IAES International Journal of Artificial Intelligence (IJ-AI)

IAES International Journal of Artificial Intelligence (IJ-AI), ISSN/e-ISSN 2089-4872/2252-8938 publishes articles in the field of artificial intelligence (AI). The scope covers all artificial intelligence (AI) and machine learning (ML) areas and theirs applications in the following topics: neural networks; fuzzy logic; simulated biological evolution algorithms (like genetic algorithm, ant colony optimization, etc); reasoning and evolution; intelligence applications; computer vision and speech understanding; multimedia and cognitive informatics, data mining and machine learning tools, heuristic and AI planning strategies and tools, computational theories of learning; technology and computing (like particle swarm optimization); intelligent system architectures; knowledge representation; bioinformatics; natural language processing; multiagent systems; supervised learning; unsupervised learning; deep learning; big data and AI approaches; reinforcement learning; and learning with generative adversarial networks; etc. This journal is indexed in Scopus and all published papers since 2018 issues were included in scopus.com.

Focus and Scope

The IAES International Journal of Artificial Intelligence (IJ-AI), ISSN/e-ISSN 2089-4872/2252-8938 covers all topics of artificial intelligence and soft computing and their applications, including but not limited to:

- Neural networks
- Reasoning and evolution
- Intelligent search
- Intelligent planning
- Intelligence applications
- Computer vision and speech understanding
- Multimedia and cognitive informatics
- Data mining and machine learning tools, heuristic and AI planning strategies and tools, computational theories of learning
- Technology and computing (like particle swarm optimization); intelligent system architectures
- Knowledge representation
- Bioinformatics
- Natural language processing
- Automated reasoning
- Logic programming
- Machine learning
- Visual/linguistic perception
- Evolutionary and swarm algorithms
- Derivative-free optimisation algorithms
- Fuzzy sets and logic
- Rough sets
- Simulated biological evolution algorithms (like genetic algorithm, ant colony optimization, etc)
- Multi-agent systems
- Data and web mining
- Emotional intelligence
- Hybridisation of intelligent models/algorithms
- Parallel and distributed realisation ofintelligent algorithms/systems
- Application in pattern recognition, image understanding, control, robotics and bioinformatics
- Application in system design, system identification, prediction, scheduling and game playing
- Application in VLSI algorithms and mobile communication/computing systems

Principal Contact
Prof. Dr. Eugene Yu-Dong Zhang
Editor-in-Chief, IJ-AI
Chair in Knowledge Discovery and Machine Learning
Associate Fellow of Higher Education Academy
IEEE Senior Member
ACM Senior Member

Contact
F26 Informatics Building
Department of Informatics
University of Leicester, University Road,
Leicester, LE1 7RH, UK
Email: ijai@iaesjournal.com

IAES International Journal of Artificial Intelligence (IJ-AI)

Editorial Team

IAES International Journal of Artificial Intelligence (IJ-AI)

Vol 11, No 2: June 2022

Table of Contents

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 110~120
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i1.pp110-120

Deep convolutional neural networks architecture for an efficient emergency vehicle classification in real-time traffic monitoring

Amine Kherraki, Rajae El Ouazzani
Image Laboratory, School of Technology, Moulay Ismail University of Meknes, Meknes, Morocco

Article Info

Article history:

Received Jun 12, 2021
Revised Dec 14, 2021
Accepted Dec 28, 2021

Keywords:

Convolution neural network
Deep learning
Emergency vehicle
Image classification
Image processing

ABSTRACT

Nowadays, intelligent transportation system (ITS) has become one of the most popular subjects of scientific research. ITS provides innovative services to traffic monitoring. The classification of emergency vehicles in traffic surveillance cameras provides an early warning to ensure a rapid reaction in emergency events. Computer vision technology, including deep learning, has many advantages for traffic monitoring. For instance, convolutional neural network (CNN) has given very good results and optimal performance in computer vision tasks, such as the classification of vehicles according to their types, and brands. In this paper, we will classify emergency vehicles from the output of a closed-circuit television (CCTV) camera. Among the advantages of this research paper is providing detailed information on the emergency vehicle classification topic. Emergency vehicles have the highest priority on the road and finding the best emergency vehicle classification model in real-time will undoubtedly save lives. Thus, we have used eight CNN architectures and compared their performances on the Analytics Vidhya Emergency Vehicle dataset. The experiments show that the utilization of DenseNet121 gives excellent classification results which makes it the most suitable architecture for this research topic, besides, DenseNet121 does not require a high memory size which makes it appropriate for real-time applications.

Corresponding Author:

Amine Kherraki
Image Laboratory, School of Technology, Moulay Ismail University of Meknes
Marjane, Meknes 50050, Morocco
Email: amine.kherraki.9@gmail.com

1. INTRODUCTION

In recent years, intelligent transportation system (ITS) has gained much importance, due to the vast increase in the number of cars, and other types of vehicle on the road [1]. Smart systems contain a large amount of high-quality information for a smart and secure use [2]. For example, the ITS provides multiple services such as transport and traffic monitoring. The aim of this monitoring is to acquire and analyze vehicle movements, provide accurate data, and important statistics about the type and shape of the vehicle, as well as the evaluation of road traffic and safety [1], [3]. Thus, transport and traffic monitoring comes to facilitate the human labor, by vision-based tasks that a computer or automated system can perform [4]. Such a system is essential for effective real-time traffic monitoring that are able to detect changes in traffic characteristics in a timely manner, allowing regulatory agencies and authorities to respond quickly to traffic situations.

In literature, many applications for traffic video surveillance make vehicle re-identification in a multi-camera environment [5]. These applications can report important information, such as traffic flow or traffic information, and travel time in a distributed traffic control system. According to [6], and with regard to object localization, the authors proposed an approach to find an instance of a vehicle by estimating its position with ratio and size, which is widely used for vehicle tracking. In [7], the authors used a fixed camera-based

application for traffic video analysis and a central processing unit (CPU) based system to count the number of passing vehicles along with their speed on a highway. The analysis of the results shows that the application will not only count the number of vehicles but also estimate the speeds of the vehicle by providing more traffic flow information. Deep learning approaches and algorithms such as convolutional neural network (CNN) are widely used in many areas, applications, and issues in computer vision like image recognition, segmentation, detection, and classification [8], [9]. In fact, vehicle detection methods must be fast enough to operate in real-time, and to be immune to changes in lighting and different weather conditions, and have the ability to separate vehicles from image sequences accurately and efficiently [10]. Image-based vehicle classification is one of the most promising techniques for large-scale traffic data collection and analysis [11], and deep learning algorithms are widely used in this topic. For example, the authors in [12] have used CNN architectures to classify road vehicle images into six categories, including large buses, cars, motorcycles, minibuses, trucks, and vans. They show that using CNN for vehicle type classification provides the most recent results on previously cropped images containing only vehicles [12].

In this paper, we will discuss an important topic to which researchers had not given much importance before, knowing that emergency vehicles have the top priority on the road. Our contribution consists of making a comparison between the results of classification of emergency vehicles by using many CNN architectures. Therefore, our paper would help researchers and developers in the implementation of some efficient real-time emergency vehicle classification applications. The rest of the paper is organized as; section 2 presents related work on the classification of vehicles in general, as well as the classification of emergency vehicles. Section 3 contains the definition of CNN, with a detail of each architecture. Section 4 provides details about the research method, especially the dataset images preprocessing, and the implementation of CNNs. Section 5 presents, the experimental results, and performance analysis discussion. Finally, section 6 contains concluding observations and future work, with guidance in this area.

2. RELATED WORKS

2.1. Vehicle classification

Vehicle classification is a promising research task in ITS, and deep learning algorithms are the most used techniques for this task. Indeed, CNN has performed well in this area, in terms of real-time speed and accuracy compared to other machine learning algorithms, such as support vector machine (SVM), decision tree (DT). According to [13], in 2015, the authors proposed a semi-supervised CNN for vehicle detection from frontal view images. The authors used Laplacian filter learning to obtain the filters of the network on a large amount of unlabeled data. Besides, they used a Softmax classifier in the output layer, and they trained their model on a small amount of labeled data. A year earlier, the authors in [14] have introduced an unattended CNN for vehicle classification. They used CNN to learn characteristics of vehicles and then classify them by using Softmax regression. The proposed network filters are learned with a sparse filtering method. In [15], the authors used CNN with low-resolution video images to detect and classify vehicles. The preprocessing operation includes resizing each image and adjusting the contrast with histogram equalization. The proposed CNN architecture detects higher-level features such as edges and corners. In addition, the authors vary the number of filters and their sizes as well as the number of hidden layers [15]. After that, the authors in [16], show that, a simple CNN surpasses scale-invariant feature transform (SIFT) and support vector machine (SVM) models applied on vehicle classification. Finally in [17], [18], the authors used you only look once (YOLO) for vehicle detection and AlexNet model for vehicle classification. Further, the authors used AlexNet as an entity extractor and performed the classification of extracted entities by using a linear SVM.

2.2. Emergency vehicle classification

In this subsection, we will present some relevant work in emergency vehicle classification. According to [19], the authors used color segmentation, which adopted the hue saturation value (HSV) and red green blue (RGB) color models to characterize the emergency vehicle siren light. Subsequently, they took the light as a detection function for the identification and classification of emergency vehicles in the input video. Finally, they used SVM classifier to perform the vehicle classification. Basically, the authors cut the image horizontally so that they can detect the siren light in the highest part. However, sometimes the ambulances are not provided with a siren. In our opinion, to train correctly a model, we need a dataset containing images of multiple angles and positions with and without light and siren. In the real world, when we put sensors in the car, we will not be able to see the complete image of the car, because the angle of view or the visual field of the cameras is very limited to the places where it is necessary to put the sensors. Thus, we can say that it is impossible to make the decision through these criteria, because it is rare to see lights and sirens in emergency vehicle images.

Later in [20], neural network (NN) and structural traits are used to train and recognize ambulance characters in emergency vehicles. The authors explore the photography processing machine through "Thinning" and "Hilditch" algorithms, and the structural features of a character in the picture. In addition, they

diagnosed some features of the widespread language while, the implemented approach is based on the detection of the characters “AMBULANCE” or “108" on the vehicle. From our point of view, we find their idea not very practical, and sometimes there are other characters or just symbols like the moon, the cross, or another language outright on the ambulance.

3. CONVOLUTION NEURAL NETWORK

In recent years, neural networks (NNs) have become one of the most widely used machine learning algorithms. The latter have been proven decisively over time that they outperform other traditional algorithms in terms of accuracy and speed. According to [21], CNNs are a version of artificial neural networks (ANNs), and they correspond to the patterns of connectivity found in the visual cortex of animals [22]. Mathematically, these models of connectivity are described by a process of convolution. CNN is an improvised variant of the multilayer perceptron, typically it is composed of an input layer, an output layer, and many hidden layers [23]. CNNs are primarily used for computer vision tasks, such as facial recognition, image classification, identification and detection, and image processing in the field of autonomous vehicles.

As shown in Figure 1, a CNN is made up of two main parts. The first one is feature extraction (feature learning), where the network will perform the convolution and grouping operations so that it can detect the features [24]. The second one is classification, where the fully connected layers will serve as a classifier on the top of these extracted features, thus, they will assign a predicted probability about the object in the image [24]. CNNs are composed of four types of layers, including convolutional, pooling, rectified linear unit (ReLU), and fully connected (FC) [25]. In regard to convolutional layers, an input image is analyzed by a set of filters that produces a feature map. This output is then sent to a grouping layer to reduce the size of feature map, thus, it reduces the processing time by mapping the feature map on the most important information [25]. The convolution and grouping processes are repeated multiple times, and these repetitions depend on the CNN architecture, later, the outputs of the condensed feature map are sent to a series of FC layers. The latter flatten the maps together and check the probabilities of each feature occurring with the others to make the best classification [26], [27]. The ReLU layer is all about adding nonlinearity to a system because the convolution performs linear operations. It is simply a multiplication and a summation by element [28]. In the following, we will highlight some famous CNN architectures.

- VGG16: VGG16 is a popular choice when extracting CNN features from images. It has 16 layers, with 13 convolutional layers separated by 5 Max Pooling layers. In addition, it has 3 FC layers with 4096 neurons for each. The final dense layer is equal to the number of classes used for the final classification [26].
- VGG19: It has almost the same principle as VGG16, except that VGG19 contains 19 convolutional layers in total. It consists of 16 convolutional layers separated by 5 Max Pooling layers, 3 FC layers at the end, 4096 neurons for the first 2 FCs, and 1000 neurons for the last FC. Finally, a dense layer which is equal to the number of classes is used for the final classification [29].
- ResNet-50: Created in 2015, ResNet-50 introduced the idea of residual learning in order to make deeper CNNs. The input of the convolutional layer is replicated and added to the output of this layer, after this process the network recognizes practically learn residuals. ResNet-50 consists of 49 convolutional layers, with a Pooling layer, an Average Pooling layer, and an FC layer with 1000 neurons. However, the ResNet-50 benefit in terms of accuracy is related to the execution time and memory requirements [30].
- Inception-V3 and Xception: The main contribution of these architectures is that they combine many different convolution filters, for example, Conv (1×1), Conv (3×3), and Conv (5×5) in a multi-extractor [31]. The Inception-V3 architecture typically consists of 22 convolutional layers, with 5 pooling layers. Its variety Inception-V3 is too demanding in terms of memory, for this reason, a more optimized variant of the creation family has been proposed, it is called Xception where separable convolutions have been proposed in an attempt to reduce computational complexity [32].
- MobileNetV2: Using the idea of separable packaging, the MobileNetV2 model family achieves optimal performance at a low computational cost. The MobileNetV2 model consists of 27 layers, 13 deep Conv (3×3), 13 Conv (1×1), 1 Conv (3×3), with an Average Pooling layer and an FC layer. The MobileNetV2 architecture is distinguished by its low requirements in terms of parameters and memory [33].
- Inception-ResNet-V2: Inception-ResNet-V2 is a CNN with 164 deep layers, it combines the Inception architecture with residual connections. Inception-ResNet-V2 is a variant of Inception-V3, and it consists of 155 Convolution layers, 3 Average Pooling Layers, 4 Max Pooling layers, and 2 FC layers [34].
- DenseNet121: Densely networks involve the use of connection hopping in a different way. DenseNet121 is a model generated with 121 layers, 120 convolution layers including Conv (1, 1), Conv (3, 3), Conv (7, 7) in different blocks of Dense, separated by transition layers and a Pooling layer. The strong

point of DenseNet121 is that it outperforms the majority of CNN architectures in terms of accuracy, and does not require large memory [35], [36].

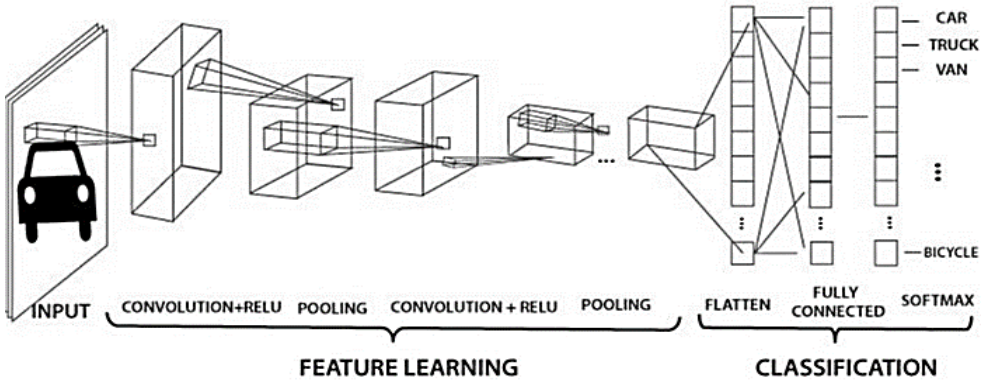


Figure 1. The Convolutional neural network architecture [23]

4. RESEARCH METHOD

In this section, we provide information about the general setup, and the dataset that we used in our experiments. Next, we present detailed information about the building and training of the CNN network. Figure 2 shows our workflow for classifying emergency vehicles.

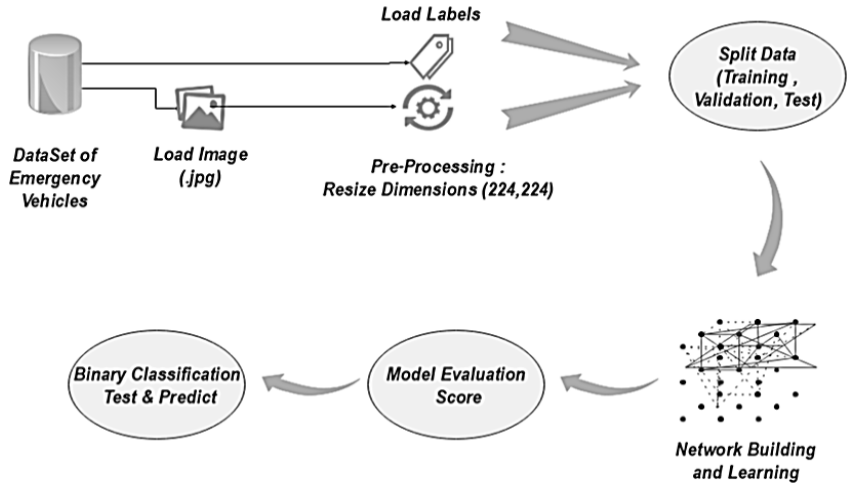


Figure 2. The workflow of the proposed method to classify emergency vehicles

4.1. Dataset and image pre-processing

For the experiments, we use a large-scale dataset named analytics vidhya emergency vehicle [37]. It contains 2352 vehicle images of different dimensions, 1361 images for normal vehicles and 991 for emergency vehicles such as police cars, ambulances, and fire brigades. Figure 3 shows some vehicle images taken in different angles and positions. All the experiments were implemented in Python language using the Keras and TenserFlow libraries on Kaggle simulator. Besides, we have used a laptop with an Intel i7-6500 Hq processor, a 2.5 GHz processor and 8 GB of memory. We have implemented several CNN architectures, including DenseNet121, MobileNetV2, Inception-ResNet-V2, Inception-V3, VGG19, VGG16, ResNet-50 and Xception to classify emergency vehicle images, and it takes about 20 hours for all models. We have performed a preprocessing on the images by resizing them to (224×224) as shown in Figure 4. After that, we randomly divided the data into two parts using the "sklearn" library. We have used 70% for training with validation, and 30% for test.

Figure 3. Examples of emergency and normal vehicle images from analytics vidhya emergency vehicle dataset [35]

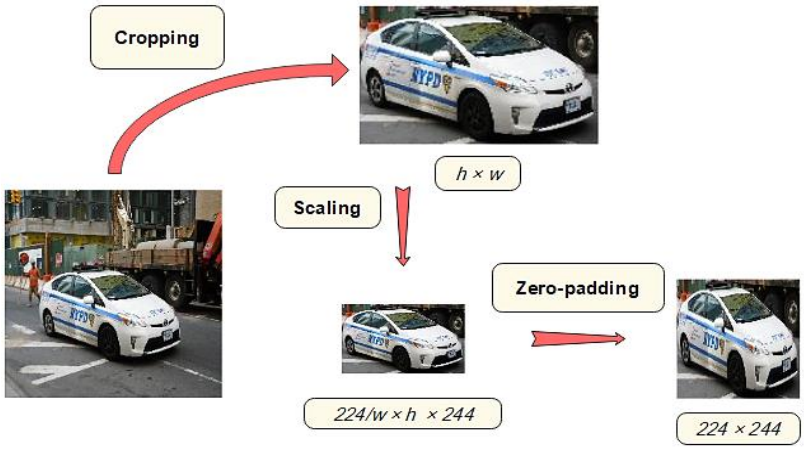


Figure 4. The workflow of image pre-processing

4.2. Network building and learning

After unifying the size of images, we have implemented eight CNN architectures, including DenseNet121, MobileNetV2, Inception-ResNet-V2, Inception-V3, VGG19, VGG16, ResNet-50, and Xception which are already predefined in the Keras. Next, we provide some parameters, including the input size with the dimensions (224×224×3) for each CNN. We have added three other layers to each of the eight implemented CNN architectures. The first additional layer is the "GlobalAveragePooling2D". It calculates the average output of each feature map in the previous layer, and it also reduces the dimensions of the input image, which accelerates the training duration. Besides, it prepares the model for the final classification layer. The second layer is "Dropout", which is proposed in 2014, and it is a regularization technique for neural network models. According to [38], deep neural networks with a large number of parameters are very powerful deep learning systems. However, overfitting is a critical problem in the training of such networks. Large networks are also slow to use, making them difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Thus, Dropout comes to address this problem by randomly dropping some units from the neural network during the training process. This operation prevents units from co-updating too much and remarkably reduces overfitting and gives great improvements over other regularization functions such as L1 and L2 regularization techniques. The third and the last additional layer is "Dense", where it is necessary to put the number of output classes. And as long as we have performed a binary classification by using two classes for normal and emergency vehicles, thus we applied "sigmoid" activation function. However, if the number of classes exceeds two, we will use the "Softmax" function. Once the three layers are added to the implemented CNN architectures, we use "Adam" as the adaptive optimizer of the learning rate. Figure 5 displays the summary of DenseNet121 architecture with the three added layers, and Figure 6 shows the detailed DenseNet121 architecture [39]. We have built the eight CNN architectures that we have already mentioned in

the previous section. We have trained each architecture on 500 epochs, and we have used the accuracy, F1, and loss metrics to evaluate the performances of our architectures.

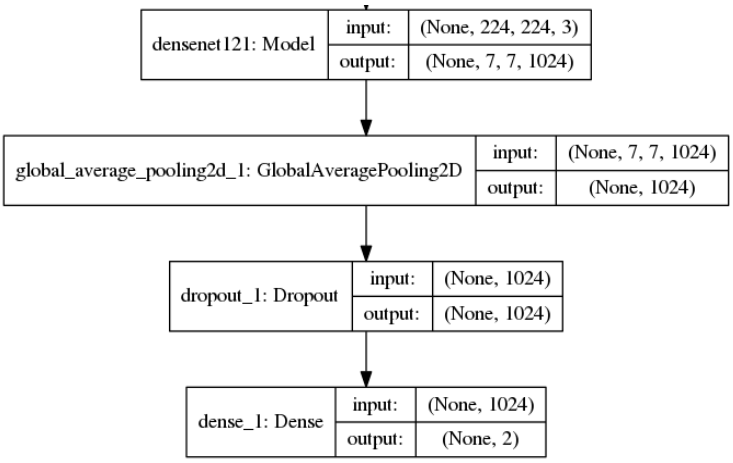


Figure 5. DenseNet121 building model summary with the three added layers

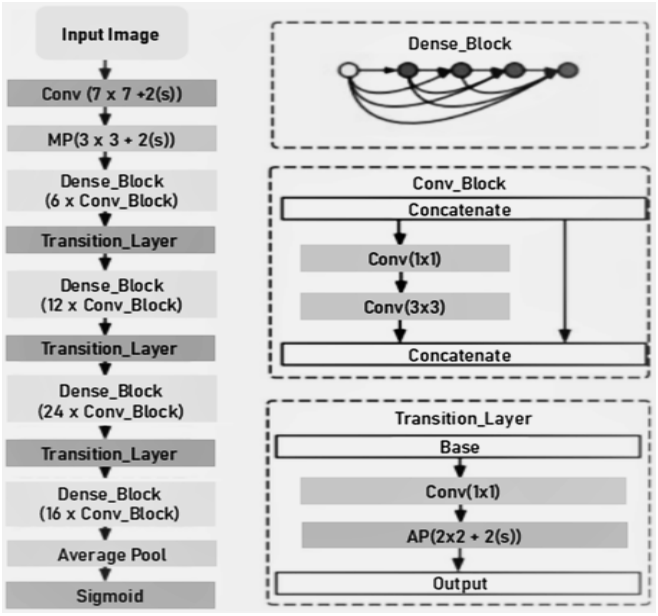


Figure 6. The used DenseNet121 architecture [37]

5. RESULTS AND DISCUSSION

In this section, we will discuss the experimental evaluation of the eight implemented CNN models that are summarized in Table 1. First, in terms of accuracy, DenseNet121 outperforms all of the other architectures with an accuracy score of 95.14%, closely followed by VGG16 with 92.3%. Simultaneously, for the F1 metric, we observe that the DenseNet121 still outperforms them, with a score of 93.87%, followed by VGG16 with 91.08%. The other models, MobileNetV2, ResNet-50, VGG19, Inception-ResNet-V2, Xception, and Inception-V3 had 91.90%, 91.90%, 91.49%, 91.09%, 90.68%, 88.25% respectively. In the end, they kept the same order of accuracy.

However, the VGG16 network needs very high computation and storage requirements, which does not always make them suitable for systems with limited resources and real-time usage. The MobileNetV2 model achieves an accuracy score of 91.9% and it does not need great requirements in terms of calculation and memory, thus, it is the most qualified model for real-time uses. Inception-ResNet-V2 offers an accuracy of 91.09%, but we observe that it surpasses all of the other models in terms of storage as well as time processing,

which makes it inappropriate for real-time applications. For the rest of the models, including ResNet-50, Xception, VGG19, and Inception-V3, they had 91.9%, 90.68%, 91.49%, and 88.25% respectively in accuracy, however, their storage and processing requirements are high.

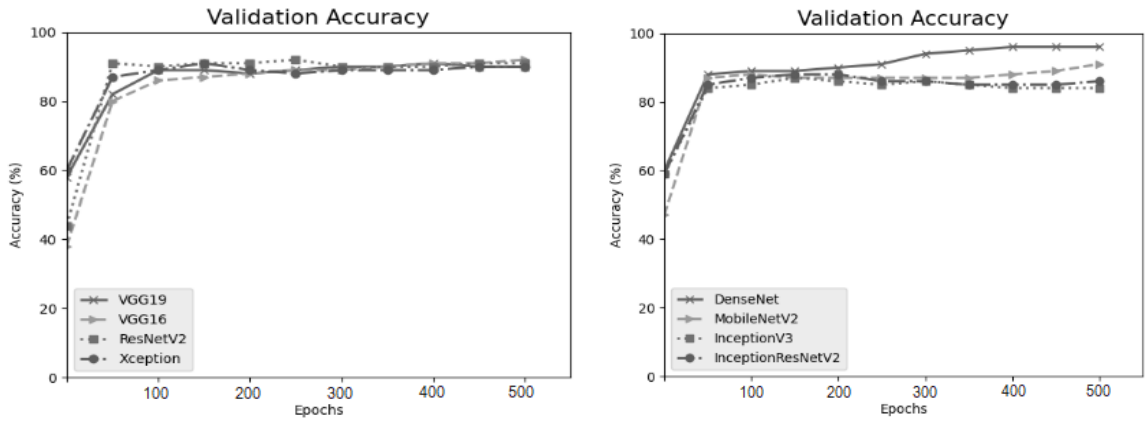
Table 1. Simulation results of CNN models

Models	Parameters (M)	Accuracy (%)	F1 (%)	Loss (%)	Memory (MB)
Inception-V3	21, 806,882	88.25	84.84	27.11	83.69
Inception-ResNet-V2	54, 339,810	91.09	88.78	26.57	208.45
MobileNetV2	**2, 260,546**	91.90	89.79	19.18	**8.92**
DenseNet121	7, 039,554	**95.14**	**93.87**	22.76	27.5
Xception	20, 865,578	90.68	89.21	23.86	79.87
ResNet-50	23, 591,810	91.90	89.58	**17.20**	90.33
VGG16	14, 715,714	92.30	91.08	17.86	56.19
VGG19	20, 025,410	91.49	89.34	25.05	76.45

In regards to the loss metric, we notice that ResNet-50 had the minimum loss score of 17.20%, closely followed by VGG16 with 17.86%. However, VGG19, Inception-V3, Inception-ResNet-V2 had high loss scores, compared to the other models. Thus, they require a big amount of memory, which does not always make them suitable for a real-time use. In terms of parameters, we notice that the MobileNetV2 had the best result with the smallest number of required parameters. And the other models kept the same order as mentioned in the memory metric, because the number of parameters has a direct relationship with the size of the required memory, so the more parameters there are, the more memory there is. And this is the power of MobileNetV2 model which performs very well with a small number of parameters and low memory.

5.1. Result analysis

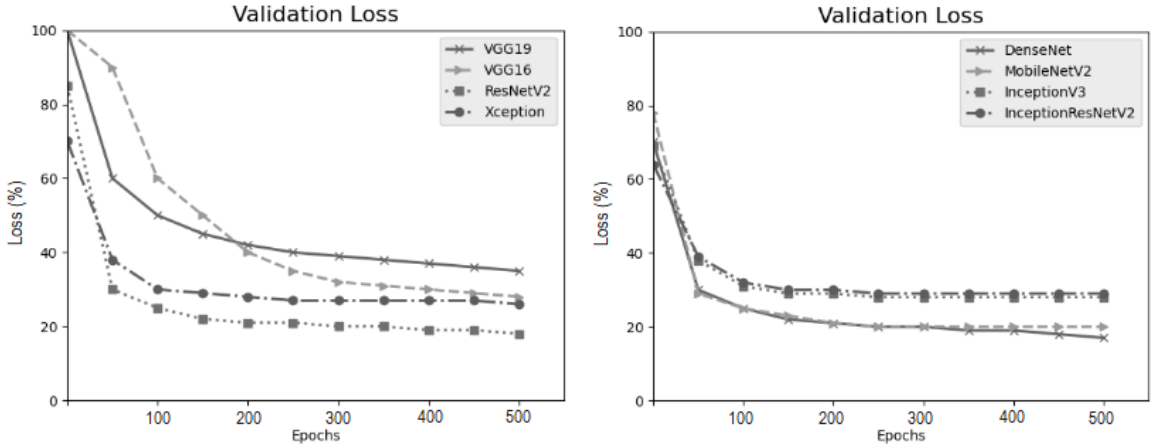
In this subsection, we will analyze reached results by using plots and confusion matrices. We drew plots for all the implemented models, and we have divided the plots into two parts as shown in Figure 7(a), Figure 7(b), Figure 8(a), and Figure 8(b). Part (a) of the two figures contains Xception, ResNet-50, VGG16, and VGG19, and part (b) contains DenseNet121, MobileNetV2, Inception-ResNet-V2, and Inception-V3. According to the curves of the whole models, we observe that ResNet-50 and MobileNetV2 exceeded all of the other models in terms of accuracy during the first 100 epochs, which means that they have made a good initialization as shown in Figure 7. However, DenseNet121 and VGG16, which had the best results, did not start learning well. We notice that the stability of the curves started after epoch 300 when the DenseNet121 was able to surpass all of the other models and it continues the progression, followed by VGG16. Regarding to the validation of the loss metric in Figure 8, we note a stability during the first 200 epochs. However, the loss curves of the DenseNet121 and VGG16 are clearly decreasing which means, that if we use more than 500 epochs, these last two models, will have good performances compared to the expected results of the other models.



(a) (b)

Figure 7. Accuracy of implemented CNN models; (a) Accuracy of Xception, ResNetV2, VGG16, and VGG19 CNN models; (b) Accuracy of InceptionV3, InceptionResNetV2, MobileNetV2, and DenseNet CNN models

Figure 9 shows the confusion matrices of the implemented CNN models. The confusion matrix draws some conclusions regarding the difficulty of the classification, and it also highlights some potential research opportunities. In the following, we will discuss the confusion matrix result of the DenseNet121 model. In total, we have used 247 images for validation, with 146 normal vehicles, and 101 emergency vehicles. In the "Normal Vehicle" class, we observe that DenseNet121 recognizes 143 images out of 146, in other words, it produced 143 correct predictions out of 146, with 97.94% of the entire normal vehicles. The analysis of misclassified images shows that missed vehicles contain advertisements, or have an abnormal shape compared to usual normal vehicles as shown in Figure 10. In "Emergency Vehicle" class, we observe that the correct prediction number is lower than that of "Normal Vehicle" class as illustrated in Figure 9, the DenseNet121 model produced 92 correct predictions out of 101, ie 91.08%. This is because some of the emergency vehicle images were taken from different distances, as well as different and difficult viewing angles. To avoid such cases, it is necessary to add several classes and other types of vehicle to the dataset.



(a) (b)

Figure 8. Loss of implemented CNN models; (a) Loss of Xception, ResNetV2, VGG16, and VGG19 CNN models; (b) Loss of InceptionV3, InceptionResNetV2, MobileNetV2, and DenseNet CNN models

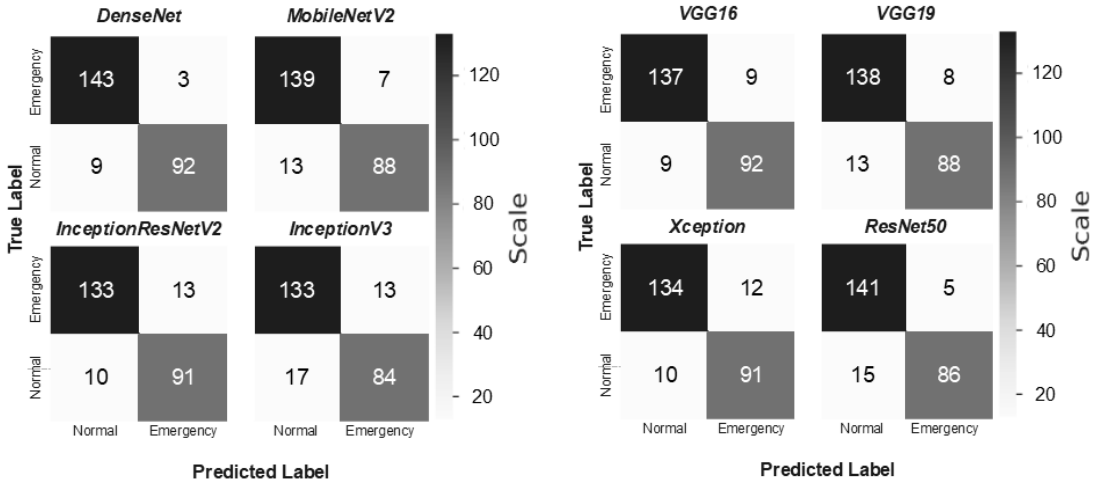


Figure 9. Confusion matrices of the implemented CNN models

5.2. Test and comparative results

The last part consists of testing all the implemented models on test images. After having trained and compiled each model on the dataset, we generated the evaluation scores for each model, we performed them on the test portion of the dataset using the TensorFlow and Keras prediction methods. Once we have a prediction, we use the Matplotlib library to display the image and its predicted class. We take as an example

the generated and trained DenseNet121 model, which had obtained the best score in terms of accuracy and F1 score. As shown in Figure 11, we have some examples of image classification into “Emergency Vehicle” and “Normal vehicle”. As we have already said, this is the first paper which makes the classification of emergency vehicle by using many CNN architectures.

There was a competition on the “Analytics Vidha platform” on this topic, where the competitor have used ResNet-18 and ResNet-34 to classify emergency vehicles on Analytics Vidhya Emergency Vehicle dataset [37]. The average of accuracy of the two models is 94.22% [40]. Details about the used architectures are not mentioned, the only available information are the name of two architectures and the utilization of pre-trained weight models from Keras. We mention that our models are trained from scratch which needs much time for training. In addition, we emphasize that our best model performances surpass those of previous work.

Figure 10. Examples of normal vehicle images with advertisements and abnormal shapes [35]

Figure 11. Results of predicted classes using DenseNet121 model

6. CONCLUSION

In this paper, we have made a comparative study using eight famous CNN architectures to classify emergency vehicle images. The design and implementation of an efficient deep learning system, have been carried out to automatically classify emergency and normal vehicles in traffic scenes. First, we did a pre-processing on the Vidhya Emergency Vehicle dataset to unify the image sizes. We have added three layers to each CNN architecture, in particular, “GlobalAveragePooling2D” layer to reduce the dimensions of the input image and accelerate the training, “Dropout” layer to avoid overfitting, and finally, “Dense” layer where we put the number of output classes. Later, we made simulations of each architecture, and we notice that DenseNet121 is the most appropriate model in real-time emergency vehicle classification with an accuracy of 95.14%, a F1 score of 93.87%, and an average order memory of 27.5 MB. Therefore, reached results are very promising and will definitely give an important added value to applications that will use our best architecture for the classification of emergency vehicles. The experiments allow us to sort the classification architectures based on different criteria like accuracy, as well as memory, thus, researchers and developers can choose the appropriate and suitable architecture for their applications. As a perspective, we plan to improve accuracy scores and time processing, we also plan to use a hybrid approach to classify emergency vehicles based on image and siren sound.

REFERENCES

[1] N. Buch, S. A. Velastin, and J. Orwell, “A review of computer vision techniques for the analysis of urban traffic,” *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 920–939, Sep. 2011, doi: 10.1109/TITS.2011.2119372.

[2] "DIRECTIVE 2010/40/EU OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 7 July 2010 on the framework for the deployment of Intelligent Transport Systems in the field of road transport and for interfaces with other modes of transport (Text with EEA relevance," 2010. Accessed: Feb. 19, 2021. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32010L0040&qid=1613756510712&from=EN.
[3] M. M. Hasan, G. Saha, A. Hoque, and M. B. Majumder, "Smart traffic control system with application of image processing techniques," 2014, doi: 10.1109/ICIEV.2014.6850751.
[4] A. Arinaldi, J. A. Pradana, and A. A. Gurusinga, "Detection and classification of vehicles for traffic video analytics," in *Procedia Computer Science*, Jan. 2018, vol. 144, pp. 259–268, doi: 10.1016/j.procs.2018.10.527.
[5] D. Zapletal and A. Herout, "Vehicle re-identification for automatic video traffic surveillance," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Dec. 2016, pp. 1568–1574, doi: 10.1109/CVPRW.2016.195.
[6] M. Ozuysal, V. Lepetit, and P. Fua, "Pose estimation for category specific multiview object localization," Mar. 2010, pp. 778–785, doi: 10.1109/cvpr.2009.5206633.
[7] S. H. Kim, J. Shi, A. Alfarrarjeh, D. Xu, Y. Tan, and C. Shahabi, "Real-time traffic video analysis using intel viewmont coprocessor," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7813 LNCS, pp. 150–160, doi: 10.1007/978-3-642-37134-9_12.
[8] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Convolutional neural networks for large-scale remote-sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 2, pp. 645–657, Feb. 2017, doi: 10.1109/TGRS.2016.2612821.
[9] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When deep learning meets metric learning: remote sensing image scene classification via learning discriminative CNNs," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 5, pp. 2811–2821, May 2018, doi: 10.1109/TGRS.2017.2783902.
[10] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos, "Detection and classification of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 37–47, Mar. 2002, doi: 10.1109/6979.994794.
[11] L. W. Tsai, J. W. Hsieh, and K. C. Fan, "Vehicle detection using normalized color and edge map," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 850–864, Mar. 2007, doi: 10.1109/TIP.2007.891147.
[12] L. Zhuo, L. Jiang, Z. Zhu, J. Li, J. Zhang, and H. Long, "Vehicle classification for large-scale traffic surveillance videos using Convolutional Neural Networks," *Mach. Vis. Appl.*, vol. 28, no. 7, pp. 793–802, Oct. 2017, doi: 10.1007/s00138-017-0846-2.
[13] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle Type Classification using a semisupervised convolutional neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2247–2256, Aug. 2015, doi: 10.1109/TITS.2015.2402438.
[14] Z. Dong, M. Pei, Y. He, T. Liu, Y. Dong, and Y. Jia, "Vehicle type classification using unsupervised convolutional neural network," in *Proceedings - International Conference on Pattern Recognition*, Dec. 2014, pp. 172–177, doi: 10.1109/ICPR.2014.39.
[15] C. M. Bautista, C. A. Dy, M. I. Mañalac, R. A. Orbe, and M. Cordel, "Convolutional neural network for vehicle detection in low resolution traffic videos," in *Proceedings - 2016 IEEE Region 10 Symposium, TENSYMP 2016*, Jul. 2016, pp. 277–281, doi: 10.1109/TENCONSpring.2016.7519418.
[16] H. Huttunen, F. S. Yancheshmeh, and C. Ke, "Car type recognition with deep neural networks," in *IEEE Intelligent Vehicles Symposium, Proceedings*, Aug. 2016, vol. 2016-August, pp. 1115–1120, doi: 10.1109/IVS.2016.7535529.
[17] Y. Zhou, H. Nejati, T. T. Do, N. M. Cheung, and L. Cheah, "Image-based vehicle analysis using deep neural network: A systematic study," in *International Conference on Digital Signal Processing, DSP*, Jul. 2016, vol. 0, pp. 276–280, doi: 10.1109/ICDSP.2016.7868561.
[18] X. Li and X. Guo, "A HOG feature and SVM based method for forward vehicle detection with single camera," in *Proceedings - 2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2013*, 2013, vol. 1, pp. 263–266, doi: 10.1109/IHMSC.2013.69.
[19] H. Razalli, R. Ramli, and M. H. Alkawaz, "Emergency vehicle recognition and classification method using HSV color segmentation," in *Proceedings - 2020 16th IEEE International Colloquium on Signal Processing and its Applications, CSPA 2020*, Feb. 2020, pp. 284–289, doi: 10.1109/CSPA48992.2020.9068695.
[20] P. Gowtham, P. Eswari, and V. P. Arunachalam, "An Investigation approach used for pattern classification and recognition of an emergency vehicle," Dec. 2018, doi: 10.1109/ICSNS.2018.8573610.
[21] "Convolutional Neural Networks (LeNet) - DeepLearning 0.1 Documentation | 2 D Computer Graphics | Cognitive Science." https://www.scribd.com/document/333051443/Convolutional-Neural-Networks-LeNet-DeepLearning-0-1-Documentation (accessed Feb. 19, 2021).
[22] M. Boukabous and M. Azizi, "Review of learning-based techniques of sentiment analysis for security purposes," Springer, Cham, 2021, pp. 96–109.
[23] A. Moubayed, M. Injadat, A. B. Nassif, H. Lutfiyya, and A. Shami, "E-learning: challenges and research opportunities using machine learning data analytics," *IEEE Access*, vol. 6, pp. 39117–39138, Jul. 2018, doi: 10.1109/ACCESS.2018.2851790.
[24] "Traffic sign detection using convolutional neural network | by Sanket Doshi | Towards Data Science." https://towardsdatascience.com/traffic-sign-detection-using-convolutional-neural-network-660fb32fe90e (accessed Feb. 19, 2021).
[25] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018, doi: 10.1016/j.patcog.2017.10.013.
[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, Accessed: Feb. 19, 2021. [Online]. Available: http://www.robots.ox.ac.uk/.
[27] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, and H. El Fadili, "Toward a deep learning-based intrusion detection system for iot against botnet attacks," *IAES Int. J. Artif. Intell.*, vol. 10, no. 1, pp. 110–120, 2021, doi: 10.11591/ijai.v10.i1.pp110-120.
[28] C. Kyrkou and T. Theocharides, "EmergencyNet: Efficient aerial image classification for drone-based emergency monitoring using atrous convolutional feature fusion," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 1687–1699, 2020, doi: 10.1109/JSTARS.2020.2969809.
[29] G. G. Dario *et al.*, "On the behavior of convolutional nets for feature extraction," *J. Artif. Intell. Res.*, vol. 61, pp. 563–592, 2018, doi: 10.1613/jair.5756.
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[31] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Oct. 2015, vol. 07-12-June-2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.
[32] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Nov. 2017, vol. 2017-January, pp. 1800–1807, doi: 10.1109/CVPR.2017.195.
[33] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv*, 2017.
[34] M. Längkvist, L. Karlsson, and A. Loutfi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," *Pattern Recognit. Lett.*, vol. 42, no. 1, pp. 11–24, 2014, [Online]. Available: http://arxiv.org/abs/1512.00567.

[35] I. Allaouzi, M. Ben Ahmed, and B. Benamrou, "An Encoder-Decoder model for visual question answering in the medical domain," *CEUR Workshop Proc.*, vol. 2380, no. September 2019, pp. 9–12, 2019.
[36] C. Ye, C. Devaraj, M. Maynord, C. Fermüller, and Y. Aloimonos, "Evenly cascaded convolutional networks," *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*, pp. 4640–4647, 2019, doi: 10.1109/BigData.2018.8622196.
[37] "JanataHack_AV_ComputerVision | Kaggle." https://www.kaggle.com/shravankoninti/janatahack-av-computervision (accessed Feb. 19, 2021).
[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res. 15 1929-1958*, vol. 299, no. 3–4, pp. 345–350, 2014, doi: 10.1016/0370-2693(93)90272-J.
[39] Q. Ji, J. Huang, W. He, and Y. Sun, "Optimized deep convolutional neural networks for identification of macular diseases from optical coherence tomography images," *Algorithms*, vol. 12, no. 3, pp. 1–12, 2019, doi: 10.3390/a12030051.
[40] "Emergency Vehicle Classification - PyTorch, ResNet | Kaggle." https://www.kaggle.com/shravankoninti/emergency-vehicle-classification-pytorch-resnet (accessed Feb. 19, 2021).

BIOGRAPHIES OF AUTHORS

Amine Kherraki was born in Meknes, Morocco, in 1994. He received the B.S degree in School of Technology from Hassan First University at Berrechid, Morocco, in 2017. He received the M.S degree in Computer Science at the National School of Applied Science, Sidi Mohamed Ben Abdellah University, Fez, Morocco. Currently, He is Ph.D. candidate at Moulay Ismail University, Meknes, Morocco. His research interests include Deep Learning, Computer Vision, Business Intelligence, and Big Data. He can be contacted at email: amine.kherraki.9@gmail.com

Rajae El Ouazzani received her Master's degree in Computer Science and Telecommunication by the Mohammed V University of Rabat (Morocco) in 2006 and the Ph.D. in Image and Video Processing by the High National School of Computer Science and Systems Analysis (Morocco) in 2010. From 2011, she is a Professor in the High School of Technology of Meknes, Moulay Ismail University in Morocco. Since 2007, she is an author of several papers in international journals and conferences. Her domains of interest include multimedia data processing and telecommunications. She can be contacted at email: elouazzanirajae@gmail.com

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 121~128
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i1.pp121-128

A sound event detection based on hybrid convolution neural network and random forest

Muhamad Amirul Sadikin Md Afendi[1], Marina Yusoff[1,2]
[1]Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Al-Khawarizmi Complex, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia
[2]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia

Article Info

Article history:

Received Jun 26, 2021
Revised Dec 23, 2021
Accepted Jan 8, 2022

Keywords:

Convolution neural network
Random forest
Sound event detection
Support vector machine
Wildlife reserve conservation

ABSTRACT

Sound event detection (SED) assists in the detainment of intruders. In recent decades, several SED methods such as support vector machine (SVM), K-Means clustering, principal component analysis, and convolution neural network (CNN) on urban sound have been developed. Advanced work on SED in a rare sound event is challenging because it has limited exploration, especially for surveillance in a forest environment. This research provides an alternative method that uses informative features of sound event data from a natural forest environment and evaluates the CNN capabilities of the detection performances. A hybrid CNN and random forest (RF) are proposed to utilize a distinctive sound pattern. The feature extraction involves mel log energies. The detection processes include refinement parameters and post-processing threshold determination to reduce false alarms rate. The proposed CNN-RF and custom CNN-RF models have been validated with three types of sound events. The results of the suggested approach have been compared with well-regarded sound event algorithms. The experiment results demonstrate that the CNN-RF assesses the superiority with remarkable improvement in performance, up to a 0.82 F1 score with a minimum false alarms rate at 10%. The performance shows a functional advantage over previous methods.

Corresponding Author:

Marina Yusoff
Institute for Big Data Analytics and Artificial Intelligence (IBDAAI)
Al-Khawarizmi Complex, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia
Email: marina998@uitm.edu.my

1. INTRODUCTION

Sound event detection (SED) recognizes a sound event's presence using artificial intelligence methods. Applying SED in detecting intrusions for wildlife reserves seems appropriate as the sound of poachers' activities are very distinctive within the natural ambience. Previous works on SED have used various types of machine learning (ML) and deep learning (DL) algorithms such as random forest (RF) and convolutional neural network (CNN). SED solutions were reviewed on an artificial dataset for detecting and classifying acoustic scenes and events [1], [2]. The database comprised isolated sound events with unique sources of background recording, baby crying, gunshots, and glass breaking. The dataset was a mixture of noises and sound events. The researchers used CNN-long short-term memory (CRNN-LSTM) to obtain 93% F1 score [3]. CNN obtained 91% [4], multilayer layer perceptron (MLP-CNN) with 84% [5] and ensemble with 78% [2]. Although these experiments used artificial datasets, they have demonstrated how SED is feasible for industrial applications. The real-world environment sounds are far from noise-free [6]. The SED performances on noisy data can be rather challenging. Sound event's effective detection rate varies n recent

SED experiments and to a certain extent, can be biased to the context [7]. Many sounds overlap on frequency, making it more perplexing to conduct SED. The noises in a forest environment are unique due to the species of animals and plants within the vicinity.

