Batch normalization stabilizes the training process so that the model converges faster
- Two dropouts of 0.5 are applied to reduce the possibility of overfitting
- Output in the form of a softmax layer to produce classification probabilities for five behavioral classes

By tailoring the architecture, we can improve model performance and achieve more accurate predictions or classifications. The final layers added are listed in Table 2.

Table 2. VGG16 Architecture Modification Summary

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
vgg16 (Functional)	(None, 7, 7, 512)	14714688
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
batch_normalization (Batch Normalization)	(None, 512)	2048
dropout (Dropout)	(None, 512)	0
dense (Dense)	(None, 512)	262656
batch_normalization_1 (BatchNormalization)	(None, 512)	2048
dropout_1 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 5)	2565
Total params : 14984005 (57 .16 MB)		
Trainable params : 7346693 (28 . 03 MB)		
Non_trainable params : 7637312 (29 . 13 MB)		

The model starts with an input layer that receives an image with a size of 224×224 pixels and 3 color channels (RGB). Then, the main part of the model uses the VGG16 base model that does not include the last classification part, only relying on convolutional and pooling layers for feature extraction, producing an output of size 7×7×512. After that, Global Average Pooling is used to flatten the output and reduce the spatial dimension to a feature vector of size 512. Next,

there is a batch normalization that normalizes the data distribution, followed by dropout to prevent overfitting. The model then has a dense layer with 512 units, followed by batch normalization 1 for training stability and dropout 1 for additional regularization. The last layer is the output layer (Dense 1) that produces predictions with 5 classes, according to the number of categories in the dataset. This model has been modified by fine-tuning certain layers to improve prediction accuracy.

3.3 Early Freeze

In the training stage, the early freeze method is applied by freezing the initial layers of the pre-trained model. Layers 1 to 12 are frozen during training so that the pre-trained weights can be maintained. In addition to maintaining the model's ability to recognize general features, early freeze also significantly reduces the number of trainable parameters by more than half the number of parameters of VGG16 [47]. Thus, the training complexity can be reduced and the transfer learning ability of the pre-trained model for high-level features can be improved.

Freezing is applied by setting the trainable property to false for all layers except the last four layers. Meanwhile, additional layers added on top of VGG16, such as GlobalAveragePooling2D, BatchNormalization, and Dense layers, are trained completely from scratch to adjust the features extracted by VGG16 to the specific classes of the new dataset. One potential challenge with this approach is that freezing too many layers might hinder the model's ability to adapt to new, domain-specific features, especially if the new dataset is significantly different from the original one. Additionally, if the frozen layers do not capture relevant features for the new task, the model's performance may be suboptimal.

3.4 Hyperparameter tuning

The hyperparameters specifically set in this study include batch size = 16, dropout = 0.5 and a learning rate that can be changed. Both initial hyperparameters are set fixed. Changes in learning rate during training use the decay learning rate method. In the early stages of training, the learning rate is set quite large, namely 0.001 to ensure the speed of steps towards convergence. Decay learning rate occurs when the model performance does not show improvement after each batch is run. Model performance is measured by validation loss and a reduction in learning rate by one-fifth of the previous learning rate. During the training process, the system experienced a decrease in learning rate which was initially 0.001 to 0.00004 at the end of training.

3.5 Model Performance

3.5.1 Training Performance

The training stage uses the training set and the model performance is measured during the

training process. Model evaluation is conducted to measure the level of accuracy and loss of the model that has been trained using the VGG16 architecture. The results of the model accuracy evaluation can be seen more clearly in Figures 5 and 6.

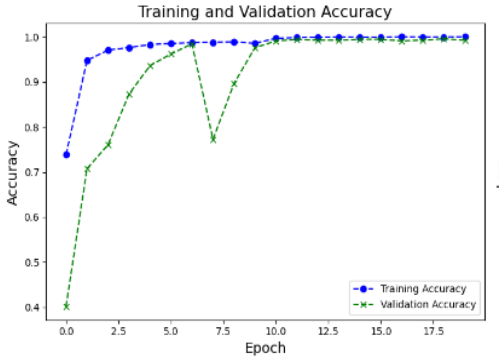

Figure 5. Training and Validation Accuracy

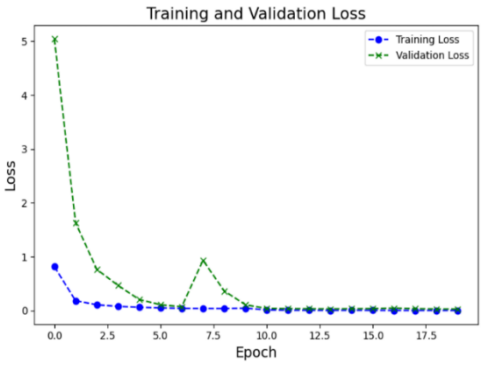

Figure 6. Training and Validation Loss

From the epoch configuration of 20 times, the validation accuracy rate slowly increased from 0.7386 at the beginning of training to 0.9997, indicating training results with almost perfect accuracy. The training loss decreased consistently from 0.8157 to 0.0015, while the validation loss also decreased from 5.0432 to 0.0278 at the 20th epoch. The validation accuracy increased from 40.20% at the beginning of training to 99.54% at the 19th epoch, although it fluctuated at the 8th epoch. Adjusting the learning rate after the 10th epoch helped the model become more stable and effective, resulting in high accuracy without overfitting. The configuration with an initial learning rate of 0.001 and a batch size of 16 proved to be optimal.

In Figure 5, the fine-tuning process plays a key role in improving model performance. By utilizing a pre-trained model such as VGG16, which was previously trained on a large dataset such as ImageNet, fine-tuning allows the model to be adjusted for more specific classification. The accuracy graph in Figure 5 shows a consistent increase of nearly 99.54%, indicating that the model can quickly adapt to new datasets through fine-tuning the last layers. The significant decrease in loss in Figure 6, with a value of 0.0278, indicates that fine-tuning is able to correct prediction errors and optimize model performance.

3.5.1 Testing Performance

Testing performance is done by testing the trained model using new data that has never been recognized by the model before, namely driver behavior data that is not included in the training data. This process involves predicting the image, analyzing its suitability to the trained model, and determining the behavior class that matches the image. From the confusion matrix in Table 3, it can be seen that the model performance in classification is very good. The level of

precision and recall in classes 3 and 4 is 100% indicating that the system can perform perfect classification for that class. While for class 2, the recall value of 98% indicates that from all data predicted as class 2, there is still an error of 2%. In general, the f1-score value of the model is very high which indicates a well-balanced performance, demonstrating that the model can concurrently attain high precision and high recall. The model accuracy is excellent at 99%.

Table 3. Confusion Matrix of Model Performance

	Srecision	**Recall**	**F1-score**	**Support**
0	0.99	1.00	0.99	221
1	0.99	1.00	0.99	221
2	0.99	0.98	0.99	212
3	1.00	1.00	1.00	206
4	1.00	1.00	1.00	217
Accuracy			0.99	1077

From Figure 7, it can be seen that the number of samples with actual label 0 as many as 220 images were successfully predicted correctly, while 1 image was incorrectly predicted as class 1. Similar errors also occur in other classes, but in general the model prediction shows a high level of accuracy with most of the values on the main diagonal, indicating correct predictions. Values outside the diagonal reflects prediction errors for certain samples.