In the SED application, features need to be extracted from the raw audio data to a piece of tailored information. This is done to distinguish sound events more effectively. A common feature used in recent SED studies is mel-log energies (MLE) features that are rich with essential values that contribute to class recognition. The methods explored by the previous research works frequently report that using MLE features with CNN produce good results [8], [9]. These features significantly support models' performance [10]. MLE features are an extraction process that includes fast Fourier transform (FFT). It separates sound frequency within a sound signal. Frequency separation helps in detection as each sound has a specific frequency range. The process would increase the feature determination among the sound events.

CNN is well known for its excellent detection of images [11], [12]. Therefore, CNN is believed to be reliable for SED. CNN uses a large amount of data to work best and overcome challenges such as overfitting, exploding gradient, and class imbalance in the training process [10], [13]. One solution created is by using data augmentation in an urban environment [14], [15]. However, limited research work has been done within the forest environment. The solutions are RF, distance-based [16] and DL methods [11], [12], [17], [18]. The overall performance of these solutions is less than 80% accuracy and has a high false alarms rate. Many solutions have been established mainly in an urban environment with a hybrid approach such as convolution recurrent neural network (CRNN) [19], LSTM-CNN, CNN-support vector machine (SVM) [20]. A study using ensemble methods obtained 85% accuracy on urban rare sound event detection [21]. Recent works on SED with hybrid approaches using CRNN with ensembles achieved 91% accuracy on the artificial and urban datasets [22]. A type of hybrid CNN acts as a feature extractor that improves the feature quality. A hybrid method of CNN and RF also has advantages. CNN-RF's attempts have improved accuracy than earlier methods on pattern recognition tasks on PlantCLEF 2019 dataset [23]. RF algorithm depends on good features like any other algorithms. Without CNN pre-processing high correlated features may affect performance [24]. Hence, the SED field requires more research to be conducted in obtaining the appropriate method of SED solution for surveillance in a forest environment. This research, then should act as an extension to increase the security performance to stop poachers and illegal activities in the forest. The contributions of this study are to provide an alternative solution for SED in the forest environment, to extract the most important features from sound events with a suitable method, to introduce an enhanced solution of hybrid CNN-RF, to provide post-processing thresholding with a minimum false alarm rate of 10%, to evaluate the proposed method using a real dataset and to compare the new solution with other machine learning methods.

This article is organized into several sections. Section 2 presents detailed research method that includes the proposed CNN-RF and post-processing thresholds. Section 3 presents the computational results, comparisons of custom CNN, pre-built CNN, and the relevant discussions. Finally, in section 4, the conclusion and future work are briefly highlighted.

2. PROPOSED MODEL

CNN model is used from a pre-trained model that has been proven to deliver performance for image recognition and the custom CNN produced by the study. The CNN model is based on the VGG16 architecture with the pre-trained weights [11], [12]. Many resources are required to optimized weights. Thus, the use of the trained weights can reduce training time. The customized CNN model used for feature extraction to compare performances of CNN developed from scratch. This is a transfer learning method. Figure 1 shows the flow of the process for the CNN-RF model.

The CNN part of the hybrid model acted as a feature extraction layer. The extracted features would be optimized to be the most useful for the sound classification. The RF performed the prediction from a CNN output. At the end of the process, we applied post-processing to optimize the results. The first step is a sound with a 5-second segment is based applied MLE extraction into MLE features. The CNN model used as a feature extraction method to obtain meaningful features from the MLE features. CNN output was used as the input for the RF model to compute the prediction. The RF ensemble size could be changed to find the best producing parameters. Once the best model was found, the post-processing was performed. The performances between models were compared to analyze the improvements of the hybrid method.

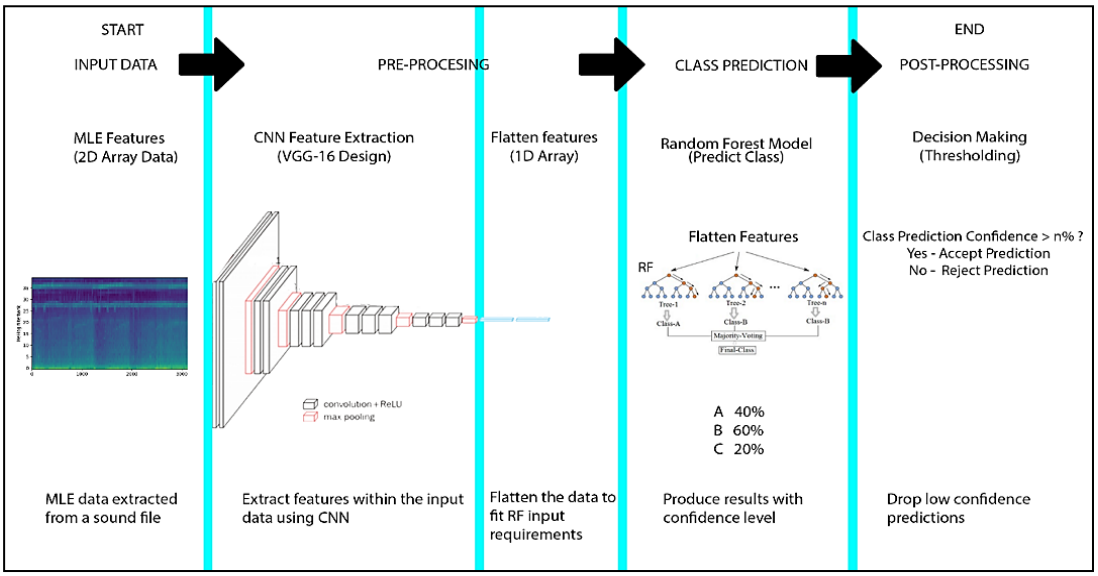


Figure 1. The process flow for the proposed CNN-RF model

3. RESEARCH METHOD

This section provides the explainations on the research steps taken. The steps included data collection, feature extraction, analysis of features, and the classification models of CNN, SVM, RF, and CNN-RF. The steps were done within the respective parameters and configurations.

3.1. Data acquisition

Wildlife reserve intrusion sound detection variables include illegal activities done by poachers. Poachers' equipment detained are mostly axes, machetes, and tools in the forest [25]. These sounds can be the solution to detect incoming threats. These sounds could be good indicators in detecting the presence of poachers in a reserve forest. Therefore, we performed data collection at Taman Negara Endau Rompin, Malaysia. Sound events were emulated to get the sound in the jungle with its current condition and environment. Each location had sound emulations of tree cutting, chainsaw, and vehicle activities. The distance from the sound source was the distance from the source emission point to the recording point in metres. It was scattered into three different distances, approximately at 30 m, 60 m, and 100 m in which were based on an average person's hearing capability [26].

3.2. Sound data feature extraction

MLE was selected as the feature extraction method for its high reliability on rare sound detection in past studies [10], [27], [28]. The steps involved were input signals, Hamming window, FFT, mel-scale filter bank and log. Hamming windows were recommended in this experiment for their properties for the frequency-selective analysis. Hamming windows and the corresponding spectrum form were adapted from [27], [28]. The cepstral features were computed by taking the FFT of the warped logarithmic spectrum. They contained each spectrum band's rate of change [27]. The pth filter bank utilized (1). This showed that f(p) was the middle frequency of the pth filter [29]. This work extended the heatmap representations of the illegal intrusion activities [9], [10].

$$Hm(k) = \begin{cases} 0, & k < f(m-1) \\ \dfrac{\mathrm{k} - \mathrm{f(m-1)}}{f(m) - f(m-1)}, & f(m-1) \le k \le f(m) \\ \dfrac{\mathrm{f(m+1)} - \mathrm{k}}{\mathrm{f(m+1)} - \mathrm{f(m)}}, & f(m) \le k \le f(m+1) \\ 0, & k > f(m+1) \end{cases} \quad (1)$$

3.3. Data preparation

For data preparation, the best practices found in recent studies and suitable for CNN training were employed. The data was divided into two subsets: in-sample and out-of-sample data. In-sample data was used

for training and validation of the CNN model. Meanwhile, the 30% out-of-sample data was used for the evaluation of the trained model. The model utilized 70% in-sample data training and 30% validation data. After the model was trained, then it was evaluated using the out-of-sample data. The out-of-sample data was based on both the thick forest and forest roadside environment.

3.4. Evaluation

The algorithms ran on the same machine to avoid any hardware performance inconsistency. The model evaluation metrics used were F1, precision and recall score performed on the out-of-sample data. The prediction results of a ML model were in the form of a collection of TP, TN, FP and FN [30], [31]. The efforts to maintain consistency were to control the hardware and software configurations with Intel Xeon E5 v4 8 Core 16 Threads, Geforce GTX 1080 Graphics Card 8GB GDDRX5, 16 GB DDR4 2666 MHz, and 1TB SATA SSD.

3.5. Thresholding prediction post-processing

Thresholding post-processing on prediction reduced the false alarm by rejecting the low confidence predictions as an additional effort. The determination of predicted class was based on the confidence probability value of each class. By default, a class with the highest value of confidence was selected. The thresholding method was expected to improve performance.

4. COMPUTATIONAL RESULTS AND DISCUSSION

This section presents the computation results of the SED performances. The experiments were conducted on all SED datasets and post-processing measurement on the false alarm rates. A detailed discussion is also provided to elaborate the findings and initiations in this section.

4.1. Model hyperparameters optimization

Several strategies were employed to avoid overfitting. This study used hyperparameters by reducing 250 batch sizes to 125, including a drop out layer of 0.5 on the dense layer, and adjusted the learning rate by decay staircase starting at 1.0 and reduced 90% at each epoch to reduce time to reach early convergence of training data on the early epochs. The design was inspired by the VGG16 model [32], which has half its previous convolutional layer. The new model 32-16-Conv 32 was compared with the previous model and it showed less overfitting. A detailed observation of the results is tabulated in Table 1. The performing model was 32-16-Conv32, 32-32-Conv 32 with early stopping at the 26th epoch with the lowest validation loss while having a considerable loss gap.

Table 1. Performance of custom CNN-RF models

Model	Stop Epoch	Validation Accuracy	Validation Loss	Training Loss	Loss Gap
16-32-Conv 32	26	0.7298	0.7487	0.7155	0.0332
32-32-Conv 32	26	0.8705	0.3824	0.2257	0.1567
32-16-Conv 32	26	0.8526	0.4659	0.5236	0.0577
16-32-Conv 64	21	0.8602	0.367	0.1036	0.2634
32-32-Conv 64	29	0.8975	0.2748	0.1087	0.1661
32-16-Conv 64	28	0.8677	0.3066	0.1759	0.1307

4.2. Computational results of the custom CNN-RF and VGG16-CNN-RF

CNN model of 32-16-Conv 32 implemented in this experiment was the model produced from tuning hyperparameters. The weights trained on this model in layer 32-16 convolutional layers were extracted to be used as a feature extraction layer for the hybrid CNN-RF model. A series of experiments was executed in search of the best parameters for an optimized model with the best performing results. Table 2 demonstrates the performance result of multiple RF models varying in ensemble size, from 10 to 1000. The peak performance of 32-16 CN-RF was achieved at the ensemble size of 500 with an accuracy of 0.7812, F1 of 0.7696, precision 0.7722 and recall of 0.7711.

The results were saturated at 0.7812 at the ensemble size of 300 to 500. The performance started dropping at 1000 ensembles, it seemed that more ensembles were not always better but may cause the model to perform otherwise. The research found that the model of 500 ensembles was the optimized model. The CNN model VGG16 implemented in this experiment was the model acquired from ImageNet [33]. It was found that the ensemble size of 400 trees had produced the best results F1 score of 0.8250, precision 0.8370, and recall 0.8187.

4.3. Post-processing results

After post-processing, the best model maintained a good performance while reducing false alarms. At 75% threshold, the model produced under 10% false alarm rate while maintaining the performance. The threshold reduced all false alarm rates to about 10% while maintaining performance. The post-processing step has improved the model's F1 score by 2.46%, precision by 0.30% and recall reduced by 0.73%. The post-processing improved the overall performance without affecting the performance on F1, precision and recall scores. Figure 2 shows the confusion matrix results of the CNN-RF model. The confusion matrix shows the performance of each class. The results after post-processing reduced the false alarm rate to about 10% and maintained the prediction rate of Hatchet class at 77.5%, Chainsaw 88.9% and Vehicle 88.8%.

Table 2. 32-16-CNN-RF results on different ensemble sizes

Ensemble Size	32-16-CN-RF				VGG16-CNN-RF			
	Accuracy (%)	F1	Precision	Recall	Accuracy (%)	F1	Precision	Recall
10	0.7705	0.7187	0.7220	0.7156	0.7725	0.7781	0.7885	0.7724
20	0.7761	0.7454	0.7548	0.7460	0.7855	0.7982	0.8072	0.7931
50	0.7759	0.7596	0.7610	0.7627	0.7933	0.8158	0.8251	0.8109
100	0.7755	0.7614	0.7626	0.7643	0.7964	0.8194	0.8299	0.8138
200	0.7793	0.7700	0.7726	0.7717	0.7962	0.8200	0.8312	0.8143
300	0.7784	0.7693	0.7716	0.7708	0.7965	0.8218	0.8331	0.8159
400	0.7802	0.7692	0.7715	0.7708	0.7969	0.8250	0.8370	0.8187
500	0.7812	0.7696	0.7722	0.7711	0.7968	0.8240	0.8358	0.8180
1000	0.7791	0.7698	0.7723	0.7710	0.7976	0.8229	0.8332	0.8177

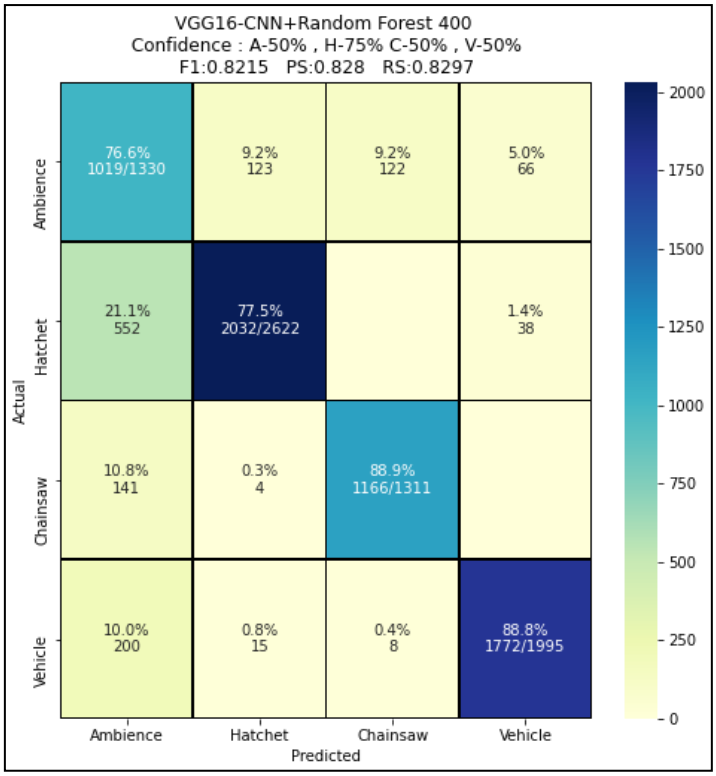


Figure 2. CNN-RF after post-processing confusion matrix

The results show a significant improvement for RF and CNN-RF, as demonstrated in Table 3. The CNN-RF had 12.55% F1 score. The ensemble size was then lowered to produce the best results. It was demonstrated that it was more efficient as it used a smaller ensemble size. The CNN-RF achieved was less than 10% FP rate at 75% Hatchet threshold than 85% on RF.

The accepted threshold value setting was the value when FP rates reached approximately 10%. The FP rate was essential for the evaluation of the algorithm. Another aspect to consider was that the ensemble size contributed to more computational time requirement. Small ensembles size should be relatively more efficient. However, this study did not include an in-depth evaluation on this aspect.

The performance of each model was compared before the post-processing and after thresholding. Table 4 presents the comparison in performance between models after the post-processing prediction. The study listed the scores in which the thresholding effort produced lower than 10% for the false alarm rate. Based on the obtained FP rate, the misclassification of classes could be identified.

Based on the observation, the average VGG16-CNN-RF model performed good results with an F1-score 0.8215, but it also reduced the performance of individual class accuracy, the Chainsaw and Hatchet class detections to 88.9% and 88.8% respectively. The 32-16-CNN-RF model acquired the best results in detecting Chainsaw and Vehicle class with 98.8% and 92.2% accuracy. The combination of CNN and RF could improve the individual class accuracy of the hatchet. The VGG16 was well-trained, but within the domain of images, perhaps a well-trained CNN for SED could be established using a more variety of real datasets in the future.

Table 3. Comparison between VGG16-CNN-RF and RF

Algorithm	Key performance scores			Threshold (%)	Individual class performance		
	F1	Precision	Recall		Hatchet	Chainsaw	Vehicle
RF	0.696	0.761	0.754	0.85	27.2	99.3	85
VGG16-CNN-RF	0.822	0.828	0.83	0.75	77.5	88.9	88.8
Difference	+0.126	+0.067	+0.076	-0.10	+50.3	-11.6	+3.0

Table 4. Comparison between models with post-processing

Algorithm	Key performance scores			False alarm rate (approx. 10%)	Individual class performance accuracy (%)		
	F1	Precision	Recall	Reliable Threshold (%)	Hatchet	Chainsaw	Vehicle
RF	0.6960	0.7606	0.7535	85	27.z2	99.3	85.0
32-16-Conv CNN-RF	0.6304	0.7112	0.6965	88	12.2	98.8	92.2
VGG16-CNN-RF	0.8215	0.8280	0.8297	75	77.5	88.9	88.8
32-16-Conv 32 CNN	0.7762	0.8007	0.8018	80	80.0	99.8	86.4
SVM	0.5694	0.6515	0.6296	92	20.7	84.4	71.3

4.4. Discussion

The research explored three methods, namely CNN, RF and SVM. A proposed method of CNN-RF has been exclusively established and studied for SED solutions in a forest environment. The results of the CNN-RF model showed considerable improvement with the F1 score of 0.7762. However, the loss function of multi-class cross-entropy was 0.42, which demonstrated that it needed more data to improve prediction quality. In contrast, the RF results were further enhanced with the CNN-RF model. The CNN-RF has shown some improvements as the CNN was used as the feature extractor. Besides, the RF was employed as the hypothesis of this combination that may improve performance. This study believes that an image similar to the one of hotspots has emerged based on the MLE pattern analysis done on the collected sound. The CNN 2D layer extracts more spatial features allowing the RF to improve performance with the tuned feature by the hybrid portion of CNN. The observation conducted has found an anomaly of difference in sound event class performance, especially at the Hatchet class. It always shows a lower detection regardless of any models. The nature hatchet sound is different compared to others. Instead of a long sustaining event like the others, it is in multiple bursts. The results show that the model can detect the hatchet event at 77% accuracy and a high FP rate at 30%. The other sound events provide better results of more than 85% and low FP under 10%. The thresholding method can be optimized on individual classes tailored to their respective difficulty in reducing FP.

Each model suffers a high degree of false prediction rate, confuses between ambience event and intruder event. The false prediction is not aligned with the intended purpose of the research for a security surveillance system. Hence, a post-processing layer is considered mandatory. The earliest result is considered too loose with false prediction rates. Thresholding post-processing is applied to the point of approximately 10% false prediction on any intruder events. It is not a good prediction with low confidence of about 51% over the other 49% [34]. Hence, increasing the threshold will avoid this problem. The variable threshold level is optimized until the target of 10% false detection rate is achieved. CNN and CNN-RF can be considered reliable for security in wildlife reserves. However, the findings do not apply in all of the SED cases. Further research is required to set the foundation to implement the use of SED in a vast area of surveillance in forests.

5. CONCLUSION

In conclusion, this study has discovered that many algorithms and techniques in solving SED have feasible application in industries. The CNN-RF has been proven to demonstrate an overall improvement in performance. It has also been discovered that it requires less configuration and optimization efforts due to the capabilities of CNN transfer learning. RF is recommended to be a suitable classifier in the SED task for forest environment compared to others because it is a hybrid approach in tackling SED in the domain. A post-processing method of thresholding has been applied to the prediction results in reducing the FP rate. Thresholding is reduced FP rate from 30% to 10% with an accuracy penalty between 94.6% and 77.5% on Hatchet class. The VGG16 based CNN-RF model and thresholding combination have enhanced performance for surveillance applications with 80% accuracy average and less than 10% of FP rate. Recommendations for subsequent research are to investigate the use of noise cancellation to isolate the featured sound events, to make use of the advantages of other deep learning methods such as Generative Adversarial Network and Attention CNN and to acquire a more optimal thresholding degree on the post-processing threshold. In addition, more experiments with real-world sound samples to understand the contributing factors such as different sound events, environment, noise, location, and weather can also be conducted.

ACKNOWLEDGEMENTS

The authors would like to thank the Research Management Center, Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Malaysia for providing essential support and knowledge for this research.

REFERENCES

[1] A. Dang, T. H. Vu, and J.-C. Wang, "Deep learning for DCASE 2017 challenge," *Detection and Classification of Acoustic Scenes and Even, t*2017.
[2] A. Mesaros *et al.*, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.
[3] A. S. Md Afendi, M. Yusoff, and M. Omar, "Mel-log energies analysis of authentic audible intrusion activities in a Malaysian forest," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 2, Apr. 2020, doi: 10.11591/eei.v9i2.2091.
[4] A. S. M. Affendi and M. Yusoff, "Review of anomalous sound event detection approaches," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 3, pp. 264–269, Dec. 2019, doi: 10.11591/ijai.v8.i3.pp264-269.
[5] B. F. Darst, K. C. Malecki, and C. D. Engelman, "Using recursive feature elimination in random forest to account for correlated variables in high dimensional data," *BMC Genetics*, vol. 19, no. S1, Art. no. 65, Sep. 2018, doi: 10.1186/s12863-018-0633-8.
[6] B. Jayaraman, L. Wang, K. Knipmeyer, Q. Gu, and D. Evans, "Revisiting membership inference under realistic assumptions," *Computer Science*, May 2020, Available: http://arxiv.org/abs/2005.10881.
[7] C. Tian, Y. Xu, and W. Zuo, "Image denoising using deep CNN with batch renormalization," *Neural Networks*, vol. 121, pp. 461–473, Jan. 2020, doi: 10.1016/j.neunet.2019.08.022.
[8] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 604–624, Feb. 2021, doi: 10.1109/TNNLS.2020.2979670.
[9] E. Cakır and T. Virtanen, "Convolutional recurrent neural networks for rare sound event detection," *Deep Neural Networks for Sound Event Detection*, vol. 12, 2019.
[10] F. Demir, M. Turkoglu, M. Aslan, and A. Sengur, "A new pyramidal concatenated CNN approach for environmental sound classification," *Applied Acoustics*, vol. 170, Art. no. 107520, Dec. 2020, doi: 10.1016/j.apacoust.2020.107520.
[11] F. J. Harris, "Multirate FIR filters for interpolating and desampling," in *Handbook of Digital Signal Processing*, Elsevier, 1987, pp. 173–287.
[12] H. Lim, J. Park, K. Lee, and Y. Han, "Rare sound event detection using 1D convolutional recurrent neural networks," *Detection and Classification of Acoustic Scenes and Events*, 2017.
[13] H. Phan, M. Krawczyk-Becker, T. Gerkmann, and A. Mertins, "DNN and CNN with weighted and multi-task loss functions for audio event detection," Aug. 2017. Available: http://arxiv.org/abs/1708.03211.
[14] J. G. Selman and N. Demir, "Automatic detection for acoustic monitoring of wild animals," 2019. Available: http://ilpubs.stanford.edu:8090/1166/.
[15] J. Li, W. Dai, F. Metze, S. Qu, and S. Das, "A comparison of deep learning methods for environmental sound detection," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 126–130, doi: 10.1109/ICASSP.2017.7952131.
[16] K. Chung, "Perceived sound quality of different signal processing algorithms by cochlear implant listeners in real-world acoustic environments," *Journal of Communication Disorders*, vol. 83, Art. no. 105973, Jan. 2020, doi: 10.1016/j.jcomdis.2019.105973.
[17] K. K. Lella and A. Pja, "Automatic COVID-19 disease diagnosis using 1D convolutional neural network and augmentation with human respiratory sound based on parameters: cough, breath, and voice," *AIMS Public Health*, vol. 8, no. 2, pp. 240–264, 2021, doi: 10.3934/publichealth.2021019.
[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014. Available: http://arxiv.org/abs/1409.1556.
[19] K. Wang, L. Yang, and B. Yang, "Audio event detection and classification using extended R-FCN approach," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, pp. 128–132, 2017.
[20] M. Z. Alom *et al.*, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, Art. no. 292, Mar. 2019, doi: 10.3390/electronics8030292.
[21] P. Patel and A. Thakkar, "The upsurge of deep learning for computer vision applications," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 1, pp. 538–548, Feb. 2020, doi: 10.11591/ijece.v10i1.pp538-548.

[22] Q. Wang, J. Du, H.-X. Wu, J. Pan, F. Ma, and C.-H. Lee, "A four-stage data augmentation approach to resnet-conformer based acoustic modeling for sound event localization and detection," Jan. 2021. Available: http://arxiv.org/abs/2101.02919.
[23] R. P. Reynolds, W. L. Kinard, J. J. Degraff, N. Leverage, and J. N. Norton, "Noise in a laboratory animal facility from the human and mouse perspectives.," *Journal of the American Association for Laboratory Animal Science: JAALAS*, vol. 49, no. 5, pp. 592–597, Sep. 2010. Available: http://www.ncbi.nlm.nih.gov/pubmed/20858361.
[24] R. Serizel, N. Turpault, A. Shah, and J. Salamon, "Sound event detection in synthetic domestic environments," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 86–90, doi: 10.1109/ICASSP40776.2020.9054478.
[25] S. Banerjee and R. Pamula, "Random forest boosted CNN: an empirical technique for plant classification," in *Advances in Intelligent Systems and Computing*, Springer Singapore, pp. 251–261, 2020.
[26] S.-H. Jung and Y.-J. Chung, "Performance analysis of the convolutional recurrent neural network on acoustic event detection," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 4, pp. 1387–1393, Aug. 2020, doi: 10.11591/eei.v9i4.2230.
[27] S. Hershey *et al.*, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 131–135, doi: 10.1109/ICASSP.2017.7952132.
[28] S. Joshi, D. K. Verma, G. Saxena, and A. Paraye, "Issues in training a convolutional neural network model for image classification," in *Communications in Computer and Information Science*, Springer Singapore, pp. 282–293, 2019.
[29] S.-Y. Jung, C.-H. Liao, Y.-S. Wu, S.-M. Yuan, and C.-T. Sun, "Efficiently classifying lung sounds through depthwise separable CNN models with fused STFT and MFCC features," *Diagnostics*, vol. 11, no. 4, Art. no. 732, Apr. 2021, doi: 10.3390/diagnostics11040732.
[30] P. Tzirakis, A. Shiarella, R. Ewers, and B. W. Schuller, "Computer audition for continuous rainforest occupancy monitoring: the case of bornean gibbons' call detection," in *Interspeech 2020*, Oct. 2020, pp. 1211–1215, doi: 10.21437/Interspeech.2020-2655.
[31] S. L. Ullo, S. K. Khare, V. Bajaj, and G. R. Sinha, "Hybrid computerized method for environmental sound classification," *IEEE Access*, vol. 8, pp. 124055–124065, 2020, doi: 10.1109/ACCESS.2020.3006082.
[32] W. Liu and J. A. Zagzebski, "Trade-offs in data acquisition and processing parameters for backscatter and scatterer size estimations," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 57, no. 2, pp. 340–352, Feb. 2010, doi: 10.1109/TUFFC.2010.1414.
[33] Z. Liu and S. Li, "A sound monitoring system for prevention of underground pipeline damage caused by construction," *Automation in Construction*, vol. 113, Art. no. 103125, May 2020, doi: 10.1016/j.autcon.2020.103125.
[34] Z. S. Bojkovic, B. M. Bakmaz, and M. R. Bakmaz, "Hamming window to the digital world," *Proceedings of the IEEE*, vol. 105, no. 6, pp. 1185–1190, Jun. 2017, doi: 10.1109/JPROC.2017.2697118.

BIOGRAPHIES OF AUTHORS

Muhamad Amirul Sadikin Md Afendi received Bachelor of Information Technology (Hons.) Intelligent System Engineering 5th July 2019. Currently, he is a Master student at Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Malaysia. He is a freelance fullstack web-system developer since 2018, working on various projects creating workflow management software. Projects previously ventures into microcontroller programming, applying ML with microcomputers, electronics hardware prototyping, and webserver cloud computing. He can be contacted at email: ai.amirul.sadikin@gmail.com.

Marina Yusoff is currently a senior fellow researcher at the Institute for Big Data Analytics and Artificial Intelligence (IBDAAI) and Associate Professor of Computer and Mathematical Sciences, Universiti Teknologi MARA Shah Alam, Malaysia. She has a Ph.D. in Information Technology and Quantitative Sciences (Intelligent Systems). She previously worked as a Senior Executive of Information Technology in SIRIM Berhad, Malaysia. She is the most interested in multidisciplinary research, artificial intelligence, nature-inspired computing optimization, and data analytics. She applied and modified AI methods in many research and projects, including deep learning, neural network, particle swarm optimization, genetic algorithm, ant colony, and cuckoo search for many real-world problems and industrial projects. Her recent projects are data analytic optimizer, audio and image pattern recognition. She has many impact journal publications and contributed as an examiner and reviewer to many conferences, journals, and universities' academic activities. She can be contacted at email: marina998@uitm.edu.my.

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 129~136
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i1.pp129-136

Transfer learning for cancer diagnosis in histopathological images

Sandhya Aneja[1], Nagender Aneja[2], Pg Emeroylariffion Abas[1], Abdul Ghani Naim[2]
[1]Faculty of Integrated Technologies, Universiti Brunei Darussalam, Bandar Seri Begawan, Brunei Darussalam
[2]School of Digital Science, Universiti Brunei Darussalam, Bandar Seri Begawan, Brunei Darussalam

Article Info

Article history:

Received Jul 19, 2021
Revised Nov 30, 2021
Accepted Dec 12, 2021

Keywords:

Cancer detection
CNN
Feature extraction
Fine tuning
Transfer learning

ABSTRACT

Transfer learning allows us to exploit knowledge gained from one task to assist in solving another but relevant task. In modern computer vision research, the question is which architecture performs better for a given dataset. In this paper, we compare the performance of 14 pre-trained ImageNet models on the histopathologic cancer detection dataset, where each model has been configured as naive model, feature extractor model, or fine-tuned model. Densenet161 has been shown to have high precision whilst Resnet101 has a high recall. A high precision model is suitable to be used when follow-up examination cost is high, whilst low precision but a high recall/sensitivity model can be used when the cost of follow-up examination is low. Results also show that transfer learning helps to converge a model faster.

Corresponding Author:

Nagender Aneja
School of Digital Science, Universiti Brunei Darussalam
Bandar Seri Begawan, Brunei Darussalam
Email: nagender.aneja@ubd.edu.bn

1. INTRODUCTION

Transfer learning focuses on transferring latent knowledge from a source domain to a target domain, to solve the issue of insufficient data on the target domain. In the case of deep transfer learning, knowledge comprises architecture and trained parameters, wherein a nonlinear function is learned for the task in the target domain. Transfer learning has been used in computer vision, natural language processing, speech recognition, and deep reinforcement learning and has been successfully deployed in industries, including health care, autonomous driving, robotics, spam filtering, search, and recommendation. Applications of transfer learning in convolutional neural network (CNN) provide the benefits of starting the training process with a better initial model, and thus, faster learning and better performance. Transfer learning also is a useful approach for data and resource-constrained applications.

Although transfer learning is advantageous in transferring features from a source domain to a target domain, it may negatively impact the performance in the target domain, especially if the source domain is not related to the target domain. This is also known as negative transfer [1]. Recently, a number of CNN architectures have been proposed, with varying width i.e., number of feature maps, and depth i.e., number of layers. The architectures also vary in other parameters such as connections between hidden layers, normalization, and non-linear function. Most of the architectures have been shown to perform well on the ImageNet dataset. As such, for a specific application, there is a need to study which of the existing network architectures generalize well on that particular application.

Transfer learning is an extension of supervised learning such that the training set is from a source domain while the test set is from a target domain [2]. In traditional supervised learning, it is assumed that the test set comes from the same joint probability distribution of features and labels, and the posterior probability $P(X \cap Y) = P(Y|X).P(X)$ can be learned. However, transfer learning relaxes the condition that the test is from the same distribution and may not have labels or the same labels. A number of studies have been published on machine learning on cancer data, as it is the second-leading cause of death in the world. Mahmood *et al.* [3] surveyed neural network techniques for the classification of different cancers, where it has been observed that on many occasions, practitioners' opinion on the detection of a disease may vary according to clinical facts. A computer-based analysis would certainly help practitioners, by providing a second expert-opinion since neural network mimics the brain process. A carefully selected training data would be needed to accurately predict the disease.

Kornblith *et al.* [4] presented performance of 16 networks on non-health datasets. The authors found a correlation between ImageNet Accuracy and transfer accuracy when used as a fixed feature extractor or fine-tuned, however, the correlation has been found to be sensitive, in the case of a fixed feature extractor. The authors also discovered that in the case of Stanford Cars and fine-grained visual classification (FGVC) Aircraft dataset, pretraining using ImageNet provides minimal benefits. Tajbakhsh *et al.* [5] presented work on transfer learning in classification, detection, and segmentation between natural and medical images from radiology, cardiology, and gastroenterology. The authors demonstrated that pretraining with fine-tuning outperforms CNN that has been trained from scratch. Layer-wise tuning has also been shown to offer the best performance.

Guo *et al.* [6] presented an algorithm to find an optimal strategy for transfer learning for each instance, whereby the authors have used a policy network to decide whether to pass the image through fine-tuned layers or pre-trained layers. Opbroek *et al.* [7] have shown that support vector machine (SVM) based classifiers can outperform and minimize classification error by 60%, with a small amount of training data by using transfer learning. Raghu *et al.* [8] explored properties of transfer learning for medical imaging datasets, pertaining to Retina and CheXpert (chest x-ray images) data, and have shown that transfer learning offers little benefit to performance, with lightweight models giving comparable performance to the more complicated ImageNet architectures. However, the authors have only used two architectures: ResNet50 and Inception-v3, in their analysis. Cheplygina *et al.* [9] discussed around 140 papers on semi-supervised, multi-instance, and transfer learning, and have found transfer learning as the most popular. The usefulness of transfer learning has been demonstrated, but also mentioned the need for future research to examine connections between learning scenarios and generalization of results.

Rustam *et al.* [10] applied regression logistics and random forest for pancreatic cancer detection and have found random forest to perform better. Al-Khowarizmi and Suherman [11] applied a simple evolving connectionist system (SECoS) on a balanced cancer dataset, whilst Dabeer *et al.* [12] applied CNN for breast cancer detection using histopathology images from a biopsy. Pratiwi *et al.* [13] ensembled three architectures: inception V3, inception ResNet V2, and DenseNet 201, with the ensembled model showing sensitivity of 90%, specificity of 97%, precision of 82%, and recall of 85%. Thus, a comprehensive study of major architectures will be helpful for researchers to select the architecture for other datasets also.

2. RESEARCH METHOD

Transfer learning is an optimization technique to apply learning of one task to another task. This is important since deep learning needs lot of data to train and collecting data for each task is a challenging problem. Transfer learning can be applied in various ways. The section below provides a few definitions to formally define transfer learning, as well as presents various transfer learning methods.

- Definition 2.1 (Feature Space). Feature space, *X*, is set of all possible images of a particular size where each pixel value ranges 0-255. Training set is assumed to be subset of feature space, $X = \{x^i, x^i \in X, 1 \leq i \leq n\}$.
- Definition 2.2 (Label Space). Label space, *Y*, includes all possible labels for samples, and $Y = \{y^i, y^i \in Y, 1 \leq i \leq n\}$. In this paper, $Y = \{0, 1\}$, where 1 indicates presence of cancer tissue and 0 otherwise.
- Definition 2.3 (Prediction Function). Our objective is to learn prediction function $\hat{y} = f(x) or\ P(\hat{y}|x)$.
- Definition 2.4 (Domain). Domain, *D*, is set of feature space and probability distribution of training set $D = \{X, P(X)\}$.
- Definition 2.5 (Task). Task, *T*, is set of label space to predict and prediction function $T = \{Y, P(Y|X)\}$.
- Definition 2.6 (Transfer Learning). Given source domain $D_S = \{X_S,\ P(X_S)\}$ and source learning task $T_S = \{Y_S,\ P(Y_S|X_S)\}$, and target domain $D_T = \{X_T,\ P(X_T)\}$ and target learning task

$T_T = \{Y_T,\ P(Y_T|X_T)\}$, transfer learning deals with learning prediction function of target $P(Y_T|X_T)$ using knowledge of source domain D_S and source task T_S such that $D_S \neq D_T, T_S \neq T_T$, or no or limited labels in target domain [14].

2.1. Classification of deep transfer learning approaches

Transfer learning can help to provide a better initial model that can be used to faster training fro a different task. Neural information processing systems (NIPS) 1995 workshop learning to learn: knowledge consolidation and transfer in inductive systems provided initial motivations for this. Transfer learning approaches can be classified based on Domain and Task; Feature space and label space; Source and Target label space; and knowledge transfer process as described below:

2.1.1. Domain and task

Transfer of knowledge is possible in a wide variety of domains. Transfer Learning aims to improve learning of target task on target domain using knowledge from source task on source domain [15]. Transfer Learning is sometimes referred as domain adaption. We can classify transfer learning techniques, based on domain and task:

a. Same Domain and Different Task. Since the task is defined by $T = \{Y, P(Y|X)\}$, either Y is different or $P(Y|X)$ is different for source and target domain (or both are different).

– $Y_S \neq Y_T$, e.g., the source data set is CIFAR-10 whilst the target data set is CIFAR-100,

– $P(Y_S|X_S) \neq P(Y_T|X_T)$ i.e., the same image may be labeled differently in the source and the target domain, and it happens when the class balance is different (prior $P(Y)$ is different in source and target.)

b. Different domain and same task. Since the domain is defined by $D = \{X, P(X)\}$, either X is different or $P(X)$ is different for source and target domain (or both are different)

– $X_S \neq X_T$, different feature space, e.g., source data set is gray-scale images whilst target data set is color images or documents in different languages, $P(X_S) \neq P(Y_T)$, different distributions, while feature space may be same, e.g., the source domain consists of hand-drawn images while the target domain consists of photographs or documents in the same language but different topics. This is the most common scenario, commonly referred to as domain adaption, e.g. An autonomous model is trained for day driving but the trained model applied in all other scenarios.

2.1.2. Feature space and label space

Zhuang *et al.* [16] proposed a classification of transfer learning as homogeneous and heterogeneous transfer learning, based on feature space X and label space Y. Homogeneous transfer learning means X_S=X_T and Y_S=Y_T. Heterogeneous transfer learning means $X_S \neq X_T$ or/and $Y_S \neq Y_T$.

2.1.3. Source and target label space

Pan and Yang [14] classified transfer learning as Transductive, Inductive, and Unsupervised transfer learning, based on the label space Y. Transductive transfer learning means Y_S exists but Y_T does not exist. Inductive transfer learning means both Y_S and Y_T exist. \Unsupervised transfer learning means both Y_S and Y_T do not exist.

2.1.4. Knowledge transfer process

Tan *et al.* [17] classified deep transfer learning in four categories based on knowledge transfer process, namely, instances based, mapping based, network-based, and adversarial based. On the other hand, Li *et al.* [18] classified deep transfer learning into three approaches: instance transfer, model transfer, and feature representation transfer.

a. Instances-based or instance transfer: selects supplementary data from the source domain, whereby the selected source instances are similar to instances in the target domain. This is different from generative data augmentation, which generates synthetic data using generative adversarial network (GAN). Instance based transfer learning avoids negative transfer by selecting similar instances.

b. Mapping-based: both source and domain instances are mapped to a third domain space, whereby both domains are similar to each other. A union deep neural network is trained on modified source and target domain.

c. Network-based or model/feature representation transfer: this is the most common approach, and it focuses on network architecture and/or trained parameters. Source network architecture may be used as a feature extractor wherein convolution layers are frozen to extract the features, or may be used to fine-tune the target model, whereby the target model is initialized with trained parameters from the source domain, and then few/all of the layers are subsequently trained for the target domain. In other cases,

only the source network architecture is used with random weights for the target domain. This approach can further be classified in four cases:

- Target data set is large but different from the source: in this case, the entire model is trained for the target domain using sequential training for the source domain or joint training for both source and target domain
- Target data set is large and similar to the source: in this case, some layers are trained while other layers are frozen
- Target data set is small but different from source: in this case, some layers are trained while leaving other layers frozen and/or the use of domain adaption to find common feature space, which may also be determined using GANs [19], and
- Target data set is small and similar to the source: in this case, convolution base is frozen (no adaption to target).

d. Adversarial-based: In this case, transferable representations are determined using GAN. These transformations are applicable to both source and target domains. The features from both domains are extracted using front layers and sent to adversarial layers. The adversarial layer is trained to discriminate origin of features. Thus, the network is trained to learn general features. The representations are called transferable when the features are discriminative for the source domain, and indiscriminate with the shift between source and target domains [20].

In this paper, network-based or model/feature representation transfer for cancer detection has been analyzed. In particular, PatchCamelyon (PCam) benchmark data set for histopathologic cancer detection competition organized by Kaggle [21], has been used. The objective of the competition was to create an algorithm to identify metastatic cancer in small image patches taken from larger digital pathology scans.

The dataset [21] is a modified version of the PatchCamelyon (PCam) benchmark dataset [22], [23], with duplicate images removed. It has a total of 220,025 images, with 130,908 (60%) benign images and 89,117 (40%) images, where the center 32×32 px region of a patch contains at least one pixel of tumor tissue. Thus, the dataset is not balanced. Eight sample images from the dataset, containing tissue images of healthy and patient with tumor, are shown in Figure 1. Digitization of microscopic glass slide images has encouraged deep learning researchers for pathology diagnosis, where manual results differ considerably. In particular, trained deep convolutional neural network has been shown to perform better than manual cancer detection by different Pathologists. In this paper, performance of transfer learning from 14 CNN architectures has been presented. research methodology comprises three model training approaches for each architecture. The three training approaches are naive, feature extraction, and fine-tuning. These approaches are based on neural network architecture that comprises of convolutional layers and fully-connected layer(s).

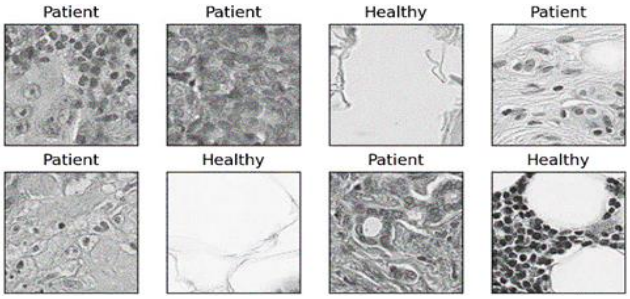


Figure 1. Sample images of healthy and patient with tumor, with center 32×32 px containing tumor tissue

In the Naive training experiment, neither the convolutional layers nor the fully connected layer(s) are trained, with only the pre-trained (trained on ImageNet dataset) layers used for testing. On the other hand, in the feature extractor experiment, the fully connected layer is trained for five epochs, whilst freezing the convolutional layers. Finally, in the fine-tune experiment, all layers from the convolutional layers and fully connected layer are trained. The fine-tune experiment is conducted for five epochs, only after conducting the second feature extractor experiment for five epochs. It has been found training for five epochs are sufficient for the model to converge, for both the convolutional and fully connected layers. At every epoch, we saved the model based on error rate i.e., after five epochs, we get the final model with the lowest error rate. The final saved model with the lowest error rate has been used to evaluate and compare different architectures.

3. RESULTS AND DISCUSSION

14 modern deep convolutional neural networks: Alexnet, Vgg16 bn, Vgg19 bn, Resnet18, Resnet34, Resnet50, Resnet101, Resnet152, Squeezenet1 0, Squeezenet1 1, Densenet121, Densenet169, Densenet201, and Densenet161, pretrained on ImageNet have been analyzed. We have considered batch size 32, input size 196×196, 80% training data, and 20% validation data, and learning rate of 0.01. Although the recommended input size is 224×224 for pre-trained models, we have reduced the input size slightly to 196×196 so that all models can fit onto our Nvidia 1080 GPU memory. Increasing input size is expected to increase Area Under the receiver operating characteristic curve (AUC) Score. Three experiments with 14 models, thus, a total of 42 sub-experiments, have been performed for this study.

3.1. Naive model

In the naive model, we have used weights from the pre-trained network. Since the model was not trained for the target task on the target domain, the model didn't generalize well. This will happen in particular when the target domain is different from source somain as is the case in this research. Results as in Figure 2 indicate that most models do not generalize well.

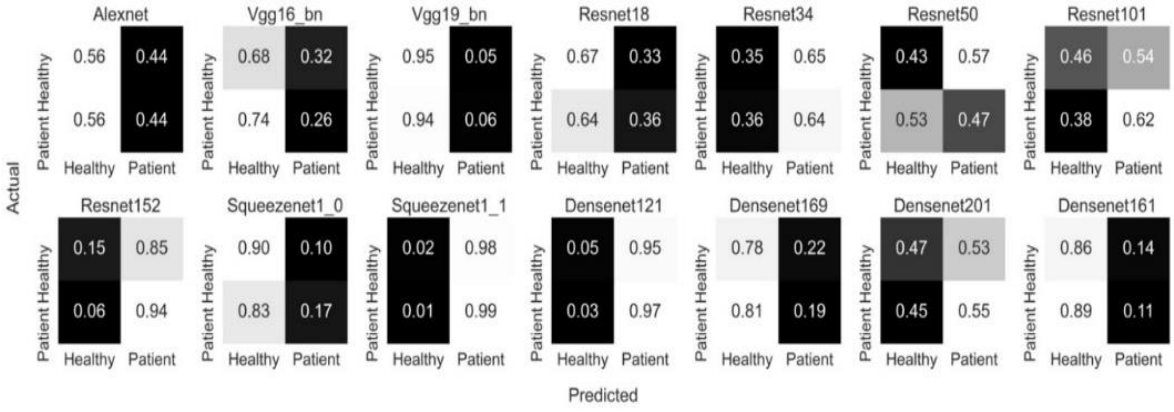


Figure 2. Naive model: confusion matrix (normalized)

3.2. Feature extractor model

In the feature extractor model, convolution layers are frozen, and the fully connected layer is trained for five epochs. Figure 3 displays the normalized confusion matrix after five epochs using the model that gives the lowest error rate during five epochs for every architecture. Figure 3 shows confusion matrix and Table 1 shows the performance of every architecture using the feature extractor model, in terms of AUC Score, F1 Score, Precision, and Recall. It can be seen that Densenet161 represents a high precision model, with precision value of 0.9710, while Resnet101 represents a high recall model, with recall value of 0.9569.

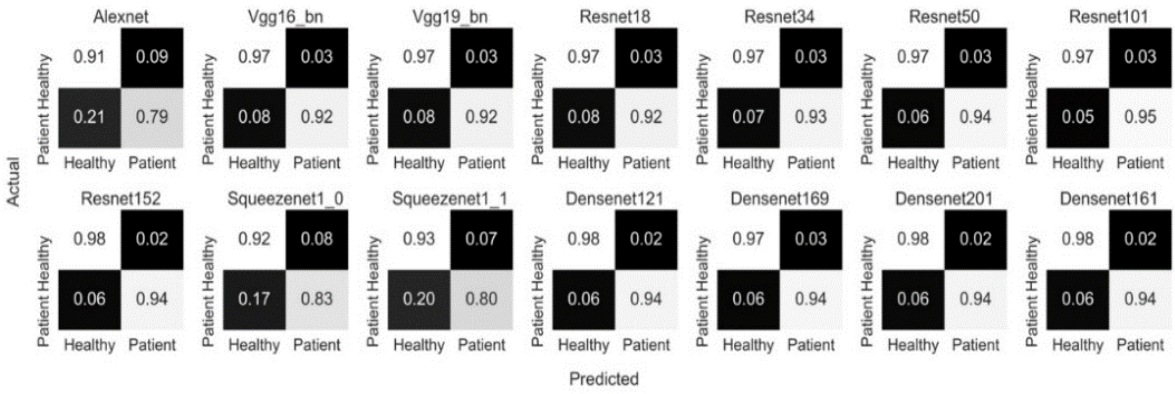


Figure 3. Feature extractor model: confusion matrix (normalized)

Table 1. Feature extractor model: AUC score, F1, precision, and recall for the 14 model

	AUC Score	F1	Precision	Recall
Alexnet	0.93872	0.82750	0.87121	0.78796
Vgg16 bn	0.98606	0.93751	0.95044	0.92492
Vgg19 bn	0.98710	0.93987	0.95831	0.92212
Resnet18	0.98640	0.93700	0.95482	0.91983
Resnet34	0.98823	0.94438	0.95614	0.93292
Resnet50	0.99046	0.94923	0.95990	0.93879
Resnet101	0.99131	0.95106	0.95685	0.94534
Resnet152	0.99131	0.95256	0.96525	0.94019
Squeezenet1 0	0.94847	0.85222	0.87606	0.82964
Squeezenet1 1	0.94659	0.84265	0.89129	0.79904
Densenet121	0.99065	0.95115	0.96355	0.93907
Densenet169	0.99151	0.95181	0.96173	0.94209
Densenet201	0.99196	0.95468	0.96566	0.94394
Densenet161	0.99240	0.95656	0.97101	0.94254

3.2. Fine tune model

In the fine tune model, we run five epochs to train the full network in addition to training a fully connected layer, with the fine-tuning experiment done only after feature extraction. Figure 4 displays normalized confusion matrix after five epochs using the models that give the lowest error rate. For this experiment, surprisingly, Squeezenet1 1 gave the worst performance, predicting most subjects as patients with cancer tissue as shown in the confusion matrix. Squeezenet1 1 has smaller number of parameters than the number of parameters in Squeezenet1 0 and other models, due to model's compression technique, as given in Iandola *et al.* [24], [25]. This trade-off has resulted in reduced performance of the model, as observed in the Figure 4. Table 2 shows that Densenet161 has a high precision value of 0.9703, whilst Densenet201 has a high recall value of 0.9458.