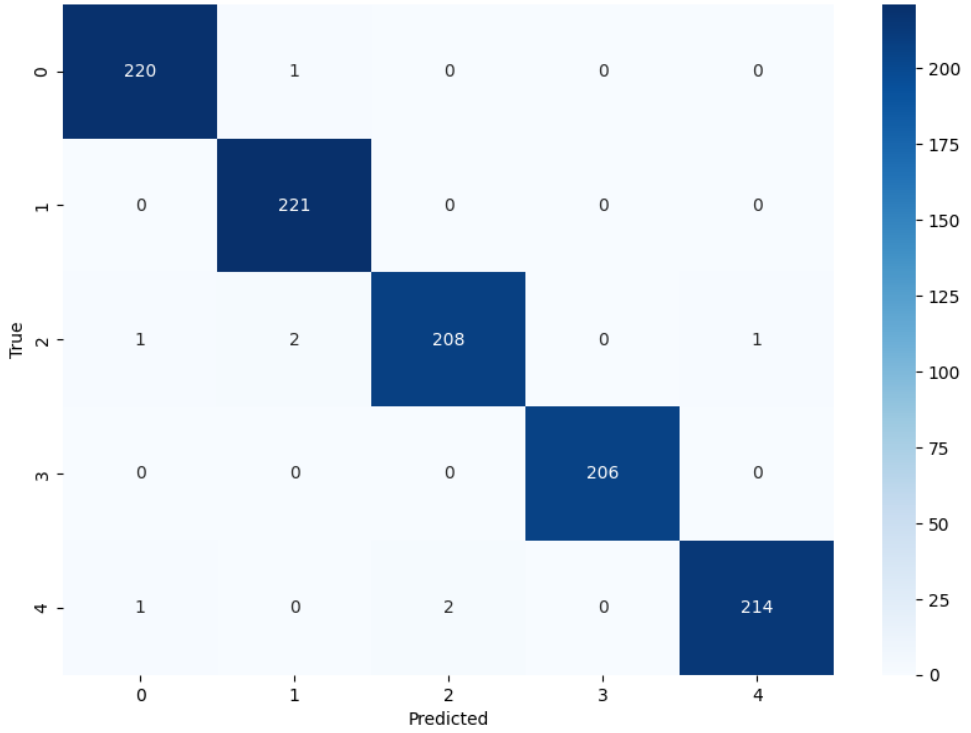

Figure 7. Heat map visualization of model performance

3.4 Discussion

3.4.1 Model Performance Comparison with VGGNet, AlexNet and RestNet34

This study will be compared with the study [47] which also uses the same dataset, epoch but with a different CNN model. The study uses the VGGNet, AlexNet and Resnet34 architectures. Figures 8 and 9 show the performance of the VGGNet. It can be seen that the learning process on VGG16 with fine-tuning is smoother with smaller accuracy and loss ripple indicators for each epoch. This shows the positive influence of the decay learning rate method applied. The final accuracy at the 20th epoch with VGG16 with fine-tuning has an accuracy of 99.54%, higher than VGGNet at 96.65%. The loss on VGGNet is still quite large, namely 7.90%. Likewise, the loss during training can be suppressed using fine-tuning. The learning process using fine-tuning is superior to transfer learning without fine-tuning for the VGGNet architecture.

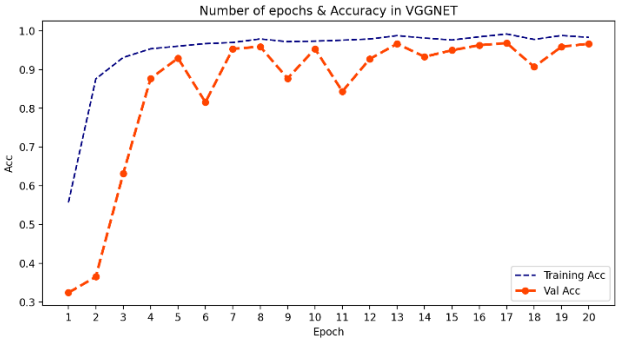

Figure 8. Accuracy with VGGNet

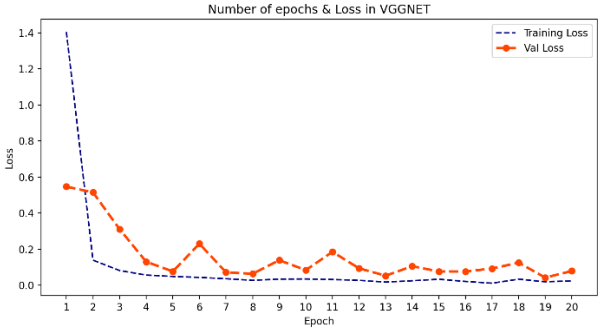

Figure 9. Loss with VGGNet

The pre-trained AlexNet model has a training accuracy of 96.46% but has a smoother graph compared to VGGNet. The accuracy is still slightly lower than VGGNet. On the other hand, the loss on AlexNet is lower than VGG16 at the 20th epoch, this can be seen in Figures 10 and 11.

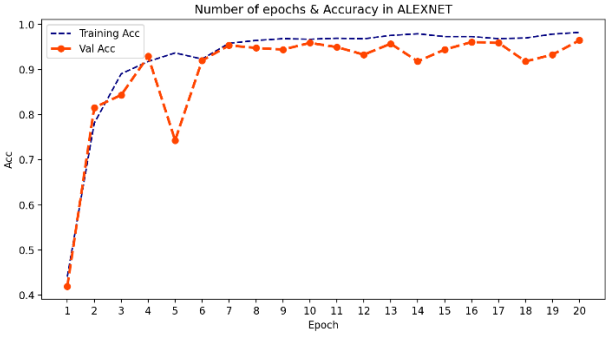

Figure 10. Accuracy with AlexNet

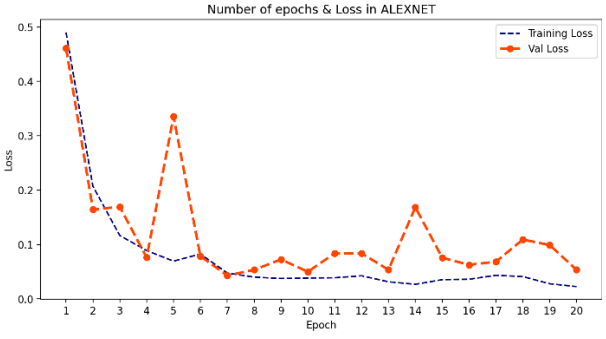

Figure 11. Loss with AlexNet

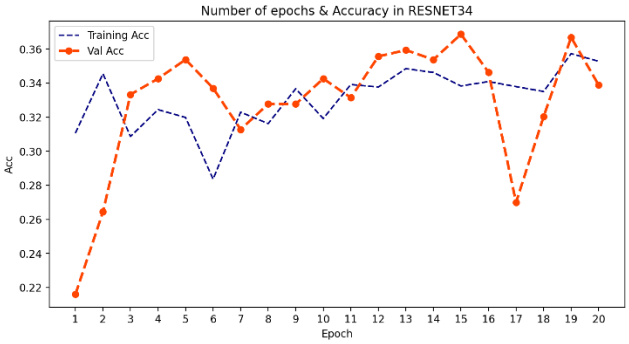

Figure 12. Accuracy with ResNet34

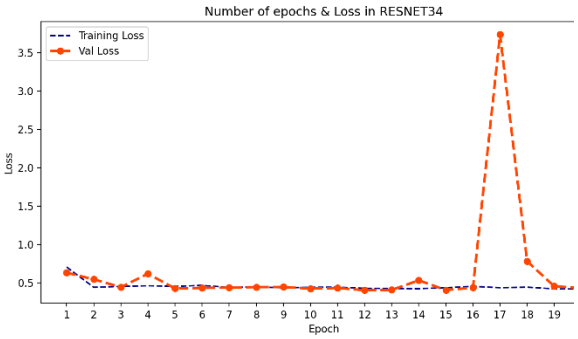

Figure 13. Loss with ResNet34

As can be seen in Figures 12 and 13, training using the ResNet34 model has not shown convergence in the 20th epoch. The ResNet34 model shows the worst training results of the other models. The training accuracy is only 33.89% and the loss is 43.15%.

3.4.2 Computational Complexity

In this study, computational complexity is measured by comparing the number of parameters in each model. Table 4 shows four models with their number of parameters, namely the VGG fine-tuned model, VGGNet, AlexNet and ResNet34. From the total parameters, it is known that the VGG16 fine-tuned model (with layer adjustment) has the lowest computational complexity followed by AlexNet. Simpler models tend to produce lower computational complexity. In addition, the application of the freezing layer to the VGG16 model makes a very large contribution to reducing the number of trainable parameters. The freezing layer cuts the number of trainable parameters by more than half. This will speed up the training time because the weights in the freezing layer will not be updated during training. By looking at the superiority of the number of parameters, it can be said that the VGG16 fine-tuned model is a lighter model even though there is an increase in the number of layers compared to the VGG16 model.