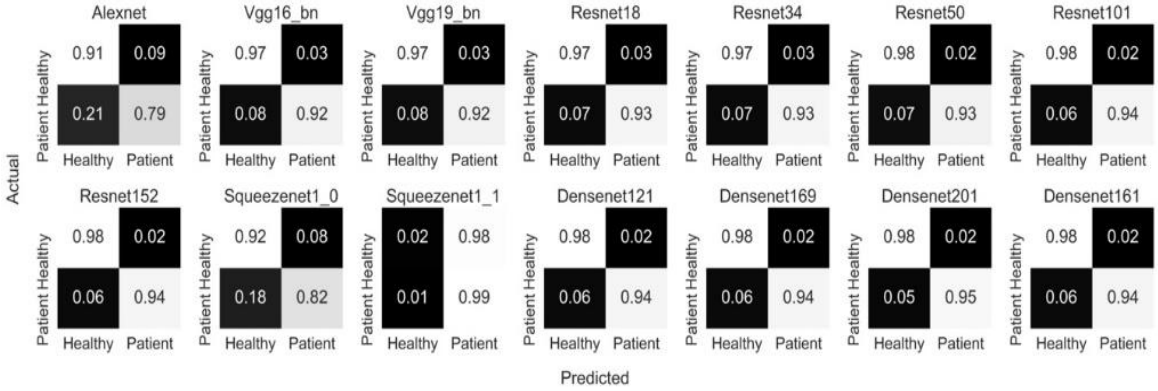


Figure 4. Fine tune model: confusion matrix (normalized)

Table 2. Fine tune model: AUC score, F1, precision, and recall for the 14 models

	AUC Score	F1	Precision	Recall
Alexnet	0.93362	0.82175	0.85449	0.79143
Vgg16 bn	0.98600	0.93684	0.95306	0.92117
Vgg19 bn	0.98707	0.93853	0.96015	0.91787
Resnet18	0.98630	0.93874	0.95239	0.92548
Resnet34	0.98844	0.94409	0.96111	0.92766
Resnet50	0.99097	0.94946	0.96852	0.93113
Resnet101	0.99149	0.95133	0.96646	0.93667
Resnet152	0.99080	0.95192	0.96612	0.93812
Squeezenet1 0	0.94676	0.85020	0.87813	0.82399
Squeezenet1 1	0.52655	0.57968	0.40978	0.99027
Densenet121	0.99092	0.95146	0.96277	0.94042
Densenet169	0.99186	0.95321	0.96653	0.94025
Densenet201	0.99235	0.95490	0.96418	0.94579
Densenet161	0.99225	0.95520	0.97028	0.94058

4. CONCLUSION

As humans learn new things by applying knowledge from previous learnings, transfer learning can help the progress of artificial general intelligence further. This research has shown that Densenet161 outperforms other architectures, with an AUC score of 0.9924 and F1 Sore 0.95 in the feature extractor model. Densenet161 has been shown to have high precision whilst Resnet101 is a high recall model. No significant performance improvement has been observed through fine-tuning, as the model has been first trained using feature extraction, such that there is limited scope for further improvement. A high precision model may be used where follow-up examination cost is high, whilst low precision but a high recall/sensitivity model can be used when the cost of follow-up examination is low. Future work may include applying pre-trained models on other categories of health datasets.

REFERENCES

[1] Z. Wang, Z. Dai, B. Poczos, and J. Carbonell, "Characterizing and avoiding negative transfer," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 11285–11294, doi: 10.1109/CVPR.2019.01155.

[2] N. Aneja and S. Aneja, "Transfer learning using CNN for handwritten devanagari character recognition," in *2019 1st International Conference on Advances in Information Technology (ICAIT)*, Jul. 2019, pp. 293–296, doi: 10.1109/ICAIT47043.2019.8987286.

[3] M. Mahmood, B. Al-Khateeb, and W. Makki Alwash, "A review on neural networks approach on classifying cancers," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 2, pp. 317–326, Jun. 2020, doi: 10.11591/ijai.v9.i2.pp317-326.

[4] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 2656–2666, doi: 10.1109/CVPR.2019.00277.

[5] N. Tajbakhsh *et al.*, "Convolutional neural networks for medical image analysis: full training or fine tuning?," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, May 2016, doi: 10.1109/TMI.2016.2535302.

[6] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, "Spottune: transfer learning through adaptive fine-tuning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 4800–4809, doi: 10.1109/CVPR.2019.00494.

[7] A. van Opbroek, M. A. Ikram, M. W. Vernooij, and M. de Bruijne, "Transfer learning improves supervised image segmentation across imaging protocols," *IEEE Transactions on Medical Imaging*, vol. 34, no. 5, pp. 1018–1030, May 2015, doi: 10.1109/TMI.2014.2366792.

[8] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, "Transfusion: understanding transfer learning for medical imaging," *Advances in Neural Information Processing Systems*, vol. 32, pp. 3342–3352, 2019.

[9] V. Cheplygina, M. de Bruijne, and J. P. W. Pluim, "Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis," *Medical Image Analysis*, vol. 54, pp. 280–296, May 2019, doi: 10.1016/j.media.2019.03.009.

[10] Z. Rustam, F. Zhafarina, G. S. Saragih, and S. Hartini, "Pancreatic cancer classification using logistic regression and random forest," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 2, pp. 476–481, Jun. 2021, doi: 10.11591/ijai.v10.i2.pp476-481.

[11] A.-K. Al-Khowarizmi and S. Suherman, "Classification of skin cancer images by applying simple evolving connectionist system," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 2, pp. 421–429, Jun. 2021, doi: 10.11591/ijai.v10.i2.pp421-429.

[12] S. Dabeer, M. M. Khan, and S. Islam, "Cancer diagnosis in histopathological image: CNN based approach," *Informatics in Medicine Unlocked*, vol. 16, 2019, doi: 10.1016/j.imu.2019.100231.

[13] R. A. Pratiwi, S. Nurmaini, D. P. Rini, M. N. Rachmatullah, and A. Darmawahyuni, "Deep ensemble learning for skin lesions classification with convolutional neural network," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 563–570, Sep. 2021, doi: 10.11591/ijai.v10.i3.pp563-570.

[14] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[15] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *International conference on artificial neural networks*, pp. 270-279. Springer, Cham, 2018, doi: 10.1007/978-3-030-01424-7_27.

[16] F. Zhuang *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555.

[17] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11141, Springer International Publishing, 2018, pp. 270–279.

[18] Y. Li, Y. Yang, S. Zhou, J. Qiao, and B. Long, "Deep transfer learning for search and recommendation," in *Companion Proceedings of the Web Conference 2020*, Apr. 2020, pp. 313–314, doi: 10.1145/3366424.3383115.

[19] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2962–2971, doi: 10.1109/CVPR.2017.316.

[20] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," in *Advances in Computer Vision and Pattern Recognition*, Springer International Publishing, 2017, pp. 189–209.

[21] B. Veeling, B. E. Bejnordi, G. Litjens, and J. V. D. Laak, "Histopathologic cancer detection," *Kaggle*. 2019.

[22] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling, "Rotation equivariant CNNs for digital pathology," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11071, Springer International Publishing, 2018, pp. 210–218.

[23] B. Ehteshami Bejnordi *et al.*, "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer," *JAMA*, vol. 318, no. 22, pp. 2199–2210, Dec. 2017, doi: 10.1001/jama.2017.14585.

[24] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," Feb. 2016, [Online]. Available: http://arxiv.org/abs/1602.07360.

[25] B. Koonce, "SqueezeNet," in *Convolutional Neural Networks with Swift for Tensorflow*, Berkeley, CA: Apress, 2021, pp. 73–85.

BIOGRAPHIES OF AUTHORS

Dr Sandhya Aneja is working as Assistant Professor of Information and Communication System Engineering at the Faculty of Integrated Technologies, Universiti Brunei Darussalam. Her primary areas of research interest include wireless networks, high-performance computing, internet of things, artificial intelligence technologies - machine learning, machine translation, deep learning, data science, and data analytics. Further info on her homepage http://expert.ubd.edu.bn/sandhya.aneja. She can be contacted at sandhya.aneja@ubd.edu.bn.

Dr Nagender Aneja is working as Assistant Professor at School of Digital Science, Universiti Brunei Darussalam. He did his Ph.D. in Computer Engineering from J.C. Bose University of Science and Technology YMCA, and M.E. Computer Technology and Applications from Delhi College of Engineering. He is currently working in the area of deep learning, computer vision, and natural language processing. He is also founder of ResearchID.co, further info on his homepage http://expert.ubd.edu.bn/nagender.aneja. He can be contacted at nagender.aneja@ubd.edu.bn

Pg Dr Emeroylariffion Abas received his B.Eng. Information Systems Engineering from Imperial College, London in 2001, before obtaining his Ph.D. Communication Systems in 2005 from the same institution. He is now working as an Assistant Professor in General Engineering, Faculty of Integrated Technologies, Universiti Brunei Darussalam. His present research interest are data analytic, energy systems and photonics. Further info on his homepage https://expert.ubd.edu.bn/emeroylariffion.abas. He can be contacted at emeroylariffion.abas@ubd.edu.bn.

Dr Abdul Ghani Naim is a Senior Assistant Professor at School of Digital Science, Universiti Brunei Darussalam. He did his Ph.D. in Information Security from the Royal Holloway College, London. His present research interests include computer security, cryptography, high performance computing, and machine learning. Further info on his homepage https://expert.ubd.edu.bn/ghani.naim. He can be contacted at ghani.naim@ubd.edu.bn.

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 137~147
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i1.pp137-147

Dataset for classification of computer graphic images and photographic images

Halaguru Basavarajappa Basanth Kumar[1], Haranahalli Rajanna Chennamma[2]
[1]DoS in Computer Science, SBRR Mahajana First Grade College (Autonomous), PG Wing, Mysuru, India
[2]Department of Computer Applications, JSS Science and Technology University, Mysuru, India

Article Info

Article history:

Received May 16, 2021
Revised Dec 24, 2021
Accepted Dec 30, 2021

Keywords:

Computer graphic images
Conventional machine learning
Deep learning
Handcrafted features
Photographic images
Photo-realistic computer graphics

ABSTRACT

The recent advancements in computer graphics (CG) image rendering techniques have made it easy for the content creators to produce high quality computer graphics similar to photographic images (PG) confounding the most naïve users. Such images used with negative intent, cause serious problems to the society. In such cases, proving the authenticity of an image is a big challenge in digital image forensics due to high photo-realism of CG images. Existing datasets used to assess the performance of classification models are lacking with: (i) larger dataset size, (ii) diversified image contents, and (iii) images generated with the recent digital image rendering techniques. To fill this gap, we created two new datasets, namely, 'JSSSTU CG and PG image dataset' and 'JSSSTU PRCG image dataset'. Further, the complexity of the new datasets and benchmark datasets are evaluated using handcrafted texture feature descriptors such as gray level co-occurrence matrix, local binary pattern and VGG variants (VGG16 and VGG19) which are pre-trained convolutional neural network (CNN) models. Experimental results showed that the CNN-based pre-trained techniques outperformed the conventional support vector machine (SVM)-based classifier in terms of classification accuracy. Proposed datasets have attained a low f-score when compared to existing datasets indicating they are very challenging.

Corresponding Author:

Haranahalli Rajanna Chennamma
Department of Computer Applications, JSS Science and Technology University
Mysuru, India
Email: hrchennamma@jssstuniv.in

1. INTRODUCTION

Rapid advancement in computer graphic (CG) image rendering techniques give birth to applications such as animations, cartoons, gaming, photo-realism, virtual reality and many more [1]. Sophisticated CG software tools allow the user to produce synthetic images which are close to the reality and are difficult to identify whether an image is a camera captured or computer generated [2], [3]. If such images are used illegally in the court-of-law, journalism, criminal investigation, and political propaganda, then it may cause serious threat to the society [4]. In such cases, verifying the authenticity of images is a big challenge in digital image forensics.

To classify photo-realistic computer graphics (PRCG) from photographic images (PG), three benchmark datasets as shown in Table 1 are used in the literature: i) Columbia Photographic Images and Photorealistic Computer Graphics Dataset–Columbia Dataset; ii) DSTok dataset; and iii) A dataset created by Rahmouni *et al.*, to evaluate the performance of classification models. The DSTok dataset is the largest dataset in the literature with 9,700 samples. The aforementioned datasets are lacking with diversified image contents, sample size and more importantly the images in these datasets were produced/captured using older

versions of CG software's/camera models. Advancement in digital image rendering techniques have made it easy for the users to capture high quality images with regard to photographs and to produce photograph-like images which cannot be compared with the graphical contents produced using older versions of CG image rendering techniques. Hence, it is needed to upgrade datasets as well to evaluate updated innovations in the field of CG and PG image classification.

Table 1. Comparison of existing CG and PG image datasets.

Dataset	Year	CG Images	PG Images	Dataset Size	Publicly Available?	Limitations
Columbia Dataset [5]	2004	800	Personal: 800 Google: 800 Recaptured PRCG: 800	3,200	Yes	Small dataset and images are restricted to few categories.
DSTok Dataset [6]	2013	4,850	4,850	9,700	No	PRCG images in CG image class are relatively small.
Rahmouni *et al.* [7]	2017	1,800	1,800	3,600	Yes	Small dataset and CG image class consist of only video game screenshots.

In this paper, we propose two new datasets, namely "JSSSTU CG and PG image dataset"and "JSSSTU PRCG image dataset". Initially the dataset is created with the intention of having diversified image contents with respect to CG and PG image categories and comprises 14,000 samples. Later dataset is created with the intention of having only photo-realistic computer graphics which are hard to distinguish with naked eyes that consist of 2,000 samples. Our new datasets would become very challenging and will be helpful for the researchers to develop efficient or improved classification models to produce better results who are working on the cutting-edge research problem: "classification of computer graphic images and photographic images". Researchers have addressed this problem in different perspectives based on conventional machine learning and deep learning approaches.

a. Conventional machine learning

Significant improvement has been made in recent years to classify CG and PG images. Existing conventional machine learning techniques can be grouped into three categories based on the features selected for classification. They are: i) camera- characteristic based approaches [4], [8]–[12]; ii) spatial feature based approaches [13]–[20], and iii) geometric feature based approaches [21]–[28].

– Camera-characteristic based approaches

Techniques used to generate CG and PG images, undergo different pipeline architectures. Since PG images are acquired using digital cameras, they must exhibit distinct intrinsic properties which are not present in CG images. Based on this fact, some identification approaches have been described in [8]–[10]. Dehnie *et al.* [8] employed pattern noise caused due to the defect in camera sensors for classification of CG and PG images. Dirik *et al.* [9] proposed the features to detect the traces of color filter array (CFA) and chromatic aberration to distinguish CG and PG images. Khanna *et al.* [10] described a method based on residual pattern noise to distinguish scanner, CG and PG images. Photo response non uniformity (PRNU) noise is used as a digital fingerprint to identify the source camera in digital forensics and this is exploited in [4], [11], [12]. Peng *et al.* [11] proposed a method based on the theory of multifractal spectrum and features of PRNU, multifractal spectrum features of PRNU are extracted from an image to distinguish PRCG and PG images. Peng and Zhou [4] examine the changes in PRNU correlations and histogram features extracted from variance histograms of PRNU are used for identification of CG and PG images. Long *et al.* [12] proposed a method based on binary measures computed from PRNU noise in RGB channels to depict the differences between CG and PG images.

– Spatial feature based approaches

Pan *et al.* [13] show that the perceptual difference between CG and PG images is generally present in color and coarseness. Former is represented using fractal dimensions and the latter is described using generalised dimensions. Wu *et al.* [14] compute the difference in histogram of images and some higher histogram bins are considered as features to perform classification. Local binary pattern (LBP) [15] is a texture descriptor majorly used in image texture analysis and this is employed in [16], [17] to classify CG and PG images. Peng *et al.* [18] proposed a method based on statistical and textural features. Tan *et al.* [19] presented a novel scheme using local ternary count (LTC) which produces 54 dimensions of features from normalized histograms. Peng *et al.* [20] proposed a hybrid feature by analysing the differences in textures of residuals of CG and PG images.

– Geometric feature based approaches

Wang and Moulin [21] used wavelet-coefficients histogram as features extracted from wavelet-based statistical model for discrimination of CG and PG images. Chen *et al.* [22] built an alpha-stable model to describe wavelet decomposition coefficients of PG images. Wavelet domain is used to extract fractional lower order moments in images. Zhang and Wang [23] found that imaging features and visual features for images produced using different image acquisition processes reveal different statistical regularities in the wavelet domain. Based on this principle, statistical features and cross correlation of wavelet coefficients are used as features extracted from each sub-bands. Guo and Wang [24] presented a method based on multiwavelets which extracts the features in wavelet subbands. Fan *et al.* [25] proposed a scheme based on modified image contour transform in HSV color space to classify CG and PG images. Statistics such as average value, variance, skewness and kurtosis are computed in the wavelet domain. Birajdar and Mankar [26] used discrete wavelet transform to extract binary statistical image features by decomposing an image into subbands. Then the fuzzy entropy measure is employed to select relevant features. Quaternion wavelet transform is presented in [27] and [28], which extracts statistical features to classify CG and PG.

b. Deep learning approaches

Rahmouni *et al.* [7] proposed a novel scheme which combines statistical feature extraction to a convolutional neural network (CNN) architecture, then class label of the entire image is predicted by using weighted voting scheme which aggregates the local estimates of the class probabilities. Nguyen *et al.* [29] customized VGG-19 architecture to extract the generic features in the first three convolutional layers, and then statistical pooling layer is constructed as proposed in [7]. Pre-trained CNN models are employed in [30]–[33], and fine-tuned through transfer learning for binary classification. Chawla *et al.* [34] proposed five layers CNN architecture by introducing a special layer which takes some prediction error filters onto the first convolutional layer to ensure the correlation between pixels in PG and CG images. To predict the outcome of the original picture, two methods are used namely, weighted voting scheme and majority voting scheme. Former is used to label the image by aggregating the class probabilities and latter is used where the label is considered that appear in the majority of the image patches. Yao *et al.* [35] employed three sorts of high-pass filters to extract sensor noise residuals then piped into the proposed five layers CNN framework. Quan *et al.* [36] proposed a new CNN framework with two CNN cascaded convolutional layers at the end of the network. He *et al.* [37] described a novel deep learning approach by combining CNN and recurrent neural network (RNN). Thereafter, He *et al.* [38] proposed an attention-based dual-branch CNN to extract the features from combined color components. Meena and Tyagi [39] proposed an ensemble model by combining the features produced by VGG-19 pre-trained CNN and noise features produced using high-pass filters to discriminate CG and PG images.

From the above study, even though much progress has been made for the classification of CG and PG images, existing techniques and datasets used to evaluate the performance is still have the following limitations: i) in the existing datasets, sample size and image contents are limited and do not make compelling high quality image content due to advancement in image rendering techniques; (ii) in prior works, accuracy of the classification model depends on choice of the feature descriptor used for classification.

In the task of distinguishing CG and PG images, the contributions of the paper are outlined as:

– Due to non-availability of large, heterogeneous dataset containing CG and PG images, 'JSSSTU CG and PG image dataset' is created. 'JSSSTU PRCG image dataset' is created which exhibits high photo-realism.
– Effectiveness of the existing texture based feature descriptors and CNN based deep learning techniques are investigated on our new datasets and benchmark datasets.

Remainder of this paper is organised as follows: section 2 presents a description of the state-of-the-art techniques based on conventional machine learning and deep learning. Section 3 demonstrates performance of the techniques on our new datasets and benchmark datasets through experimental results. Conclusion is given in section 4.

2. RESEARCH METHOD

In this section, state-of-the-art conventional machine learning and CNN based deep learning techniques are developed for the task of classifying CG and PG images. Handcrafted textural features are considered for conventional machine learning and VGG variants CNN based pre-trained models, are used for deep learning. They are described in the following sections.

2.1. Conventional machine learning techniques

Texture describes surface characteristics of an image. Most widely used texture descriptors such as gray level co-occurrence matrix (GLCM) [40] and LBP [15] are employed to analyse the surface texture of CG and PG image. Texture surface of CG image appear smoother than those of PG image, which exhibits the basic differences between them. Hence, the aforementioned texture descriptors are used in our work.

2.1.1. GLCM descriptor

GLCM is a statistical method which computes the occurrence of pairs of pixels or gray levels in a particular orientation over all in an image or image region. This is represented by using parameters (Θ, d) where 'Θ' represents orientation and 'd' is the distance between two picture elements. The GLCM descriptor allows rotational invariance and it is defined by 8 orientations separated by Π/4 radians. Haralick *et al.* [40] defined 14 statistical properties computed from the normalized GLCM matrix. In this work, we employed four properties such as contrast, correlation, energy and homogeneity.

– Contrast: contrast computes local intensity variation between a picture element and to its neighbor for the entire image as given in (2). The range is calculated using the (1).

Range = [0 (size(GLCM,1)-1)^2] (1)

Where, GLCM represents normalized matrix. I_{mn} and variables m and n from (2)-(5) represent (m, n)th entry and a value at (m, n) in a normalized GLCM.

$$Contrast = \sum_{m,n} |m-n|^2 I_{mn} \quad (2)$$

– Correlation: correlation computes correlation of a picture element to its neighbor over the entire image. It returns a value between -1 and 1for a positively or negatively correlated image. Otherwise, return unrepresentative value for a constant image as given in (3). Where, 'μ' and 'σ′ indicates mean and standard deviation of the marginal distributions associated with I_{mn}/R, and R is a normalized constant.

$$Correlation = \sum_{m,n} I_{mn} \frac{(m-\mu)(n-\mu)}{\sigma^2} \quad (3)$$

– Energy: energy is also termed as angular second moment which computes the sum of squared elements. It returns a value between 0 and 1, otherwise, returns 1 for a constant image as given in (4).

$$Energy = \sum_{m,n} (I_{mn})^2 \quad (4)$$

– Homogeneity: Homogeneity computes the closeness of elements diagonally in GLCM. It returns 1 for diagonal elements, otherwise, returns a value between 0 and 1 as given in (5).

$$Homogeniety = \sum_{m,n} \frac{I_{mn}}{1+|m-n|} \quad (5)$$

2.1.2. LBP descriptor

LBP is a texture descriptor operator proposed by Ojala [15], which encodes each pixel value of an image by comparing its neighborhood pixels with the center pixel value. If the intensity of neighboring pixel is greater than or equal to the intensity of center pixel mark the neighboring pixel as 1, otherwise, mark as 0 which result in a binary sequence. Then, a bit vector is converted into decimal number and is replaced with center pixel value. LBP descriptor of every pixel in an image is computed using (6) and f(s) is given in. (7).

$$LBP(N,R) = \sum_{N=0}^{N-1} f(I_n - I_c)\, 2^N \quad (6)$$

$$f(s) = \begin{cases} 1, & if\ s \geq 0 \\ 0, & Otherwise. \end{cases} \quad (7)$$

Where, I_n and I_c indicate intensity of neighboring and current pixel respectively. N represents the number of neighbors chosen at a radius of R. In this work, we choose P=8 neighbors with a radius R=1.

2.2. Support vector machines (SVM) classifier

SVM [41] are the most widely used and effective supervised machine learning algorithm for classification problems. It can be used to perform linear and non-linear classification. In this work, we perform non-linear classification, when the feature vectors could not be separated linearly. Radial basis function (RBF) kernel is chosen in the experimentation and it is described in (8). Where, ||u1-u2|| in (8) represent euclidean distance between two feature vector points u1 and u2 and 'σ' represent variance.

$$k(u1, u2) = exp\left(-\frac{||u1 - u2||^2}{2\sigma^2}\right) \quad (8)$$

2.3. CNN based deep learning techniques

Among other deep neural networks, CNN based deep learning techniques have shown its effectiveness by obtaining general features to specific features automatically based on the image content. CNN based pre-trained neural network models such as AlexNet [42], VGG (VGG16 and VGG19) [43], GoogLeNet [44], and ResNet [45], have shown great performance in classifying images into 1000 object categories such as pencil, keyboard, mouse, etc., and have become standard models for classification tasks. These models are trained on millions of images and have learnt rich feature representations for a wide range of PG images in the ImageNet database.

Training a deep ConvNet model from scratch takes several days or weeks or even months on a large dataset. A pre-trained neural network model would be a better choice to solve similar kinds of problems for a smaller dataset. In this work, two variants of VGG pre-trained neural network models such as VGG16 and VGG19 are adopted. Because, these neural network models have fixed size kernels which take less time to process and easily capture small patterns. Transfer learning is applied to perform classification on a new dataset which contains CG and PG images. It can be carried out in two ways: feature extraction and fine-tuning. Latter is adopted in our work and is performed by replacing the last three layers of pre-trained neural network models and these layers are fine-tuned for classification of CG images and PG images.

2.3.1. Visual geometry group (VGG) architecture

VGG architecture can be viewed as an input layer, feature extraction layers and classification layers. In the input layer, a color image of fixed size 227×227 is input to the architecture during training. The image is pre-processed by subtracting the mean RGB value from each pixel on the training set. During feature extraction, the image is moved through a series of convolutional layers, where the filters of fixed size 3×3 are used. Spatial padding and stride is fixed to 1 pixel which preserves the spatial dimension after convolution. The depth of the convolutional layers begins from 64 in the first layer and increases by a factor of 2 after every maximum pooling layer until it attains 512. Spatial dimension of the image is reduced by maximum pooling layers and this is done by using a filter of size 2 and a stride of 2. Five maximum pooling layers are used in the architecture which follows some convolutional layers. Classification layers consist of three fully connected layers: first two have 1024 neurons each and third contain two neurons and at the end sigmoid activation function is used to perform binary classification which produces the value in the range 0 and 1and it is described in (9). VGG variants CNN architecture is presented in Table 2 (the parameters of convolutional layers are denoted as conv(block)-(number of filters)_ layer number at each block. ReLU is not shown for brevity).

$$s(t) = \frac{1}{1 + e^{-x}} \quad (9)$$

For training, binary cross entropy loss function is used and is given in (10).

$$Binary\ cross\ entropy = -\frac{1}{M}\sum_{j=1}^{M} -(y_j \times log(p_j) + (1 - y_j) \times log(1 - p_j) \quad (10)$$

Where, M is the number of categories, $(1 - p_j)$ is the probability of class CG. $p_j$is the probability of class PG and y is the binary indicator (0 or 1) if category label is the correct classification for sample. Rectified linear unit (ReLU), a non-linear activation function, i.e. f(x)=max(0, x) is used in all hidden layers of VGG variants.

Table 2. VGG variant CNN configuration: output volume and parameters for VGG16 and VGG19 architecture

VGG variant CNN configuration					
VGG16	Output volume	Parameters	VGG19	Output volume	Parameters
Image input (227×227 color image)					
conv1-64_1	227×227×64	1792	conv1-64_1	227×227×64	1792
conv1-64_2	227×227×64	36928	conv1-64_2	227×227×64	36928
Maximum pooling					
conv2-128_1	113×113×128	73856	conv2-128_1	113×113×128	73856
conv2-128_2	113×113×128	147584	conv2-128_2	113×113×128	147584
Maximum pooling					
conv3-256_1	56×56×256	295168	conv3-256_1	56×56×256	295168
conv3-256_2	56×56×256	590080	conv3-256_2	56×56×256	590080
conv3-256_3	56×56×256	590080	conv3-256_3	56×56×256	590080
			conv3-256_4	56×56×256	590080
Maximum pooling					
conv4-512_1	28×28×512	1180160	conv4-512_1	28×28×512	1180160
conv4-512_2	28×28×512	2359808	conv4-512_2	28×28×512	2359808
conv4-512_3	28×28×512	2359808	conv4-512_3	28×28×512	2359808
			conv4-512_4	28×28×512	2359808
Maximum pooling					
conv5-512_1	14×14×512	2359808	conv5-512_1	14×14×512	2359808
conv5-512_2	14×14×512	2359808	conv5-512_2	14×14×512	2359808
conv5-512_3	14×14×512	2359808	conv5-512_3	14×14×512	2359808
			conv5-512_4	14×14×512	2359808
Maximum pooling					
FC – 1024					
FC – 1024					
FC – 1					
Sigmoid					
Trainable parameters (in millions)	-	41,343,873	-	-	46,653,569

3. RESULTS AND DISCUSSION

3.1. Dataset collection

'JSSSTU CG and PG image dataset' consists of image categories: CG and PG images with 7,000 samples in each class, containing diversified contents. CG images are collected from various reliable computer graphics websites and PG images are captured from different camera models (standalone, in-built mobile cameras) as the camera specifications for each model vary in terms of megapixel count, image quality, sensor type and so on. To improve the diversity of PG image contents, they are collected from other sources INRIA [46], ICCV09 [47], and McGill calibrated colour image database [48]. Contents of the CG image class include 3D model, architecture, cartoon, digital art, non-PRCG images, object, people, PRCG images, texture, trademark, vector maps and video gaming. PG image class cover a wide range of image contents: animals, buildings, man-made objects, indoor scenes, outdoor scenes, nature, vehicles and so on. 'JSSSTU PRCG image dataset' contains 2,000 samples which demonstrate high photo-realism.

Online sources used to create CG image dataset are given in Table 3. Camera models and other sources used to create PG image dataset are given in Tables 4 and 5. Various online sources used to create PRCG image dataset are given in Table 6. The aforementioned datasets are made publicly available to the research community at the following link: https://sites.google.com/view/hrchennamma/research-activities/jssstu-data-sets.

Table 3. Online sources used to create CG image dataset

Image class	Online sources		
Computer graphic images	https://pixbay.com	https://www.grabcad.com	http://www.3dlinks.com
	http://wallpaperlepi.com	https://freestocktextures.com	http://www.realsoft.com
	http://www.cgw.com	http://www.acitymap.com	www.digitalrepose.com
	http://www.cgsociety.org	http://www.cadnav.com	www.google.com/imghp?hl=EN
	https://free3d.com	https://www.nexusmods.com	
	http://fantasyartdesign.com	https://wallpaper.mob.org	

In addition to these datasets, existing benchmark datasets presented in [5]–[7], are used for the experimentation. Sample size pertaining to each class and each dataset are shown in Table 7. Image samples

from JSSSTU CG and PG image dataset and JSSSTU PRCG image dataset is shown in Figures 1(a)-(c) respectively.

All the images are resized to a dimension of 227×227 pixels. Textual information present in some of the computer graphic images is cropped. Datasets are randomly partitioned into 80% for training (70% for training and 10% for validation in case of pre-trained neural network model) and 20% for testing.

Table 4. Camera models used to create PG image dataset

Image class	Camera models
Photographic images	Canon PowerShot A2200, NIKON D7100, NokiaC6-01, Canon PowerShot SX200 IS, Vivo 1714, Canon PowerShot SD1100 IS, SAMSUNG GT-S7262, Lenovo K50a40, SONY DSC-WX7, Vivo 1718, Canon PowerShot A400, Canon EOS 1100D, Canon EOS 1000D.

Table 5. PG image sources used to create PG image dataset

PG image source	Count
Personal collection	4,019
INRIA	1,280
ICCV09	622
McGill	1,079
Total	7,000

Table 6. Online sources used to create JSSSTU PRCG image dataset

Image class	Online sources		
PRCG images	https://www.3dartistonline.com	https://www.gamespot.com	https://www.chaosgroup.com
	https://3dexport.com	https://gizmodo.com	https://app.easyrender.com
	https://archicgi.com	http://www.graphicmania.net	https://evermotion.org
	https://archvizcamp.com	https://lumion.com	https://www.freepik.com
	https://area.autodesk.com	https://www.maxon.net	https://www.ronenbekerman.com
	https://www.blenderguru.com	http://www.nextlimit.com	https://www.thearender.com
	https://www.cgmeetup.com	https://www.blog.poliigon.com	https://forums.unrealengine.com
	https://www.cgtrader.com	https://www.vizpark.com	http://www.3dlinks.com

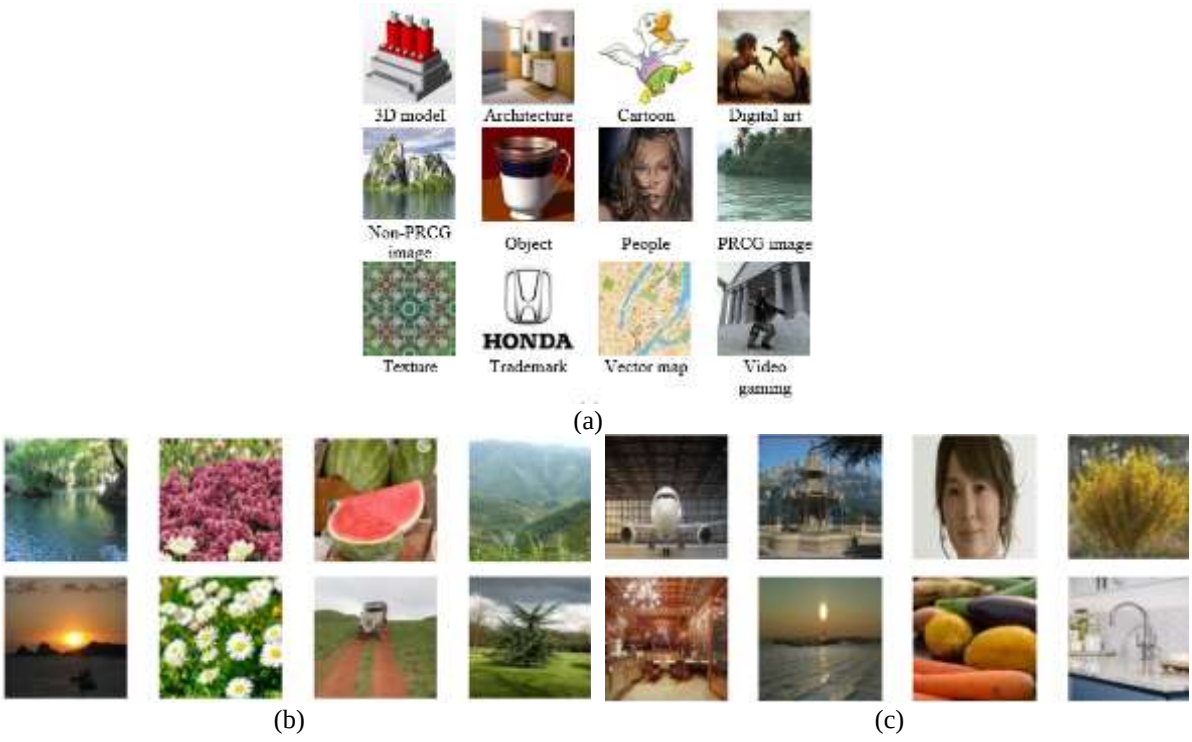


Figure 1. Image samples from JSSSTU CG and PG image dataset and JSSSTU PRCG image dataset (a) CG images, (b) PG images, and (c) PRCG images

Table 7. Datasets used in the experiment

Dataset	CG images	PG images	Dataset size
Columbia dataset	800	800	1,600
DSTok dataset	4,850	4,850	9,700
Rahmouni *et al.* dataset	1,800	1,800	3,600
JSSSTU CG and PG image dataset proposed	7,000	7,000	14,000
JSSSTU PRCG image dataset proposed	2,000	2,000	4,000

3.2. Experiments

3.2.1. Experiment setup for conventional machine learning techniques

Experiments are conducted using MATLAB R2018a with Intel Core-i3 4005U processor, 1.70 GHz and 8 GB RAM. Texture features such as GLCM and LBP are extracted independently from an image which consists of a feature dimension 4 and 59 respectively. SVM classifier is used in the experimentation.

3.2.2. Experiment setup for pre-trained neural network models

VGG variants (VGG16 and VGG19) are implemented with Google Colaboratory platform on the free 'Tesla K80 GPU' with 25 GB RAM using Keras. Regularization techniques such as early stopping (monitor=validation loss and patience=10), data augmentation and dropout (dropout probability 0.5) [49] are used to prevent the pre-trained neural network models from overfitting. Hyper-parameters such as stochastic gradient descent (SGD) optimizer with default momentum value of 0.9 and maximum number of epochs 100 are used for all datasets during training. Other hyper-parameters such as batch size and initial learn rate and learn rate drop factor used for different datasets are given:

- Columbia dataset, DSTok Dataset, Rahmouni *et al.* Dataset and JSSSTU CG and PG image dataset
  VGG variants (VGG16 and VGG19): batch size of 32 images, an initial learn rate of 1e-4, learn rate drop factor (monitor=validation loss, factor=0.1, and patience=5) are used.
- JSSSTU PRCG image dataset
  VGG variants (VGG16 and VGG19): batch size of 16 images, an initial learn rate of 1e-4 and learn rate drop factor (monitor=validation loss, factor=0.1, and patience=5) are used.
- Data augmentation
  To increase the diversity in content of images, we employed data augmentation techniques such as translation, rotation, shear, reflection and zooming. These techniques are used during training of VGG16 and VGG19 pre-trained CNN on JSSSTU CG and PG image dataset, JSSSTU PRCG image dataset and DSTok Dataset respectively. Aforementioned random transformations help the model to expose to more aspects of data and yield better generalization.

3.2.3. Experiment results

Average classification accuracies obtained using handcrafted texture features and pre-trained CNN on our new datasets are tabulated in Table 8. As shown in Table 8, CNN based pre-trained techniques outperformed the classification accuracy performance against the conventional SVM-based classifier. VGG19 has attained better classification results when compared to the handcrafted texture features and VGG16.

Table 8. Average classification accuracies of handcrafted texture features and pre-trained CNN on our new datasets

Dataset	Average classification accuracy in %			
	Handcrafted features		Pre-trained CNN	
	GLCM	LBP	VGG16	VGG19
JSSSTU CG and PG image dataset	63.67	85.75	94.42	94.46
JSSSTU PRCG image dataset	63.50	75.62	89.12	89.37

3.2.4. Comparative analysis of benchmark datasets used to evaluate classification models

Existing methods based on conventional machine learning and deep learning are used to compare their performances on existing benchmark datasets. Average accuracies attained are listed in Table 9. Performance of the methods is given from highest to lowest. As seen from Table 9, VGG19 pre-trained CNN has achieved cent percent classification results on Columbia dataset. Feature fusion method based on conventional machine learning proposed by Tokuda *et al.* has obtained better identification accuracy on DSTok dataset. Further, techniques presented in [29], [34], [35] have attained cent percent accuracy on a Rahmouni *et al.* dataset.

Table 9. Comparative analysis of benchmark datasets used to evaluate classification models

Columbia dataset		DSTok dataset		Rahmouni *et al.* dataset	
Method	Average accuracy (%)	Method	Average accuracy (%)	Method	Average accuracy (%)
VGG19	100	Tokudal *et al.* [6]	97	Yao *et al.* [35]	100
VGG16	99.37	Ming He [30]	96	Chawla *et al.* [34]	100
Cui *et al.* [31]	98	Rezende *et al.* [33]	94	Nguyen *et al.* [29]	100
LBP	97.50	VGG19	90.92	VGG19	95.96
Fan *et al.* [25]	93.51	VGG16	89.94	VGG16	95.42
GLCM	78.75	LBP	75.61	Rahmouni *et al*. [7]	93.2
		GLCM	61.85	LBP	87.22
				GLCM	66.52

3.2.5. Performance metrics

The metrics such as precision, recall and f-score [50] are used to assess the performance of VGG19 pre-trained neural network model as it yields best classification accuracy against handcrafted features and VGG16 pre-trained neural network model on existing and proposed datasets. Macro average of aforementioned metrics is computed for two classes. Table 10 shows the evaluation metrics used to assess the performance of VGG19 pre-trained neural network model on existing and proposed datasets.

As seen from Table 10, low f-score is obtained on JSSSTU PRCG image dataset when compared to other datasets. The difference in f-score of JSSSTU CG and PG image dataset and DSTok dataset is only 0.3. Hence, we conclude that, our new datasets are very challenging and the DSTok dataset is as good as JSSSTU CG and PG image dataset but it is lacking with larger dataset size, contain limited number of PRCG images and images produced using recent rendering technology.

Table 10. Performance metrics used to assess the performance of VGG19 pre-trained neural network model on different datasets

Dataset	Performance metrics		
	Precision	Recall	F-score
Columbia Dataset	1	1	1
Rahmouni *et al*. Dataset	0.96	0.96	0.96
JSSSTU CG and PG image dataset – Proposed	0.94	0.94	0.94
DSTok Dataset	0.91	0.91	0.91
JSSSTU PRCG image dataset – Proposed	0.89	0.89	0.89

4. CONCLUSION

This work is aimed at creating two new datasets, namely 'JSSSTU CG and PG image dataset' a heterogeneous dataset which comprises 14,000 samples and 'JSSSTU PRCG image dataset' which exhibits photo-realism with 2,000 samples. Further, we implemented state-of-the-art techniques based on handcrafted texture features and deep learning. Performance of these techniques is evaluated on our new datasets and benchmark datasets. Experimental results show that CNN based pre-trained techniques outperformed the classification accuracy performance against the conventional SVM-based classifier. Further, we found that the choice of handcrafted features used for classification has achieved better results on the Columbia Dataset when compared to other benchmark datasets and our new datasets. The performance of VGG19 pre-trained neural network technique has attained significant results on 'JSSSTU CG and PG image dataset' but still the accuracy can be improved. On the other hand, its performance on 'JSSSTU PRCG image dataset' has achieved low detection rate due to the high-realism images present in the dataset. Hence, an efficient and robust technique is needed to solve this problem and our new datasets will be helpful for the researchers who are working on the cutting-edge research problem: "classification of computer graphic images and photographic images" to evaluate their classification models. To the best of our knowledge, these kinds of datasets do not exist in the literature.

REFERENCES

[1] "Applications of computer graphics." https://www.geekforgeeks.org/applications-of-computer-graphics/ (accessed Jan. 05, 2021).
[2] O. Holmes, M. S. Banks, and H. Farid, "Assessing and improving the identification of computer-generated portraits," *ACM Transactions on Applied Perception*, vol. 13, no. 2, pp. 1–12, Mar. 2016, doi: 10.1145/2871714.
[3] S. Lyu and H. Farid, "How realistic is photorealistic?," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 845–850, Feb. 2005, doi: 10.1109/TSP.2004.839896.
[4] F. Peng and D. Zhou, "Discriminating natural images and computer generated graphics based on the impact of CFA interpolation on the correlation of PRNU," *Digital Investigation*, vol. 11, no. 2, pp. 111–119, Jun. 2014, doi: 10.1016/j.diin.2014.04.002.
[5] T. Ng, S. Chang, J. Hsu, and M. Pepeljugoski, "Columbia photographic images and photorealistic computer graphics dataset," *Columbia University, ADVENT Tecnical Report*. pp. 1–23, 2004.

[6] E. Tokuda, H. Pedrini, and A. Rocha, "Computer generated images vs. digital photographs: a synergetic feature and classifier combination approach," *Journal of Visual Communication and Image Representation*, vol. 24, no. 8, pp. 1276–1292, Nov. 2013, doi: 10.1016/j.jvcir.2013.08.009.
[7] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen, "Distinguishing computer graphics from natural images using convolution neural networks," in *2017 IEEE Workshop on Information Forensics and Security, WIFS 2017*, Dec. 2017, pp. 1–6, doi: 10.1109/WIFS.2017.8267647.
[8] S. Dehnie, T. Sencar, and N. Memon, "Digital image forensics for identifying computer generated and digital camera images," in *2006 International Conference on Image Processing*, Oct. 2006, pp. 2313–2316, doi: 10.1109/ICIP.2006.312849.
[9] A. E. Dirik, S. Bayram, H. T. Sencar, and N. Memon, "New features to identify computer generated images," in *2007 IEEE International Conference on Image Processing*, 2007, vol. 4, pp. 433–436, doi: 10.1109/ICIP.2007.4380047.
[10] N. Khanna, G. T. C. Chiu, J. P. Allebach, and E. J. Delp, "Forensic techniques for classifying scanner, computer generated and digital camera images," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Mar. 2008, pp. 1653–1656, doi: 10.1109/ICASSP.2008.4517944.
[11] F. Peng, J. Shi, and M. Long, "Identifying photographic images and photorealistic computer graphics using multifractal spectrum features of PRNU," in *2014 IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2014, pp. 1–6, doi: 10.1109/ICME.2014.6890296.
[12] M. Long, F. Peng, and Y. Zhu, "Identifying natural images and computer generated graphics based on binary similarity measures of PRNU," *Multimedia Tools and Applications*, vol. 78, no. 1, pp. 489–506, Jan. 2019, doi: 10.1007/s11042-017-5101-3.
[13] F. Pan, J. Chen, and J. Huang, "Discriminating between photorealistic computer graphics and natural images using fractal geometry," *Science in China, Series F: Information Sciences*, vol. 52, no. 2, pp. 329–337, Feb. 2009, doi: 10.1007/s11432-009-0053-5.
[14] R. Wu, X. Li, and B. Yang, "Identifying computer generated graphics VIA histogram features," in *2011 18th IEEE International Conference on Image Processing*, Sep. 2011, pp. 1933–1936, doi: 10.1109/ICIP.2011.6115849.
[15] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, Jul. 2002, doi: 10.1109/TPAMI.2002.1017623.
[16] Z. Li, J. Ye, and Y. Q. Shi, "Distinguishing computer graphics from photographic images using local binary patterns," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7809, Springer Berlin Heidelberg, 2013, pp. 228–241.
[17] Z. Li, Z. Zhang, and Y. Shi, "Distinguishing computer graphics from photographic images using a multiresolution approach based on local binary patterns," *Security and Communication Networks*, vol. 7, no. 11, pp. 2153–2159, Nov. 2014, doi: 10.1002/sec.929.
[18] F. Peng, J. ting Li, and M. Long, "Identification of natural images and computer-generated graphics based on statistical and textural features," *Journal of Forensic Sciences*, vol. 60, no. 2, pp. 435–443, Mar. 2015, doi: 10.1111/1556-4029.12680.
[19] D. Q. Tan, X. J. Shen, J. Qin, and H. P. Chen, "Detecting computer generated images based on local ternary count," *Pattern Recognition and Image Analysis*, vol. 26, no. 4, pp. 720–725, Oct. 2016, doi: 10.1134/s1054661816040167.
[20] F. Peng, D. Zhou, M. Long, and X. Sun, "Discrimination of natural images and computer generated graphics based on multi-fractal and regression analysis," *AEU - International Journal of Electronics and Communications*, vol. 71, pp. 72–81, Jan. 2017, doi: 10.1016/j.aeue.2016.11.009.
[21] Y. Wang and P. Moulin, "On discrimination between photorealistic and photographic images," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2006, vol. 2, pp. II-161-II–164, doi: 10.1109/icassp.2006.1660304.
[22] D. Chen, J. Li, S. Wang, and S. Li, "Identifying computer generated and digital camera images using fractional lower order moments," in *2009 4th IEEE Conference on Industrial Electronics and Applications, ICIEA 2009*, May 2009, pp. 230–235, doi: 10.1109/ICIEA.2009.5138202.
[23] R. Zhang and R. Wang, "Distinguishing photorealistic computer graphics from natural images by imaging features and visual features," in *2011 International Conference on Electronics, Communications and Control (ICECC)*, Sep. 2011, pp. 226–229, doi: 10.1109/ICECC.2011.6067631.
[24] K. Guo and R. Wang, "A new method for detecting computer-generated images based on multiwavelets," *Journal of Information and Computational Science*, vol. 8, no. 8, pp. 1449–1456, 2011.
[25] S. Fan, R. Wang, Y. Zhang, and K. Guo, "Classifying computer generated graphics and natural images based on image contour information," *Journal of Information and Computational Science*, vol. 9, no. 10, pp. 2877–2895, 2012.
[26] G. K. Birajdar and V. H. Mankar, "Computer graphic and photographic image classification using local image descriptors," *Defence Science Journal*, vol. 67, no. 6, pp. 654–663, Nov. 2017, doi: 10.14429/dsj.67.10079.
[27] J. Wang, T. Li, Y.-Q. Shi, S. Lian, and J. Ye, "Forensics feature analysis in quaternion wavelet domain for distinguishing photographic images and computer graphics," *Multimedia Tools and Applications*, vol. 76, no. 22, pp. 23721–23737, Nov. 2017, doi: 10.1007/s11042-016-4153-0.
[28] J. Wang, T. Li, X. Luo, Y. Q. Shi, and S. K. Jha, "Identifying computer generated images based on quaternion central moments in color quaternion wavelet domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 9, pp. 2775–2785, Sep. 2019, doi: 10.1109/TCSVT.2018.2867786.
[29] H. H. Nguyen, N. D. T. Tieu, H. Q. Nguyen-Son, V. Nozick, J. Yamagishi, and I. Echizen, "Modular convolutional neural network for discriminating between computer-generated images and photographic images," in *ACM International Conference Proceeding Series*, Aug. 2018, pp. 1–10, doi: 10.1145/3230833.3230863.
[30] M. He, "Distinguish computer generated and digital images: a CNN solution," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 12, Jun. 2019, doi: 10.1002/cpe.4788.
[31] Q. Cui, S. McIntosh, and H. Sun, "Identifying materials of photographic images and photorealistic computer generated graphics based on deep CNNs," *Computers, Materials and Continua*, vol. 55, no. 2, pp. 229–241, 2018, doi: 10.3970/cmc.2018.01693.
[32] K. B. Meena and V. Tyagi, "A deep learning based method to discriminate between photorealistic computer generated images and photographic images," in *Communications in Computer and Information Science*, Springer Singapore, 2020, pp. 212–223.
[33] E. R. S. de Rezende, G. C. S. Ruppert, A. Theóphilo, E. K. Tokuda, and T. Carvalho, "Exposing computer generated images by using deep convolutional neural networks," *Signal Processing: Image Communication*, vol. 66, pp. 113–126, Aug. 2018, doi: 10.1016/j.image.2018.04.006.
[34] C. Chawla, D. Panwar, G. S. Anand, and M. P. S. Bhatia, "Classification of computer generated images from photographic images using convolutional neural networks," in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, Oct. 2018, pp. 1053–1057, doi: 10.1109/ICACCCN.2018.8748829.