Table 4. Number of Parameters in Each Model

	VGG16 fine-tuned	**VGGNet**	**AlexNet**	**ResNet34**
Total Params	14,984,005	194,312,261	24,748,805	160,858.117
Trainable params	7,346,693	194,311,365	24,748,101	160,850,437
Non-trainable params	7,637,312	896	704	7,680

CONCLUSIONS

The results showed that the VGG16 model with fine-tuning gave the best performance, with a test accuracy reaching 99% and superior to the VGGNet, AlexNet and ResNet34 models. The use of transfer learning from ImageNet training weights resulted in a more adaptive model for application in detecting driving distractions. The early freezing method ensures an increase in the model's ability to recognize complex spatial features of the problem domain and maintains the general feature recognition capability of VGG16. In addition, the freezing layer also reduces computational complexity because it reduces the number of trainable parameters. The use of the decay learning rate method during the learning process results in an increase in accuracy and a smoother decrease in loss during the learning process. To improve the model's ability to recognize new features that have never been learned before, dropout is used during the training process.

It should also be noted that the fine tuning conducted in this study is empirical. The focus of the research will be increased on more in-depth fine tuning, especially to explore the ability of each layer to recognize features. Optimization methods can be developed for fine tuning freezing layers and other hyperparameters.

ACKNOWLEDGEMENT

This research used datasets from Driver Behavior Detection | CNN, available on Kaggle, accessed on September 23, 2024 (https://www.kaggle.com/datasets/robinreni/revitsone-5class). It is an open dataset and available for the public in the purpose of ensuring transparency and reproducibility.

AUTHOR CONTRIBUTIONS

Conceptualization, S.A.S.M. and T.D.I.D.O.; methodology, S.A.S.M.; software, A.L.H. and T.D.I.D.O.; validation, A.S.K. and A.L.H.; formal analysis, S.A.S.M. and N.F.T.U.; investigation, S.A.S.M. and N.F.T.U.; resources, S.A.S.M. and T.D.I.D.O.; data curation, T.D.I.D.O. and S.A.S.M.; writing—original draft preparation, A.S.K., N.F.T.U., B.P. and S.A.S.M.; writing—review and editing, N.F.T.U., S.A.S.M., A.S.K., T.D.I.D.O. and A.L.H.; visualization, A.S.K. and N.F.T.U.; supervision, S.A.S.M. and B.P.; project administration, S.A.S.M.; funding acquisition, S.A.S.M.. All authors have read and agreed to the published version of the manuscript.

CONFLICT OF INTERESTS

The authors declare that there is no conflict of interests.

REFERENCES

[1] A. Kashevnik, R. Shchedrin, C. Kaiser, et al. Driver Distraction Detection Methods: A Literature Review and Framework, IEEE Access 9 (2021), 60063–60076. https://doi.org/10.1109/ACCESS.2021.3073599.

[2] E. Michelaraki, C. Katrakazas, S. Kaiser, et al. Real-Time Monitoring of Driver Distraction: State-of-the-Art and Future Insights, Accid. Anal. Prev. 192 (2023), 107241. https://doi.org/10.1016/j.aap.2023.107241.

[3] M.A. Regan, J.D. Lee, K. Young, eds., Driver Distraction: Theory, Effects, and Mitigation, CRC Press, 2009. https://doi.org/10.1201/9781420007497.

[4] M. Née, B. Contrand, L. Orriols, et al. Road Safety and Distraction, Results from a Responsibility Case-Control Study among a Sample of Road Users Interviewed at the Emergency Room, Accid. Anal. Prev. 122 (2019), 19–24. https://doi.org/10.1016/j.aap.2018.09.032.