[35] Y. Yao, W. Hu, W. Zhang, T. Wu, and Y.-Q. Shi, "Distinguishing computer-generated graphics from natural images based on sensor pattern noise and deep learning," *Sensors*, vol. 18, no. 4, Apr. 2018, doi: 10.3390/s18041296.
[36] W. Quan, K. Wang, D.-M. Yan, and X. Zhang, "Distinguishing between natural and computer-generated images using convolutional neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2772–2787, Nov. 2018, doi: 10.1109/TIFS.2018.2834147.
[37] P. He, X. Jiang, T. Sun, and H. Li, "Computer graphics identification combining convolutional and recurrent neural networks," *IEEE Signal Processing Letters*, vol. 25, no. 9, pp. 1369–1373, Sep. 2018, doi: 10.1109/LSP.2018.2855566.
[38] P. He, H. Li, H. Wang, and R. Zhang, "Detection of computer graphics using attention-based dual-branch convolutional neural network from fused color components," *Sensors*, vol. 20, no. 17, pp. 1–15, Aug. 2020, doi: 10.3390/s20174743.
[39] K. B. Meena and V. Tyagi, "Distinguishing computer-generated images from photographic images using two-stream convolutional neural network," *Applied Soft Computing*, vol. 100, Mar. 2021, doi: 10.1016/j.asoc.2020.107025.
[40] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 6, pp. 610–621, Nov. 1973, doi: 10.1109/TSMC.1973.4309314.
[41] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/bf00994018.
[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.
[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2015, [Online]. Available: http://arxiv.org/abs/1409.1556.
[44] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[46] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5302, Springer Berlin Heidelberg, 2008, pp. 304–317.
[47] S. Gould, R. Fulton, and D. Koller, "Decomposing a scene into geometric and semantically consistent regions," in *2009 IEEE 12th International Conference on Computer Vision*, Sep. 2009, pp. 1–8, doi: 10.1109/ICCV.2009.5459211.
[48] "McGill calibrated colour image database," *McGill Vision Research*. http://tabby.vision.mcgill.ca/html/browsedownload.html (accessed Jan. 10, 2021).
[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[50] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.

BIOGRAPHIES OF AUTHORS

Halaguru Basavarajappa Basanth Kumar is currently working as Assistant Professor in the Department of Studies in Computer Science, SBRR Mahajana First Grade College (Autonomous), PG Wing, Mysuru. He received MCA from Visvesvaraya Technological University, Karnataka, India in 2007. He is currently pursuing Ph.D. at Sri Jayachamarajendra College of Engineering, Visvesvaraya Technological University, Karnataka, India. His research interest is machine learning. He can be contacted at email: basanth.10@gmail.com.

Haranahalli Rajanna Chennamma received Ph.D. in Computer Science from the University of Mysore in the area of Digital Image Forensics in 2011. Subsequently, she served as a Post Doctoral fellow in the Department of Computer Science and Engineering, University of North Texas, USA in 2012. Currently, she is an Associate Professor in the Department of Computer Applications, JSS Science and Technology University, Mysuru. Previously, she has been awarded a prestigious Fellowship under the scheme of Research and Development in Forensics from the Department of Home Affairs, Government of India in 2005. Chennamma served as a Senior Research Fellow (SRF) in Central Forensic Science Laboratory, Hyderabad. She served as a Project Trainee for a year at the National Aerospace Laboratory (NAL), Bangalore and she also served as a software engineer for a year in a multinational software company, Bangalore. She is the recipient of two "Best Scientific Paper Awards". Her research areas of interest are Image and Video forensics, pattern recognition and Computer Vision. She can be contacted at email: hrchennamma@jssstuniv.in.

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 148~160
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i1.pp148-160

Systematic development of real-time driver drowsiness detection system using deep learning

Tarig Faisal, Isaias Negassi, Ghebrehiwet Goitom, Mohammed Yassin, Anees Bashir, Moath Awawdeh
Department of Electrical Engineering, Higher Colleges of Technology, Abu Dhabi, United Arab Emirates

Article Info

Article history:

Received May 30, 2021
Revised Dec 14, 2021
Accepted Dec 30, 2021

Keywords:

Classification
Convolutional neural network
Driver drowsiness
Haar cascade
Hyperparameters optimization

ABSTRACT

Advancements in globalization have significantly seen a rise in road travel. This has also led to increased car accidents and fatalities, which become a global cause of concern. Driver's behavior, including drowsiness, contributes to many of the road deaths. The main objective of this study is to develop a system to diminish mishaps caused by the driver's drowsiness. Recently deep convolutional neural networks have been used in multiple applications, including identifying and anticipate driver drowsiness. However, limited studies investigated the systematic optimization of convolutional neural networks (CNNs) hyperparameters, which could lead to better anticipation of driver drowsiness. To bridge this gap, a holistic approach based on the deep learning method is proposed in this paper to anticipate the drivers' drowsiness and provide an alerting mechanism to prevent drowsiness related accidents. To ensure optimal performance achievement by the system, a database of real-time images preprocessed via Haar cascade's classifiers is used to systematically optimize the CNN model's hyperparameters. Different metrics, including accuracy, precision, recall, F1-score, and confusion matrix, are used to evaluate the performance of the model. The training evaluation results of the optimal model achieved an accuracy of 99.87%, while the testing results accurately classify the drowsy driver with 97.98%.

Corresponding Author:

Tarig Faisal
Department of Electrical Engineering, Higher Colleges of Technology
Ruwais, Al Dhafrah, Abu Dhabi, United Arab Emirates
Email: tfaisal@hct.ac.ae

1. INTRODUCTION

With advancements in world development, vehicular movement is becoming prominent, contributing significantly to traffic accidents, traffic injuries, and loss of lives. Various human factors contribute to these accidents, including the recklessness and fatigue of drivers. Drowsiness related accidents have all the earmarks of being more dangerous because of the higher speeds involved in distraction and the driver being unable to take any counteraction. The World Health Organization reported that more than 1.35 million are killed every year, and up to 50 million are injured in road traffic crashes [1]. It's reported by the National Highway Traffic Safety Administration (NHTSA) that drowsy driving contributed to 2.3% to 2.5% of all fatal crashes [2]. The development of a system for the recognition or identification of driver drowsiness is crucial for accident prevention and road safety at large. In general, the loss of awareness due to tiredness causes a few changes in the human body and activities. These side effects motivated many researchers to develop strategies to measure the drowsiness level and monitoring of the behavior of drivers [3]–[5].

Object detection is one of the most common tasks in computer vision and states to determine the existence or absence of prominent features in image data. Once the features are detected, an object is

classified as belonging to one of a predefined set of classes. This later operation is known as object classification. Object detection and object classification are fundamental building blocks of artificial intelligence. The introduction of the deep learning technique has created a significant impact on object detection and classification due to its comparatively high accuracy and speed [6]–[8]. Convolutional neural networks (CNN) is one type of deep learning (DL) model for the processing of data like the detection and classification that has a grid pattern. Inspired by the animal visual cortex organization [9], [10], CNN is designed to learn features from low- to high-level patterns. Deep convolutional neural networks have been used recently in multiple applications for identifying and anticipate driver drowsiness [11]–[13]. In order to detect the driver's drowsiness, a deep neural network-based model is proposed in [13]. The driver's face was extracted from the video and passed as input to the model containing three stages of CNN, a convolutional control gate-based recurrent neural network (ConvCGRNN), and a voting layer. In the proposed system, the CNN layers interpret various facial expressions based on the global data sets while the ConvCGRNN determines the temporal dependencies, and finally, the voting layer determines the level of drowsiness. Experimental analysis based on real-time environment showed an accuracy of almost 85%. For identifying the sleepiness of drivers, the CNN based recurrent neural network (RNN) and long short-term memory (LSTM) were found to be very successful [12]. In this study, the information was extracted from the input images through the CNN and the features were interpreted across consecutive frames via the LSTM. The system achieved an accuracy of 88.5%. In [11], object recognition was used along with deep learning in order to detect the distraction and drowsiness of drivers in real time. Ear aspect ratio (EAR) and mouth aspect ratio (MAR) were computed using 68 facial landmarks to identify the driver's sleep probability. Object detection technique, namely, single shot multi-box detector (SSD) based on convolution neural networks, was used to determine the objects or environmental causes that may cause a distraction to the driver. The system offered an accuracy of 90% under experimental conditions.

Viola-Jones object detection framework proposed in 2001 [14] is a machine-learning object detection technique that has been used successfully to identify objects in an image or video. In drowsiness detection, the Viola–Jones object detection framework was implemented in many applications [15], [16]. A real-time system to detect drowsiness using Andriod mobile application was discussed in [15]. The study used Haar feature-based cascades for extracting landmark coordination from images and transfer them to the multilayer perceptron classifiers to identify drowsy drivers. 80% accuracy was achieved by the proposed model. One of the aspects of monitoring the behavior of the driver for drowsiness or sleepiness is yawing detection. Using Viola-Jones computational theory, a smart camera with computer vision was utilized to detect yawning by detecting the changes in the driver's mouth [16]. The model was tested on a real platform named CogniVue APEX, which shows improvements in reliability and accuracy when compared to existing methods in the detection of yawning. In this study, the camera for yawning detection was placed on the front mirror or the dashboard, and its output was given to the embedded platform, which displays the results on the monitor. The combination of CNN and Viola–Jones object detection Framework has also been in some applications for drowsiness detection [17], [18]. Viola-Jones face and eye detection algorithm along with deep CNN were used to develop a system for the identification of driver drowsiness [18]. 96.42% accuracy achieved by the proposed system utilizing 2,850 images for training, validation, and testing data. A similar technique with more features was used by [17]. The system achieved an accuracy of 88% for subjects with glasses and more than 85% for category night without glasses. The study indicated that on average more than 83% accuracy was achieved in all categories.

Even though some of those studies have achieved good results for identifying and anticipate driver drowsiness, however, limited studies have investigated in-depth systematic optimization of the CNN hyperparameters, which could lead to higher drowsiness identification performance as well as practical implementation. Accordingly, the main objective of this study is to develop an effective driver drowsiness identification system considering the optimization of the CNN hyperparameters as well as the incorporation of the Viola–Jones object detection framework. The developed system will assist in diminishing avoidable mishaps cased by the driver's drowsiness and save many lives, and this has served as one of our motivations to propose this work. The paper has four sections. In the first section, works related to driver drowsiness detection techniques were reviewed. The methodology in the second section discusses the proposed system, implemented techniques, database, and system evaluation methods. The third section presents the results and is followed by the discussion and conclusion.

2. METHODOLOGY

2.1. Proposed system

In this research, the primary aim is to develop a Real-Time driver drowsiness detecting system based CNN system to identify whether the driver is drowsy or not. The proposed system flowchart is shown in Figure 1. As shown in the figure, the system starts by receiving inputs from a camera mounted in front of

the driver. The camera continuously feeds a stream of frames of the face of the driver. Then the face feature of the driver is detected, and the location of the eyes is identified. The eye's location is then extracted from the face and preprocessed. The processed image is fed to a trained CNN model to classify it into two categories: “open eyes state” and “closed eyes state.” Based on predefined criteria, the system determines whether the output obtained from the CNN model belongs to a drowsy driver or not. Finally, a prototype alert system/mechanism is used to notify the driver in case the driver is found drowsy.

Figure 2 shows the process followed for identifying the optimal CNN model. The first phase starts with the image extraction and preprocessing, which was accomplished using the Viola-Jones object detection framework. The second stage is identifying the optimal CNN hyperparameters, leading to determining the optimal model that accomplishes the best performance. In this stage, four hyperparameters, namely kernal size, Maximum pooling size, learning rate, and a number of epochs, were investigated. Finally, the system was tested experimentally using a real-time date.

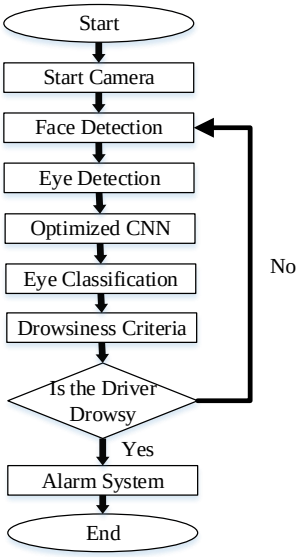


Figure 1. Overall flowchart of the system

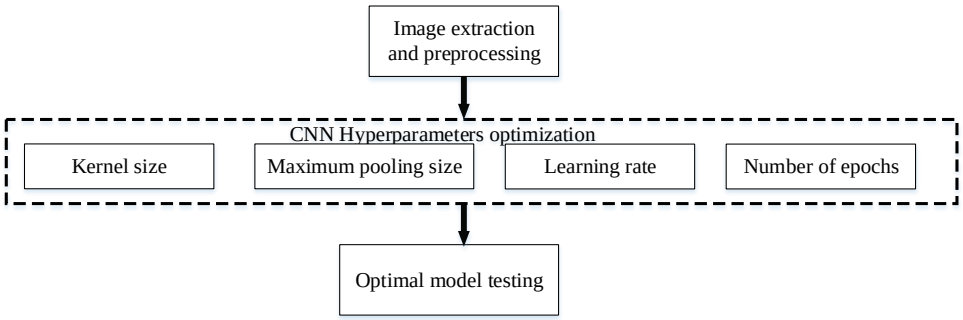


Figure 2. CNN hyperparameters optimization

2.2. Image extraction, preprocessing, and database

The flowchart for the procedures of collecting and preprocessing the database is shown in Figure 3. In preparing our data set, we used 20 volunteers from those 7 of them were with eyeglasses. Initially, two types of videos were recorded for each person; one when they are awake and the other when they are feeling drowsy. From the recorded videos of each person, eye images were detected and extracted using Viola-Jones object detection Framework and our advanced formula, which will be discussed later. In total, 4,000 images per person were collected via the camera. Each extracted image has a size of 300×300 pixels. In addition, 20,000 more foreign images are added from the internet with good resolution [19], to obtain better accuracy.

In order for the image to fit the proposed model, the size of the image should be reduced. Therefore, the Numpy library was used to convert the images to an array with a dimension of 100×100 pixels. Then the dataset was normalized to expedite the training. The normalization of the dataset was performed by dividing the RGB image pixel by 255 to get a grayscale image that simplifies the algorithm and reduces the computational requirements. The OPEN CV library with python was used to implement the Viola-Jones object detection framework with the Haar cascade algorithm.

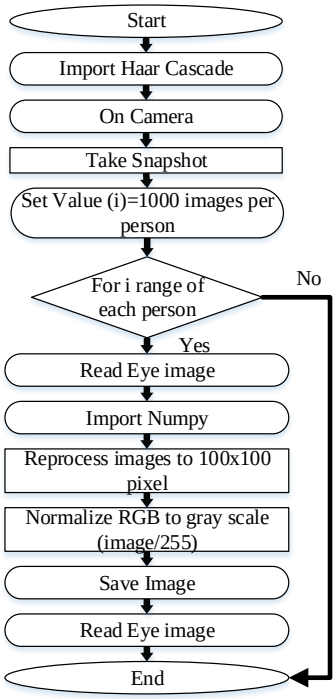


Figure 3. Dataset flowchart

The haar cascade algorithm was used for face detection. Then the eyes, which are the region of interest (ROI) was estimated and extracted since we are interested only in the eye portion of the image rather than the whole face because they are more informative. Haar algorithm provides four parameters to identify the position of the faces namely x_0, y_0, W and H, x_0, and y_0 are the starting point for the used rectangle to identify the face. W is the width of the face rectangle, formed by the horizontal distance from the point (x_0, y_0) to the right. H is the height of the face rectangle, which formed by the vertical distance from the point (x_0, y_0) downward direction. Figure 4 displays those parameters.

As shown in the Figure 4, to estimate the eyes region, the following positions need to be identified: (x_1, y_1) which is the starting point for the eye rectangle, the second vertex point (x_2, y_1) found to the right of the starting point, and the third vertex point (x_1, y_2) found below the starting point. Those positions of the three vertices are enough to draw a rectangle around the eyes. In this study, (1)-(4) is used to identify the eye's rectangular vertices based on the four parameters obtained from the Haar cascade algorithm. The coefficients shown in the above equations were determined empirically in this study.

$$x_1 = x_0 + 0.084\text{W} \quad (1)$$

$$x_2 = x_0 + 0.916\text{W} \quad (2)$$

$$y_1 = y_0 + 0.25\text{H} \quad (3)$$

$$y_2 = y_0 + 0.5\text{H} \quad (4)$$

It is worth mentioning that the estimated and extracted eyes region size and position change proportionally with the identified face since the size of the detected face changes with the distance of the person in front of the camera (the identified face rectangle enlarges if a person is close to the camera and vice

versa). Finally, only the identified eye ROI was extracted from the face and fed to the CNN for training and testing. Figure 5 shows samples from the database of extracted awake and sleep images. Accordingly, around 100,000 images were collected. From these, 62,000 images were without eyeglasses, and the rest 38000 images were collected with eyeglasses to ensure the system's efficiency. 80% of the data were used for the training process and 10% for the validating; the remaining 10% was used for the testing dataset.

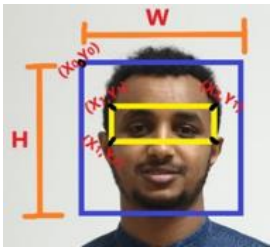


Figure 4. Sample photo showing parameters and points used for faces and eyes

Figure 5. Randomly taken samples from the dataset of wake and sleep

2.3. Convolutional neural network structure

Convolutional layers, pooling layers, and fully connected layers are the main types of layers included in the CNN. It primarily starts with the convolutional layer, which is the most important layer in CNN. In this layer, the basic different visual features are extracting from the input image. Two different operations are performed in the Convolutional layer; linear operations are represented by the convolution operation, and the activation function reflects the nonlinear operations. The convolution operation aims to extract different input image features by kernel convolving (sliding) throughout the whole input image.

The convolution process begins from the left-hand corner top portion of the input image and gradually and sequentially moves to the top right and then moves downwards in the same manner, covering the entire image. The stride determines the number of elements that would be convolved by the kernel. Between the tensor (which refers to the portion of the input image) and the kernel, an element-wise multiplication and accumulation operation would occur at every stride. The feature map, $Y(i,j)$ would be produced from the output values of this operation and is defined by the following equation,

$$Y(i,j) = (I * F)\,[i,j] = \sum_a \sum_b I[a,b]F\,[i-a, j-b] \quad (5)$$

where F represents the kernel, I represents the input image, j is the convolved matrix's index columns, and i is the convolved matrix's index rows.

The extracted features are determined by the number of kernels; 1 mapped feature would be produced from 1 kernel, whereas T mapped features would be produced from T kernels. Hence the input image, kernel size, and stride size would determine the size of the feature map. If the input image I has a size of $m \times n$, the Kernel (filter) F has a size of $u \times v$, and the number of strides is g, the Y feature map size is calculated as:

$$Y(i,j) = Y\left(\frac{m-u}{g} + 1,\ \frac{n-v}{g} + 1\right) \quad (6)$$

The activation function is the nonlinear operation of the CNN, which represents the second stage of the convolution operation. It is used to improve the learning capability of the neural network for any complex data based on the input. Although many activation functions exist, the rectified linear unit (ReLU) is a very popular type used as it is speedy and efficient [20]. It gives a null value when the input value is negative, and it is represented by the (7):

$$H(z) = \begin{cases} 0 & if\ z < 0 \\ z & if\ z \geq 0 \end{cases} \quad (7)$$

where z is the input of the function.

The pooling layer is located between the convolution layers. It is used to preserve the significant information that minimizes the computation time for the following layer and reducing the features map.

Likewise, the overfitting of the network is also distorted when the learnable parameters are reduced [21], [22]. Several pooling techniques have been used, including maximum pooling, average pooling, and sum pooling. To discard the insignificant features and acquire the maximum value within the pooling window, the maximum pooling is used in this study.

The above convolutional and pooling process is shown in Figure 6, which is the schematic representation of the network architecture used in our proposed system to identify the drowsiness drivers. The network comprises three convolutional blocks (gray) and three max-pooling (orange) layers. An input image of size 100×100 pixels is convolved with thirty-two 10×10 filters (with unitary strides) to produce thirty-two 100×100 feature maps (C1 in gray). These feature maps are then passed to 15×15 max-pooling operations that downsize the feature maps to thirty-two 64×64 feature maps (P1 in orange). Next, these feature maps are passed through the second convolution layer of sixty-four 10×10 filters (with unitary strides), yielding sixty-four 32×32 features maps (C2 in gray). The second 15×15 max-pooling operations downsize the produced feature maps to sixty-four 22×2 feature maps (P2 in orange). The last convolution operation resulted in two hundred fifty-six 18×18 feature maps (C3 in gray), which is then downsized to two hundred fifty-six 9×9 feature maps (P3 in orange) using the last pooling operation. The nonlinear operation in all convolution blocks is performed using the ReLU activation functions. The third and final layer of the CNN is the fully connected or dense layer. It receives the input volume of information from the output of the preceding layer and generates a Q-dimensional vector output by calculating the probability score. Q represents the number of output classes, which is 2 classes for our study. This process is accomplished by flattening the input volume received from the preceding layer to a vector which is then fed to single or multiple fully-connected layers. The fully connected layers comprise multiple neurons fully connected to each and every input and output through learnable weight. The fully-connected layers comprise multiple neurons entirely connected to all inputs and outputs via learnable weight. The fully-connected layers' output is enjoined to an activation function that yields Q-dimensional vector probability scores output equivalent to the classification target by normalizing the outputs. One of the most commonly used activation functions is the Softmax, and its output estimation is given in the (8).

$$\emptyset(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \quad for\ i = 1, \dots, K \tag{8}$$

where $\emptyset()$ represents Softmax activation function, x is the input and K in the number of the classes.

The above process is presented in the seventh layer in the proposed system, as shown in Figure 6. The feature maps produced from the sixth layer are flattened passed through one fully connected layer with 512 nodes. Finally, the states of the eye open or close take place in the output layer via a softmax activation function. Table 1 summarize CNN layers parameters.

CNN training aims to reduce the variance between the desired and the generated outputs by obtaining suitable learnable parameters values, including kernels, weights, and biases. Calculating these variances can be accomplished by many types of loss functions; however, mean square error (MSE) loss functions are used in our application, as it is the most commonly used. The MSE can be calculated by (9):

$$\Psi = \frac{1}{r} \sum_{n=1}^{r} \left(O_n - \hat{O}_n\right)^2 \tag{9}$$

where r is the number of the samples, $\hat{O}_n$ is the desired output and O_n is the calculated output.

The training is commonly performed using the stochastic gradient descent (SGD) and the stander backpropagation [23]. The SGD is a commonly used optimization algorithm since other algorithms is computationally expensive [19], [21]. The training starts by forward propagation, of which the MSE is calculated and followed by the back propagation, where the learnable parameters are updated. It worth mentioning that the selected training algorithm determines the ways to update the learnable parameters. The iteration is increased by one after all the parameters are updated. The preceding steps are repeated until a given number of iterations are completed or a satisfactorily small MSE value is attained. The SGD is updating the learnable parameters to reduce the loss function using the following formula:

$$P \leftarrow P - \xi \frac{\partial \Psi}{\partial P} \tag{10}$$

where P is the learnable parameters, and ξ is the learning rate.

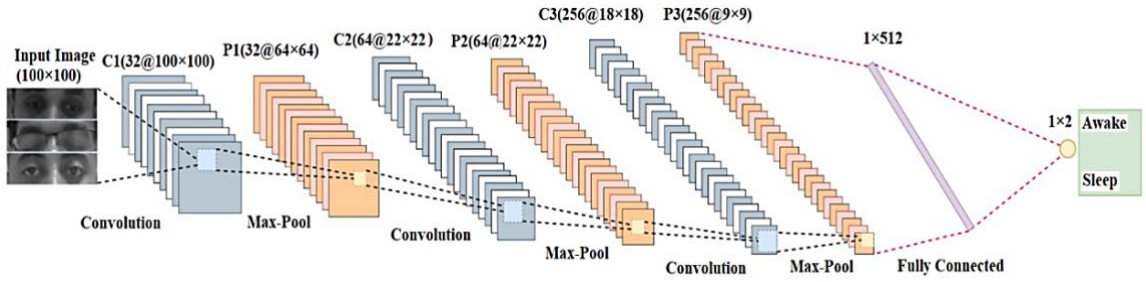


Figure 6. CNN architecture of the system

Table 1. Description of the CNN parameters

Layer	Size	Other parameters
Input	100×100×1	Grayscale image=100,000 images
Convolution C1	100×100×32	Window size=15
Max Pooling P1	64×64×32	Window size=10
Convolution C2	48×48×64	Window size=15
Max Pooling P2	22×22×64	Window size=10
Convolution C3	18×18×256	Window size=15
Max Pooling P3	9×9×256	Window size=10
Fully Connected	1×512	

2.4. CNN hyperparameters optimization

Since the tunable hyperparameters are not updated during the training process, the CNN model optimization was performed by obtaining the optimal hyperparameters, which could lead to effective performance. The following hyperparameters were examined in our study: kernel size (convolutional window size), maximum pooling size, learning rate, and the number of epochs. Hence, four steps were iterated to attain the optimal model by fine-tune the hyperparameters. At every step, we varied the hyperparameters that needed to be optimized and fixed the other parameters. The optimal parameters selection criteria were based on the highest training and validation accuracy and the minimum MSE square error.

In the testing phase, the confusion matrix was used to assess the model performance by summarizing the optimal model predictions. The use of the confusion matrix reduces any prejudices that may arise due to unbalanced data [24]. Based on the confusion matrix, the precision, accuracy, recall, and F1 score were calculated. Those parameters have been used in many applications to calculate the effectiveness for the classification of the CNN [25]–[27]. The accuracy calculates the percentage of images correctly predicted by the model, which can be calculated as:

$$Accuracy = \frac{TP + TN}{TP+TN+FP+FN} \times 100\% \quad (11)$$

where: TN is True negatives, TP is True positives, FN is False negatives, and FP is False positives.

In our model, the open eyes were labeled as positive, while the closed eyes were labeled as negative. TP means that the model correctly predicted the open eyes as open eyes. In contrast, FP indicates that the model predicted the closed eyes as the open eyes, and the same thing applies to TN and FN.

The correctness of the classification was calculated using the precision (12):

$$Precision = \frac{TP}{TP+FP} \times 100\% \quad (12)$$

the effectiveness for the classification was calculated using the recall (13):

$$Recall = \frac{TP}{TP+FN} \times 100\% \quad (13)$$

finally, the F1 score was calculated (14):

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \times 100\% \quad (14)$$

2.5. Drowisiness identification criteria

In the development of the proposed system, it's very crucial to differentiate whether the eyes are blinking or totally closed. Generally, a single blink can be divided into four phases: closing, closed, early opening, and late opening [28]. Calculating the duration of the human blink is challenging, mainly calculating the duration of the eye closing during blinking. Many experiments were conducted in our study to calculate the duration of the eyes closing during blinking. In those experiments, the eyes were considered as a closed eye state, if it's fully closed or more than 90% closed. Based on this assumption, 0.2 seconds were found as an average duration a person can close eyes during blinking. In the experiment, the Logitech webcam camera was used, which was capable of taking 30 frames per second. Accordingly, based on the estimated eyes blinking duration, the camera can take up to 6 frames in 0.2 seconds. As a result of this calculation, it can be estimated that a person can be considered asleep if the eye closes for seven consecutive frames (0.234 seconds) or more (one frame was added for confirmation). Therefore, the drowsy criteria were set in the proposed system in a way that if seven consecutive frames are classified by the CNN model as "closed eyes state," it reflects the drowsiness of the driver.

3. RESULTS

3.1. Hyperparameters optimization

The optimization results of the CNN Hyperparameters are shown in Figures 7 to 10. Figure 7 shows the results of determining the optimal number of epochs. As shown in the figure, the initial training accuracy was 50.42%, and validation accuracy was 45%. The initial loss of the training and validation was 10.985 and 11.985, respectively. From Figure 7, it can be observed that the training and validation accuracy is increased, and the loss is decreased when the number of epochs is increased. The highest training and validation accuracy with the lowest loss was achieved with the number of epochs of 100. The attained training accuracy was 70.219%, while the validation accuracy was 65.12%. Therefore, the model with 100 epochs was chosen.

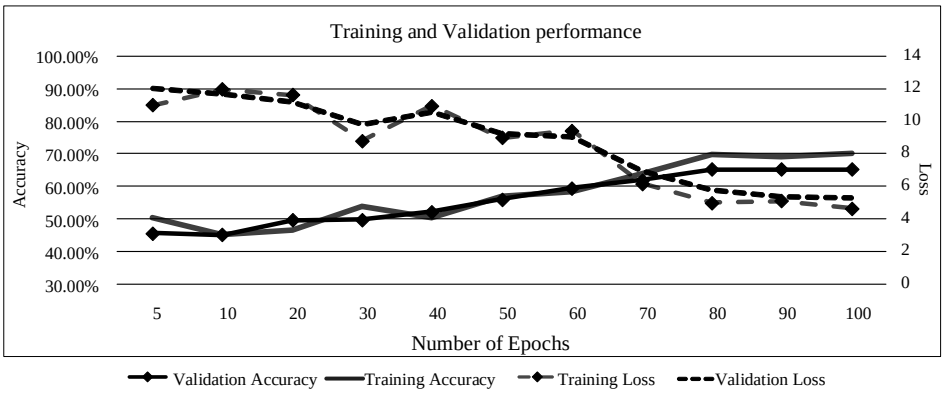


Figure 7. Training and validation performance with varying the number of epochs

Figure 8 shows the effects of varying the learning rate. The learning rate value was varied from 0.0000001 to 1. Figure 8 depicts that the learning rate of 0.01 produced the highest training accuracy of 84.348% and validation accuracy of 82.106%. Additionally, with the same learning rate, the network produced the lowest training and validation loss of 3.04. Accordingly, 0.01 was chosen as the optimal learning rate.

Determining the optimal convolutional window size is shown in Figure 9. The figure shows that convolutional window size of 15 produced training accuracy of 84.723%, validation accuracy of 86.9%, training loss of 2.813, and validation loss 2.9. Increasing the window size for more than 15 doesn't improve the network accuracy; therefore, it was chosen as an optimal value.

Figure 10 shows network performance for various max-pooling window size. The value of the max-pooling window was varied from 3 to 12. As shown in Figure 10, with a max-pooling window size of 10, the highest training and validation accuracy of 99.87% and 99.63% were achieved, respectively. On the other hand, the training loss is 0.871, while the validation loss is 0.991.

In light of the above results, the hyperparameters optimization effectively increased the training and validation accuracy from 50.42% and 45% to 99.87% and 99.63%, respectively. Accordingly, the optimal

hyperparameters were chosen as 100, 0.01, 15, and 10 for the number of epochs, learning rate, Convolutional window size, and max-pooling window size, respectively.

3.2. Model Performance evaluation

Tables 2 and 3, present the confusion matrix and the testing performance results obtained by using the optimal network and 10000 unused images in the testing phase. Those images are comprised of 5000 open eyes images and 5000 closed eyes images of different individuals in the data set. In the confusion matrix, each row represents the actual class, and each column represents the predicted class. As shown in the tables, the model perfectly classified the input data with 97.98% accuracy, 98.06% precision, 97.903% recall, and 97.981% F1 Score.

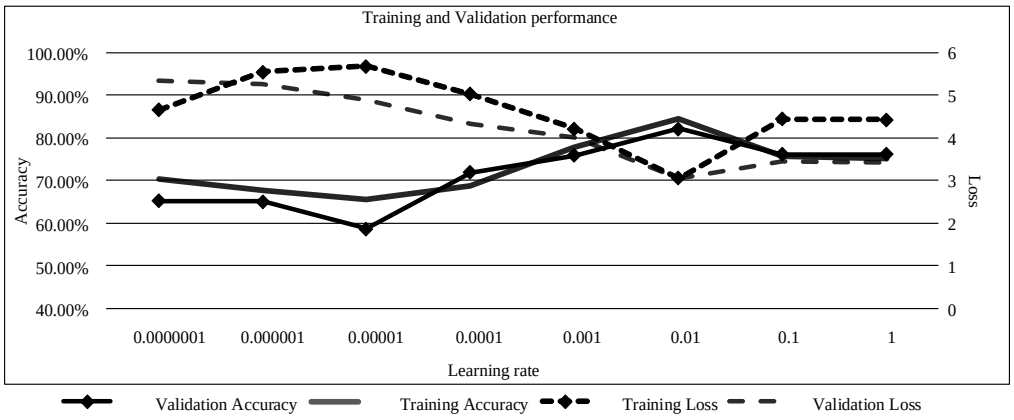


Figure 8. Training and validation performance with varying the learning rate

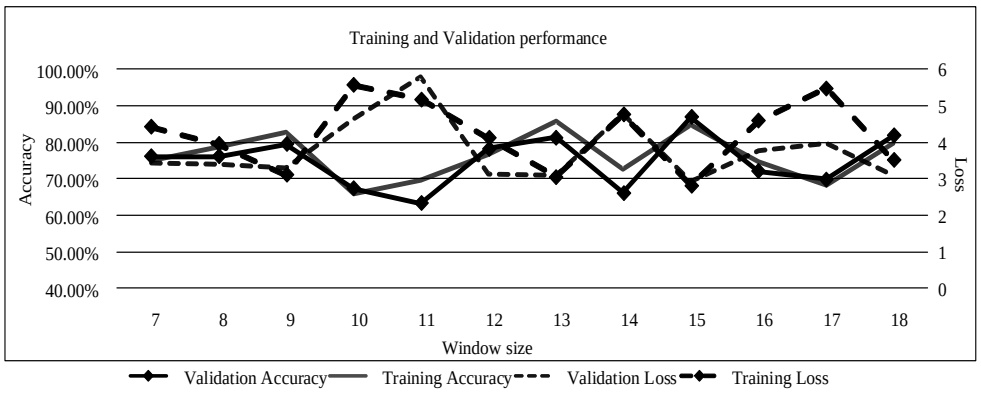


Figure 9. Training and validation performance with varying the convolutional window size

3.3. Prototype development

The major components used in the developed prototype system include Logitech webcam, NVidia Jatson Nano, LCD monitor, and Alert system (alarm speaker and Car Seat Vibrator). NVidia Jatson nano microprocessor is one of the powerful microprocessors available in the market which are capable of processing real time video for artificial intelligence systems. It has a 128 core NVidia Maxwell processor with a speed of 1.43 GHz. It has 4 GB 64-bit low-power double data rate (LPDDR4) memory capacity. The interface of the Nvidia Jetson Nano started by initially preparing a 64 GB microSD card to serve as a boot device and main storage. Then the image that contains bootable files and an ubuntu operating system are copied into the SD card using launcher Etcher before inserting the memory card into Nvidia Jetson Nano. Subsequently, the peripherals connected to the Nvidia Jetson Nano and the system was booted the model was uploaded our model.

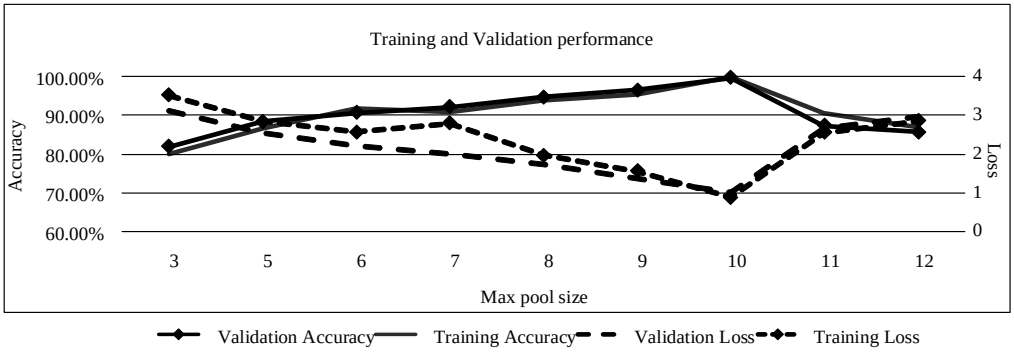


Figure 10. Training and validation performance with varying the Max_pool window size

Table 2. Confusion matrix of the proposed model

			Predicted Class	
			Positive	Negative
Actual Class	Awake	Positive	4903	105
	Drowsy	Negative	97	4895

Table 3. Testing performance

Parameters	Values
Number of Images	10000
Accuracy(%)	97.98%
Precision(%)	98.06%
Recall(%)	97.903%
F_1 Score(100%)	97.981%

Accordingly, the developed system operates: Firstly, the system is powered, and it takes three seconds to boot. A mounted camera starts to continuously record video and monitors the person (driver) present in front of it. The captured raw video is then sent to the NVidia Jatson nano for feature extracting and processing.

The processing starts by extracting frames from the series of frames included in the captured video and then reshape and resize them. The processed image is then fed to the developed CNN model to identify the state of the eyes, whether they are closed or opened. If the eyes are identified as open, the above process restarted again. However, if the state of the eye is closed, the consecutive frame is used to differentiate whether the driver is blinking or drowsy. This process is continued until the number of consecutive frames is equal to the predefined ones. Once confirming the eyes are closed, the alert system composes a seat with a motor (to simulated the car seat vibrator) and an alarm speaker is activated, and a warning sign is displayed on the LCD monitor. The alert system is interfaced with NVidia via relays. The alert system is deactivated if the eyes are opened or the engine is turned off.

4. DISCUSSION

Table 4 compare the proposed system with other latest developed systems. We have considered the latest studies that have similar feature to our system to identify the gaps and optimal solutions. Some of those systems used available online database where other system including our system, developed their own dataset. Studies used the online dataset used subjects with and without eyeglasses, while the other studies used their own developed dataset used only subjects without eyeglasses. However, in our studies, we used subjects with and without eyeglasses even though we developed our dataset. It can also be noticed that most of those studies didn't present in-depth any optimization for the CNN Hyperparameters or the experimental identification haar cascade classifier parameters. They rather focus on the application of the method. On the other hand, our study presented systematic optimization of the CNN Hyperparameters and the empirical determination of the haar cascade algorithm parameters. As a result, as shown in Table 4 our system outperformed other system's accuracy. On the other hand, one of the limitations of our system compared with other systems, especially those studies that used the available online database, is that our study was based on daytime lighting since we did not use a night vision camera.

Table 4. Brief comparison of the existing and the proposed system

Reference	Implemented system	Feature	Performance
R. Jabbar *et al.*, 2018 [15]	▪ Driver sleepiness identification method using CNNs to extract information from images and long short term memory networks (LSTM). Data for subjects with and without eyeglasses was used	▪ Dlib library is used to extract landmark coordination from images ▪ Android mobile application, Java Native Interface (JNI) Available online National Tsing Hua University (NTHU) Driver Drowsiness Detection Dataset was used	▪ 80.9% accuracy was achieved
H. Vanjani *et al.*, 2019 [12]	▪ Deep neural network (DNN) for detecting driver drowsiness in videos. ▪ Subjects included with and without eyeglasses	▪ Stander Viola-Jones Haar-Feature was used for face detection. ▪ Adam Optimizer was used the CNN model for feature extraction and LSTM for interpreting the features across consecutive frames.	▪ 88.5% accuracy was achieved
T. VU *et al.*, 2019 [13]	▪ Driver drowsiness detection based on eye state. ▪ Viola-Jones face detection algorithm with CNN classifier. ▪ Subjects included without eyeglasses	▪ MTCNN for face detection, and a correlation function in dlib for tracking ▪ CNN, a ConvCGRNN, and a voting layer are used to developed the system. ▪ Available online NTHU was used	▪ 84.81% accuracy was achieved
V. Chirra *et al.*, 2019 [18]	▪ Anticipation of driver drowsiness by applying a recurrent neural network over a sequence frame driver's face. ▪ Data for subjects with and without eyeglasses and sunglasses was used	▪ Adam method is used for CNN Optimization. ▪ Stander Haar cascade classifier implemented using OPEN CV with python.	▪ 96.42% accuracy was achieved
Y. Ed-Doughmi *et al.*, 2020 [29]	▪ Drowsiness detection system based on CNN-based Machine Learning. ▪ Data for subjects with and without eyeglasses was used	▪ 3D Convolutional Networks was used ▪ Keras framework with the Python programming language was used. ▪ mobile application. ▪ Available online NTHU was used	▪ 97% accuracy was achieved
R. Jabbar *et al.*, 2020 [17]	▪ Driver drowsiness and object distraction detection system using facial landmarks	▪ Comparative analysis of four pertained models MLP-Model, ▪ CNN Model, VGG-16, AlexNet. ▪ Available online NTHU was used ▪ Dlib library is used to extract landmark coordination from images ▪ Android mobile application, Java Native Interface was used for implantation	▪ Accuracy achieved by MLP, CNN, VGG-16, and AlexNet models were 80.9%, ▪ 83.3, 90.5, and 82. 8 respectively.
Komal *et al.*, 2020 [11]	▪ Driver drowsiness detection based on eye state. CNN hyperparameters with Viola-Jones object detection. ▪ Subjects included with and without eyeglasses	▪ Single shot multi-box detector Object detection technique was used. ▪ Tensor flow deep learning framework is used to implement the model.	▪ 90% accuracy was achieved
Proposed system	▪ Driver sleepiness identification method using CNNs to extract information from images and LSTM. ▪ Subjects included without eyeglasses	▪ Systematic Hyperparameters optimization were conducted for the CNN. ▪ OPEN CV with python is used for implementing Haar cascade classifier ▪ Empirically determination of Haar cascade algorithm parameters to identify open and close eyes ▪ Logitech webcam, NVidia Jatson Nano, LCD monitor were used	▪ 97.98% accuracy was achieved

5. CONCLUSION

A significant percentage of car accidents are traced back to sleeping while driving. To address this issue, this study proposed an intelligent driver drowsiness detecting system based CNN model. The system was developed systematically by using; Haar cascade algorithm to preprocess the images, optimization of the CNN hyperparameters for precise classification, and defining the drowsiness criteria. This study showed that systematic optimization of the CNN hyperparameters could tremendously increase the system's accuracy. Overall, promising results were achieved from the proposed model for identifying the drowsiness of the drivers. The experimental results performed shown that the model achieved 99.87%, 99.63%, and 97.98% accuracy for training, validation, and testing, respectively. The high accuracy levels achieved by the system indicate the probability and need to deploy such systems for real-time application. The main challenge to developing a comprehensive real-time drowsiness detecting system seems unaccomplished yet. Even though our system demonstrated high accuracy compared with other developed systems, we could still add additional features to increase its reliability and practicality. Accordingly, the scope of our future work is divided into two areas; first, including an extra feature in drowsiness identification such as Yawning facial landmarks and abnormal head movement as well as night detection. Secondly, real-time implementation of the system.

REFERENCES

[1] "Global status report on road safety 2018," 2018. Accessed: Aug. 08, 2020. [Online]. Available: https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/.

[2] "Drowsy driving 2015." Accessed: Aug. 08, 2020. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812446.

[3] K. Al Hammadi, M. Ismaeel, and T. Faisal, "Intelligent car safety system," in *2016 IEEE Industrial Electronics and Applications Conference (IEACon)*, Nov. 2016, pp. 319–322, doi: 10.1109/IEACON.2016.8067398.

[4] A. Picot, S. Charbonnier, and A. Caplier, "On-line detection of drowsiness using brain and visual information," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 42, no. 3, pp. 764–775, May 2012, doi: 10.1109/TSMCA.2011.2164242.

[5] J. I. Pilataxi, W. M. Vinan, and D. C. Garcia, "Design and implementation of a driving assistance system in a car like robot when fatigue in the user is detected," in *2014 IEEE ANDESCON*, Oct. 2014, pp. 1–1, doi: 10.1109/ANDESCON.2014.7098558.

[6] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," in *1st International Conference on Learning Representations, ICLR 2013 - Conference Track Proceedings*, Jan. 2013, pp. 1–8, [Online]. Available: http://arxiv.org/abs/1301.3572.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8691, no. 3, pp. 346–361, Jun. 2014, doi: 10.1007/978-3-319-10578-9_23.

[8] Q. Wang, C. Rasmussen, and C. Song, "Fast, deep detection and tracking of birds and nests," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10072 LNCS, Springer, Cham, 2016, pp. 146–155.

[9] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, Mar. 1968, doi: 10.1113/jphysiol.1968.sp008455.

[10] K. Fukushima and S. Miyake, "Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, no. 6, pp. 455–469, Jan. 1982, doi: 10.1016/0031-3203(82)90024-3.

[11] Komal, P. Sharma, A. Lamba, B. Nagpal, and S. Chauhan, "Driver drowsiness detection system using convolutional neural network," *International Journal on Emerging Technologies*, vol. 11, no. 3, pp. 502–506, 2020.

[12] H. B. Vanjani and U. Varyani, "Identify dozyness of person using deep learning," *International Journal of Applied Engineering Research*, vol. 14, no. 4, pp. 845–848, 2019, [Online]. Available: http://www.ripublication.com.

[13] T. H. Vu, A. Dang, and J.-C. Wang, "A deep neural network for real-time driver drowsiness detection," *IEICE Transactions on Information and Systems*, no. 12, pp. 2637–2641, Dec. 2019, doi: 10.1587/transinf.2019EDL8079.

[14] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001, vol. 1, pp. I-511-I–518, doi: 10.1109/CVPR.2001.990517.

[15] R. Jabbar, K. Al-Khalifa, M. Kharbeche, W. Alhajyaseen, M. Jafari, and S. Jiang, "Real-time driver drowsiness detection for android application using deep neural networks techniques," *Procedia Computer Science*, vol. 130, pp. 400–407, Jan. 2018, doi: 10.1016/j.procs.2018.04.060.

[16] M. Omidyeganeh *et al.*, "Yawning detection using embedded smart cameras," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 3, pp. 570–582, Mar. 2016, doi: 10.1109/TIM.2015.2507378.

[17] R. Jabbar, M. Shinoy, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, "Driver drowsiness detection model using convolutional neural networks techniques for android application," *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020*, pp. 237–242, 2020, doi: 10.1109/ICIoT48696.2020.9089484.

[18] V. Chirra, S. ReddyUyyala, and V. KishoreKolli, "Deep CNN: a machine learning approach for driver drowsiness detection based on eye state," *Revue d'Intelligence Artificielle*, vol. 33, no. 6, pp. 461–466, Dec. 2019, doi: 10.18280/ria.330609.