[5] D.L. Strayer, J. Turrill, J.M. Cooper, et al. Assessing Cognitive Distraction in the Automobile, Human Factors 57 (2015), 1300–1324. https://doi.org/10.1177/0018720815575149.

[6] P.A. Hancock, M. Lesch, L. Simmons, The Distraction Effects of Phone Use during a Crucial Driving Maneuver, Accid. Anal. Prev. 35 (2003), 501–514. https://doi.org/10.1016/S0001-4575(02)00028-3.

[7] Md.M. Haque, S. Washington, The Impact of Mobile Phone Distraction on the Braking Behaviour of Young Drivers: A Hazard-Based Duration Model, Transp. Res. Part C: Emerg. Technol. 50 (2015), 13–27. https://doi.org/10.1016/j.trc.2014.07.011.

[8] O. Oviedo-Trespalacios, M.M. Haque, M. King, et al. Understanding the Impacts of Mobile Phone Distraction on Driving Performance: A Systematic Review, Transp. Res. Part C: Emerg. Technol. 72 (2016), 360–380. https://doi.org/10.1016/j.trc.2016.10.006.

[9] A.J. Allen, S.A. Meda, P. Skudlarski, et al. Effects of Alcohol on Performance on a Distraction Task During Simulated Driving, Alcoholism: Clin. Exp. Res. 33 (2009), 617–625. https://doi.org/10.1111/j.1530-0277.2008.00876.x.

[10] E.L.R. Harrison, M.T. Fillmore, Alcohol and Distraction Interact to Impair Driving Performance, Drug Alcohol Depend. 117 (2011), 31–37. https://doi.org/10.1016/j.drugalcdep.2011.01.002.

[11] F. Alonso, C. Esteban, S.A. Useche, et al. Smoking while Driving: Frequency, Motives, Perceived Risk and Punishment, World J. Prev. Med. 5 (2017), 1-9.

[12] G. Yannis, E. Papadimitriou, C. Bairamis, et al. Is It Risky to Talk, Eat or Smoke While Driving? Findings from a Driving Simulator Experiment, in: Proceedings of the third International Conference on Road Safety and Simulation, Indianapolis, 2011.

[13] T. Chen, O. Oviedo-Trespalacios, N.N. Sze, S. Chen, Distractions by Work-Related Activities: The Impact of Ride-Hailing App and Radio System on Male Taxi Drivers, Accid. Anal. Prev. 178 (2022), 106849. https://doi.org/10.1016/j.aap.2022.106849.

[14] J.Y. Lee, J.D. Lee, J. Bärgman, et al. How Safe Is Tuning a Radio?: Using the Radio Tuning Task as a Benchmark for Distracted Driving, Accid. Anal. Prev. 110 (2018), 29–37. https://doi.org/10.1016/j.aap.2017.10.009.

[15] Y. Qi, R. Vennu, R. Pokhrel, Distracted Driving: A Literature Review, Illinois Center for Transportation, 2020. https://doi.org/10.36501/0197-9191/20-005.

[16] A.D. McDonald, T.K. Ferris, T.A. Wiener, Classification of Driver Distraction: A Comprehensive Analysis of Feature Generation, Machine Learning, and Input Measures, Human Factors 62 (2020), 1019–1035. https://doi.org/10.1177/0018720819856454.

[17] S. Ahangari, M. Jeihani, A. Ardeshiri, M.M. Rahman, A. Dehzangi, Enhancing the Performance of a Model to Predict Driving Distraction with the Random Forest Classifier, Transp. Res. Rec. 2675 (2021), 612–622. https://doi.org/10.1177/03611981211018695.

[18] T. Abbas, S.F. Ali, M.A. Mohammed, et al. Deep Learning Approach Based on Residual Neural Network and SVM Classifier for Driver's Distraction Detection, Appl. Sci. 12 (2022), 6626. https://doi.org/10.3390/app12136626.

[19] Y. Ma, G. Gu, B. Yin, et al. Support Vector Machines for the Identification of Real-Time Driving Distraction Using in-Vehicle Information Systems, J. Transp. Saf. Secur. 14 (2022), 232–255. https://doi.org/10.1080/19439962.2020.1774019.