[19] R. Ghoddoosian, M. Galib, and V. Athitsos, "A realistic dataset and baseline temporal model for early drowsiness detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Apr. 2019, pp. 178–187, doi: 10.1109/CVPRW.2019.00027.

[20] W. Guo, L. Huang, and L. Liang, "A weld seam dataset and automatic detection of welding defects using convolutional neural network," in *Advances in Intelligent Systems and Computing*, vol. 905, Springer, Cham, 2020, pp. 434–443.

[21] D. Sirohi, N. Kumar, and P. S. Rana, "Convolutional neural networks for 5G-enabled intelligent transportation aystem: a systematic review," *Computer Communications*, vol. 153, pp. 459–498, Mar. 2020, doi: 10.1016/j.comcom.2020.01.058.

[22] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/s13244-018-0639-9.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[24] T. Faisal, M. N. Taib, and F. Ibrahim, "Neural network diagnostic system for dengue patients risk classification," *Journal of Medical Systems*, vol. 36, no. 2, pp. 661–676, Apr. 2012, doi: 10.1007/s10916-010-9532-x.

[25] N. N. A. Mangshor, I. A. A. Majid, S. Ibrahim, and N. Sabri, "A real-time drowsiness and fatigue recognition using support vector machine," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 4, pp. 584–590, Dec. 2020, doi: 10.11591/ijai.v9.i4.pp584-590.

[26] Q. A. Abed, O. Mohammed Fadhil, and W. L. Al-Yaseen, "Data mining in web personalization using the blended deep learning model," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 3, pp. 1507–1512, Dec. 2020, doi: 10.11591/ijeecs.v20.i3.pp1507-1512.

[27] R. I. Farhan, A. T. Maolood, and N. F. Hassan, "Performance analysis of flow-based attacks detection on CSE-CIC-IDS2018 dataset using deep learning," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 3, pp. 1413–1418, Dec. 2020, doi: 10.11591/ijeecs.v20.i3.pp1413-1418.

[28] K.-A. Kwon *et al.*, "High-speed camera characterization of voluntary eye blinking kinematics," *Journal of The Royal Society Interface*, vol. 10, no. 85, Aug. 2013, doi: 10.1098/rsif.2013.0227.

[29] Y. Ed-Doughmi, N. Idrissi, and Y. Hbali, "Real-time system for driver fatigue detection based on a recurrent neuronal network," *Journal of Imaging*, vol. 6, no. 3, Mar. 2020, doi: 10.3390/jimaging6030008.

BIOGRAPHIES OF AUTHORS

Dr. Tarig Faisal received his master degree of Mechatronics Engineering from IIUM university in 2006 and his Ph.D. degree in Signal Processing from University of Malaya, Malaysia, in 2011. He has been the Dean of Academic Operations at Higher Colleges of Technology since 2018. He has published more than 35 research articles with more than 400 citations and H-index of 11. His research focuses on biomedical signal processing, intelligent systems, robotics, control, embedded system design, IoTs, machine learning, and outcome based education. He has been a reviewer for multiple journals including IEEE, Elsevier, Taylor & Francis, and Springer Nature. He has more than 20 years of academic and industry experience of which he worked as an engineering, assistant professor, Programs Chair, Head of department, and Division Chair. He is also a charted engineering as well as Senior Fellow of Higher Education Academy. Dr. Tarig can be contacted at email: tfaisal@hct.ac.ae.

Isaias Negassi is alumni from the Electrical Engineering Department of Higher Colleges of Technology, UAE. Isaias has been working as research assistant in the area of embedded system design and Artificial intelligence. He has won many awards including 1st prize on Engineering Project Competition in HCT AL Dhafrah colleges in 2017, 2nd prize on Projects and innovative idea competition 2018, 1st prize on AL Dhafrah innovation Carnival competition 2019, and Bronze Award in HCT Ibtikar Innovation and Entrepreneur Competition 2020. He can be contacted at email: Isaiastesfu27@gmail.com.

Ghebrehiwet Goitom is alumni from the Electrical Engineering Department of Higher Colleges of Technology, UAE. Ghebrehiwet has been working as research assistant in the area of embedded system design and intelligent system. He has won many awards including 1st prize on Engineering Project Competition in HCT AL Dhafrah colleges in 2017, 2nd prize on Projects and innovative idea competition 2018, 1st prize on AL Dhafrah innovation Carnival competition 2019, and Bronze Award in HCT Ibtikar Innovation and Entrepreneur Competition 2020. He can be contacted at email: gebrishfet@gmail.com.

Mohammed Yassin is alumni from the Electrical Engineering Department of Higher Colleges of Technology, UAE. Mohammed has been working as research assistant in the area of embedded system design and intelligent system. He has won many awards including 1st prize on Engineering Project Competition in HCT AL Dhafrah colleges in 2017, 2nd prize on Projects and innovative idea competition 2018, 1st prize on AL Dhafrah innovation Carnival competition 2019, and Bronze Award in HCT Ibtikar Innovation and Entrepreneur Competition 2020. He can be contacted at email: mohammedyssn2019@gmail.com.

Anees Bashir received Bachelor in electronics and communication engineering from Anna University, India in 2012, and master degree in applied electronics engineering from M.S. University India, in 2014. Her main research interests include embedded system design, signal processing, outlier detection, data mining. She is currently working on her research based on Moving-horizon strategy for anomaly detection in discrete LTI systems. She can be contacted at email: abashir1@hct.ac.ae.

Dr. Moath Awawdeh (Member, IEEE) received the B.S. in communication and software engineering from Al Balqa Applied University for Engineering Technology, Jordan, in 2009, and Ph.D. in mathematical engineering and simulation with specialization in control engineering from the University of Genoa, Italy, in 2014. He is currently an applied research coordinator at Higher Colleges of Technology. Beside his active work as a reviewer for different international journals, He is on the Editorial Board of ASET conference and JAETS journal. His main research interests include outlier detection, data mining, machine learning, signal processing, and control engineering applications. He is currently working on the problem of outlier detection and correction in LTI systems with a specific approach of state estimation and measurements validation. He can be contacted at email: mawawdeh@hct.ac.ae.

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 161~172
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i1.pp161-172

Artificial speech detection using image-based features and random forest classifier

Choon Beng Tan[1], Mohd Hanafi Ahmad Hijazi[1], Frazier Kok[2], Mohd Saberi Mohamad[3], Puteri Nor Ellyza Nohuddin[4]
[1]Faculty of Computing and Informatics, Universiti Malaysia Sabah, Kinabalu, Malaysia
[2]Bayurini Sdn Bhd, Penampang, Kinabalu, Malaysia
[3]College of Medicine and Health Sciences, United Arab Emirates University, Abu Dhabi, United Arab Emirates
[4]Institute of IR4.0, Universiti Kebangsaan Malaysia, Bangi, Malaysia

Article Info

Article history:

Received Jul 15, 2021
Revised Nov 26, 2021
Accepted Dec 11, 2021

Keywords:

Anti-spoofing voice recognition
Artificial speech detection
Speaker recognition
Speaker verification
Voice presentation attack detection

ABSTRACT

The ASVspoof 2015 Challenge was one of the efforts of the research community in the field of speech processing to foster the development of generalized countermeasures against spoofing attacks. However, most countermeasures submitted to the ASVspoof 2015 Challenge failed to detect the S10 attack effectively, the only attack that was generated using the waveform concatenation approach. Hence, more informative features are needed to detect previously unseen spoofing attacks. This paper presents an approach that uses data transformation techniques to engineer image-based features together with random forest classifier to detect artificial speech. The objectives are two-fold: (i) to extract image-based features from the mel-frequency cepstral coefficients representation of the speech signal and (ii) to compare the performance of using the extracted features and Random Forest to determine the authenticity of voices with the existing approaches. An audio-to-image transformation technique was used to engineer new features in classifying genuine and spoof voices. An experiment was conducted to find the appropriate combination of the engineered features and classifier. Experimental results showed that the proposed approach was able to detect speech synthesis and voice conversion attacks effectively, with an equal error rate of 0.10% and accuracy of 99.93%.

Corresponding Author:

Mohd Hanafi Ahmad Hijazi
Faculty of Computing and Informatics, Universiti Malaysia Sabah
Jalan UMS, Sabah, Malaysia
Email: hanafi@ums.edu.my

1. INTRODUCTION

Voice recognition, often known as speaker recognition, is the act of identifying and verifying a speaking human. It is divided into two categories: speaker identification and speaker verification. Speaker identification is the process of determining a speaking individual's identity, whereas speaker verification is the act of verifying that individual's claimed identity. Figure 1 depicts the distinction between speaker identification and speaker verification. In recognizing and validating the identity of a person from voice, speaker recognition employs both physiological and behavioral components.

Automatic speaker verification (ASV) is the process of verifying the claimed identity of a speaking individual automatically. In most ASV systems, the speaker enrolment phase and the speaker verification phase are the two key phases. During speaker enrolment, the ASV system captures the speaker's voice and extracts attributes that are utilized to create a speaker model of the speaking individual. The speaker model is

then registered with the ASV system. During speaker verification, the speaking individual's voice is recorded to create a speaker model for verification. After that, the speaker model is compared to the claimed identity's speaker model in the ASV system. Finally, the matching will generate a score, with the claim being accepted if the score is equal to or greater than the ASV system's threshold. Otherwise, the claim will be turned down. Numerous types of features have been deployed for ASV systems. Gaussian mixture models (GMM) were extensively used in the past for feature extraction to produce robust ASV systems. In speaker verification, the universal background model (UBM) is a speaker model that represents broad attributes and characteristics that can be compared to the specific person being verified [1]. Later, i-vector and x-vector [2] based ASV systems were introduced to replace the gaussian mixture model-universal background model (GMM-UBM) based ASV systems. Deep learning approaches [3] such as recurrent neural network (RNN) [4] as a backend classifier was shown the capability in speaker verification with a low equal error rate (EER).

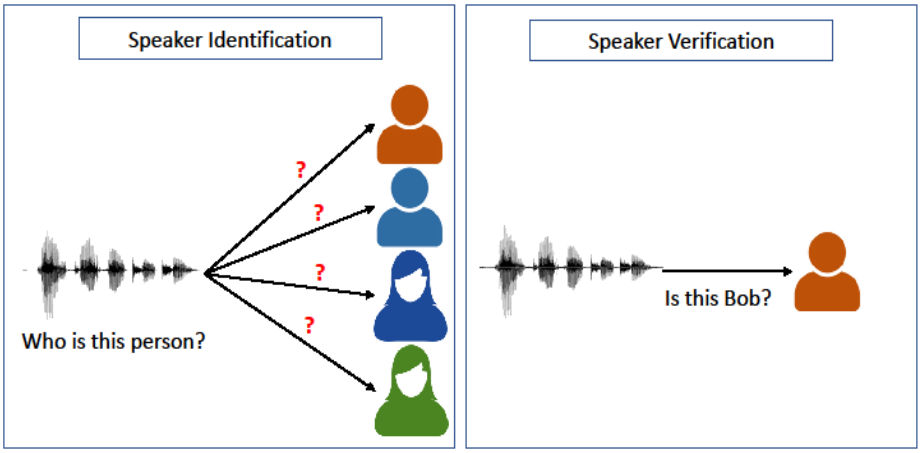


Figure 1. The illustration of speaker identification versus speaker verification

To mitigate the issue of security threats to the ASV systems, voice presentation attack detection (PAD) was introduced. Voice PAD can be categorized into two major types, namely artificial and replayed speech detection. The artificial speech was generated by speech synthesis and voice conversion, whereas replayed speech was generated by replaying the recordings of human speech. Several efforts can be seen to foster the development of countermeasures against spoofing attacks on ASV systems. First, the building of public datasets such as the dataset, which is made up of a collection of genuine and replayed speech [5]. In the ReMASC dataset, the human speech captured by the microphone array was labeled as genuine speech, whereas the playback of the replay source recordings generated in different replay settings was labeled as replayed speech. In particular, the ReMASC dataset made up of 9,240 genuine and 45,472 replayed recordings. The speech corpus was collected from a total of 50 speakers, in particular, 22 female and 28 male speakers with ages range from 18 to 36. Among the 50 speakers, there were 36 English native speakers, 12 Chinese native speakers, and 2 Indian native speakers. About 132 voice commands made up of 273 unique words were used as recording materials to provide reasonable phonetic diversity. Four different recording environments with different noise levels were used, namely one outdoor environment, two indoor environments (quiet and noisy), and one vehicle environment. The building of a public dataset allowed the community of spoofing and anti-spoofing for ASV to develop robust PAD for ASV systems.

Second, the ASVspoof challenges were held to encourage the development of voice spoofing countermeasures. Standard datasets, techniques, and evaluation criteria were utilized in the ASVspoof Challenge series. The first ASVspoof Challenge, which covered speech synthesis and voice conversion attacks, was held in 2015. In the ASVspoof 2015 Challenge, the rating was based on 16 primary submissions. The best system in the ASVspoof 2015 Challenge had an EER of 1.21% on average [6]. Then, to emphasize replay attacks, the ASVspoof 2017 Challenge was launched. There was a much higher number of submissions received in the ASVspoof 2017 Challenge, recorded 49 submissions compared to the previous challenge. The best performing system in the ASVspoof 2017 challenge has achieved 6.73% EER [7]. The ASVspoof 2019 challenge was later organized to include speech synthesis, voice conversion, and replay attacks. The ASVspoof 2019 dataset can be divided into two types of attacks: logical attacks (LA) and

physical attacks (PA). Genuine speech, speech synthesis, and voice conversion attacks were included in the LA dataset, whereas genuine speech and replay attacks were included in the PA dataset. For the LA and PA scenarios, the best submissions achieved 0.22% and 0.39% EER, respectively [8]. However, only 56.25% and 64% submissions for the LA and PA scenarios had outperformed the baseline system, respectively. Nonetheless, in both the LA and PA scenarios, the EER of the majority of the submissions had not been less than 5%. Due to the ease of obtaining biometric data, especially through social media, the security threat to the ASV systems is significant. Publicly available biometric data in social media can be used by security adversaries to launch presentation attacks such as speech synthesis and voice conversion to spoof the ASV systems. Furthermore, a large amount of artificial speech can be generated using state-of-the-art speech synthesis and voice conversion algorithms to spoof ASV systems. Whereby in this paper, the focus is on artificial speech due to it is the common spoofing attack as it can be generated in a short time to spoof the ASV system. Several voice PAD systems were introduced to detect artificial speech. As most of the artificial speech was produced using parametric vocoders, phase information was an effective feature to detect speech synthesis attacks. As a result, phase-based voice PAD for detecting artificial speech has become state-of-the-art [9]. However, ASV systems are still prone to attacks from artificial speech as most of the phase-based voice PAD introduced were only effective against artificial speech generated using minimum-phase filters based parametric vocoders [10].

There were numerous works found in the literature whereby the application of image classification in the signal domain was shown to be effective. To apply an image classification approach, audio data were pre-processed and transformed into image data. For example, features extracted from the Spectrogram image were shown to improve the performance of acoustic event classifications [11]. Besides, Spectrogram images were also being used for rapid speaker recognition and artificial speech detection [12]. The recent work [12], which used raw Spectrogram image as input for an end-to-end Light-ResNet-34 model, has outperformed the conventional approach of using constant q cepstral coefficients (CQCC) and GMM in artificial speech detection. Another recent work that applied deep neural network (DNN) architecture as a backend classifier with constant-q equal subband transform (CQ-EST) features [13] was shown to outperform most of the state-of-the-art approaches with an EER of 0.06%. Other than backend classifiers, deep learning architecture such as convolutional neural network (CNN) was also used as a feature extractor in recent works. In other work, a light gated CNN was used as a feature extractor to extract features from spectrogram image and probabilistic linear discriminant analysis (PLDA) as backend classifier to achieve an EER of 0.16% in artificial speech detection [14].

A fused system using short time fourier transform (STFT) and modified group delay (MGD) features were introduced recently and produced a 0.02% EER in detecting artificial speech. The advantage of this kind of fused system [15] is that both magnitude and phase spectral features were used together. This method yielded better performance than a fusion of independent systems with one feature for each system. Although most of the recent works achieved good EER, however, most of them did not perform well in detecting an S10 attack, one of the attack scenarios of the ASVspoof 2015 dataset. This indicates that more generalized models of artificial speech attacks are needed. In the context of artificial speech detection, the most recent works were using CNN as a feature extractor to extract image-based features from the spectrogram. Nonetheless, CNN usually requires a large number of training samples, computing time, and resources for better performance and generalization. However, similar performance can be achieved by utilizing handcrafted features for image classification, dealing with the abovementioned drawbacks. Moreover, work that utilized handcrafted image-based features in detecting artificial speech was limited in the literature. It is conjectured that using similar approaches to extract image-based features (color, texture, or edges) could be useful to generate more generalized features for artificial speech detection.

Despite the advancements made possible by speech identification technology, spoofing attacks by security adversaries to evade ASV systems is always a problem. To spoof ASV systems, state-of-the-art speech synthesis and voice conversion algorithms could easily generate artificial speech in massive quantities. Furthermore, because it is so easy to get biometric data via social media, spoofing attacks on ASV systems are becoming more common. As a result, robust spoofing countermeasures are required. These countermeasures are commonly known as voice PAD. Voice PAD has been the subject of various research studies, which may be found in the literature. Recent voice PADs, on the other hand, were vulnerable to unknown spoofing techniques [6]. The voice PADs submitted in the ASVspoof 2015 competition demonstrate this. System A, the best system in the ASVspoof 2015, proposed using two features for artificial speech detection: mel-frequency cepstral coefficients (MFCC) and cochlear filter cepstral coefficients plus instantaneous frequency (CFCCIF) using GMM classifier. Although system A performed with an average of 1.21% EER, the average EER for known and unknown attacks were 0.41% and 2.01%, respectively. Similarly, most systems submitted to the ASVspoof 2015 challenge encountered a similar circumstance in which they were unable to identify the S10 attack effectively, which was the sole attack produced using the waveform concatenation method. This pattern can be interpreted as possible overfitting in the proposed voice

PADs. Hence, more informative features are needed to generalize voice PAD against unseen spoofing attacks [16]. One solution is to produce new features using feature engineering, in which new descriptive features are constructed to be used to train a predictive model. This paper is written to propose a new feature engineering approach using data transformation techniques for artificial speech detection. In this work, rather than using conventional signal processing to extract features from speech, we proposed to use data transformation to apply the techniques from other domains such as image processing for artificial speech detection. The proposed approach is motivated by the success of deploying image classification techniques to sounds classification and speaker recognition [11]. An ensemble classifier in the form of random forest (RF) was used to generate the artificial speech model. The performance of the proposed approach is detailed, along with the results and discussion. Then, issues and future work to mitigate the limitation of the proposed approaches are described in this paper.

The key contributions of this paper are:

- Application of data transformation techniques to engineer image-based features to detect artificial speech
- Application of RF to be used with the new features engineered to detect artificial speech
- Empirical evaluation of the proposed approach with the existing work found in the literature

2. THE PROPOSED METHOD

In this paper, data transformation is considered to generate potential generalized features for voice PAD. In the conventional approach, features such as MFCC and CQCC are extracted directly from the speech signal to determine the genuineness of the speech. Recently, deep learning approaches, including CNN, were frequently being used to automatically extract features from image representation of speech signals. Unlike conventional and deep learning feature extraction approaches, the work presented in this paper proposed to use handcrafted features extracted from the image and hexadecimal frequency representation of the speech signal. In this paper, audio recordings were first transformed into images. Then, the image-based features are extracted from the transformed data to form the feature vectors. Figure 2 shows the differences between the conventional approach and the proposed feature engineering for artificial speech detection. The proposed feature engineering allows new features to be extracted from the speech data. Subsection 2.1 describes the generation of image-based features considered in this paper. Subsection 2.2 presents the RF classifier used for artificial speech detection in this work.

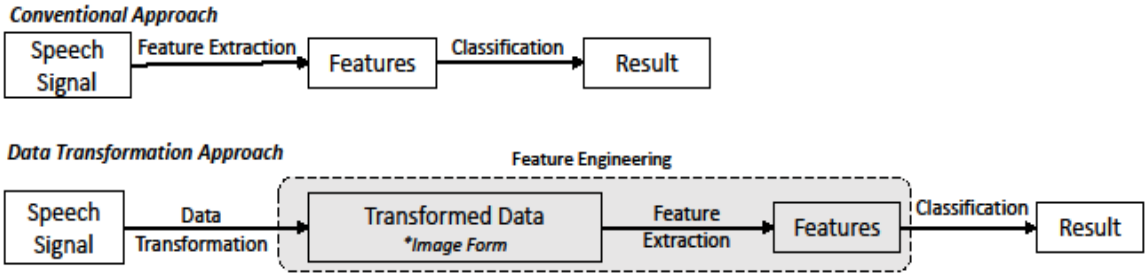


Figure 2. The comparison of the conventional approach and the proposed data transformation approach for artificial speech detection

2.1. Transformation of audio data into image representation and extraction of image-based features

Spectrogram and MFCCs are two common forms used to represent audio data [17], [18]. However, little attention has been paid to use both as images, whereby image-based features could be extracted for voice PAD. In work presented in this paper, the audio signals are represented as Spectrogram and MFCCs images. Different features were then extracted from each of the generated images. Figure 3 shows the process of the feature extraction from Spectrogram and MFCC images proposed in this paper. The speech signal is first transformed into spectrogram and MFCC images. Then, the color layout filter (CLF) and local binary patterns (LBP) features are extracted from the spectrogram to form the spectrogram-based features. concerning MFCC images, the CLF features are extracted to form the MFCC-based features.

A spectrogram is a representation of a signal that shows the signal's spectral information as frequency over time in the form of visual. Figure 4 shows how spatial differences between genuine and spoof voices using Spectrogram image representation could be observed. In this example, a genuine voice contains

less background noise in a certain region of the spectrogram, while the spoof voice contains more background noise. To detect more differences between genuine and spoof voices in the voice-transformed images such as spectrogram, image classification techniques could be used as suggested in [17]. MFCC is an audio feature commonly used for signal processing, especially speech recognition [18], [19]. Figure 5 shows the generated MFCC images of genuine and spoof voices. From Figure 5, a slightly different color intensity in the region of a non-speech segment can be observed when comparing the MFCC images of genuine and spoof speech.

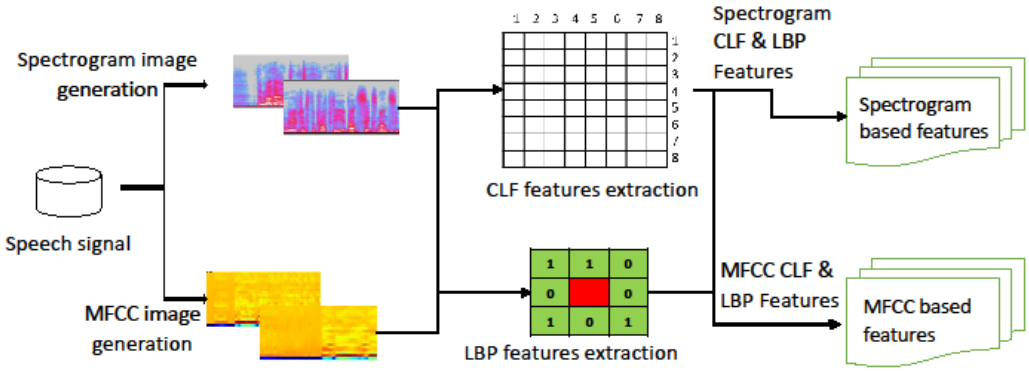


Figure 3. The feature extraction process

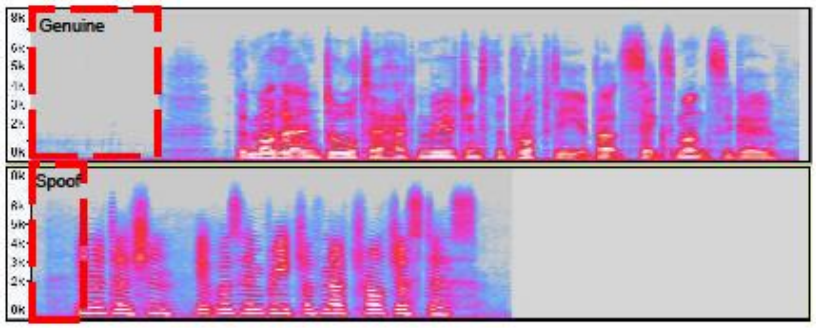


Figure 4. The observable spatial differences between genuine and spoof voices using spectrogram generated from the audacity tool

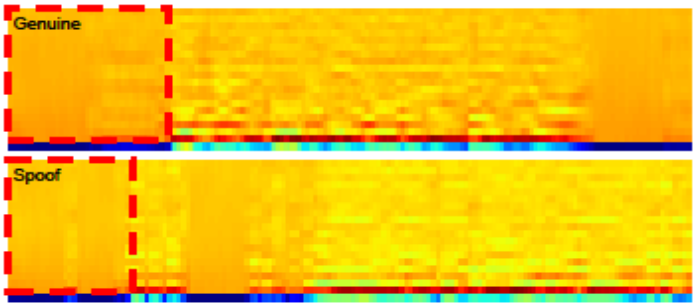


Figure 5. An example of MFCC images generated for genuine and spoof voices of a speaker

CLF was selected for feature extraction as it describes the spatial distribution of colors in an image and it works well in image classification when applied on color spectrogram [20]. In the CLF algorithm, the input image was divided into 64 blocks. Then, the values of all pixels within each block were averaged to obtain a representative color, resulting in three 8×8 arrays, collectively representing *YCbCr* color space.

Then, discrete cosine transform (DCT) was applied to the three 8×8 arrays, resulting in three DCT matrices, one for each *YCbCr* component. The CLF descriptor was formed by reading the coefficients from the matrices in zigzag order. The CLF descriptor contains a total of 33 features generated. As shown in Figures 4 and 5, genuine and artificial voices may be distinguished by the differences in the spatial distribution of color in certain regions of the generated Spectrogram and MFCC images. Figure 6 shows the process of CLF features extraction.

LBP is chosen in this paper as it is commonly used and produced good descriptors of texture in image classification. Figure 7 shows the process of LBP feature extraction. To extract LBP features from a Spectrogram image, the 3D color pixels were converted into 2D grayscale values. For each pixel in the converted grayscale image, a neighborhood radius *r* surrounding the center pixel was selected. Then, the LBP value was calculated for this center pixel and stored as a 2D array with the same height and width as the converted grayscale image. Then, the center pixel was compared to the surrounding neighborhood pixels, whether the neighbor pixels were greater-than-or-equal-to the center pixel. If the neighbor pixel was greater than or equal to the center pixel, then the value will be set as 1; otherwise, 0 will be set. The possible number of combinations of LBP codes was 2^p, where p is the number of neighborhood pixels. In original LBP, with neighborhood radius, =1 and p=8, there were 2^8=256 possible number combinations of the LBP codes, ranged 0-255. A frequency histogram of LBP codes was computed as LBP features. Details of LBP can be found in [21].

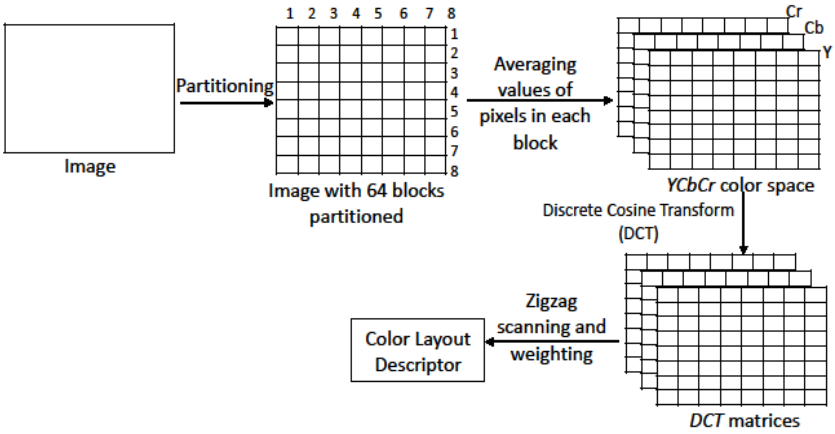


Figure 6. The process of CLF features extraction

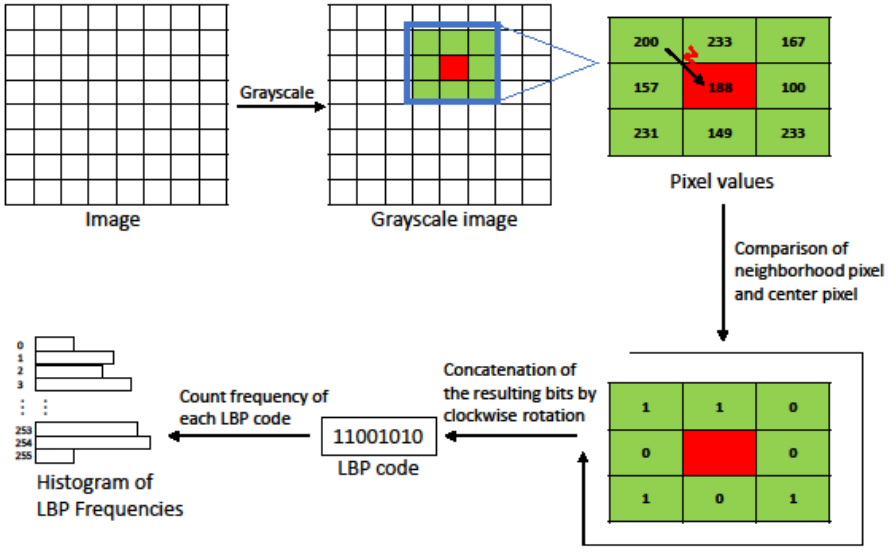


Figure 7. The process of LBP features extraction

In this paper, the Spectrogram and MFCC images were generated from audio recordings using Python. Spectrogram and MFCC images were plotted using *pyplot* and *librosa* libraries, respectively. The generated images were then saved in PNG format with a size of 640×480 pixels. The CLF implementation in Weka [22] was used to extract the features. The *cvtColor()* algorithm of the *OpenCV* library (*cv2*) was employed for grayscale conversion in Python. Concerning LBP, a neighborhood radius r=1 was chosen as it was the setting used in the original LBP [21] and most used in the literature, in which there were eight neighboring pixels in a 3×3 pixels window. The frequency histogram computed from LBP was used as LBP features in this work. In total, 322 features were extracted from the spectrogram and MFCC represented audio data. A total of 289 features were generated from the spectrogram; 33 were CLF and 256 were LBP features. Concerning MFCC, a total of 33 CLF features were generated.

2.2. Random forest (RF) classifier for artificial speech detection

Features were extracted from data samples and automatically learned using a deep learning process, which was then used to predict the data samples' class labels in end-to-end learning. Unlike the end-to-end approach, a backend classifier is needed to differentiate between genuine and spoof speech using the proposed handcrafted features. In this work, an ensemble classifier is selected as it shows good classification results when applied with handcrafted features [23]–[25].

RF is a supervised, ensemble learning model where decision trees are bagged for classification and regression. In an RF model, multiple decision trees based on randomly selected training subsets were trained and merged to get a more accurate and stable prediction via votes aggregation.

The use of the greedy algorithm to select the best split point at each step in the tree building process will lead to similar resulting trees for bagged decision trees. This resulting in the reduction in the variance of the predictions of the bagged decision trees. To mitigate this issue, RF is an improved version of bagged decision trees that disrupt the greedy splitting algorithm during tree creation. When the greedy splitting algorithm is disrupted during tree creation in RF, the split points of decision trees can only be chosen from a subset of the input features at random. As a result, the similarity between the bagged decision trees decreased and led to lower bias and higher variance of the predictions. Due to its simplicity and predictive performance, RF was chosen as a backend classifier for artificial speech detection in this work. Details of RF can be found in [26]. The Weka implementation of the identified classifiers was used in work presented in this paper.

3. RESEARCH METHOD

An experiment was executed to test the proposed approach's generalization capability in identifying artificial speech as an independent system rather than as part of an integrated ASV system. The ASVspoof 2015 dataset, the largest and most used public dataset for artificial speech detection, was used to measure the performance of the proposed approach. The recent ASVspoof 2019 dataset was not included as it was designed to evaluate the impact of the countermeasures on the reliability of an ASV system when subjected to spoofing attacks [27], which is out of the scope of the work presented in this paper.

The ASVspoof 2015 dataset used in the experiment was made up of speech synthesis and voice conversion attacks in addition to genuine speeches. The ASVspoof 2015 dataset was collected and generated from a total of 106 speakers, specifically 45 male and 61 female speakers. The genuine speeches of the ASVspoof 2015 were recorded in a semi-anechoic chamber having a solid floor, whereas the spoof speeches were generated using ten different common speech synthesis and voice conversion algorithms. These algorithms produced ten different categories of attacks (S1-S10). The known attacks in the ASVspoof 2015 dataset were made up of S1-S5 attacks, which used common voice conversion and speech synthesis algorithms. The unknown attacks in the ASVspoof 2015 dataset were made up of S6-S10 attacks. S1, S2, and S6-S9 attacks were generated using voice conversion algorithms, whereas S3, S4, and S10 attacks were generated using speech synthesis algorithms. Details on each of the ten spoofing algorithms used in the production of spoof speeches in the ASVspoof 2015 dataset are available in [6].

Four types of features were extracted from the audio recordings of the ASVspoof 2015 dataset, whereas the classifications were conducted using the weka tool. Most of the parameters set in the Weka tool were empirically found to be working well in most cases; hence this work uses the suggested parameters by Weka. There were training, development, and evaluation sets in the ASVspoof 2015 datasets. As described in [6], the training set is to train and build a PAD model, whereas the development set is for model tuning and refinement, and the evaluation set is for model evaluation. An experiment was conducted to evaluate the performances of the different combinations of the extracted features and classifiers. The experiment was conducted for each combination of features and classifiers, where both training and development sets were used to train the model, whereas the evaluation set was used for validation. The experiment was conducted using a machine with specifications: Intel i5-3210M processor, 2.50 GHz, 8 GB of RAM, Windows 10 (64-bit) OS.

4. RESULTS AND DISCUSSION

In this work, we showed the accuracy and F1-score of each model as a supporting metric in addition to the EER for a better comparison of the model performances. This is because a lower EER may not necessarily indicate that a model predicted more instances correctly. F1-score is often used in binary classification to evaluate how good the classifier is in detecting positive cases. Table 1 shows the results of experiment 1, whereby the detection was performed to identify genuine or artificial speech, with the best-performed combination of features and classifiers using the four features proposed. Several recent works that used both training and development sets for model training, namely $Model_1$-$Model_3$ [28]–[30], were used for comparison to present the competitiveness of the proposed approach to the state-of-the-art. In the remainder of this section, the combination of features and classifiers is represented using the model number, as shown in Table 1. Figure 8 shows detection error tradeoff (DET) curves for $Model_4$-$Model_7$ in experiment 1.

Table 1. The performances of the models trained with both train set and development set and tested with the evaluation set of the ASVspoof 2015 dataset

Model	Experiment 1 (Eval Set)		
	EER (%)	Accuracy (%)	F1-Score (%)
$Model_1$: Scattering cepstral coefficients (SCC)+GMM-UBM [28]	0.18	-	-
$Model_2$: Compressed sensing for high dimensional features (CS-HD)+i-vector [29]	0.24	-	-
$Model_3$: CQCC+SCC+GMM-UBM [30]	0.10	-	-
$Model_4$: Spectrogram image CLF+RF	17.61	93.02	96.42
$Model_5$: Spectrogram image LBP+RF	17.09	94.35	97.11
$Model_6$: MFCC image CLF+RF	0.10	99.93	99.96
$Model_7$: MFCC image LBP+RF	30.01	95.01	97.45

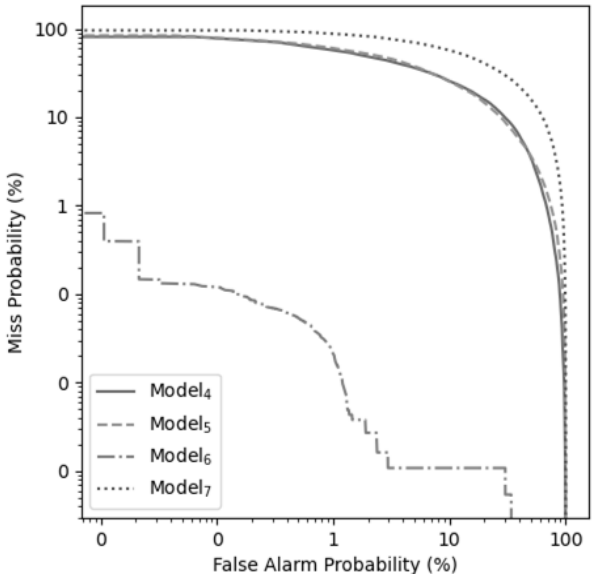


Figure 8. DET curves of $Model_4$-$Model_7$ in experiment 1

As the primary metric used in the ASVspoof 2015 was EER only, hence the accuracy of the $Model_1$-$Model_3$ are not shown in Table 1. From Table 1, most of the proposed models performed with over 17% EER in experiment 1. All the proposed models ($Model_4$-$Model_7$) achieved accuracy over 90% in experiment 1. In experiment 1, $Model_6$ achieved the lowest EER and the highest accuracy among the proposed models. It can be observed clearly in Figure 8 that the performance of $Model_6$ was far better than other models. On the other hand, all the proposed models ($Model_4$-$Model_7$) achieved over 96% F1-score. This indicates that the proposed models were very good in detecting the spoof voices while at the same time has low misclassification of genuine instances as a spoof.

The combination of the MFCC image with the CLF feature extractor has produced a robust feature that enabled the $Model_6$ to perform the best in experiment 1. MFCC uses a Mel scaling that produces a series of coefficients resembling the resolution of the human auditory system, which is different from spectrogram that uses a linear frequency scaling. In addition, the differences in the spatial distribution of color in the

MFCC images between genuine and spoof could be detected by the CLF feature extractor. Therefore, the MFCC based features performed the best when using the CLF feature extractor for artificial speech detection. Other than the robustness of the MFCC CLF feature, the use of RF may be one of the factors that lower EER was achieved by $Model_6$ in experiment 1. Due to the nature of RF, decision trees with more variation were built when the number of instances in training data to be randomly selected increases. Eventually, this produces a more generalized predictive model as the similarity between the bagged trees decreased. Therefore, having more data in model training may improvise the detection rate, though it may not always be the case.

In terms of detection error trade-off, a DET curve was presented in Figure 8 using the results obtained in experiment 1. A DET curve shows the detection error trade-off between the false-negative rate (miss probability) and false positive rate (false alarm probability) of a binary classification model. From Figure 8, $Model_6$ has a significantly lower detection error trade-off than other models in experiment 1. This indicates $Model_6$ performed significantly better than other models in experiment 1.

Besides, the robustness of $Model_6$ can also be seen by looking at the ISO/IEC standard metrics, namely, attack presentation classification error rate (APCER) and bonafide presentation classification error rate (BPCER) of the model. In total, only 130 out of 193,404 instances (0.07%) in the ASVspoof 2015 evaluation set were misclassified by $Model_6$. The APCER of $Model_6$ was 0.02%, given 29 out of 184,000 spoof instances were misclassified as genuine. The BPCER of $Model_6$ was 1.07%, given 101 out of 9,404 genuine instances were misclassified as a spoof. Nonetheless, the difference between APCER and BPCER was about 1%. To further compare our best model, $Model_6$, with recent works, the comparison of artificial speech detection by category of attacks (S1-S10) is presented in Table 2.

Table 2. The comparison of the performance of our best model with recent works on the evaluation set of the ASVspoof 2015 dataset by category of attacks (S1-S10)

Model	EER (%)									
	Known Attack						Unknown Attack			
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
$Model_1$		0.02					0.33			
$Model_2$	0.02	0.03	0.01	0.01	0.02	0.01	0.00	0.01	0.00	26.28
$Model_3$	0.00	0.01	0.00	0.00	0.02	0.01	0.01	0.00	0.00	0.95
$Model_6$	0.14	0.10	0.02	0.02	0.21	0.21	0.14	0.02	0.12	0.03

From Table 2, it can be observed that our model, $Model_6$, significantly outperformed $Model_1$ - $Model_3$ for the S10 attacks scenario, the most difficult spoofing attack. $Model_6$ also produced comparable performances on other categories of attack. Besides, it can be observed that $Model_6$ recorded 0.02-0.21% EER across S1-S10 attacks. $Model_6$ that performed with an overall EER of 0.10%, recorded a significantly higher EER of 0.95% on the S10 attack despite the model achieved below 0.02% EER in other attacks (S1-S9). This indicates that $Model_6$ is more generalized than the others.

There were 9,404 genuine instances and 18,400 spoof instances of each attack type (S1-S10) in the ASVspoof 2015 evaluation set. The misclassified instances for genuine, known, and unknown attacks by $Model_6$ were 101, 11, and 18 instances, respectively. An interesting observation is that there were no instances from S10 attacks being misclassified as genuine by $Model_6$, in which the 0.03% EER for S10 attacks were incurred by the false alarm. Unlike known attacks (S1-S5), which were generated using a vocoder, S10 attacks were generated without a vocoder. This was the factor that has caused most of the state-of-the-art voice PAD systems to suffer from significantly higher EER on S10 attacks. Comparably, $Model_6$ has successfully identified all S10 attacks as a spoof; hence it was more generalized and effective in detecting artificial speech regardless of the use of the vocoder.

To prevent unreliable performance evaluation, EER, accuracy, and F1-score were used in this work. From the results shown in Table 1, the performance of $Model_6$ is reliable as the EER, accuracy, and F1-score were good. Very high accuracy and F1-score but high EER can be obtained if the proportion of either class of the test set was overwhelming and the model biased toward one of the classes with overwhelming proportion. For example, a test set was made up of 100 instances with 90 spoof instances and ten genuine instances. If the model was bias and overfitting, it might predict all instances of the test set as a spoof to achieve high accuracy and F1-score. In this case, the accuracy of the model would be 90%, whereas the EER would be 50%. However, $Model_6$ was not the case. From the results, as shown in Tables 1 and 2, the low EER achieved by $Model_6$ indicated that the high accuracy achieved was neither due to bias nor overfitting. The combination of RF and MFCC image-based CLF features was shown to be effective in detecting artificial speech as $Model_6$ produced a low EER of 0.10% while achieving high accuracy and F1-score of 99.93% and 99.96%,

respectively. It is also shown to be able to produce similar detection performance on all categories of attacks (S1-S10).

5. CONCLUSION

In this paper, a feature engineering approach to produce handcrafted features for artificial speech detection was proposed. The contribution of this paper is in the proposed combination of features engineered using data transformation approaches and RF classifier for artificial speech detection. Four types of image-based spectrogram and MFCC features were extracted to classify genuine and spoof speeches. The ASVspoof 2015 dataset was used in the experiment to determine the effectiveness of the proposed approach against artificial speech. An experiment was run to compare the performance of the new features with the RF classifier for artificial speech detection. From the experiment, the results showed that the proposed approach could produce a model ($Model_6$) which used an Image Filter called CLF to extract features from MFCC images and a Random Forest as the classifier. The combination of the MFCC CLF feature and RF classifier generated a well-performed model, which yields good EER, accuracy, and F1-score of 0.10%, 99.93%, and 99.96%, respectively, in detecting artificial speech. However, in a real-world scenario, speech data were always exposed to various noises that deteriorate the audio quality. As the ASVspoof 2015 dataset contained only clean audio recording, the proposed approach may not be able to achieve similar performance when tested on the noise added dataset. Hence, future work is directed to test the proposed approach on the noise added dataset. Then, the investigation of feature fusion and ensemble classifiers to improve the performance further and to expand the detection to other types of presentation attacks such as replay attacks. In addition, more datasets will be used to evaluate the generalization capability of feature engineered using a data transformation approach against previously unseen spoofing attacks. Lastly, the integration of the proposed approach with the ASV systems will be conducted and tested on the ASVspoof 2019 dataset.

REFERENCES

[1] A. A. Mallouh, Z. Qawaqneh, and B. D. Barkana, "New transformed features generated by deep bottleneck extractor and a GMM–UBM classifier for speaker age and gender classification," *Neural Computing and Applications*, vol. 30, no. 8, pp. 2581–2593, Oct. 2018, doi: 10.1007/s00521-017-2848-4.

[2] A. I. Abdurrahman and A. Zahra, "Spoken language identification using i-vectors, x-vectors, PLDA and logistic regression," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2237–2244, Aug. 2021, doi: 10.11591/eei.v10i4.2893.

[3] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 5115–5119, doi: 10.1109/ICASSP.2016.7472652.

[4] S.-H. Yoon and H.-J. Yu, "A simple distortion-free method to handle variable length sequences for recurrent neural networks in text dependent speaker verification," *Applied Sciences*, vol. 10, no. 12, Jun. 2020, doi: 10.3390/app10124092.

[5] Y. Gong, J. Yang, J. Huber, M. MacKnight, and C. Poellabauer, "REMASC: realistic replay attack corpus for voice controlled systems," in *Interspeech 2019*, Sep. 2019, pp. 2355–2359, doi: 10.21437/Interspeech.2019-1541.

[6] Z. Wu *et al.*, "ASVspoof: the automatic speaker verification spoofing and countermeasures challenge," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 588–604, Jun. 2017, doi: 10.1109/JSTSP.2017.2671435.

[7] T. Kinnunen *et al.*, "The ASVspoof 2017 challenge: assessing the limits of replay spoofing attack detection," in *Interspeech 2017*, Aug. 2017, pp. 2–6, doi: 10.21437/Interspeech.2017-1111.

[8] M. Todisco *et al.*, "ASVSpoof 2019: future horizons in spoofed and fake audio detection," in *Interspeech 2019*, Sep. 2019, vol. 2019-Septe, pp. 1008–1012, doi: 10.21437/Interspeech.2019-2249.

[9] M. Pal, D. Paul, and G. Saha, "Synthetic speech detection using fundamental frequency variation and spectral features," *Computer Speech and Language*, vol. 48, pp. 31–50, Mar. 2018, doi: 10.1016/j.csl.2017.10.001.

[10] C. Demiroglu, O. Buyuk, A. Khodabakhsh, and R. Maia, "Postprocessing synthetic speech with a complex cepstrum vocoder for spoofing phase-based synthetic speech detectors," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 671–683, Jun. 2017, doi: 10.1109/JSTSP.2017.2673807.

[11] I. Ozer, Z. Ozer, and O. Findik, "Lanczos kernel based spectrogram image features for sound classification," *Procedia Computer Science*, vol. 111, no. 2015, pp. 137–144, 2017, doi: 10.1016/j.procs.2017.06.020.

[12] P. Parasu, J. Epps, K. Sriskandaraja, and G. Suthokumar, "Investigating light-ResNet architecture for spoofing detection under mismatched conditions," in *Interspeech 2020*, Oct. 2020, pp. 1111–1115, doi: 10.21437/Interspeech.2020-2039.

[13] J. Yang, R. K. Das, and H. Li, "Significance of subband features for synthetic speech detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2160–2170, 2020, doi: 10.1109/TIFS.2019.2956589.

[14] A. Gomez-Alanis, A. M. Peinado, J. A. Gonzalez, and A. M. Gomez, "A light convolutional GRU-RNN deep feature extractor for ASV spoofing detection," in *Interspeech 2019*, Sep. 2019, vol. 2019-Septe, pp. 1068–1072, doi: 10.21437/Interspeech.2019-2212.

[15] A. Gomez-Alanis, A. M. Peinado, J. A. Gonzalez, and A. M. Gomez, "A gated recurrent convolutional neural network for robust spoofing detectionn," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 27, no. 12, pp. 1985–1999, Dec. 2019, doi: 10.1109/TASLP.2019.2937413.

[16] C. B. Tan *et al.*, "A survey on presentation attack detection for automatic speaker verification systems: state-of-the-art, taxonomy, issues and future direction," *Multimedia Tools and Applications*, vol. 80, no. 21–23, pp. 32725–32762, Sep. 2021, doi: 10.1007/s11042-021-11235-x.

[17] A. M. Badshah, J. Ahmad, N. Rahim, and S. W. Baik, "Speech emotion recognition from spectrograms with deep convolutional neural network," in *2017 International Conference on Platform Technology and Service, PlatCon 2017 - Proceedings*, Feb. 2017, vol. 24, no. 6, pp. 1–5, doi: 10.1109/PlatCon.2017.7883728.

[18] C. K. On, P. M. Pandiyan, S. Yaacob, and A. Saudi, "Mel-frequency cepstral coefficient analysis in speech recognition," in *2006 International Conference on Computing and Informatics*, Jun. 2006, pp. 1–5, doi: 10.1109/ICOCI.2006.5276486.
[19] D. Y. Mohammed, K. Al-Karawi, and A. Aljuboori, "Robust speaker verification by combining MFCC and entrocy in noisy conditions," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2310–2319, Aug. 2021, doi: 10.11591/eei.v10i4.2957.
[20] M. Towsey *et al.*, "Long-duration, false-colour spectrograms for detecting species in large audio data-sets," *Journal of Ecoacoustics*, vol. 2, no. 1, pp. 1–1, Apr. 2018, doi: 10.22261/JEA.IUSWUI.
[21] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, Jan. 1996, doi: 10.1016/0031-3203(95)00067-4.
[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, Nov. 2009, doi: 10.1145/1656274.1656278.
[23] J. Monteiro, J. Alam, and T. H. Falk, "An ensemble based approach for generalized detection of spoofing attacks to automatic speaker recognizers," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 6599–6603, doi: 10.1109/ICASSP40776.2020.9054558.
[24] M. H. A. Hijazi, S. Kieu Tao Hwa, A. Bade, R. Yaakob, and M. Saffree Jeffree, "Ensemble deep learning for tuberculosis detection using chest X-ray and canny edge detected images," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 4, pp. 429–435, Dec. 2019, doi: 10.11591/ijai.v8.i4.pp429-435.
[25] T. R. Jayanthi Kumari and H. S. Jayanna, "Limited data speaker verification: fusion of features," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 6, pp. 3344–3357, Dec. 2017, doi: 10.11591/ijece.v7i6.pp3344-3357.
[26] O. Sagi and L. Rokach, "Ensemble learning: a survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, pp. 1–18, Jul. 2018, doi: 10.1002/widm.1249.
[27] X. Wang *et al.*, "ASVspoof 2019: a large-scale public database of synthetized, converted and replayed speech," *Computer Speech and Language*, vol. 64, Nov. 2020, doi: 10.1016/j.csl.2020.101114.
[28] K. Sriskandaraja, V. Sethu, E. Ambikairajah, and H. Li, "Front-end for antispoofing countermeasures in speaker verification: scattering spectral decomposition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 1–1, 2017, doi: 10.1109/JSTSP.2016.2647202.
[29] Y. Zhao, R. Togneri, and V. Sreeram, "Compressed high dimensional features for speaker spoofing detection," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Dec. 2017, pp. 569–572, doi: 10.1109/APSIPA.2017.8282108.
[30] Y. Zhao, R. Togneri, and V. Sreeram, "Spoofing detection using adaptive weighting framework and clustering analysis," in *Interspeech 2018*, Sep. 2018, vol. 2018-Septe, pp. 626–630, doi: 10.21437/Interspeech.2018-1042.

BIOGRAPHIES OF AUTHORS

Choon Beng Tan received his B.Comp.Sc. and M.Sc. degrees from Universiti Malaysia Sabah (UMS) in 2016 and 2018. He is now doing his PhD study in Computer Science in Universiti Malaysia Sabah (UMS). His recent work includes malware classification using machine learning and ensemble techniques, and cloud data integrity scheme; he is now working on voice presentation attack detection. His research interests include information security, cloud computing, and voice biometric security. He can be contacted at email: tanchoonbeng@ums.edu.my.

Mohd Hanafi Ahmad Hijazi is an Associate Professor of Computer Science at the Faculty of Computing and Informatics, Universiti Malaysia Sabah in Malaysia. His research work addresses the challenges in knowledge discovery and data mining to identify patterns for prediction on structured and/ or unstructured data; his particular application domains are medical image analysis and understanding and sentiment analysis on social media data. He has authored/ co-authored more than 50 journals/ book chapters and conference papers, most of which are indexed by Scopus and ISI Web of Science. He also served on the program and organizing committees of numerous national and international conferences. He is the leader of data technologies and applications research group at the faculty. He can be contacted at email: hanafi@ums.edu.my.

Frazier Kok holds a Degree in Computer Science from Universiti Teknologi Malaysia (UTM) 2001 and has 20 years' experience in ICT industry. He has completed various projects ranging from being a programmer, designer, solution architect, project coordinator and project manager. Now as an Executive Director in Bayurini Sdn Bhd, he is also extensively involved in Major Product Deployment in various customer sites. He can be contacted at email: frazier@bayurini.com.

Dr. Mohd Saberi Mohamad is a Professor of Bioinformatics and Artificial Intelligence at the Department of Genetics and Genomics, the College of Medicine and Health Sciences, UAE University. His research areas are Bioinformatics, Artificial Intelligence, Data Science, and Computational Biology. He has received the 2018 Malaysia's Research Star Award in the category of young researchers by the Malaysia Ministry of Higher Education. He has been the project leader for 19 research grants and the project member for 20 research grants. He has also published 282 articles in the international refereed journal, international conferences, and book chapters; and produced 14 books (research books, edited books, and original book). Previously before joining UAUE, he had experience as a Director for the Institute For Artificial Intelligence and Big Data, a founder of the Department of Data Science, and a head of the Artificial Intelligence and Bioinformatics Research Group. Besides, he was also a head and member of 65 academic and research committees at the faculty, university, national, and international levels. He also has been appointed as an advisory board for Artificial Intelligence Research Institute and IoT Digital Innovation Hub in Europe. He served as seven chairman and 39 members of the committees for several international conferences in Europe, ASEAN, Japan, US, Australia, China, and Malaysia. He has become an accreditation panel to Malaysia Qualification Agency (MQA) to review academic programs in the field of Bioinformatics since 2013. He can be contacted at email: saberi@uaeu.ac.ae.

Dr. Puteri Nor Ellyza binti Nohuddin received her B.Sc. in Computer Science from University of Missouri-Columbia, USA and her M.Sc. IT from Universiti Teknologi MARA. In 2012, she was awarded her Ph.D. in Computer Science from the University of Liverpool, UK. Puteri joins Institute of IR4.0 (IIR4.0), Universiti Kebangsaan Malaysia as a Research Fellow in July 2015. Prior to coming to IIR4.0, she was lecturer at the Universiti Pertahanan Nasional Malaysia, Kuala Lumpur. Prior to her academic career, she worked with several conglomerates such as ExxonMobil, Sime Darby, Shell IT and Malaysian Resources Corporation Berhad as System Analyst. Puteri's teaching interests include Programming, Database systems and Data mining. Her primary research interests are in the field of Big Data, Data Mining and Knowledge Engineering. Specifically, she is interested in Time Series Clustering, Trend mining, Tacit Knowledge, and Social Network Analysis. She can be contacted at email: puteri.ivi@ukm.edu.my.

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 173∼178
ISSN: 2252-8938, DOI: 10.11591/ijai.v11i1.pp173-178

Adaptive weight assignment scheme for multi-task learning

Aminul Huq[1], **Mst. Tasnim Pervin**[2]
[1]Department of Computer Science and Engineering, Brac University, Dhaka, Bangladesh
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China

Article Info

Article history:

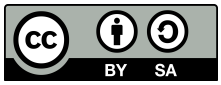
Received Jul 31, 2021
Revised Nov 20, 2021
Accepted Dec 4, 2021

Keywords:

Adaptive weight assignment
Deep learning
Dynamic weight average
Multi-task learning
Uncertainty weights

ABSTRACT

Deep learning based models are used regularly in every applications nowadays. Generally we train a single model on a single task. However, we can train multiple tasks on a single model under multi-task learning (MTL) settings. This provides us many benefits like lesser training time, training a single model for multiple tasks, reducing overfitting, and improving performances. To train a model in multi-task learning settings we need to sum the loss values from different tasks. In vanilla multi-task learning settings we assign equal weights but since not all tasks are of similar difficulty we need to allocate more weight to tasks which are more difficult. Also improper weight assignment reduces the performance of the model. We propose a simple weight assignment scheme in this paper which improves the performance of the model and puts more emphasis on difficult tasks. We tested our methods performance on both image and textual data and also compared performance against two popular weight assignment methods. Empirical results suggest that our proposed method achieves better results compared to other popular methods.



Corresponding Author:

Aminul Huq
Department of Computer Science and Engineering,Brac University
66 Mohakhali, Dhaka-1212, Bangladesh
Email: aminul.huq@bracu.ac.bd

1. INTRODUCTION

From the beginning of the last decade deep learning methods has been used vastly in various applications. The reach of it has exceeded tremendously not only in the field of computer science but also in electrical engineering, civil engineering, mechanical engineering and other fields as well. It is due to the fact that deep neural networks (DNN), have achieved human level competence in various applications like image classification [1], question answering [2], lip reading [3], and video games [4]. Deep neural networks have the capability to find out complex and hidden features of the input data without any assistance. Previously these models were depended on hand crafted features [5]-[10].

Human beings have the capability to perform multiple tasks simultaneously without harming performance of any tasks. Humans do this regularly and are able to decide which tasks can be done at the same time. That is why in recent years a lot of focus have been put into multi-task learning using deep neural network methods. Generally, a single model is devoted to performing a single task. However, performing multiple tasks increases the performance of the model, reduces training time and overfitting [11]. Often we find small insufficient datasets for individual tasks but if the tasks are related somehow then we can use this shared information and build a large enough dataset which will reduce this problem. Currently in the field of mult-task learning, several research work is going on to create new deep neural network architectures for multi-task learning setting [12],[13], deciding which tasks should be learned together [14] and how to assign weights to the loss values

[15],[16]. In this research work we focus on creating a dynamic weight assignment technique which will assign different weights to the loss values in each epoch during training. In our research work, we propose a new method for assigning weights to all loss values and test it against two datasets which are used in both image and text domain. The contributions of our research work are: i) We propose an intuitive loss weighting scheme for multi-task learning; ii) We tested our method against both image and text domain by using two different dataset. We did this to ensure that our method performs well across all domains; and iii) We compared our method against two popular weight assigning schemes for comparing the performance of our method

2. RESEARCH METHOD

In this section we will provide a discussion about previous research work performed in this field. Next, we will provide our proposed method.

2.1. Literature review

One of the earliest papers on multi-task learning is provided by R. Caruana [11]. In the manuscript, the author explored the idea of multi-task learning and showed it's effectiveness under different datasets. The author also explained how multi-task learning works and how it can be used in backpropagation. To train a deep neural network based on multi-task learning setting we need to consider which layers of network are shared among all the tasks and which layers are used for individual tasks. Previously, most of the research work has been focused on the concept of hard parameter sharing concept [17]-[19]. In this scenario, the user defines the shareable layers up to a particular point after which all layers are assigned per each task. There is also the concept of soft-parameter sharing where a single column exists for all the tasks in the network. A special mechanism is designed to share the parameter across all the network. Popular approaches for this method is cross-stitch [13] and sluich [20]. A new approach named Ada-share has been proposed recently where the model learns dynamically which layers to share for all tasks and which layers to be used for single tasks [14]. The authors also proposed a new loss function which ensures the compactness of the model as well as the performance of it.

Weight assignment is a very crucial task in the field of multi-task learning. Previously weights either had equal values or some hand-tuned value was assigned by the researchers [18],[21],[22]. However in scenarios where a large number of tasks existed for the multi-task learning model to perform, such approaches fall short. A method based on uncertainty was proposed by [15]. Later a revised method of this approach was proposed by [12]. In this paper, the authors improved the previous uncertainty based method by adding a positive regularization term. Dynamic weight average method was proposed by [12]. In this method the authors calculated the relative change in loss values in previous two epochs and used softmax function on these values to get the weights. In the paper, Gong *et al.* [23] performed a comparative study of different weight assigning scheme. However, they didn't study these methods in any domain other than images. Also, the dataset they used had only 2 tasks.

2.2. Adaptive weight assignment

Our proposed method is simple and it takes into account of the loss value of each task in each epoch. Compared to other methods our method is easy to implement. Generally, in multi-task learning settings to train the model we need to sum up all the loss values with their weights and then perform backpropagation for updating the weights of the model. This summation of losses can be expressed as (1),

$$\sum_{i=1,2,..n} W_i L_i = W_1 L_1 + W_2 L_2 + ... + W_n L_n. \tag{1}$$

here, W corresponds to the weight of the loss and L represents the loss for each task. In vanilla multi-task learning setting all the weights are set to 1. However, we must keep in mind that all the tasks are not the same. Some are more difficult than others so we need to provide more weights on difficult tasks to improve performance of the overall multi-task learning system. That is why we propose algorithm 1.

Our algorithm is based on the simple concept that difficult tasks will have more loss values than the easier ones. So we should put more emphasis or weights on those loss values while assigning less weights to the smaller loss values. What we do is take the sum of the loss values for each tasks and use it to figure out the ratio of how much a single tasks loss value contributes to the total loss. We multiply this value with the total number of tasks. Generally, in vanilla multi-task learning setting all loss values have equal weights 1. So

the total weight is then n for n number of tasks. That is why we multiply our ratios with n. Finally, we use these weights and using (1) compute the total loss for the multi-task learning model. Figure 1 provides a visual representation of the method.

Algorithm 1

Inputs: Loss values $L_1, L_2, .., L_n$, total no. of tasks n
Outputs: Total loss

1: **for** $t = 1, 2, \ldots n$ **do**
2: $TempLoss$ += L_t
3: **end for**
4: **for** $t = 1, 2, \ldots n$ **do**
5: $weights_t$ = $L_t/TempLoss$
6: $TotalLoss$ += $weights_t$ x L_t x n
7: **end for**

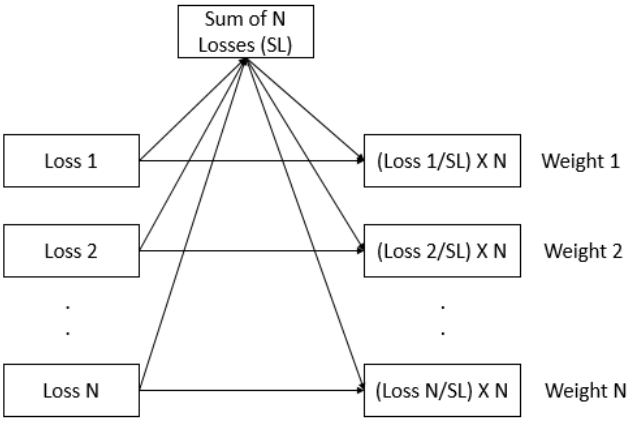


Figure 1. Flow diagram of our proposed method

One of the important things about designing loss weighting schemes is that we need to ensure that these weight calculating methods should not take a lot time because it will increase the training time. Table 1 provides a chart about the time required to execute these schemes including our method. From the table we can see that though our method is not the fastest method to compute weights but it certainly is not the slowest. Also, the time difference between the quickest method and our method is very small.

Table 1. Time required(s) for executing loss weighting schemes on CIFAR-100 and AGNews dataset

	CIFAR-100	AGNews
Re_Uncertainty	0.001	0.0004
DWA	0.0004	0.0002
Ours	0.0006	0.0003

3. RESULTS AND DISCUSSION

We will discuss about the dataset, experimental setup and results of the experiments in this section.

3.1. Dataset description

We used two different datasets in our experiment. They are CIFAR-100 [24] and AGNews [25]. The formal one is image based and the later one is text based. Since these datasets are designed for single task learning we created artificial tasks for multi-task learning settings. We created 5 different tasks from CIFAR-100 and 2 tasks from AGNews dataset. All the tasks were created based on the original tasks labels and we grouped different labels together to form multiple tasks. The tasks were created to ensure that no class imbalance exists for all tasks.

3.2. Experimental setup

We used two different deep neural network models for our experiment. We used wide resnet-28-10 (WRN) [26] for CIFAR-100 and a custom deep neural network for AGNews dataset. We split the final layer of the WRN model into 5 output layers for CIFAR-100 and 2 output layers for AGNews dataset. we trained WRN model for 100 epochs using SGD optimizer and set the learning rate to 0.001. We also used one cycle learning rate scheduler [27]. In order to train the AGNews dataset we at first tokenize the dataset and create a vocabulary dictionary based on it. Then we perform embedding of the text which is going to be the input of the model. Our custom deep neural network consists of two fully connected layers. We trained this model using SGD optimizer. To ensure the effectiveness of our method, we compared our proposed method against two state-of-the-art methods namely dynamic weight average (DWA) and uncertainty method. We also compared against single task learning and vanilla multi-task setting.

3.3. Experimental results

We will discuss about the performance of our method against two datasets in this section. Tables 2 and 3 represents the results of our overall experiment. In Figure 2 We have plotted test loss curves for the dataset as shown in Figures 2(a) and 2(b) CIFAR 100 and AGNew. In Table 2, we have the results on running experiments on CIFAR-100 dataset which is an image dataset. At the beginning we have results for all the five tasks in a single task learning settings. That is five different models were trained to get the results of these five tasks. Next under multi-task learning setting we trained four methods for these tasks. In vanilla multi-task learning we have assigned equal weights to each task for each epoch. Other methods Uncertainty, DWA and our method updates weights in each epoch. From this table we can see our proposed method out performs other methods in three out of five tasks. Also our method achieved second best performance in the rest of the two tasks. We can see that multi-task learning models performed better than STL models and also we needed to train only one single model for all five of these tasks.

Table 2. Accuracy (%) comparison of different methods and showing best scores (bold) and second best scores (italic)

	2 Class Classification	3 Class Classification	4 Class Classification	5 Class Classification	100 Class Classification
STL	74.52	*75.70*	*74.02*	**72.81**	**76.56**
MTL - Vanilla	79.97	74.36	70.97	67.95	60.23
MTL - Uncertainty	69.47	59.52	55.42	50.21	34.91
MTL - DWA	*80.33*	74.57	71.37	68.41	60.40
MTL - Ours	**81.68**	**77.01**	**74.41**	*72.07*	*66.81*

Table 3. Accuracy (%) comparison of different methods on AGNews dataset and showing best scores (bold) and second best scores (italic)

	2 Class Classification	4 Class Classification
STL	84.00	79.13
MTL - Vanilla	**86.57**	*80.11*
MTL - Uncertainty	84.56	75.94
MTL - DWA	85.86	79.77
MTL - Ours	*86.02*	**81.18**

We evaluate our methods performance on AGNews dataset which contains textual data. We have two tasks and at the beginning we train two individual models for these two tasks. After that we train four multi-task learning models with different weight assignment schemes. We can observe from the table that our proposed method performs well under one task and achieves second best score in the other one. Compared to other popular methods we can see that our proposed method is performing much better. If we look closely at the values we will see that other methods fail to achieve the best results. In some cases these approaches even fail to attain better performance than single task learning approach. We believe this is due to the fact the model architecture has a big impact on the performance of multi-task learning settings. In our experiment we focused on uniform deep neural network architecture for evaluation but some tasks might need a few extra convolutional or fully connected layers. If we put further emphasis on the deep neural network architecture then the performance of our proposed method would definitely be better in both tasks. We believe that a simpler

approach should be taken while assigning weights. As this step is performed in each iteration, too much parameterized and complex approach mind hinder the performance of the model and increase time complexity.

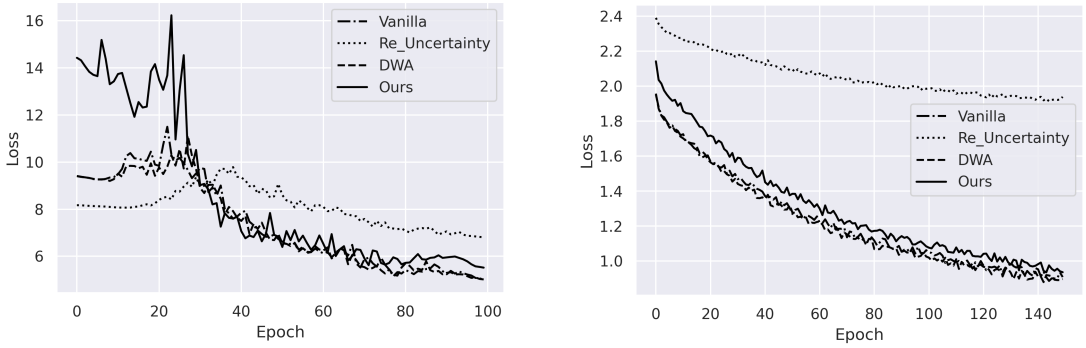


Figure 2. Loss vs epoch curve (a) CIFAR-100 and (b) AGNews

4. CONCLUSION

Understanding and properly executing different hyper-parameters is extremely crucial in training a deep neural network model for the best results. Multi-task learning settings have the upper-hand on single task learning when it comes to amount of data needed, time to train the model, reducing overfitting and increasing model performance. In multi-task learning settings since not all tasks are of equal difficulties assigning weight to the loss values is important to put more emphasis on difficult task. In this paper, we propose a new weight assignment scheme which aids in improving the performance of the multi-task learning model. Our proposed method out-performs other state-of-the-art weight assigning schemes in both image and textual domain and boosts the performance of the model.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 248–255, doi: 10.1109/cvpr.2009.5206848.

[2] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuad: 100,000+ questions for machine comprehension of text," in *Conf. on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392, doi: 10.18653/v1/d16-1264.

[3] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, "LipNet: End-to-End Sentence-level Lipreading," *arXiv preprint arXiv:1611.01599*, Nov 2016.

[4] J. X. Chen, "The evolution of computing: AlphaGo," *Computing in Science Engineering*, vol. 18, no. 4, pp. 4–7, Jul. 2016, doi: 10.1109/MCSE.2016.74.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proc. IEEE Comp. Society Conf. on Computer Vision and Pattern Recognition, CVPR*, vol. I, 2005, pp. 886–893, doi: 10.1109/CVPR.2005.177.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[7] R. Mehrotra, K. R. Namuduri, and N. Ranganathan, "Gabor filter-based edge detection," *Pattern Recognit.*, vol. 25, no. 12, pp. 1479–1494, Dec. 1992, doi: 10.1016/0031-3203(92)90121-X.

[8] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 5, pp. 489–497, May 1990, doi: 10.1109/34.55109.

[9] M. T. Pervin, S. Afroge, and A. Huq, "A feature fusion based optical character recognition of Bangla characters using support vector machine," in *3rd International Conference on Electrical Information and Communication Technology, EICT 2017*, Dec. 2018, vol. 2018-January, pp. 1–6, doi: 10.1109/EICT.2017.8275138.

[10] A. Huq, S. Afroge, and M. T. Pervin, "Combined zemike moments, binary pixel and histogram of oriented gradients feature extraction technique for recognizing hand written Bangla characters," in *3rd Int, Conference on Electrical Information and Communication Technology*, EICT 2017, vol. 2018, 2018, pp. 1–5, doi: 10.1109/EICT.2017.8275144.

[11] R. Caruana, "Multitask Learning," in textslLearning to Learn, Springer US, 1998, pp. 95–133.

[12] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *Proc. IEEE Comp. Society Conf. on Comp. Vision and Pattern Recognition*, vol. 2019, 2019, pp. 1871–1880, doi: 10.1109/CVPR.2019.00197.

[13] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-Stitch Networks for Multi-task Learning," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2016, vol. 2016-December, pp. 3994–4003, doi: 10.1109/CVPR.2016.433.
[14] X. Sun, R. Panda, R. Feris, and K. Saenko, "AdaShare: Learning what to share for efficient deep multi-task learning," in *Advances in Neural Information Processing Systems*, 2020, vol. 2020-December, [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/634841a6831464b64c072c8510c7f35c-Abstract.html.
[15] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 7482–7491, doi: 10.1109/CVPR.2018.00781.
[16] L. Liebel and M. Körner, "Auxiliary Tasks in Multi-task Learning," *arxiv.org/abs/1805.06334*, 2018.
[17] J. Huang, R. Feris, Q. Chen, and S. Yan, "Cross-domain image retrieval with a dual attribute-aware ranking network," in *Proc. of the IEEE Int. Conf. on Comp. Vision*, vol. 2015, 2015, pp. 1062–1070, doi: 10.1109/ICCV.2015.127.
[18] I. Kokkinos, "UberNet: Training a universal convolutional neural network for Low-, Mid-, and high-level vision using diverse datasets and limited memory," in *Proc. 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, Jul. 2017, vol. 2017, pp. 5454–5463, doi: 10.1109/CVPR.2017.579.
[19] B. Jou and S. F. Chang, "Deep cross residual learning for multitask visual recognition," in *MM 2016 - Proceedings of the 2016 ACM Multimedia Conference*, Oct. 2016, pp. 999–1007, doi: 10.1145/2964284.2964309.
[20] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard, "Latent multi-task architecture learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4822–4829, Jul. 2019, doi: 10.1609/aaai.v33i01.33014822.
[21] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE international conference on computer vision*, vol. 2015, 2015, pp. 2650–2658, doi: 10.1109/ICCV.2015.304.
[22] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arxiv.org/abs/1312.6229v4*, 2013.
[23] T. Gong et al., "A comparison of loss weighting strategies for multi task learning in deep neural networks," *IEEE Access*, vol. 7, pp. 141627–141632, 2019, doi: 10.1109/ACCESS.2019.2943604.
[24] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images.(2009)," *Technical report, University of Toronto*, 2009. [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html.
[25] X. Zhang, J. Zhao, and Y. Lecun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 2015-January, pp. 649–657, Sep. 2015.
[26] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference 2016*, BMVC 2016, 2016, vol. 2016-September, pp. 87.1-87.12, doi: 10.5244/C.30.87.
[27] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations App.*, May 2019, doi: 10.1117/12.2520589.

BIOGRAPHIES OF AUTHORS

Aminul Huq is a lecturer at Brac University. He received his Master's in Computer Science and Technology from Tsinghua University in 2021. He completed his Bachelor's from Rajshahi University of Engineering & Technology in 2017. His research interest lies in the field of Multi-task Learning and Adversarial Machine Learning. He could be contacted via email: aminul.huq@bracu.ac.bd.

Mst. Tasnim Pervin has completed her Master's in Computer Science and Technology from Tsinghua University in 2021. Her completed her Bachelor's from Rajshahi University of Engineering & Technology in 2017. Her research interest lies in the field of Medical Image Analysis, Adversarial Machine Learning and Domain Adaptation. She could be contacted via email: pervinmt10@mails.tsinghua.edu.cn.

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 179~188
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i1.pp179-188

Labeling of an intra-class variation object in deep learning classification

Putri Alit Widyastuti Santiary, I Ketut Swardika, Ida Bagus Irawan Purnama, I Wayan Raka Ardana, I Nyoman Kusuma Wardana, Dewa Ayu Indah Cahya Dewi
Department of Electrical Engineering, State Polytechnic of Bali, Bali, Indonesia

Article Info

Article history:

Received Jul 26, 2021
Revised Dec 19, 2021
Accepted Dec 31, 2021

Keywords:

Class labelling
Classification
Deep learning
Flower dataset
Intra-class variation

ABSTRACT

Machine orientation learning had demonstrated that deep learning (DL)-convolutional neural networks (CNNs) were robust image classifiers with significant accuracy. Although to been functional, DL scope classification as tight, well-defined as possible uses a 2-class object, for instance, cats and dogs. The DL classification faced many challenges, e.g., variation factors, the intra-class variation. This nature is presented in every object, its diversity of an object. The label was an exact given name of an intra-class variation object. Unfortunately, not every object had a specific name, in exceptionally high similarity inside the category. This paper explored those problems in flower plants' taxonomy naming. In supervised learned of DL, image datasets musted labeled with a meaningful word or phrase that humans are familiar with, a taxonomy naming. Labeled with visual feature extraction brought a fully automatic classification. Flower Plumeria L labeling extracted from perspective dimension scale of petal flower which automatically obtained by contour detection, and peaks of blue green red (BGR) histogram channels from bins histogram after object masked. Dataset collected on photography workbench equipped with webcam and ring light. Results showed labels for intra-class variation of Plumeria L in form of dimension-scale and BGR-peaks. The result of this study presented a novelty in building datasets for intra-class variation for the DL classification.

Corresponding Author:

I Ketut Swardika
Department of Electrical Engineering, State Polytechnic of Bali
Kampus Bukit Jimbaran, 80361, Indonesia
Email: swardika@pnb.ac.id

1. INTRODUCTION

Artificial intelligence (AI) technology has been widely applied without users realizing it. All devices are equipped with intelligent drive-by software that can recognize user-environment characteristics and connect to the networks. The global goal of AI is to provide a method to solve problems that humans perform intuitively and near automatically. AI-related to work in inference, planning, heuristics, or automatic machine reasoning [1]–[4]. Machine learning (ML) is a subfield of AI technology. ML more specifically focuses on pattern recognition that is learning from a dataset. ML allows a device to take actions or decisions based on optimal analysis results obtained after learning from the trained datasets. This allows users to get results or decisions quickly and are autonomous. ML has been widely applied in various fields to solve complex problems involving large volumes of events or data and requiring fast optimal decisions. One of the main things about ML is the object classification process [5]–[7]. Deep learning (DL) is a subfield of ML. DL uses a raw input to self-learning from data and not selected features that have been designed. Features then automatically gain from learning in the training process [8]–[13]. The convolutional neural networks (CNNs)

is a computer vision subfield that leverage ML to be the most powerful object image classifier [14]–[18]. Considering a large of neural networks, the term of deep consists of more than two hidden layers that involve a huge dataset to be trained. DL now is a thriving field with many practical applications and active research topics [19]–[22]. Although DL and CNNs have demonstrated that powerful image classification is robust under various challenges, to be accurate and functioning, DL scope object classification as tight and well-defined as primarily possible uses a 2-Class object for instances using mask or no-mask. However, development on the DL object classification faced many challenges, i.e., semantic gap (difference perceives, the human versus computer represented an object), and variation factors (the intra-class variation) [23]–[28].

In promotes deep learning forward to the state-of-the-art, researchers through internet communities develop network models and datasets. The first network model is LENET architecture with modified national institute of standards and technology (MNIST) dataset. Intending to classify handwriting (numbers 0-9) with significant accuracy usually uses as a benchmark of ML algorithms. Another dataset for the image classification algorithm is ImageNet, which consists of more than a thousand objects of everyday lives. The intra-class variation present in every object, its nature of diversity of an object. Every object has a class, for instance, class object dog. If an intra-class variation is used, the class object becomes a name of an object, for instance, a dalmatian (a white-dog with black-spot). Unfortunately, not every object has a specific name, in exceptionally high similarity inside the category. This paper explores this problem in flower plants' taxonomy naming, but the possibility also for intra-class variation of person, despite the fact person already has a given name or ID-number; instead, a visual feature can be a substitute for future works [29]–[31].

Building of image classifier model can be used available dataset through the DL internet communities. These datasets (MNIST, CIFAR, Flowers, Caltech, ImageNet, CVPR, STANFORDCARS) generally consist of a massive image of a noun class, i.e., cat, dog, or panda in the animals' category or others, and uses as a benchmark for machine learning algorithm study. Class labeling and categorizing these datasets are organized according to the WordNet hierarchy called a synonym set or synonym for short. As seen, object labeling with their actual name, i.e., the name species of flower in Flowers-17 dataset or year-maker-model-car in STANFORDCARS dataset [32]–[35].

The objects that have huge intra-class variation are difficult to have a given name. For the sake of simplicity generally uses their major-class name as a label. Indeed, the DL classification needs the image datasets to have labels associated with them. In the supervised learning algorithm of DL, the process needs to see these labels to teach itself how to recognize each class. The different classes must have unique labels, i.e., incremental integers. However, labels must be meaningful of a word or phrase that humans are familiar [36]–[38].

This study explores the intra-class variation of the flower Plumeria L (local name Bunga Jepun), a genus of flowering plants in the family Apocynaceae. In personal communication, the local botanical claim has collected about 400 varieties of this flower. Few varieties have registered in the standardized taxonomy and nomenclature database (Interagency Taxonomic Information System, itis.org), remain has a no-given name [39].

2. RESEARCH METHOD

2.1. Method of labeling

This study proposes an ID system for the intra-class variation of flower Plumeria L labeling for the DL classification. The term flower in this study meaning a single piece of flower, not a bunch of flowers bound at their plant stem. Hence, image pre-processing such as image segmentation is out of the scope of this study. The ID label consists of feature extraction from the object flower that can be retrieved using computer vision during the preparation stage in DL classification. Therefore, this method can fully automatically [38], [40], [41]. The feature that can be extracted from flower Plumeria L:

i) Shape. The flower Plumeria L in general has five petals and clearly can be used to distinguish them from other flowers. In an intra-class variation, the shape of individual petals, their configuration, and the overall shape of the flower are similar. Hence, the shape features are not considering in creating the flower ID label [7], [25], [32], [33].
ii) Scale. The scale of flower Plumeria L has variation in petal dimension, varies from has small to a big petal, some flowers have wrapped up the petal and others have blooming petal. Moreover, found in one stem, there is variation in scale as well. The dimension of the flower will vary in viewpoint. For the sake of simplicity, viewpoint only two will be considered, i.e., a natural or non-pose and petal pose [2], [6], [15], [31].
iii) Texture. The flower Plumeria L in general has a similar texture of the petal. Some variety has softer fine thin of petal texture. However, it cannot capture detail within a tiny image size needed for the DL classification. Hence, the texture feature is not considering in creating the flower ID label [41]–[43].

iv) Color. The main characterizes of the flower Plumeria L is petal color. The dominant petal color is bright white to bright red, dark color scarce, i.e., green to blue. Some variety has dark deep red color. The ID color of the petal will be extracted from the peak of their BGR histogram color [33]–[35], [44].

Therefore, the ID label will consist of scale and color features of petals in format ss-peakBGR.

Figure 1 shows four stages of the DL classification in general; at the preparation stage (block number 1), the proposed method of creating ID label is most useful, where the class images must be associated with a label. The remaining stages, i.e., dataset splitters, train networks, and evaluation progress as it is [42]–[44]. In the preparation stage, the flower Plumeria L will be processed to extract the perspective dimensions of the petal in cm unit. The extracted peak of blue green red (BGR) color histogram in the 8-bit unit. Creating ID label for each variant or class of flower Plumeria L. Archives hundred of a pose, non-pose photograph of flowers. Creating dataset completely with a label in CSV format.

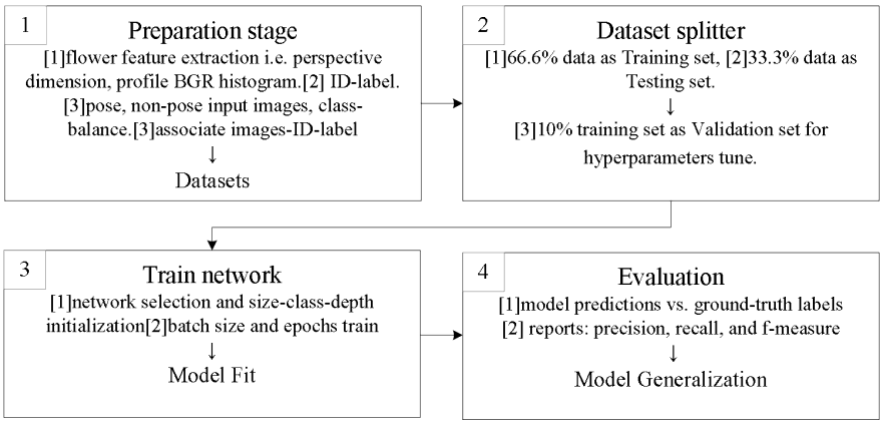


Figure 1. Propose method of ID-label extraction, where processes integrated within DL classification in the preparation stage (block 1)

2.2. Dataset collector

Flowers collect directly from the plant and process to the image-workbench not within five hours to ensure petal is fresh to prevent bias in measurement. The image workbench is set up with a camera and light. Exposer light ensures no shadow effect and camera setting with a minimum of lens distortion to minimize interference or noise image output. For consistency of the image output image workbench equipped with a 15×15 cm of green canvas paper ink-jet printing to void reflected light to the camera, finally image workbench operated by python code running on pc with NVidia graphics processing unit (GPU).

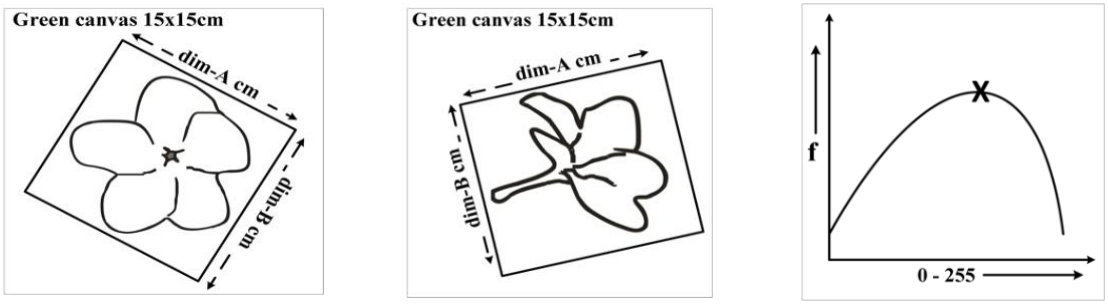
Camera position calibrated exactly perpendicular and center above the object. The edge of the green canvas makes a constraint controller for camera position and area to captures. The pixel metric (size of a pixel in cm) was obtained by calibrating the camera with a known dimension object in cm, such as a 7×7 cm glass cap. The code automatically extracts scale features by a rectangular bound of flower in a perspective way, including flower dimensions. The code also automatically extracts color features by computing the peak of BGR color histogram and return intensity bins value in 8-bit format. The images for one variety of flowers captured in a non-pose and petal pose within 200 times vary in viewpoints and flowers. The image dataset has 340 by 340 pixels with 24-bit color depth and 170 kB file size. The averages of the processing time for labeling of one variety of the flower Plumeria L with 200 images capture approximately 23 minutes 45 seconds or just for one image approximately 7 seconds.

Figure 2 shows the illustration at image workbench capture progress to feature extraction. Figure 2(a) and Figure 2(b) show the scale or the flower dimension feature extraction in pose and non-pose of the flower petal. The python code will create a rectangle bounding box in cm unit after scale calibration process (in figure mark as dim A and B in cm unit). Note that results will be dynamic according to the viewpoint. Figure 2(c) shows the peak of histogram per channel BGR of flower under evaluation. In axis-x, an 8-bit color resolution, and axis-y is a histogram frequency. In figure, a peak mark as letter x.

Figure 3 shows the Author operates a python code to creates the flower Plumeria L dataset for deep learning on a set of photography workbench that uses in this study. In figure, the photography workbench consists of a webcam installed on a rounded LED light on an adjustable vertical tripod. The height and orientation of the webcam from the table are adjusted using a green canvas as a reference that displays on a pc monitor. Then, the flower is placed over the green canvas for measurement with computer vision.

2.3. Accuracy of labeling

The labeling process in this research is limited to the preparation stage in Figure 1 (block 1). So that, the evaluation stage (block 4) in Figure 1 to obtains the accuracy of test and validation of classification results is out of the scope of this research. The flower Plumeria L ID label results from section 2.1. were obtained from computer vision evaluated by comparison with manual measurement of petal flower dimension. This measurement uses a laser distance meter (LDM) tool that has 2.0 mm measuring accuracy. A set of equipment was prepared for that purpose shown in Figure 4. The LDM measures the distance between laser sources with a reflector where the flower being measured is placed. The flower position is adjusted to get 2 flower dimensions i.e., the shortest and longest distance between laser source and reflector. The LDM is set to automatic mode to measure the distance of each in 1 second. The flower petal measured in 5 seconds, result in the dimension maximum, minimum, and averaged value for 5 times measurement. The accuracy of labeling is evaluated using the root mean square error (RMSE) and Pearson's coefficient of correlation (r) of output between the computer vision with LDM labeling [45], [46].



(a) (b) (c)

Figure 2. Image workbench scale feature extraction capture (a) pose, (b) non-pose petal, and (c) color peak feature extraction histogram

Figure 3. The author creating the flower Plumeria L dataset for deep learning study on a set of photography workbench

Figure 4. Manual measurement of petal flower dimension using laser distance meter (LDM)

3. RESULTS AND DISCUSSION

Unfortunately, the flower collectors cannot collect about 400 varieties of flower Plumeria L due to a different blooming session and time-consuming issue. For an initial study step, about 28 varieties of flower Plumeria L have been collected and processing. See section discussion for further, how flower Plumeria L has reacted 400 child varieties and compared with taxonomy databases naming [28], [36], [38].

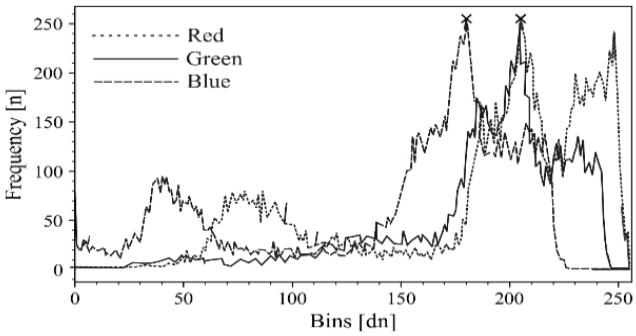
Table 1 shows the results of scale-color feature extraction of flowers from the image workbench. This table shows four representing the smallest to biggest petal flower collected during the first periods of research. Other 24 remain variant datasets not possible to show in this paper. In Table 1 column header contains Item ID (necessary marks of flower), computer vision (CV) results of scale dimensions of flowers in a perspective way in cm unit (dim A and B), the peak of the color histogram channel (BGR), LDM results of scale measurement in cm unit (dim A and B), RMSE and Accuracy (r). All values are computes and averaged from 200 image samples (n=200) [32], [34], [35]. The result shows overall accuracy from the comparison between CV with LDM method has reached 99% with RMSE in 3 mm. So that this proposed method of labeling of intra-class variation of flower Plumeria L can be justified.

Table 1. Tabulation of scale-color feature extraction of Plumeria L

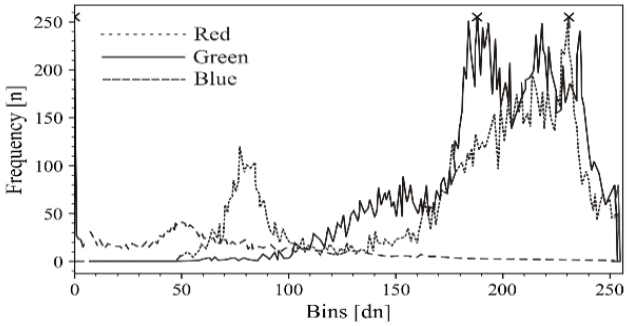
Item ID:	Viewpoint	CV		B	G	R	LDM		RMSE	Accuracy (r)
		dim A	dim B				dim A	dim B		
04-04-2021	Non-pose	4.45	4.56	0	226	236	4.60	4.80		
Genus: none*	Pose	4.23	4.35	0	242	253	4.40	4.40	0.316	0.99
Class ID:6		4.35	4.45	0	234	244				
Label		4.3-4.4-0-234-244								
03-04-2021	Non-pose	5.27	5.42	0	223	254	5.30	5.50		
Genus: none*	Pose	5.88	5.86	0	232	254	5.80	5.90	0.082	0.99
Class ID:5		5.58	5.64	0	227	254				
Label		5.5-5.6-0-227-254								
02-04-2021	Non-pose	5.60	5.47	0	217	225	5.70	5.50		
Genus: none*	Pose	5.72	5.69	0	242	253	5.80	5.90	0.224	0.99
Class ID:4		5.66	5.59	0	230	239				
Label		5.6-5.5-0-230-239								
11-04-2021	Non-pose	8.67	9.15	0	142	254	8.80	9.30		
Genus: none*	Pose	10.10	9.90	0	152	254	10.40	10.20	0.468	0.99
Class ID:16		9.39	9.53	0	147	254				
Label		9.3-9.5-0-147-254								
Overall RMSE	0.306									
Overall Accuracy (r)	0.99									

Using this method for labeling, images will be associated with class ID and their label ID, e.g., class id 6 labels, 4.3-4.4-0-234-244, and so on. After 75% of the flower collection and processing are complete, the scale features of petal flowers will be categorized into five Linkert scales, i.e., biggest, big, medium, small, smallest (bb, b, m, s, ss) for short in labeling [36]. Figure 5 shows the histograms of petal flower of the Plumeria L obtained from computer vision. Figure 5(a) for class ID: 6 (the flower variant in Figure 6), Figure 5(b) for class ID: 4 (the flower variant in Figure 7), and histogram Figure 5(c) for class ID:16 (the flower variant in Figure 8). In Figure 5, the peaks of BGR channel shows in black-cross marks (x). The bins in digital number of axis x obtained after traces histogram values with axis x range. The peak values in Table 1 result is a mean peak value computes from 200 image samples. As seen from Figure 5, the flower variant that has high similarity can be distinguished from their peaks of color channel profile [32], [34].

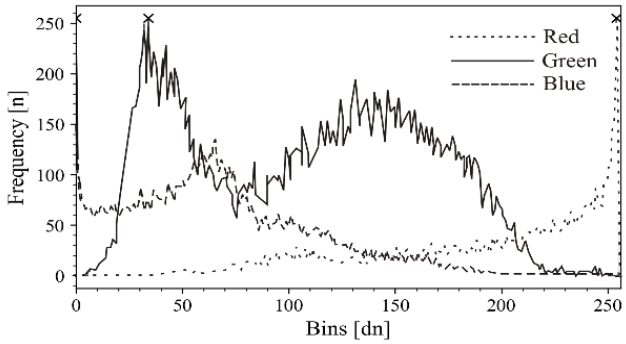
Figures 6-8 show the result of scale feature extraction capture from the image workbench. Figures 6(a)-8(a) are for non-pose of petal flower, while Figures 6(b)-8(b) are for pose of petal flower. In Figures 6-8 show of each image have four red dots that creates a rectangle with green color line. The red dot is computed from the rotated bounding box of the specified contour flower. Then added four blue dots at half of the green line length with the magenta color line where the dim A and dim B are obtained. In Table 1 genus is mask as none that means no associates of taxonomy naming in the database for this variety of flower Plumeria L [39], [44]. The world checklist of selected plant's families (wcsp.science.kew.org) registered 160 proposals of genus Plumeria taxonomy names. However, only 21 approved remain not accepted by status. For instance, the Plumeria Tourn. ex L. is approved [18], [23], [31], [41], [42]. The Plumeria is a flowering plant that is easy to be cultivated, by grafting technique; hence cross cultivation can be with any varieties, resulting in about a hundred (400 variety, personal communication) varieties of petal flowers.



(a)

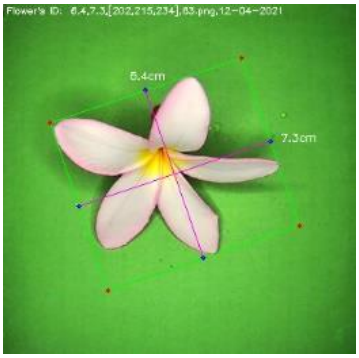


(b)

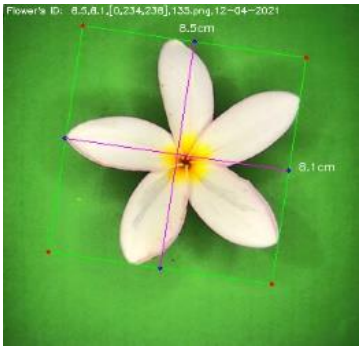


(c)

Figure 5. Histograms of petal flower of the Plumeria L obtained from computer vision, (a) Class ID: 6, (b) Class ID: 5, and (c) Class ID: 4



(a)



(b)

Figure 6. Scale feature extraction capture from image workbench for class ID: 6, (a) non-pose and (b) pose petal

(a) (b)

Figure 7. Same as Figure 5, but for class ID:5, (a) non-pose and (b) pose petal

(a) (b)

Figure 8. Same as Figure 5, but for class ID:4, (a) non-pose and (b) pose petal

4. CONCLUSION

The flower Plumeria L has a huge of intra-class variation, from scale to the color feature of a petal flower. Not of each variety of flower Plumeria L has a taxonomy naming. Few varieties have been approved. In the supervised learning algorithm of DL, the image datasets must be labeling with a meaningful word or phrase that humans are familiar with, the taxonomy naming. Labeling with visual feature extraction brings a fully automatic classification, which can be processed in the initial preparation stage. The flower Plumeria L labeling extracted from the perspective scale of petal dimension and the peaks of BGR color histogram channels of a petal flower. For short, the perspective scale of the petal dimension is categorized into a five of the Linkert scale. This method presents a novelty in building datasets for intra-class variation for the DL classification.

ACKNOWLEDGEMENTS

This research was funded by the Budget Implementation Registration Form of the State Polytechnic of Bali. The contract number of this project is SP DIPA-023.18.2.677608/2021 (November 23, 2020).

REFERENCES

[1] J. Y. W. Jien, A. Baharum, S. H. A. Wahab, N. Saad, M. Omar, and N. A. M. Noor, "Age-based facial recognition using convoluted neural network deep learning algorithm," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 3, pp. 424–428, Sep. 2020, doi: 10.11591/ijai.v9.i3.pp424-428.

[2] S. Z. M. Zaki, M. Asyraf Zulkifley, M. Mohd Stofa, N. A. M. Kamari, and N. Ayuni Mohamed, "Classification of tomato leaf diseases using MobileNet v2," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 2, pp. 290–296, Jun. 2020, doi: 10.11591/ijai.v9.i2.pp290-296.