[20] D. Ariansyah, R. Rahutomo, G.N. Elwirehardja, et al. AI-Based Video Analysis for Driver Fatigue Detection: A Literature Review on Underlying Datasets, Labelling, and Alertness Level Classification, in: S.C. Mukhopadhyay, S.M.N.A. Senanayake, P.W.C. Withana (Eds.), Innovative Technologies in Intelligent Systems and Industrial Applications, Springer Nature Switzerland, Cham, 2023: pp. 251–261. https://doi.org/10.1007/978-3-031-29078-7_22.

[21] Z. Wang, L. Yao, Recongnition of Distracted Driving Behavior Based on Improved Bi-LSTM Model and Attention Mechanism, IEEE Access 12 (2024), 67711–67725. https://doi.org/10.1109/ACCESS.2024.3399789.

[22] T.W. Cenggoro, F. Tanzil, A.H. Aslamiah, et al. Crowdsourcing Annotation System of Object Counting Dataset for Deep Learning Algorithm, IOP Conf. Ser.: Earth Environ. Sci. 195 (2018), 012063. https://doi.org/10.1088/1755-1315/195/1/012063.

[23] J.M.D. Bruxella, J. Kanimozhi, An Efficient FWA–RNN Algorithm for the Driver Distraction Classification, Malaya J. Mat. S (2021), 576–580.

[24] K. Srinivasan, L. Garg, D. Datta, et al. Performance Comparison of Deep CNN Models for Detecting Driver's Distraction, Comput. Mater. Continua 68 (2021), 4109–4124. https://doi.org/10.32604/cmc.2021.016736.

[25] H. Kang, C. Zhang, H. Jiang, Advancing Driver Behavior Recognition: An Intelligent Approach Utilizing ResNet, Autom. Control Comput. Sci. 58 (2024), 555–568. https://doi.org/10.3103/S0146411624700664.

[26] A. Budiarto, R. Rahutomo, H.N. Putra, et al. Unsupervised News Topic Modelling with Doc2Vec and Spherical Clustering, Procedia Comput. Sci. 179 (2021), 40–46. https://doi.org/10.1016/j.procs.2020.12.007.

[27] U. Narayanan, P. Prajith, R.T. Mathew, et al. Real Time Distracted Driver Detection Using Xception Architecture and Raspberry Pi, Intel. Artif. 28 (2024), 15–29. https://doi.org/10.4114/intartif.vol28iss75pp15-29.

[28] S. Anber, W. Alsaggaf, W. Shalash, A Hybrid Driver Fatigue and Distraction Detection Model Using AlexNet Based on Facial Features, Electronics 11 (2022), 285. https://doi.org/10.3390/electronics11020285.

[29] F. Sajid, A.R. Javed, A. Basharat, et al. An Efficient Deep Learning Framework for Distracted Driver Detection, IEEE Access 9 (2021), 169270–169280. https://doi.org/10.1109/ACCESS.2021.3138137.

[30] L. Goel, S. Chennamaneni, A. Golla, et al. Transfer Learning-Based Driver Distraction Detection, in: 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), IEEE, Erode, India, 2023: pp. 58–63. https://doi.org/10.1109/ICSCDS56580.2023.10104662.

[31] C.H. Praharsha, A. Poulose, CBAM VGG16: An Efficient Driver Distraction Classification Using CBAM Embedded VGG16 Architecture, Comput. Biol. Med. 180 (2024), 108945. https://doi.org/10.1016/j.compbiomed.2024.108945.

[32] C. Ou, C. Ouali, F. Karray, Transfer Learning Based Strategy for Improving Driver Distraction Recognition, in: A. Campilho, F. Karray, B. Ter Haar Romeny (Eds.), Image Analysis and Recognition, Springer, Cham, 2018: pp. 443–452. https://doi.org/10.1007/978-3-319-93000-8_50.