[3] H. Lu, Y. Li, M. Chen, H. Kim, and S. Serikawa, "Brain Intelligence: go beyond artificial intelligence," *Mobile Networks and Applications*, vol. 23, no. 2, pp. 368–375, Apr. 2018, doi: 10.1007/s11036-017-0932-8.

[4] R. Vaishya, M. Javaid, I. H. Khan, and A. Haleem, "Artificial intelligence (AI) applications for COVID-19 pandemic," *Diabetes and Metabolic Syndrome: Clinical Research and Reviews*, vol. 14, no. 4, pp. 337–339, Jul. 2020, doi: 10.1016/j.dsx.2020.04.012.

[5] T. Meng, X. Jing, Z. Yan, and W. Pedrycz, "A survey on machine learning for data fusion," *Information Fusion*, vol. 57, pp. 115–

129, May 2020, doi: 10.1016/j.inffus.2019.12.001.
[6] A. M. Abdu, M. M. M. Mokji, and U. U. U. Sheikh, "Machine learning for plant disease detection: an investigative comparison between support vector machine and deep learning," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 4, pp. 670–683, Dec. 2020, doi: 10.11591/ijai.v9.i4.pp670-683.
[7] L. N and K. K. Saju, "Classification of macronutrient deficiencies in maize plant using machine learning," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, Art. no. 41974203, Dec. 2018, doi: 10.11591/ijece.v8i6.pp4197-4203.
[8] L. Deng, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014, doi: 10.1561/2000000039.
[9] N. H. Ali, M. E. Abdulmunem, and A. E. Ali, "Constructed model for micro-content recognition in lip reading based deep learning," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2557–2565, Oct. 2021, doi: 10.11591/eei.v10i5.2927.
[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
[11] R. A. Pratiwi, S. Nurmaini, D. P. Rini, M. N. Rachmatullah, and A. Darmawahyuni, "Deep ensemble learning for skin lesions classification with convolutional neural network," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 563–570, Sep. 2021, doi: 10.11591/ijai.v10.i3.pp563-570.
[12] M. Y. Kamil, "A deep learning framework to detect Covid-19 disease via chest X-ray and CT scan images," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 1, pp. 844–850, Feb. 2021, doi: 10.11591/ijece.v11i1.pp844-850.
[13] A. A. Abdulmunem, Z. A. Abutiheen, and H. J. Aleqabie, "Recognition of corona virus disease (COVID-19) using deep learning network," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 1, pp. 365–374, Feb. 2021, doi: 10.11591/ijece.v11i1.pp365-374.
[14] N. Wang, Y. Wang, and M. J. Er, "Review on deep learning techniques for marine object recognition: Architectures and algorithms," *Control Engineering Practice*, vol. 118, Art. no. 104458, Jan. 2022, doi: 10.1016/j.conengprac.2020.104458.
[15] S. B. Jadhav, "Convolutional neural networks for leaf image-based plant disease classification," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 4, pp. 328–341, Dec. 2019, doi: 10.11591/ijai.v8.i4.pp328-341.
[16] R. Poojary, R. Raina, and A. Kumar Mondal, "Effect of data-augmentation on fine-tuned CNN model performance," *IAES International Journal of Artificial Intelligence (IJ-AI)*, Mar. 01, 2021. https://doi.org/10.11591%2Fijai.v10.i1.pp84-92 (accessed Sep. 20, 2021).
[17] R. Binti Roslan, I. N. Mohd Razly, N. Sabri, and Z. Ibrahim, "Evaluation of psoriasis skin disease classification using convolutional neural network," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 2, pp. 349–355, Jun. 2020, doi: 10.11591/ijai.v9.i2.pp349-355.
[18] I. A. Md Zin, Z. Ibrahim, D. Isa, S. Aliman, N. Sabri, and N. N. A. Mangshor, "Herbal plant recognition using deep convolutional neural network," *Bulletin of Electrical Engineering and Informatics*, Oct. 01, 2020. https://doi.org/10.11591%2Feei.v9i5.2250 (accessed Sep. 20, 2021).
[19] K. Singh, S. Kumar, and P. Kaur, "Automatic detection of rust disease of Lentil by machine learning system using microscopic images," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 660–666, Feb. 2019, doi: 10.11591/ijece.v9i1.pp660-666.
[20] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection," *Sustainable Cities and Society*, vol. 65, Art. no. 102600, Feb. 2021, doi: 10.1016/j.scs.2020.102600.
[21] H. Panwar, P. K. Gupta, M. K. Siddiqui, R. Morales-Menendez, and V. Singh, "Application of deep learning for fast detection of COVID-19 in X-Rays using nCOVnet," *Chaos, Solitons & Fractals*, vol. 138, Art. no. 109944, Sep. 2020, doi: 10.1016/j.chaos.2020.109944.
[22] S. Jitanan and P. Chimlek, "Quality grading of soybean seeds using image analysis," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 3495–3503, Oct. 2019, doi: 10.11591/ijece.v9i5.pp3495-3503.
[23] S. A. Dhole and R. P. Shaikh, "Review of leaf unhealthy region detection using image processing techniques," *Bulletin of Electrical Engineering and Informatics*, Dec. 01, 2016. https://doi.org/10.11591%2Feei.v5i4.498 (accessed Sep. 20, 2021).
[24] H. Asil and J. Bagherzadeh, "A new approach to image classification based on a deep multiclass AdaBoosting ensemble," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 5, pp. 4872–4880, Oct. 2020, doi: 10.11591/ijece.v10i5.pp4872-4880.
[25] A. R. Luaibi, T. M. Salman, and A. H. Miry, "Detection of citrus leaf diseases using a deep learning technique," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 2, Art. no. 17191727, Apr. 2021, doi: 10.11591/ijece.v11i2.pp1719-1727.
[26] N. Sabri, N. Shafekah Kassim, S. Ibrahim, R. Roslan, N. N. A. Mangshor, and Z. Ibrahim, "Nutrient deficiency detection in Maize (Zea mays L.) leaves using image processing," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 2, pp. 304–309, Jun. 2020, doi: 10.11591/ijai.v9.i2.pp304-309.
[27] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8, doi: 10.1109/ICCV.2007.4409066.
[28] L. Santos, F. N. Santos, P. M. Oliveira, and P. Shinde, "Deep learning applications in agriculture: a short review," in *Advances in Intelligent Systems and Computing*, Springer International Publishing, 2020, pp. 139–151.
[29] N. Ali, A. H. AbuEl-Atta, and H. H. Zayed, "Enhancing the performance of cancer text classification model based on cancer hallmarks," *IAES International Journal of Artificial Intelligence (IJ-AI)*, Jun. 01, 2021. https://doi.org/10.11591%2Fijai.v10.i2.pp316-323 (accessed Sep. 20, 2021).
[30] M. A. Ihsan Aquil and W. H. Wan Ishak, "Evaluation of scratch and pre-trained convolutional neural networks for the classification of Tomato plant diseases," *IAES International Journal of Artificial Intelligence (IJ-AI)*, Jun. 01, 2021. https://doi.org/10.11591%2Fijai.v10.i2.pp467-475 (accessed Sep. 15, 2021).
[31] M. Morgan, C. Blank, and R. Seetan, "Plant disease prediction using classification algorithms," *IAES International Journal of Artificial Intelligence (IJ-AI)*, Mar. 01, 2021. https://doi.org/10.11591%2Fijai.v10.i1.pp257-264 (accessed Sep. 15, 2021).
[32] M.-E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 2 (CVPR'06)*, vol. 2, pp. 1447–1454, doi: 10.1109/CVPR.2006.42.
[33] M.-E. Nilsback, "An automatic visual flora-segmentation and classification of flower images," *University of Oxford*, 2009. https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.504504 (accessed Apr. 23, 2021).
[34] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian*

Conference on Computer Vision, Graphics and Image Processing, Dec. 2008, pp. 722–729, doi: 10.1109/ICVGIP.2008.47.
[35] M.-E. Nilsback and A. Zisserman, "Delving deeper into the whorl of flower segmentation," *Image and Vision Computing*, vol. 28, no. 6, pp. 1049–1062, Jun. 2010, doi: 10.1016/j.imavis.2009.10.001.
[36] R. Bavishi, M. Pradel, and K. Sen, "Context2Name: a deep learning-based approach to infer natural variable names from usage contexts," Aug. 31, 2018. http://arxiv.org/abs/1809.05193 (accessed Apr. 15, 2021).
[37] X. Sun, J. Xu, C. Jiang, J. Feng, S.-S. Chen, and F. He, "Extreme learning machine for multi-label classification," *Entropy*, vol. 18, no. 6, p. 225, Jun. 2016, doi: 10.3390/e18060225.
[38] Y. Sun, E. Lank, and M. Terry, "Label-and-Learn," in *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, Mar. 2017, pp. 523–534, doi: 10.1145/3025171.3025208.
[39] Y. Sun, Y. Liu, G. Wang, and H. Zhang, "Deep learning for plant identification in natural environment," *Computational Intelligence and Neuroscience*, vol. 2017, pp. 1–6, 2017, doi: 10.1155/2017/7361042.
[40] B. Pukhrambam and R. Rathna, "A smart study on medicinal plants identification and classification using image processing techniques," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, Feb. 2021, pp. 956–962, doi: 10.1109/ICICV50876.2021.9388566.
[41] S. Donesh and U. P. Ishanka, "Plant recognition system based on leaf images," *A Systematic Literature Review*, 2020, Accessed: Jun. 28, 2021. [Online]. Available: http://ir.kdu.ac.lk/handle/345/2945.
[42] S. Aggarwal, M. Bhatia, R. Madaan, and H. M. Pandey, "Optimized sequential model for plant recognition in Keras," *IOP Conference Series: Materials Science and Engineering*, vol. 1022, no. 1, Art. no. 012118, Jan. 2021, doi: 10.1088/1757-899X/1022/1/012118.
[43] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, Sep. 2016, doi: 10.3389/fpls.2016.01419.
[44] S. H. Lee, C. S. Chan, S. J. Mayo, and P. Remagnino, "How deep learning extracts and learns leaf features for plant classification," *Pattern Recognition*, vol. 71, pp. 1–13, Nov. 2017, doi: 10.1016/j.patcog.2017.05.015.
[45] I. K. Swardika, P. A. W. Santiary, I. B. I. Purnama, and I. W. Suasnawa, "Development of green zone energy mapping for community-based low carbon emissions," *International Journal on Advanced Science, Engineering and Information Technology*, Dec. 22, 2020. https://doi.org/10.18517%2Fijaseit.10.6.12642 (accessed Mar. 08, 2021).
[46] I. K. Swardika, P. A. W. Santiary, and I. N. E. Indrayana, "Radiance threshold of nighttime satellite data for green zone energy mapping," in *2019 International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*, Oct. 2019, pp. 1–6, doi: 10.1109/ICEEIE47180.2019.8981451.

BIOGRAPHIES OF AUTHORS

Putri Alit Widyastuti Santiary. The first author works as a lecture at the Department of Electrical Engineering, the State Polytechnic of Bali. She holds a master's degree in computer engineering from Udayana University and teaching computer programming language for more than twenty years at the State Polytechnic of Bali. She wrote many teaching materials about programming in C++, Java, and Python. She had to author more than eleven journal papers. Her current research interest is on botanical classification with deep learning. She can be contacted by email at: putrialit@pnb.ac.id.

I Ketut Swardika. The second author is an associate professor at the Department of Electrical Engineering, the State Polytechnic of Bali. He received his Ph.D. degree from Yamaguchi University in 2012. He had experience in Japanese's satellite remote sensing program. He wrote many teaching materials from the topic in RS, electrical to computer programming. He had the authoring of more than twenty journal papers. His current research interests include nighttime RS observation for the sustainability of energy and the environment. He can be contacted by email at: swardika@pnb.ac.id

Ida Bagus Irawan Purnama. The third author is an associate professor at the Department of Electrical Engineering, the State Polytechnic of Bali. He received his Ph.D. degree from Queensland University in computer science. His concentration research is in machine learning on an embedded system. He is familiar with single board computers such as raspberry pi, google coral dev. board and Arduino MCU. He had tutoring many practical materials for digital automation study program students. He wrote many teaching materials and authoring more than twenty journal papers. He can be contacted by email at: ida.purnama@pnb.ac.id.

I Wayan Raka Ardana. The fourth author works as a lecture at the Department of Electrical Engineering, the State Polytechnic of Bali. He holds a master's degree in electrical engineering from Udayana University and teaching digital telecommunication system for more than twenty years at the State Polytechnic of Bali. He wrote many teaching materials and laboratory's job sheets. He had to author more than ten journal papers. His current research interest is on hybrid solar power. He can be contacted by email at: rakawyn@pnb.ac.id.

I Nyoman Kusuma Wardana. The fifth author is an assistant professor at the Department of Electrical Engineering, the State Polytechnic of Bali. He received the B.Eng. degree in engineering physics and the M.Eng. degree in engineering system from the University of Gadjah Mada, Indonesia, in 2008 and 2010, respectively. He received the M.Sc. degree in electronic engineering from Politecnico di Torino, Italy, in 2012. He is currently working toward a PhD degree at the School of Engineering, the University of Warwick, U.K. His research interests include embedded computing, edge computing, and the Internet of Things. He has been working on distributed machine learning at the edge for air quality monitoring. He can be contacted by email at: kusumawardana@pnb.ac.id.

Dewa Ayu Indah Cahya Dewi. The sixth author work as a lecture at Department of Electrical Engineering, the state Polytechnic of Bali. She received her master's degree in computer engineering from Udayana University in 2018. Her current research interest in data mining. She can be contacted by email at: ayuindahcahyadewi@pnb.ac.id.

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 189~200
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i1.pp189-200

Oil palm unstripped bunch detector using modified faster regional convolutional neural network

Wahyu Sapto Aji, Kamarul Hawari Bin Ghazali, Son Ali Akbar
Fakulty of Electrical and Electronics Engineering Technology, Universiti Malaysia Pahang, Pekan, Malaysia

Article Info

Article history:

Received Sep 27, 2021
Revised Dec 9, 2021
Accepted Dec 29, 2021

Keywords:

Faster regional convolutional neural network
Object detector
Unstripped bunch
Visual geometry group 16

ABSTRACT

The palm oil processing industry in Malaysia and Indonesia is significant and plays a vital role in the community's welfare. The efficiency of palm oil mills is characterized by the low number of unstripped bunch (USBs), so USB detection is essential in the palm oil production process. So far, USB detection is done manually and is often ignored because it is labor-intensive. We developed a USB detector based on faster regional convolutional neural network with a modified visual geometry group 16 (VGG16) backbone to solve this problem. To see the performance of our proposed USB detector, we compared it to the faster region based convolutional neural networks (R-CNN) USB detector with the VGG16 standard backbone. Based on the validation test, the USB faster R-CNN detector with modified VGG16 can improve the performance of the USB faster R-CNN detection system based on the original VGG 16 backbone. The proposed system can work faster (100% faster) with an mAP value of 0.782 (7.42% more precise) than the USB Detector with the original VGG16. In the training process, the proposed system on the speed parameter has better training parameters, which is 58.9% faster, the total loss is smaller (43.4% smaller), and the proposed system has better best accuracy (98%) than the previous system (93%). Still, it has a smaller overlap bounding box (23.91% less).

Corresponding Author:

Wahyu Sapto Aji
Faculty of Electrical and Electronics Engineering Technology, University Malaysia of Pahang
Pekan Campus, 26600 Pekan, Malaysia
Email: wahyusa@ee.uad.ac.id

1. INTRODUCTION

One of the causes of losses in palm oil mills is oil palm unstripped bunch (USB). In Malaysia, losses caused by USB are estimated at an average of 0.05% [1]. In some cases where the bunches are not appropriately processed, losses due to USB can be up to 40% [2]. Based on this, it can be concluded that the primary system performance of palm oil mills can be seen based on the presence of USB. Unfortunately, no method can perform USB observations automatically. So far, USB monitoring is still done manually and is often ignored. An automatic USB counting system must be developed to solve this problem as the first step in USB monitoring; a faster region-based convolutional neural networks (R-CNN) object detection system with a modified visual geometry group 16 (VGG16) feature extractor is proposed. The performance of the proposed USB detection system will be compared with the faster R-CNN USB detector system with the original VGG16.

USB is loosely defined as an oil palm fruit bunch that still has oil palm fruit fruitlets. Figure 1 shows an example of a USB, while Figure 2 shows an example of an empty fruit bunch (EFB). EFB itself is in oil palm bunches without sticking fruit, as shown in Figure 2. Some researchers have given their definitions of USB specifically. According to Hassan *et al*., a USB is an empty bunch with more than 20 fruits still attached [3].

Figure 1. A USB with fruitlets (red box) attached to it

Figure 2. An EFB

2. RELATED WORK

Deep learning has gained much interest in artificial intelligence (AI) applications, especially object detection systems. That is inseparable from the ability of deep learning to recognize objects, where their abilities exceed human abilities [4]. Several researchers realized this deep learning ability and developed an object recognition application based on deep learning. In the health sector, Ghaderzadeh *et al*. take advantage of deep learning for the detection of the coronavirus disease (COVID-19), wherein their research, deep learning-based object recognition can reduce false positives and negatives in the detection and diagnosis of the COVID-19 virus and offers a unique opportunity to provide fast, inexpensive diagnostic services, and safe for the patient [5]. Deep learning applications for object detection in animal husbandry and agriculture also attracted the attention of several researchers. Barbedo *et al*. in 2019, succeeded in developing a deep learning-based cattle detection system using Unmanned aerial vehicle (UAV) [6]. In 2019, Aravind *et al.* developed deep learning-based eggplant disease detection [7]. In a recent effort, Liu *et al.* successfully developed palm tree detection based on faster R-CNN [8]. However, no researcher has yet developed a USB detection system that oil palm mills operators need.

3. RESEARCH METHOD

The research methodology carried out includes several steps. The first step is the creation of the USB dataset; the second step is fine-tuning and original VGG16 modification. After the above process, training and validation are carried out using both the original R-CNN Faster and the modified RCNN faster. The results of the training and validation processes are then compared.

3.1. USB dataset

USB detection research requires the availability of a USB database, and the USB database is not yet publicly available. Therefore, the first step in carrying out this research is to create a USB database. The USB database was compiled using image data obtained from surveillance cameras. This surveillance camera is placed above a USB conveyor.

The custom datasets used are the USB and EFB datasets. The images of the datasets are captured from video surveillance mounted on top of the exhaust conveyor of a palm oil mill. This video is taken from PT. Sawit Arum Madani, a palm oil mill located in Blitar, Indonesia in May 2019. The number of images available is 500. Four hundred sixty images (92%) were used as train data, and 40 (8%) images were used as test data. Figures 3 and 4 respectively show examples of USB and EFB labeling. For USB and EFB classes, the EFB class is 255, and the USB class is 245. LabelImg [9] is used for labeling images.

Figure 3. USB labeling example

Figure 4. EFB labeling example

3.2. Faster R-CNN

Faster R-CNN is one well-known two-stage object detector [10]. A faster R-CNN object detection network is composed of a feature extraction network which is typically a pre-trained convolutional neural networks (CNN), followed by a proposal network (RPN), which is, as its name suggests, used to generate object proposals, and the second is used to predict the actual class of the object. The final layer is the detection network or classification network, as shown in Figure 5, which depicts faster R-CNN's main component [11]. This study uses a modified VGG16 as a feature extractor.

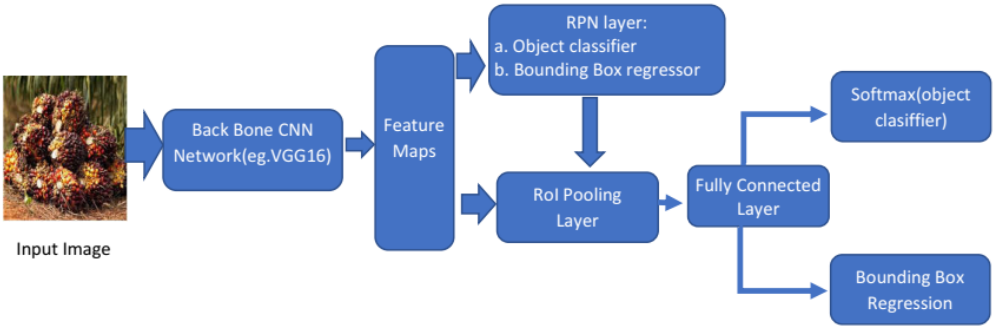


Figure 5. Faster R-CNN basic architecture

3.3. Original VGG16

VGG16 is a convolutional neural network model proposed by Simonyan and Zisserman [12]. The architecture can be seen in Table 1. It consists of five blocks, where there are piles of convolutional layers and a max-pooling layer. That is followed by three fully connected layers and a softmax layer. All convolutional layers are also equipped with ReLU, which stands for rectified linear unit. In the original configuration, VGG16 does not have a batch normalization layer. The original VGG16 model has 136,688,504 parameters, all trainable parameters.

Table 1. Original VGG16 architecture

Block	Layer	Kernel size	Number of kernels	Activation	Padding
1	Conv2D	3×3	64	Relu	Same
	Conv2D	3×3	64	Relu	Same
	MaxPooling2D				
2	Conv2D	3×3	128	Relu	Same
	Conv2D	3×3	128	Relu	Same
	MaxPooling2D				
3	Conv2D	3×3	256	Relu	Same
	Conv2D	3×3	256	Relu	Same
	Conv2D	3×3	256	Relu	Same
	MaxPooling2D			Relu	Same
4	Conv2D	3×3	512	Relu	Same
	Conv2D	3×3	512	Relu	Same
	Conv2D	3×3	512	Relu	Same
	MaxPooling2D				
5	Conv2D	3×3	512	Relu	Same
	Conv2D	3×3	512	Relu	Same
	Conv2D	3×3	512	Relu	Same

3.4. Fine-tuning and modified faster RCNN

To get a better USB detector, we propose improvements and modifications of Faster R-CNN. We propose resizing the image, adding an HPF filter, and adding a batch normalization layer. In detail, the above steps are:

a. Minimal resizing of the sides of the image. According to Huang *et al.* reducing the size of the input image by half will speed up the inference time, although it will decrease the accuracy [13]. To increase the inference speed, in this study, we propose the minimum size of the image size is 300 pixels from the previous 600 pixels. That is important because, in its application, this system will be applied to moving images or video files.

b. Image augmentation using Gaussian high pass filter (HPF). The accuracy of detection and classification in the object detection process usually depends on the feature expression of the detection object. The accuracy of detection and classification in the object detection process usually depends on the feature expression of the object [14]. The detection performance is also related to whether the features are diminished after a certain number of traditional convolutional operations [15]. Augmentation techniques can be used to increase the expression of the features of an object. The augmentation technique is commonly used to increase the performance of a object detector [16]. Some researchers have used this technique to increase the mAP value of the object detector [17]. In the proposed model, additional image augmentation is carried out in the form image sharpening process. Image sharpening refers to any enhancement technique that highlights an image's edges and fine details. Widely used image sharpening enhances local contrast and amplifies high-frequency pixels. This process highlights the unique features found in USB, namely the features that appear with high frequencies. As seen in Figures 1 and 2, EFB is dominated by pixels that form a line, while USB is dominated by pixels that form a circle. The image sharpening method used is the sharpening method with a Gaussian filter; this is because the high-frequency gaussian filter Gaussian high pass filter (GHPF) can sharpen images better than some other filters butterworth high pass filter (BHPF) [18]. The GHPF matrix used in this study is shown in Figure 6.

$$\begin{vmatrix} 0.11074074 & 0.11129583 & 0.11074074 \\ 0.11129583 & 0.1118537 & 0.11129583 \\ 0.11074074 & 0.11129583 & 0.11074074 \end{vmatrix}$$

Figure 6. Gaussian high pass filter (GHPF) kernel with size 3x3 and standard deviation 10

c. Added a batch normalization layer in VGG16. Batch normalization [19] is another regularization technique that normalizes the set of activations in layers. Normalization works by subtracting the batch average from each activation and dividing by the batch standard deviation [20]. This normalization is a standard technique in pixel value pre-processing. The addition of batch normalization in several studies able to speed up the training process by eliminating internal covariate shift problems [21], is also able to increase the mAP value by more than 2% [22]. Table 2 shows the modified VGG architecture used as the backbone of USB detection with Faster RCNN. In this modified VGG16 architecture, there are 136,705,400 parameters with 136,696,952 trainable parameters and 8448 non-trainable parameters.

Table 2. Modified VGG16 with batch normalization layers added

Block	Layer	Kernel size	Dropout	Activation	Batch normalization	Padding
1	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	MaxPooling2D					
2	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	MaxPooling2D					
3	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	MaxPooling2D			Relu		Same
4	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	MaxPooling2D					
5	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same

With Conv_block as a layer wrapper of: x = dense(), x= batch_normalization()
x =drop_out(), x=activation()

3.5. Training performance parameters

The researchers used several parameters to assess the training performance of the deep learning-based object detector model. This study uses parameters such as loss, accuracy, and speed. A brief explanation of the parameter we:

a. Region proposal network (RPN) layer loss: RPN layer loss is the sum of classification loss and bounding box regression loss. Classification loss uses cross-entropy loss to punish misclassified boxes. Regression loss uses the distance function between the true regression coefficients (calculated using the closest foreground anchor box to match the ground truth box) and the regression coefficients predicted by the network.
b. Detection network losses: detection network losses also consist of two parts, regression loss, and classification loss. The regression and classification losses are also computed similarly to the RPN, except the regression coefficients are class-specific. The network calculates the regression coefficient for each object class. The cross-entropy loss is directly used for the classification loss, and the regression loss, the smooth L1 loss, is used, but the loss is only calculated for the positive sample.
c. Total loss: he total loss is the sum of the losses in the RPN and detection network layers. As shown in (1) the equation for calculating the total loss.

$$Total_{loss} = RPN_{loss} + Detection\_Network_{loss} \quad (1)$$

d. Training speed: Speed is an indicator of object detection system [23]. The speed indicator is important, especially for hardware systems with limited resources. Unfortunately, this indicator is often opposite to other indicators such as accuracy and accuracy, so it is often necessary to balance the choice of performance indicators.

3.6. Testing (validation) performance parameters

The parameters used to assess validation performance are slightly different from those used to assess training performance. Here we use the mAP (mean precision average) parameter. This mAP parameter aims to assess the precision of the model for all classes. The following is an explanation of the parameters we use.

a. Mean average precision (mAP): mean average precision (mAP) is a popular metric in measuring the accuracy of object detectors such as Faster R-CNN and SSD [24]. Mean Average Precision is used to measure the accuracy of object detectors (object position and object class) in all classes in a specific database [25]. As shown in (2) is the basic equation for calculating mAP [26].

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \quad (2)$$

where:
AP = Average Precision Class i
K = The number of classes evaluated
In (2), it can be seen that mAP is the average value of AP. Average precision (AP) is the most commonly used metric, derived from precision and recall, for evaluating object detection performance [27]. AP is evaluated on a specific object category. That is means that it is computed for each separate object category.

b. Speed: inference speed is one of the performance parameters of the detection system. Of course, the ideal system is a system that is instantaneous but has high precision. To measure speed, the standard unit used is fps (frames per second) [28], [29]. Also, this is called the frame rate or frame frequency. If the speed is higher, it has better performance for handling more images. As shown in (3) and (4) are the equations used to calculate the detection time/image.

$$T = \left(\frac{test_{time}}{image}\right) = (total \frac{time}{amount\ of\ data\ in\ the\ test\ dataset} \quad (3)$$

$$FPS = \frac{1}{T} \quad (4)$$

4. RESEARCH IMPLEMENTATION

The research was conducted in three steps. The first step is training the faster R-CNN USB detector system with the original VGG16 feature extractor. The second step is to train the USB detector system with a modified VGG16 and Gaussian filter addition during fine-tuning. The third step is testing and comparison between the two USB detector models. All these steps use the same dataset

The software used to compile is Anaconda3-2019 with Python 3.6.8, supported by the Python library for machine learning (Numpy, Panda, and Scikit), Keras version 2.2.4, and the artificial intelligence framework TensorFlow with version 1.14.0. The hardware uses an Intel core i-7 desktop, 8 GB RAM, and Nvidia RTX2080 graphic card. Figures 7 and 8 show the flow of the training and validation processes of the proposed USB detector system.

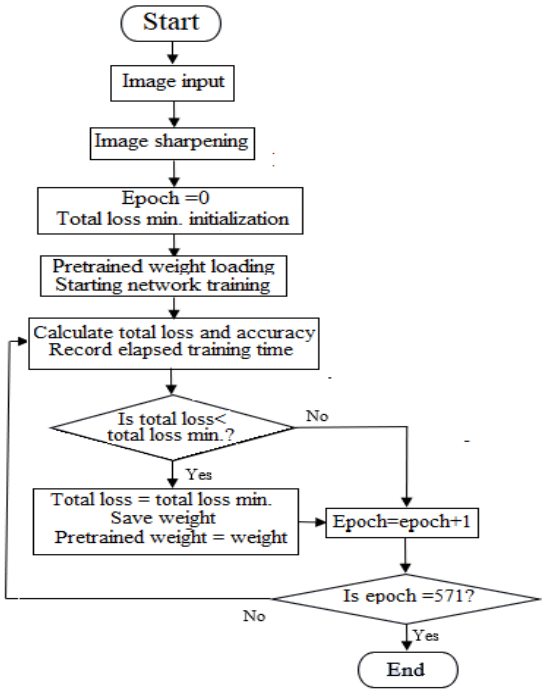


Figure 7. Workflow for USB detector training

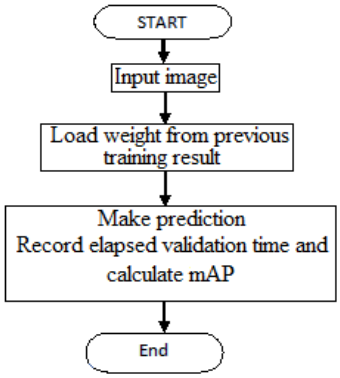


Figure 8. Workflow for USB detector validation

5. RESULTS AND DISCUSSION

5.1. The performance of USB detector using faster R-CNN based on original VGG16

UBS detection performance based on the original VGG16 is divided into training and validation performance. Figures 9 to 18 show a performance graph of the USB detection system using the original VGG16. The training performance is shown in Figures 9 to 16, while the validation performance is shown in Figures 17 and 18.

5.1.1. Original VGG16 based faster R-CNN USB detector training performance

The training performance parameters for the RPN network layer are shown in Figures 9 to 16. Based on Figures 9 to 16, it can be seen that the losses during the training process have successfully converged. Figure 9 shows the initial RPN classifier loss is 2; this loss consistently decreases and reaches a value of 0.01 after 500 epochs and is stable. The same results are also shown for RPN regression loss; the initial value of RPN regression loss is 0.11, then consistently decreases to 0.02 after 500 epochs as shown in Figure 10.

The losses of the detection network layer also show the same pattern, where the initial regression loss of the detection network layer is 0.35; this loss consistently continues to decrease and is stable at 0.05

after 500 epochs as shown in Figure 11. The classification loss also converges; as shown in Figure 12, the initial classification loss is 0.8. This value continued to decrease and stabilize at a value of 0.14 after 500 epochs. It causes the total loss during training to be also convergent. The initial total loss value is 3.3 and converges to a total loss of 0.203 as shown in Figure 13.

The time required for 570 epochs (460 images per epoch) was 2349.1 minutes or 39.15 hours as shown in Figure 14; this means the average time required for one epoch is 4.12 minutes or 0.54 seconds per image (1.85 fps). Figure 15 shows the accuracy performance during training, and it can be seen in Figure 15 that the class detection accuracy is 93%. The average bounding box with a score of 24 was obtained during training, as shown in Figure 16.

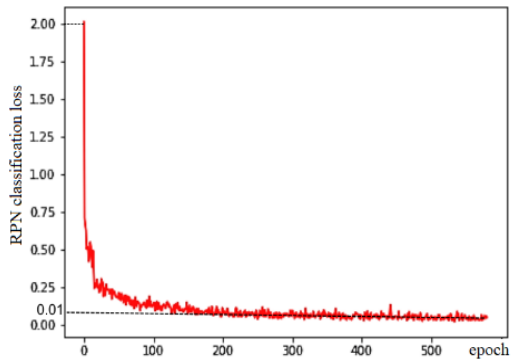


Figure 9. RPN classification loss of original Faster R-CNN during training

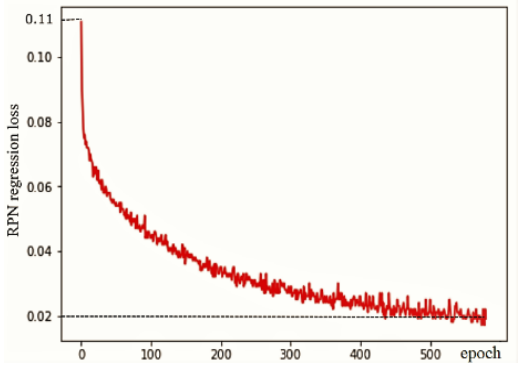


Figure 10. RPN regression loss of original Faster R-CNN during training

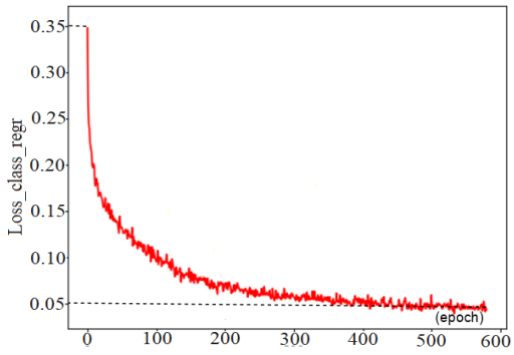


Figure 11. Regression loss of the original faster R-CNN detection network layer

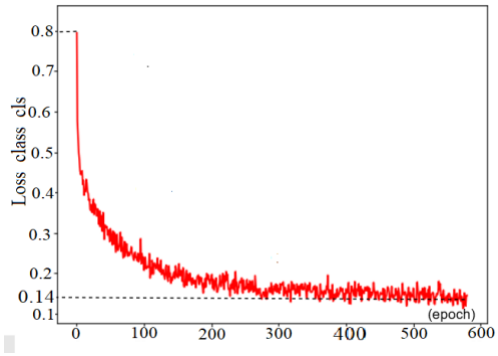


Figure 12. Classification loss of the original faster R-CNN detection network layer

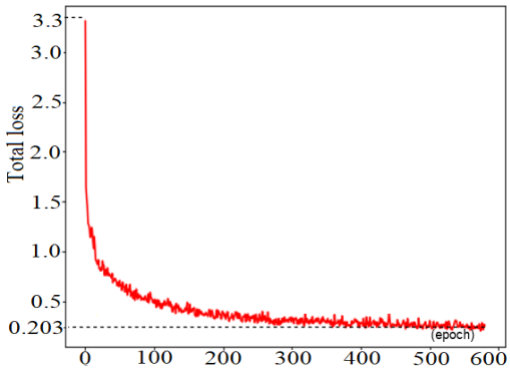


Figure 13. Total loss loss of original faster R-CNN

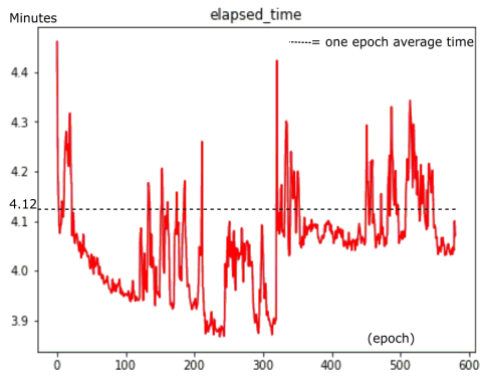


Figure 14. Time elapsed/epoch of original faster R-CNN

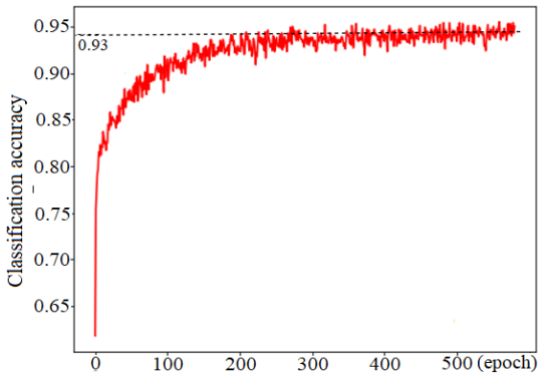


Figure 15. Classification accuracy of original faster R-CNN

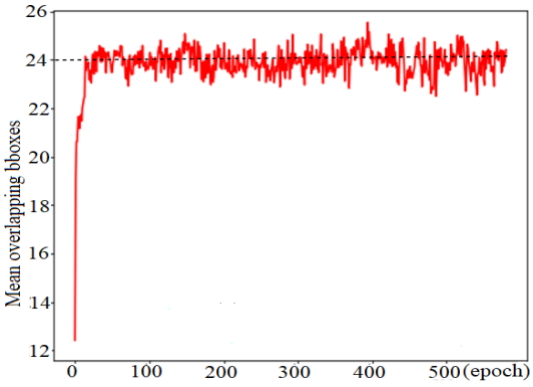


Figure 16. Mean overlapping BBox of original faster R-CNN

5.1.2. Original VGG16 based Faster R-CNN USB detector validation performance

Validation was carried out using test data which contained 40 images. In this test, the mAP average data obtained was 0.728 as shown Figure 17. The time needed to detect the testing dataset is 21.61 seconds as shown in Figure 18, which means that the average detection time for one image is 0.54 seconds (1.85 fps)

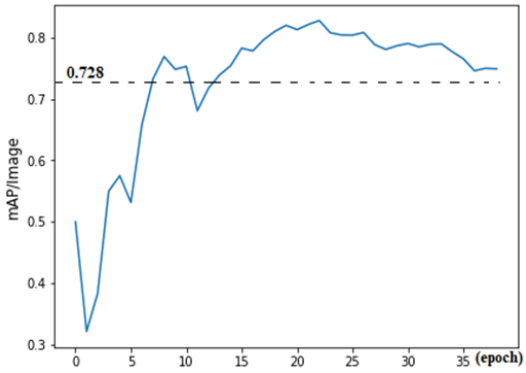


Figure 17. mAP/Image during testing

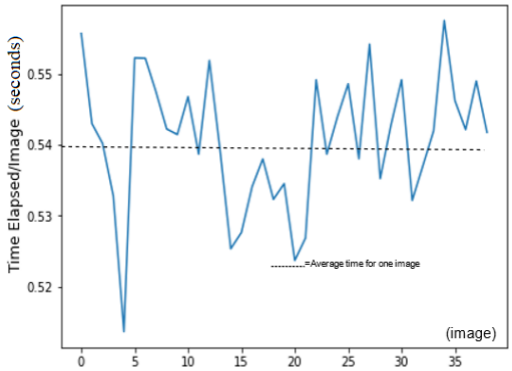


Figure 18. Time elapsed/image during testing

5.2. USB detector performance using modified VGG16 based faster R-CNN

As with the performance of the original VGG16-based UBS detector, the modified VGG16-based USB detector is also divided into training and validation performance. Figures 19 through 28 show a performance graph of the USB detection system using modified VGG16, image filtering, and image resizing. The training performance is shown in Figures 19 to 26, while the validation performance is shown in Figures 27 and 28.

5.2.1. Modified VGG16 based faster R-CNN USB detector training performance

The USB detector system training was carried out using the same dataset as the previous training. The tools and datasets used are identical to the tools and datasets used in the previous training. The hyperparameters used are also the same except for the input image size. Figures 19 to 26 show the training performance (regression and classification loss, respectively) of the USB detector based on the modified VGG16 Faster R-CNN.

RPN network layer training performances are depicted in Figures 19 to 20. Figure 19 shows that the initial value of the RPN classification loss is 1.695. This loss decreases consistently, reaches a value of 0.014 after 500 epochs, and is stable. The same results are also shown for RPN regression losses. The initial value of RPN losses is 0.092, then consistently decreases to 0.005 after 500 epochs as shown in Figure 20.

The losses of the classifier layer also show the same pattern, where the initial value of the regression loss of the classifier layer is 0.424. This loss consistently decreases and is stable at 0.03 after 500 epochs as

shown in Figure 21. The classification loss during training is convergent as shown in Figure 22, the initial classification loss of the detection network layer is 0.883, and this value continues to decrease and stabilize at a value of 0.104 after 500 epochs. Its causes the overall loss during training for (RPN layer and detection network layer losses) is also convergent. The initial total loss is 3.1 and converges to 0.103 as shown in Figure 23. The total time required for 570 epochs is 1509.52 minutes or 25.15 hours as shown in Figure 24; this means that the average time needed for one epoch is 2.64 minutes or 0.34 seconds per image (2.94 fps). It can be seen in Figure 25 that the best class detection accuracy is 98%, for the average overlapping bounding box is 17.8, as depicted seen in Figure 26.

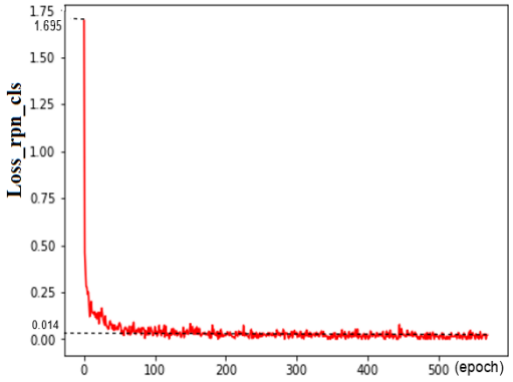


Figure 19. RPN layer classification loss

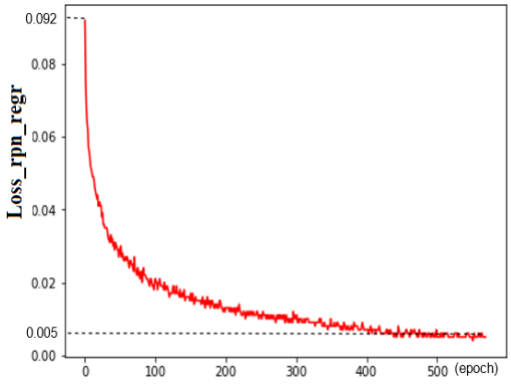


Figure 20. RPN layer regression loss

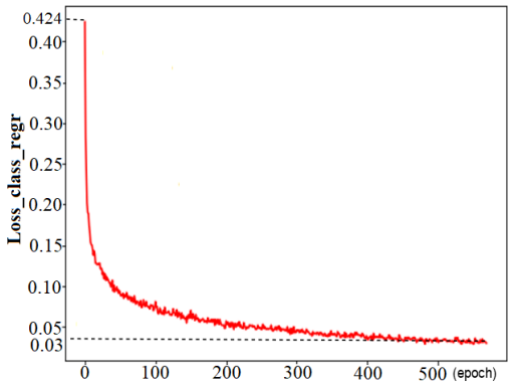


Figure 21. Regression losses of the detection network layer

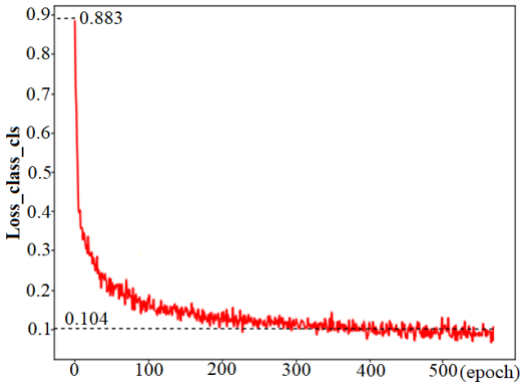


Figure 22. Classification losses of the detection network layer

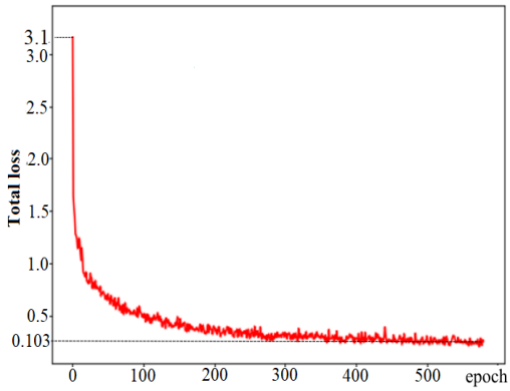


Figure 23. Total loss loss of modified faster RCNN

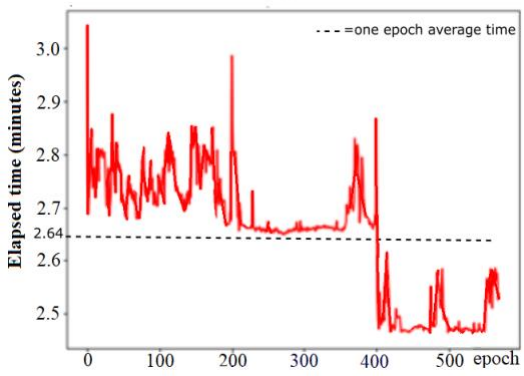


Figure 24. Time elapsed/epoch of the modified faster R-CNN

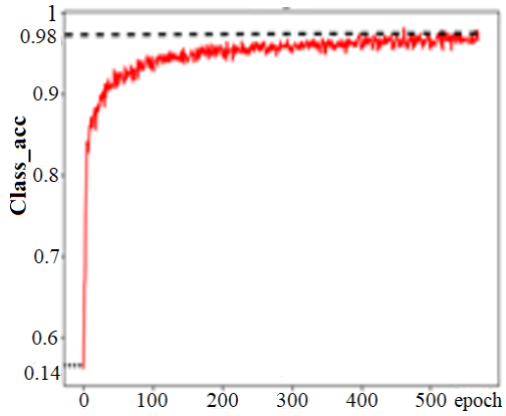


Figure 25. Classification accuracy of the modified faster R-CNN

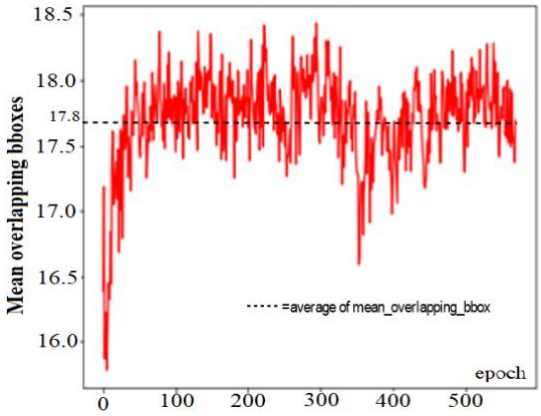


Figure 26. Mean overlapping bboxes

5.2.2. Modified VGG16 based Faster R-CNN USB detector validation performance

Tests to obtain training performance are carried out using the same dataset for the previous validation test. Figures 27 and 28 show the performance of the USB detector system validation test after being modified. In this test, the average mAP data obtained is 0.782 Figure 27 with an overall detection time of 10.92 seconds Figure 28, which means the average detection time for one image is 0.27 seconds (3.7 fps). The values above are improvements from the values obtained previously.

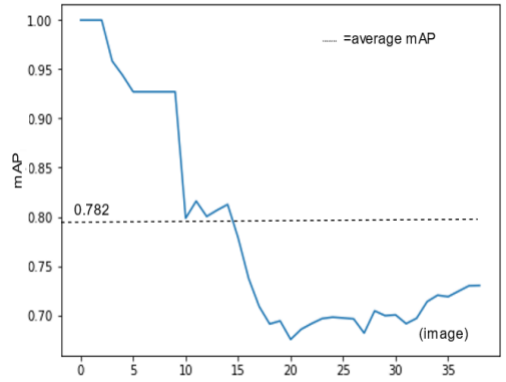


Figure 27. mAP/Image of the modified faster R-CNN

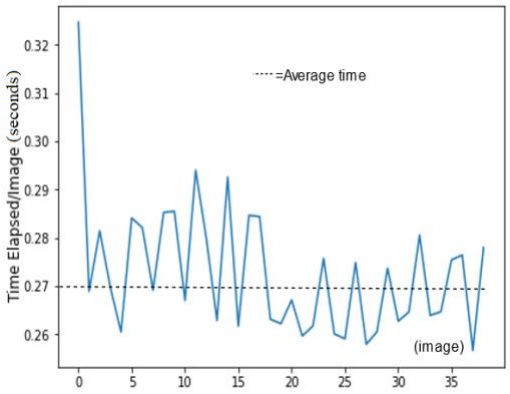


Figure 28. Time elapsed/image of the modified faster R-CNN

5.3. Performance comparison

The training performance and validation performance of the original and modified VGG16-based Faster R-CNN USB detector are shown in Tables 2 and 3. Based on the training performance table as shown Table 2, it can be seen that the proposed system has improved performance, namely 58.9% faster, 43.4% smaller total losses with 5.4% better class prediction accuracy. However, the bbox (bounding box) overlap is 23.91% less.