[33] V. Eswarakrishnan, P. Singla, An Intelligent System for Driving Distraction Identification by Fine-Tuned Modified Transfer Learning Techniques, in: 2024 5th International Conference on Smart Electronics and Communication (ICOSEC), IEEE, Trichy, India, 2024: pp. 2042–2046. https://doi.org/10.1109/ICOSEC61587.2024.10722137.

[34] B. Pardamean, T. Suparyanto, T.W. Cenggoro, et al. AI-Based Learning Style Prediction in Online Learning for Primary Education, IEEE Access 10 (2022), 35725–35735. https://doi.org/10.1109/ACCESS.2022.3160177.

[35] A.A. Sheikh, I.Z. Khan, Enhancing Road Safety: Real-Time Detection of Driver Distraction through Convolutional Neural Networks, arXiv:2405.17788 [cs.CV] (2024). https://doi.org/10.48550/ARXIV.2405.17788.

[36] B. Qin, J. Qian, Y. Xin, et al. Distracted Driver Detection Based on a CNN With Decreasing Filter Size, IEEE Trans. Intell. Transp. Syst. 23 (2022), 6922–6933. https://doi.org/10.1109/TITS.2021.3063521.

[37] A. Ezzouhri, Z. Charouh, M. Ghogho, et al. Robust Deep Learning-Based Driver Distraction Detection and Classification, IEEE Access 9 (2021), 168080–168092. https://doi.org/10.1109/ACCESS.2021.3133797.

[38] T.W. Cenggoro, A. Budiarto, R. Rahutomo, B. Pardamean, Information System Design for Deep Learning Based Plant Counting Automation, in: 2018 Indonesian Association for Pattern Recognition International Conference (INAPR), IEEE, Jakarta, Indonesia, 2018: pp. 329–332. https://doi.org/10.1109/INAPR.2018.8627019.

[39] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556 [cs.CV] (2014). https://doi.org/10.48550/ARXIV.1409.1556.

[40] H. Soeparno, A.S. Perbangsa, B. Pardamean, Best Practices of Agricultural Information System in the Context of Knowledge and Innovation, in: 2018 International Conference on Information Management and Technology (ICIMTech), IEEE, Jakarta, 2018: pp. 489–494. https://doi.org/10.1109/ICIMTech.2018.8528187.

[41] J.W. Baurley, A. Budiarto, M.F. Kacamarga, et al. A Web Portal for Rice Crop Improvements, Int. J. Web Portals 10 (2018), 15–31. https://doi.org/10.4018/IJWP.2018070102.

[42] M.F. Kacamarga, B. Pardamean, H. Wijaya, Lightweight Virtualization in Cloud Computing for Research, in: R. Intan, C.-H. Chi, H.N. Palit, L.W. Santoso (Eds.), Intelligence in the Era of Big Data, Springer, Berlin, Heidelberg, 2015: pp. 439–445. https://doi.org/10.1007/978-3-662-46742-8_40.

[43]K.W. Church, Z. Chen, Y. Ma, Emerging Trends: A Gentle Introduction to Fine-Tuning, Nat. Lang. Eng. 27 (2021), 763–778. https://doi.org/10.1017/S1351324921000322.

[44] M. Heydarian, T.E. Doyle, R. Samavi, MLCM: Multi-Label Confusion Matrix, IEEE Access 10 (2022), 19083–19095. https://doi.org/10.1109/ACCESS.2022.3151048.

[45] R.E. Caraka, M. Tahmid, R.M. Putra, et al. Analysis of Plant Pattern Using Water Balance and Cimogram Based on Oldeman Climate Type, IOP Conf. Ser.: Earth Environ. Sci. 195 (2018), 012001. https://doi.org/10.1088/1755-1315/195/1/012001.

[46] B. Pardamean, H. Soeparno, A. Budiarto, et al. Quantified Self-Using Consumer Wearable Device: Predicting Physical and Mental Health, Healthc. Inform. Res. 26 (2020), 83–92. https://doi.org/10.4258/hir.2020.26.2.83.

[47] M. Momeni, Driver Behavior Detection | CNN, 2023. https://www.kaggle.com/code/imtkaggleteam/driver-behavior-detection-cnn/notebook.