Table 2. Comparison of training performance between USB detection systems using the original faster R-CNN and with the proposed system

Performance parameters	USB detector based original system	USB detector based proposed system	Remark
Speed (FPS)	1.85	2.94	58.9% faster
Total loss	0.203	0.103	43.4% smaller
Class accuracy	93%	98%	5.4% more accurate
BBox Overlap	23	17,5	23.91% less

While the test performance as shown Table 3, the proposed system also shows a significant improvement where the proposed system has a speed of 3.7 fps which is faster than the previous system, which has a speed of 1.85 fps. The difference of 1.85 fps means that the proposed system is 100% faster. The proposed system has better performance for accuracy, with an mAP value of 0.782; this value is higher than the previous system, which has an mAP value of 0.728. This difference of 0.054 means an increase in accuracy performance of 7.42%.

Table 3. Comparison of validation performance between USB detection systems using the original faster R-CNN and with the proposed system

Performance parameters	USB detection system using original Faster R-CNN VGG16	USB detection system using Faster R-CNN based on modified VGG16	Comparison of the proposed system with the original system
Speed (fps)	1.85	3.7	100% faster
mAP	0.728	0.782	7.42% more precise

6. CONCLUSION

From the results, it can be concluded that the augmentation method, minimal image resizing, and adding a batch normalization layer can improve the performance of the faster R-CNN-based USB detection system using the modified VGG16 as a feature extractor. The proposed system on the speed parameter gives better results in the training process, which is 58.9% faster; the total loss is lesser (43.4% less) and has better best accuracy (98%) but had a lower average bounding box overlap (BBox) 23.91% less. While the testing (validation) performance, the proposed system can work faster (100% faster) and with better precision (7.42% more precise).

ACKNOWLEDGEMENTS

The author would like to thank Nuryono Satya Widodo. ST, M.Eng, Head of the Department of Electrical Engineering, Ahmad Dahlan University, Yogyakarta, facilitated this research. We also thank the management of PT. Sawit Arum Madani allowed us to retrieve data.

REFERENCES

[1] U. K. H. M. Nadzim, R. Yunus, R. Omar, and B. Y. Lim, "Factors contributing to oil losses in crude palm oil production process in malaysia: a review," *International Journal of Biomass and Renewables*, vol. 9, no. 1, 2020.
[2] O. Walat and N. S. Bock, "Palm oil mill OER and total oil losses," *POEB*, no. 108, pp. 11–16, 2013.
[3] A. Hassan, N. H. Muhammad, Z. A. Rahman, R. M. Halim, H. Alias, and M. Sabtu, "Improving mill oil extraction rate under the malaysian national key economic area," *POEB*, no. 103, pp. 33–47, 2012.
[4] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1026–1034, doi: 10.1109/ICCV.2015.123.
[5] M. Ghaderzadeh and F. Asadi, "Deep learning in the detection and diagnosis of COVID-19 using radiology modalities: a systematic review," *Journal of Healthcare Engineering*, vol. 2021, pp. 1–10, Mar. 2021, doi: 10.1155/2021/6677314.
[6] J. G. A. Barbedo, L. V. Koenigkan, T. T. Santos, and P. M. Santos, "A study on the detection of cattle in uav images using deep learning," *Sensors*, vol. 19, no. 24, p. 5436, Dec. 2019, doi: 10.3390/s19245436.
[7] K. R. Aravind, P. Raja, R. Ashiwin, and K. V Mukesh, "Disease classification in Solanum melongena using deep learning," *Spanish Journal of Agricultural Research*, vol. 17, no. 3, Nov. 2019, doi: 10.5424/sjar/2019173-14762.
[8] X. Liu, K. H. Ghazali, F. Han, and I. I. Mohamed, "Automatic detection of oil palm tree from UAV images based on the deep learning method," *Applied Artificial Intelligence*, vol. 35, no. 1, pp. 13–24, Jan. 2021, doi: 10.1080/08839514.2020.1831226.
[9] M. License, "Tzutalin LabelImg." https://github.com/tzutalin/labelImg (accessed Dec. 20, 2020).
[10] L. Du, R. Zhang, and X. Wang, "Overview of two-stage object detection algorithms," *Journal of Physics: Conference Series*, vol. 1544, no. 1, May 2020, doi: 10.1088/1742-6596/1544/1/012033.
[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.
[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, Apr. 2014.
[13] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3296–3297, Apr. 2017.
[14] Y. Xiao, X. Wang, P. Zhang, F. Meng, and F. Shao, "Object detection based on faster R-CNN algorithm with skip pooling and fusion of contextual information," *Sensors*, vol. 20, no. 19, p. 5490, Sep. 2020, doi: 10.3390/s20195490.
[15] H. K. Leung, X.-Z. Chen, C.-W. Yu, H.-Y. Liang, J.-Y. Wu, and Y.-L. Chen, "A deep-learning-based vehicle detection approach for insufficient and nighttime illumination conditions," *Applied Sciences*, vol. 9, no. 22, p. 4769, Nov. 2019, doi: 10.3390/app9224769.
[16] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V Le, "Learning Data Augmentation Strategies for Object Detection," 2020, pp. 566–583.
[17] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learnin," *Journal of Big Data*, vol. 6, no. 1,

p. 60, Dec. 2019, doi: 10.1186/s40537-019-0197-0.
[18] A. Dogra and P. Bhalla, "Image sharpening by gaussian and butterworth high pass filter," *Biomedical and Pharmacology Journal*, vol. 7, no. 2, pp. 707–713, Dec. 2014, doi: 10.13005/bpj/545.
[19] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, 2015, vol. 37, pp. 448–456.
[20] S. Ioffe and C. Szegedy, "renormalization: towards reducing minibatch dependence in batch-normalized models," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1942–1950.
[21] Y. Zhou, C. Yuan, F. Zeng, J. Qian, and C. Wu, "An object detection algorithm for deep learning based on batch normalization," 2018, pp. 438–448.
[22] L. Jiao *et al.*, "A Survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.
[23] W. Li, "Analysis of object detection performance based on faster R-CNN," *Journal of Physics: Conference Series*, vol. 1827, no. 1, Mar. 2021, doi: 10.1088/1742-6596/1827/1/012085.
[24] H. Mao, X. Yang, and W. J. Dally, "A delay metric for video object detection: what average precision fails to tell," *arXiv:1908.06368*, Nov. 2019.
[25] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, p. 279, Jan. 2021, doi: 10.3390/electronics10030279.
[26] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Jul. 2020, pp. 237–242, doi: 10.1109/IWSSIP48289.2020.9145130.
[27] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "One metric to measure them all: localisation recall precision (LRP) for evaluating visual detection tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1, 2021, doi: 10.1109/TPAMI.2021.3130188.
[28] L. Liu *et al.*, "Deep Learning for generic object detection: a survey," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, Feb. 2020, doi: 10.1007/s11263-019-01247-4.
[29] L. Lang, K. Xu, Q. Zhang, and D. Wang, "Fast and accurate object detection in remote sensing images based on lightweight deep neural network," *Sensors*, vol. 21, no. 16, p. 5460, Aug. 2021, doi: 10.3390/s21165460.

BIOGRAPHIES OF AUTHORS

Wahyu Sapto Aji is a Ph.D. student in the Faculty of Electrical and Electronic Engineering Technology, University of Malaysia Pahang, Malaysia. Wahyu S Aji obtained a bachelor's and master's degree in Electrical Engineering from the Electrical Engineering Department of Gadjah Mada University, Yogyakarta, Indonesia. He has been working as a lecturer in the electrical engineering department at Ahmad Dahlan University since 2014. His current research interests include Image Processing, Deep Learning. He can be contacted at email: wahyusa@ee.uad.ac.id.

Kamarul Hawari bin Ghazali is a Professor in the Faculty of Electrical and Electronic Engineering Technology, Universiti Malaysia Pahang. His major research areas include Machine Vision System, Image Processing, Signal Processing, Intelligent system, Vision Control, Thermal Imaging Analysis. (in all related applications – Electrical, Medical, Environment), Deep Learning for Image and Signal Classification. Major sponsor research includes Fundamental Research Grant Scheme (FRGS), International Grant, Long Term Research Grant Schemes (LRGS) and short-term grant from UMP. Currently a Professional Engineer (Ir), Board of Engineer Malaysia (BEM). He can be contacted at email: kamarul@ump.edu.my

Son Ali Akbar is a Ph.D. student in the Faculty of Electrical and Electronics Engineering Technology, University Malaysia of Pahang, Malaysia. He has been working as a lecturer in the electrical engineering department at Ahmad Dahlan University since 2014. His current research interests include Image Processing, Deep Learning. He can be contacted at email: son.akbar@te.uad.ac.id.

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 11, No. 2, June 2022, pp. 201~209
ISSN: 2252-8938, DOI: 10.11591/ijai.v11.i1.pp201-209

An optimization clustering and classification based on artificial intelligence approach for internet of things in agriculture

Sakchai Tangwannawit[1], Panana Tangwannawit[2]
[1]Department of Information Technology Management, Faculty of Information Technology and Digital Innovation, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand
[2]Department of Mathematics and Computing Science, Faculty of Science and Technology, Phetchabun Rajabhat University, Phetchabun, Thailand

Article Info

Article history:

Received Aug 1, 2021
Revised Dec 13, 2021
Accepted Dec 29, 2021

Keywords:

Agriculture
Artificial intelligence
Classification
Internet of things
Optimize clustering

ABSTRACT

This research focused on testing with maize, economical crop grown in Phetchabun province, Thailand, by installing a total of 20 sets of internet of things (IoT) devices which consist of soil moisture sensors and temperature and humidity sensors (DHT11). Data science tools such as rapidminer studio was used for data cleansing, data imputation, clustering, and prediction. Next, these data would undergo data cleansing in order to group them to obtain optimization clustering to identify the optimum condition and amount of water required to grow the maize through k-mean technique. From the analysis, the optimization result showed 3 classes and these data were further analyzed through prediction to identify precision. By comparing several algorithms including artificial neural network (ANN), decision tree, naïve bayes, and deep learning, it was found that deep learning algorithm can provide the most accurate result at 99.6% with root mean square error (RMSE)=0.0039. The algorithm obtained was used to write function to control the automated watering system to make sure that the temperature and humidity for growing maize is at appropriate condition. By using the improved watering system, it improved the efficacy of watering system which saves more water by 13.89%.

Corresponding Author:

Panana Tangwannawit
Faculty of Sciences and Technology, Phetchabun Rajabhat University
Phetchabun, Thailand
Email: panana.t@gmail.com

1. INTRODUCTION

Artificial intelligence (AI) is a technology to enable machines, computers, and statistical tools and equipment to create software that can imitate human capabilities especially on the very complex tasks e.g., memories, classification, reasoning, decision, prediction, and even communication with human beings, all through algorithms. In some cases, AI can be improved through self-learning which consists of 3 levels: machine learning [1], machine intelligence [2], and machine consciousness [3]. Machine learning is one of AI capabilities which the machine can learn on its own.

This research used a combination of internet of things (IoT), big data, and AI technology and integrated into agricultural system as a more effective alternative to help solve existing problems the farmers are facing. This could improve the speed, increase crop yields, and enable more effective use of natural resources for the users, or farmers which are the main objectives for this study. This research integrateds AI, IoT, and big data with 3 main actions: i) to analyze the classification of data to find optimum amount of

water required, ii) to compare the algorithm of watering system, and iii) to improve the automated watering system based on the algorithm obtained.

2. THEORETICAL BACKGROUND AND RELATED RESEARCH

2.1. Artificial intelligence (AI)

AI technology works in between automated and intelligence system, it is crucial that the data obtained must be analyzed with the appropriate data classification, data types, and the relationship of the information from big data to use this analysis to decide instead of using human judgement to decide. This type of classification is suitable to create predictive modeling called supervised learning [4]. This type of classification of the data is to create model in grouping the data into assigned groups by create sample groups with data in advance and predict the data groups that have not been classified. The sample groups might be in a form of artificial neural network (ANN) [5], decision tree [6], naïve bayesian, and deep learning. This research compares the algorithms for the best efficiency.

2.1.1. Artificial neural network (ANN)

ANN is a type of self-learning similar to human brain system. Process involves bringing neural networks into the prediction which requires keying in the data to build simulation in data prediction in the future [7]. The artificial neural network will try to predict based on the built simulation to make sure there's as little error as possible, with the (1):

$$n = \sum_{i=1}^{z} = x_i w_i + b \quad (1)$$

n=sum of function, x_i=input i, w_i=weight i, z=number of input layer, b=bias, i=1 to z.

2.1.2. Decision tree (DT)

Decision tree is a type of self-learning through mathematics to identify the best choice by creating data prediction in a form of tree structure which is learned through supervised learning. This enables clustering through training data set automatically and enable the group prediction of the data that has not been grouped before. In [8] the rank widget scores the attributes according to their correlation with the class. Attribute scoring methods that can be used in rank widget are information gain, information gain ratio and gini [9], [10].

2.1.3. Naïve bayesian (NB)

Naïve Bayesian (NB) is a type of learning through probability based on bayes theorem with a more straight forward, non-complex algorithm. It is a type of process in classifying information by learning from the problems occurred and utilize these conditions for re-classifying the data. It is a type of data classification that uses probability and computation to classify based on the hypothesis created with the data. The new calculated models will be used to adjust and re-classified which could either increase or reduce the probability of the information [11]. New information generated and set samples are then adjusted in combination with existing data, with the (2):

$$P(A|B) = \frac{P(B|A)\times \mathrm{P}(A)}{P(B)} \quad (2)$$

where A and B are the events and $P(B) \neq 0$
$P(A|B)$ the likelihood of event A occurring given that B is true.
$P(B|A)$the likelihood of event B occurring give that A is true.
$P(A)$ and $P(B)$ are probabilities of observing A and B independently of each other.

2.1.4. Deep learning

Deep learning is a type of automated learning that imitates the function of human neural networks (neurons) by overlapping these neural networks into several layers and create learning of the sample data. These data mentioned will be used to detect patterns or arrange into category or classify the data. This use a multi layer feed forward neural network which uses back propagation for building the deep learning environment [12].

2.1.5. Algorithm performance

After that, testing data is conducted to evaluate the accuracy of the model (model evaluation) to be used to analyze the effectiveness of the algorithm operation obtained. In this research, the algorithm

performance metrics used for experiment is accuracy and the evaluation of the algorithm operation is based on root mean square error (RMSE). Accuracy, with the (3):

$$Accuracy = \left(\frac{TP+TN}{TP+TN+FN+FP}\right) \times 100 \quad (3)$$

RMSE, with the (4):

$$RMSE = \sqrt{\frac{(a_1-y_1)^2+\ldots+(a_n-y_n)^2}{n}} \quad (4)$$

2.2. Internet of thing (IoT)

IoT is a paradigm which connects sensors to the internet. This enables us to order and control the operation of different equipment [13] or circuit which detect the changes and showing the results as signals. [14] sensor samples such as humidity sensors, temperature sensors, immersion sensors, vibration sensors, current leakage sensors, and intelligent video sensors through smart phones or computer. This process then creates a smart system such as smart agriculture, smart device [15], smart grid [16], smart home, smart city [17], or smart transportation.

2.3. Maize

Maize has tall stems which can range from 60 cm up to 6meters tall depending on the species. The diameter of the stem ranges from 1.27-5.08 cm. It takes approximately 100-120 days to be fully grown [18]. It is considered one of the economic crops in Thailand. In 2019-2020, it was found that the land for growing these crops throughout the country were declining. This is due to the fact that the land for growing these crops were left empty and farmers were unable to grow these crops due to drought from delays of rain and insufficient amount of natural water reserve. While the growing or crop is declining, there are still increasing demands of the maize in the country especially in farming business where maize is the main ingredient to make animal food for up to 7.41 million tons per year, while the annual supply is only 4.62 million tons [19].

2.4. Literature review

From Table 1 the review of all papers found that the fundamental of IoT has connecting physical objects, sensors, and other smart technologies altogether into one system. IoT is being utilized to view sensor values via the internet. Compilation of data obtained from IoT is used for analysis, clustering, classification, or prediction. Pulling these important data, big data, from several sensors from edge computing and data pre-processing allow information from data collection to be classified and completed. These data can either be with structure or without structure.

Table 1. Literature review methodology

Methodology	Subject	Reference
– Cycle of smart farming.	IoT	[13]–[15]
– Sensing and monitoring: robotics and sensors (temperature, humidity, and CO2), greenhouse computers.		[17] [20]–[22]
– Analysis and planning: seeding, planting, soil typing, crop, health, yield modelling, lighting, energy, and management.		
– Control: precision farming, climate control.		
– IoT definitions, technologies, and applications.		
– IoT solution, devices, platform.		
– Data collection: capture, storage, and transfer of data	Big data	[23], [24]
– Data pre-processing: transformation, marketing, and analysis of Big Data, Big Data greening, caching, remote sensing, real-time, analysis.		
– Choosing patten for AI; machine learning, machine intelligence, and machine consciousness	AI	[2], [3] [25]–[28]

The data classification methodology, it will be divided into 2 data sets. First, training set is a set of data used to train AI. This type of data is used to find variable which help the model function properly. While test set can measure the efficiency before bringing them to build AI. After that, choosing pattern for AI is conducted to select the appropriate algorithm.

3. RESEARCH METHOD

From Figure 1 showed the steps to find algorithm is as follows:

3.1. IoT

Based on previous research, devices for the router node were installed which consists of Arduino Uno R3, soil moisture sensor, temperature and humidity sensor (DHT11), NODEMCU ESP8266 module, and battery pack as one set of installation. The entire research required 20 sets of the mentioned system to cover 1,200 square meters of land for the experiment with one set of the device with coordinator node, consisted of Arduino Uno R3, NODEMCU ESP8266, and water pump, shown in Figure 2. Each type of sensors receives the data in a form of analog and digital which required python language to improve by collecting data from the sensors through arduino board. Arduino board was used to read and deliver the data to cloud system through web service. Cloud system received the data and record them into database, firebase, by collecting, separating the data based on different measurements such as humidity and temperature from soil moisture sensor and DHT11. This information was passed through a wireless model NODEMCU ESP8266. The humidity and temperature sensors operated by having NODEMCU ESP8266 linked with wifi, sending the information through to firebase with data collection conditions to capture at 8 different times which was pre-designed since the start of the programming of arduino ide.

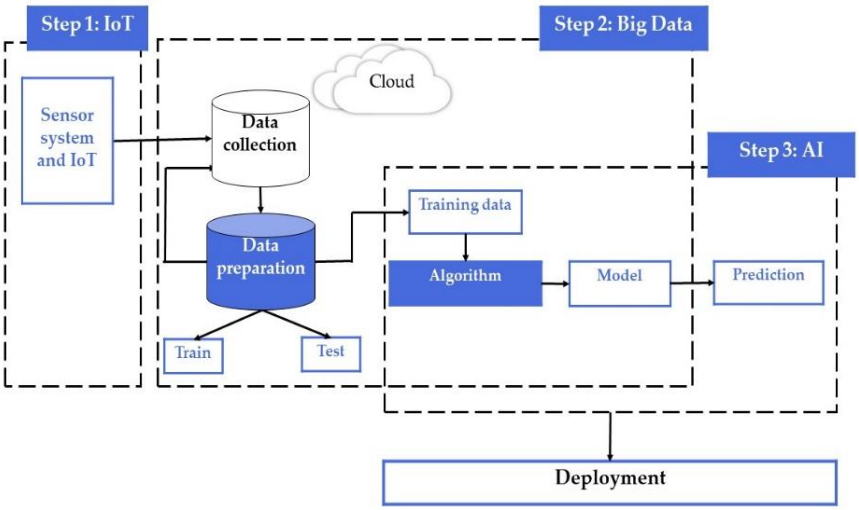


Figure 1. Framework of research method

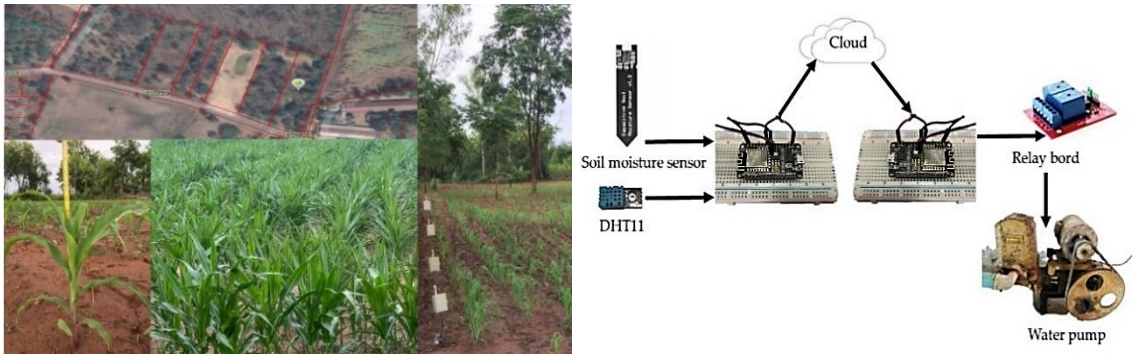


Figure 2. The prototype IoT and design of hardware sensors for maize area

3.2. Big data

Once data was received from the sensor (in step 1), application programming interface (API) was designed to receive the data from the devices and record them into the database by collecting them into the cloud computing. After that, improvement of cloud system was started by building the database for collecting

humidity and temperature from each station. Data was then pulled from cloud system to analyze the appropriate value to predict the right amount of water to grow the crops in the future.

3.3. AI

From Figure 3, the steps to finding best algorithm for prediction. There are several tools available for the application of machine learning algorithms to data cleansing, imputation data, clustering, and prediction data. After a review of data science tools, RapidMiner is one of the best tools for data science and machine learning because it allows extremely fast and easy in data analytics [29].

Sensors captured the data every day for 30 consecutive days at 8 different time points: 1 am, 4 am, 7 am, 10 am, 1 pm, 4 pm, 7 pm, and 10 pm. In total, there will be 20 data sets x 8 different time points x 30 days = 4,800 records. There might be some data lost when collecting from the sensors, so this research used cleansing and imputation with deep learning algorithm and assigned rectified linear unit (ReLU) technique to collect temperature and humidity data.

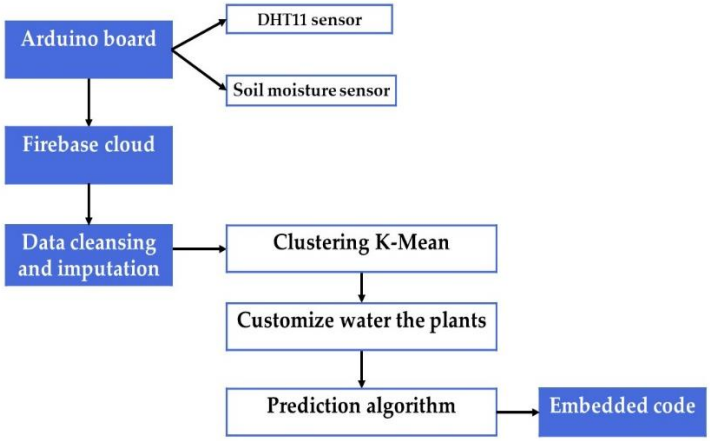


Figure 3. Steps for predictive model

After that, classifying the quality of the plantation with attributes such as time, station, temperature, humidity, and soil moisture were then tested with the k-mean clustering technique to identify the appropriate cluster. The result showed that the most appropriate cluster is cluster 3 where the elbow point and the best point from average within centroid distance is 34.643 and from davies bouldin is 0.959 as shown in Table 2. After all 3 points were identified, different classes were created by giving high, medium, and no water to the crops.

Table 2. Result of optimize cluster

Cluster	Davies bouldin	Average within centroid
2.0	1.138	54.710
3.0	0.959	34.643
4.0	1.023	29.628
5.0	1.093	25.188

Next, these 3 different classes were used in the experiment to provide the appropriate amount of water and monitor by using several sensors consisted of soil moisture sensor, DHT11 sensors to measure the temperature and soil moisture. It was found that for the class that require high amount of water, it needs a total of 15 minutes to water them, while 10 minutes is required for the class with medium amount of water. These conditions would create the soil that is moist enough to provide the appropriate condition for plantation. By analyzing the data with data selection, training data and test data were used with cross validation method to compare each model and create optimized parameter to find the most appropriate value in each model. From Table 3, the results from optimization of artificial neural network showed that the most appropriate value that was used to compare with other algorithms is the value of number of folds at 7 folds and training cycle at 18 cycles with accuracy at 99.37% and RMSE value=0.0166. The results from optimization of decision tree showed that the most appropriate value that was used to compare with other

algorithms is the value of number of folds at 7 folds and gain ratio as criterion with the accuracy at 99.35% and RMSE=0.0053.

Table 3. Results from the algorithm comparison of the analysis of data on humidity and temperature

Algorithm	Accuracy	RMSE
Artificial neural network	99.37%	0.0166
Decision tree	99.35%	0.0053
Naïve bayesian	97.46%	0.0338
Deep learning	99.60%	0.0039

The results from optimization of naïve bayes showed that the most appropriate value that was used to compare with other algorithms is the value of number of folds at 11 folds with accuracy at 97.46% and RMSE=0.0338. The results from optimization of deep learning showed that the most appropriate value that was used to compare with other algorithms is the value of number of folds at 9 folds with accuracy at 99.60% and RMSE=0.0039. From the optimization, the best model was then created by looking at the accuracy and root mean square error, RMSE. The results showed that deep learning algorithm provided the highest accuracy at 99.60% with root mean square error at 0.0039. From Figure 4, the design and improvement of the automated watering system was divided into 2 main parts:

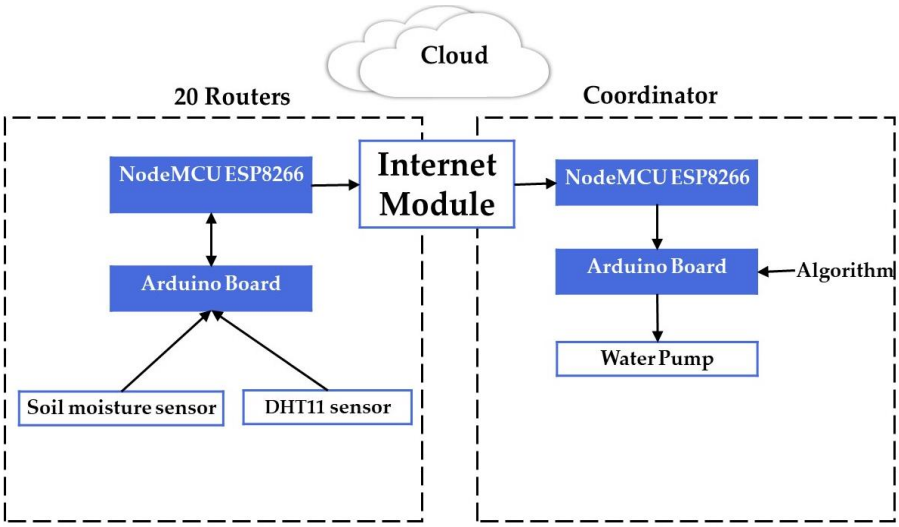


Figure 4. Data transmission between routers and coordinator

3.3.1. Router node design

Router node consisted of Arduino IDE board which connects to two different types of sensors. First was soil moisture sensor to measure the level of humidity in soil by installing the sensors at the roots of the maize at the 15 cm depth into the soil. Second, humidity and temperature sensor were used to measure the temperature. Both sensors indicated the needs of water for the maize through connecting with the wireless communication module, NODEMCU ESP8266, to send the sensor signals read from the arduino board to the coordinator node.

3.3.2. Coordinator node

The coordinator node consisted of two types of micro-controller boards which were arduino IDE and Node MCU with NODEMCU ESP8266 as communication module to receive the sensors from router node for the input of algorithm controller. This was to control the regulation of turning on and off of the water pump. The signal was sent to control the relay coordinator board which connected to the water pump to pump water into the watering system in the maize fields. Sensors received from the router nodes along with the status of all the pumps and solenoid valves was then sent up to the cloud service to record all the data and the status can be checked at monitored at real-time. Coordinator would receive data on temperature and humidity from the routers through NODEMCU ESP8266. These data would be analyzed with algorithm that

was written in the Arduino board to control the water pump system in releasing the appropriate amount of water into the experiment area.

4. RESULTS AND DISCUSSION

4.1. Measure the amount of water used to grow maize through automated algorithm control

a. Measure the amount of water sprinklers in experimentation field with dimension of 60 meters in length ×20 meters width: growing 1 row of maize with 60 meters length, distance in between each water sprinkler is at 30 centimeters. So, total of water sprinkler required for one row is equal to 6,000/30=200 sprinklers. Growing total of 25 rows of maize will require a total of 200×25=5,000 water sprinklers for the experiment.
b. Measure the rate of water used per hour: total of 5,000 water sprinklers. Water flow rate is at 2.5 liters per hour per water sprinkler. So, the total water flow rate is 5,000×2.5=12,500 liters per hour.
c. From 30 days of data collection: it was found that the total duration required to water the entire area is 420 minutes or 6 hours and 30 minutes. So, the water rate is equivalent to 12,500×6.5=81,250 liters per month.

4.2. Calculate the amount of water required to grow the maize through statistical methodology

Watering the crops through statistical methodology required the calculation of amount of water needed for each type of plants with details as follows: evapotranspiration means the total amount of water lost from the planting area to the atmosphere in the form of steam which consisted of transpiration and evaporation process. The coefficient of water usage in each type of plants is unequal. The same type of plant might also have different water usage coefficient depending on the age as well. The water usage coefficient of maize can be referred from the data from department of agriculture [30] as shown in Table 4.

Table 4. Water usage coefficient of maize at different month of age [30]

Month of age	1	2	3	4	5	6	7	8	9	10	11	12	13	14
K_c	0.63	0.72	0.86	1.13	1.35	1.52	1.61	1.63	1.58	1.50	1.38	1.15	0.90	0.67

Finding the amount of water usage of the plants referring to ET_p can be calculated by relying on the statistic of the climate in Thailand from Meteorological Department, Ministry of Digital Economy and Society [31]. ET_p of the planting area in Phetchabun province [32] was calculated as shown in Table 5.

Table 5. Amount of water used by plants referred to the planting area in Phetchabun province [31]

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
ET_p	3.33	4.05	4.96	5.18	4.16	3.69	3.58	3.43	3.22	3.69	3.73	3.41

The value then was calculated with this (5):

$$ET = K_c \times ET_p \quad (5)$$

ET=evapotranspiration, K_c =crop coefficient, ET_p =potential evapotranspiration.

The calculation of the amount of water required by maize was based on the duration of the experiment in starting in May 2021 which lasted for 1 month. Referring to the Table 4 and 5, K_c equals 0.63, ET_p equal 4.16 mm per day.

$$= 0.63 \times 4.16$$
$$= 2.6208\ mm\ per\ day$$

The amount of water required to grow maize in a month is equal to 2.6208×30= 78.624 mm. Total amount of water needed for 1,200 square meters=1,200 square meters x (78.624/1000 mm)=94.3488 cubic meters per month or approximately 94,349 liters per month which when compared to the amount of water used through the automated watering controller, it would require significantly less amount of water to grow the plants in the same area, with only 13,099 liters required for the automated water control. This can help save up to 13.89% of water usage within the 1,200 square meter planting area.

5. CONCLUSION

This research used deep learning to identify the missing value which is a popular technique used for this type of research for data cleansing and imputation purpose. After that, the data underwent clustering to different groups with k-mean technique. Based on the temperature and soil moisture data, 3 classes of optimized data were obtained and used to identify the appropriate prediction value. By comparing the best accuracy value and RMSE value, it was found that deep learning algorithm resulted in the best value. Once algorithm was received, it was then written into the arduino board of the coordinator router to regulate the watering system. Sensor data from the router was further analyzed to provide the right amount of water based on the class predicted. Results from the experiment showed that after 30 days of testing, the new system can save 13.89% water more than the conventional system. For future work, more sensors should be installed to collect more information for further analysis such as sunlight sensors and mineral detectors in soil. This methodology can be applied to other types of plants and drone system can also be incorporated to collect data in a form of videos to analyze the growth of the crops or identify pests as well.

ACKNOWLEDGEMENTS

This research is supported by Faculty of Information Technology and Digital Innovation, KMUTNB and Faculty of Sciences and Technology, PCRU. This research is supported by the Thailand Science Research and Innovation, and Phetchabun Rajabhat University, with project No. 65A145000002.

REFERENCES

[1] P. Hebron, *Machine learning for designers*. O'Reilly Media, 2016.
[2] N. Buduma, *Fundamentals of deep learning: designing next-generation machine intelligence algorithms*. O'Reilly Media, 2017.
[3] A. Julianto and A. Sunyoto, "A performance evaluation of convolutional neural network architecture for classification of rice leaf disease," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 4, pp. 1069–1078, Dec. 2021, doi: 10.11591/ijai.v10.i4.pp1069-1078.
[4] B. Annanurov and N. Noor, "A compact deep learning model for Khmer handwritten text recognition," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 584–591, Sep. 2021, doi: 10.11591/ijai.v10.i3.pp584-591.
[5] S. Verma, G. T. Thampi, and M. Rao, "ANN based method for improving gold price forecasting accuracy through modified gradient descent methods," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 1, pp. 46–57, Mar. 2020, doi: 10.11591/ijai.v9.i1.pp46-57.
[6] C. C. Aggarwal, "Training deep neural networks," in *Neural Networks and Deep Learning*, Cham: Springer International Publishing, 2018, pp. 105–167.
[7] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: a," *Heliyon*, vol. 4, no. 11, Nov. 2018, doi: 10.1016/j.heliyon.2018.e00938.
[8] L. Kanagasabai, "Shrinkage of power loss by enriched brain storm optimization algorithm," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 1, pp. 1–6, Mar. 2019, doi: 10.11591/ijai.v8.i1.pp1-6.
[9] R. S. El-Sayed and M. N. El-Sayed, "Classification of vehicles' types using histogram oriented gradients: comparative study and modification," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 4, pp. 700–712, Dec. 2020, doi: 10.11591/ijai.v9.i4.pp700-712.
[10] M. Morgan, C. Blank, and R. Seetan, "Plant disease prediction using classification algorithms," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 1, pp. 257–264, Mar. 2021, doi: 10.11591/ijai.v10.i1.pp257-264.
[11] T. A. Assegie, R. L. Tulasi, and N. K. Kumar, "Breast cancer prediction model with decision tree and adaptive boosting," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 1, pp. 184–190, Mar. 2021, doi: 10.11591/ijai.v10.i1.pp184-190.
[12] S. Marzukhi, N. Awang, S. N. Alsagoff, and H. Mohamed, "RapidMiner and machine learning techniques for classifying aircraft data," *Journal of Physics: Conference Series*, vol. 1997, no. 1, Aug. 2021, doi: 10.1088/1742-6596/1997/1/012012.
[13] M. H. Ali and N. K. Ali, "IoT based security system and intelligent home automation multi monitoring and control systems," *IAES International Journal of Robotics and Automation (IJRA)*, vol. 8, no. 3, pp. 205–210, Sep. 2019, doi: 10.11591/ijra.v8i3.pp205-210.
[14] A. H. Ali, A. H. Duhis, N. A. L. Alzurfi, and M. J. Mnati, "Smart monitoring system for pressure regulator based on IOT," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 3450–3456, Oct. 2019, doi: 10.11591/ijece.v9i5.pp3450-3456.
[15] V. S. Padala, K. Gandhi, and P. Dasari, "Machine learning: the new language for applications," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 4, pp. 411–421, Dec. 2019, doi: 10.11591/ijai.v8.i4.pp411-421.
[16] S. Chen *et al.*, "Internet of things based smart grids supported by intelligent edge computing," *IEEE Access*, vol. 7, pp. 74089–74102, 2019, doi: 10.1109/ACCESS.2019.2920488.
[17] S. Chatterjee, A. K. Kar, and M. P. Gupta, "Success of IoT in smart cities of india: an empirical analysis," *Government Information Quarterly*, vol. 35, no. 3, pp. 349–361, Sep. 2018, doi: 10.1016/j.giq.2018.05.002.
[18] "Agricultural economics of mize." Office of Agricultural Economics, Ministry of Agriculture and Cooperatives, 2020.
[19] "The study report project on suatainable consumtion and production of maize supply chain in Thailand," Mimistry for the Environment, Nature Conservation, 2018.
[20] P. Tangwannawit, "Development of of smart internet of things (IoT) for local vegetables," in *The 15th National Conference and International Conference on Applied Computer Technology and Information Systems*, 2019, pp. 134–146.
[21] A. Riansyah, S. Mulyono, and M. Roichani, "Applying fuzzy proportional integral derivative on internet of things for figs greenhouse," *IAES International Journal of Artificial Intelligence*, vol. 10, no. 3, pp. 536–544, Sep. 2021, doi: 10.11591/ijai.v10.i3.pp536-544.

[22] P. Tangwannawit and K. Saengkrajang, "An internet of things secosystem for planting of coriander (Coriandrum sativum L.)," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 5, pp. 4568–4576, Oct. 2021, doi: 10.11591/ijece.v11i5.pp4568-4576.
[23] N. Seman and N. Atiqah Razmi, "Machine learning-based technique for big data sentiments extraction," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 3, pp. 473–479, Sep. 2020, doi: 10.11591/ijai.v9.i3.pp473-479.
[24] S. Tangwannawit and P. Tangwannawit, "Artificial intelligence theory and applications." Artificial Intelligence Association of Thailand, pp. 36–80, 2020.
[25] A. M. Abdu, M. M. M. Mokji, and U. U. U. Sheikh, "Machine learning for plant disease detection: an investigative comparison between support vector machine and deep learning," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 4, pp. 670–683, Dec. 2020, doi: 10.11591/ijai.v9.i4.pp670-683.
[26] A. Ullah, "Artificial bee colony algorithm used for load balancing in cloud computing: review," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 2, pp. 156–167, Jun. 2019, doi: 10.11591/ijai.v8.i2.pp156-167.
[27] N. A. K. Rosili, R. Hassan, N. H. Zakaria, S. Kasim, F. Z. C. Rose, and T. Sutikno, "A systematic literature review of machine learning methods in predicting court decisions," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 4, pp. 1091–1102, Dec. 2021, doi: 10.11591/ijai.v10.i4.pp1091-1102.
[28] N. A. Mashudi, N. Ahmad, and N. M. Noor, "Classification of adult autistic spectrum disorder using machine learning approach," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 743–751, Sep. 2021, doi: 10.11591/ijai.v10.i3.pp743-751.
[29] "RapidMiner reviews," *Gartner*. https://www.gartner.com/reviews/market/data-sciencemachine-learning-platforms/vendor/rapidminer/reviews (accessed Feb. 28, 2021).
[30] "Reference crop evapotranspiration by penman monteith." Department of Agriculture, Thailand, 2019.
[31] "Water usage of the plants." Meteorological Department, Ministry of Digital Economy and Society, 2019.
[32] P. Tangwannawit and K. Saengkrajang, "Technology accept ance model to evaluate the adoption of the internet of things for planting maize," *Life Sciences and Environment Journal*, vol. 22, no. 2, pp. 262–273, 2021, doi: 10.14456/lsej.2021.13.

BIOGRAPHIES OF AUTHORS

Sakchai Tangwannawit received the professional education with B.S. (Computer Science), M.S. (Information Technology), and Ph.D. (Computer Education) from King Mongkut's University of Technology North Bangkok (KMUTNB), Bangkok, Thailand. He has been working as a lecturer for more than 10 years in the field of Information Technology in the Faculty of Information Technology and Digital Innovation, KMUTNB. He can be contacted at email: sakchai.t@itd.kmutnb.ac.th.

Panana Tangwannawit received the professional education with B.S. (Computer Science), M.S. (Information Technology), and Ph.D. (Information Technology) from King Mongkut's University of Technology North Bangkok (KMUTNB), Bangkok, Thailand. She has been working as a lecturer for more than 24 years in the field of Computer and Information Technology in the Faculty of Sciences and Technology, Phetchabun Rajabhat University, Thailand. She can be contacted at email: panana.t@gmail.com.

IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 99, No. 1, Month 2099, pp. 1~1x
ISSN: 2252-8938, DOI: 10.11591/ijai.v99.i1.pp1-1x

Paper's title should be the fewest possible words that accurately describe the content of the paper (Center, Bold, 16pt)

Abdel-Rahman Hedar[1,2], Patricia Melin[3], Kennedy Okokpujie[4] (10 pt)
[1]Department of Computer Science, Faculty of Computers & Information, Assiut University, Assiut, Egypt (8 pt)
[2]Department of Computer Science in Jamoum, Umm Al-Qura University, Makkah, Saudi Arabia
[3]Division of Graduate Studies, Tijuana Institute of Technology, Tijuana, Mexico
[4]Department of Electrical and Information Engineering, College of Engineering, Covenant University, Ogun State, Nigeria

Article Info

Article history:

Received month dd, yyyy
Revised month dd, yyyy
Accepted month dd, yyyy

Keywords:

First keyword
Second keyword
Third keyword
Fourth keyword
Fifth keyword

ABSTRACT (10 PT)

An abstract is often presented separate from the article, so it must be able to stand alone. A well-prepared abstract enables the reader to identify the basic content of a document quickly and accurately, to determine its relevance to their interests, and thus to decide whether to read the document in its entirety. The abstract should be informative and completely self-explanatory, provide a clear statement of the problem, the proposed approach or solution, and point out major findings and conclusions. **The Abstract should be 100 to 200 words in length.** References should be avoided, but if essential, then cite the author(s) and year(s). Standard nomenclature should be used, and non-standard or uncommon abbreviations should be avoided, but if essential they must be defined at their first mention in the abstract itself. No literature should be cited. The keyword list provides the opportunity to add 5 to 7 keywords, used by the indexing and abstracting services, in addition to those already present in the title (9 pt).

Corresponding Author:

Kennedy Okokpujie
Department of Electrical and Information Engineering, College of Engineering, Covenant University
Km. 10 Idiroko Road, Canaan Land, Ota, Ogun State, Nigeria
Email: kennedy.okokpujie@covenantuniversity.edu.nga

1. INTRODUCTION (10 PT)

The main text format consists of a flat left-right columns on A4 paper (quarto). The margin text from the left and top are 2.5 cm, right and bottom are 2 cm. The manuscript is written in Microsoft Word, single space, Time New Roman 10 pt, and maximum 12 pages for original research article, or maximum 16 pages for review/survey paper, which can be downloaded at the website: http://ijai.iaescore.com.

A title of article should be the fewest possible words that accurately describe the content of the paper. The title should be succinct and informative and no more than about 12 words in length. Do not use acronyms or abbreviations in your title and do not mention the method you used, unless your paper reports on the development of a new method. Titles are often used in information-retrieval systems. Avoid writing long formulas with subscripts in the title. Omit all waste words such as "*A study of ...*", "*Investigations of ...*", "*Implementation of ...*", "*Observations on ...*", "*Effect of.....*", "*Analysis of ...*", "Design of…", etc.

A concise and factual abstract is required. The abstract should state briefly the purpose of the research, the principal results and major conclusions. An abstract is often presented separately from the article, so it must be able to stand alone. For this reason, References should be avoided, but if essential, then cite the author(s) and year(s). Also, non-standard or uncommon abbreviations should be avoided, but if essential they must be defined at their first mention in the abstract itself. Immediately after the abstract, provide a maximum of 7 keywords, using American spelling and avoiding general and plural terms and

multiple concepts (avoid, for example, 'and', 'of'). Be sparing with abbreviations: only abbreviations firmly established in the field may be eligible. These keywords will be used for indexing purposes.

Indexing and abstracting services depend on the accuracy of the title, extracting from it keywords useful in cross-referencing and computer searching. An improperly titled paper may never reach the audience for which it was intended, so be specific.

The Introduction section should provide: i) a clear background, ii) a clear statement of the problem, iii) the relevant literature on the subject, iv) the proposed approach or solution, and v) the new value of research which it is innovation (within 3-6 paragraphs). It should be understandable to colleagues from a broad range of scientific disciplines. Organization and citation of the bibliography are made in Institute of Electrical and Electronics Engineers (IEEE) style in sign [1], [2] and so on. The terms in foreign languages are written italic (*italic*). The text should be divided into sections, each with a separate heading and numbered consecutively [3]. The section or subsection headings should be typed on a separate line, e.g., 1. INTRODUCTION. A full article usually follows a standard structure: **1. Introduction, 2. The Comprehensive Theoretical Basis and/or the Proposed Method/Algorithm** *(optional)***, 3. Method, 4. Results and Discussion, and 5. Conclusion.** The structure is well-known as **IMRaD** style.

Literature review that has been done author used in the section "INTRODUCTION" to explain the difference of the manuscript with other papers, that it is innovative, it are used in the section "METHOD" to describe the step of research and used in the section "RESULTS AND DISCUSSION" to support the analysis of the results [2]. If the manuscript was written really have high originality, which proposed a new method or algorithm, the additional section after the "INTRODUCTION" section and before the "METHOD" section can be added to explain briefly the theory and/or the proposed method/algorithm [4].

2. METHOD (10 PT)

Explaining research chronological, including research design, research procedure (in the form of algorithms, Pseudocode or other), how to test and data acquisition [5]–[7]. The description of the course of research should be supported references, so the explanation can be accepted scientifically [2], [4]. Figures 1-2 and Table 1 are presented center, as shown below and cited in the manuscript [5], [8]–[13]. The settlement curves produced at SG1 has been illustrated in Figure 2(a) and SG2 has been illustrated Figure 2(b).

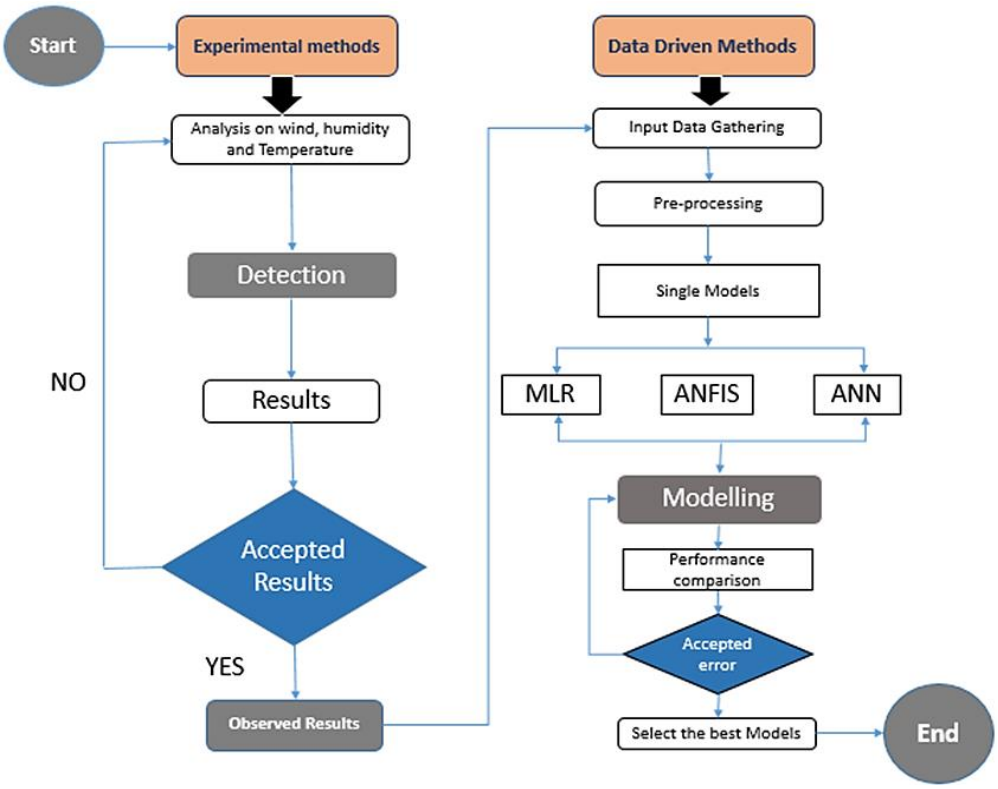
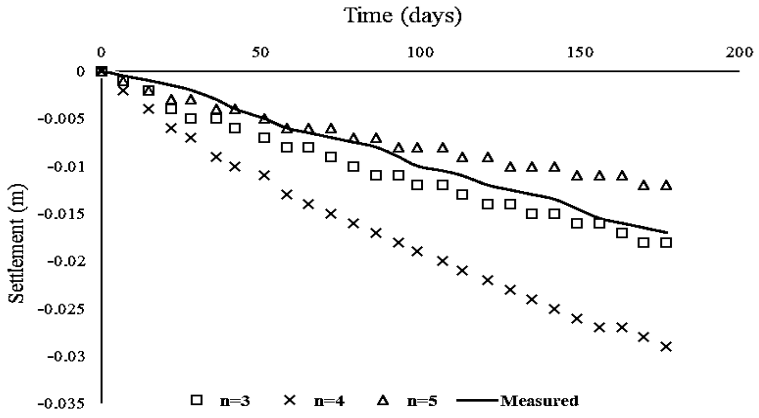
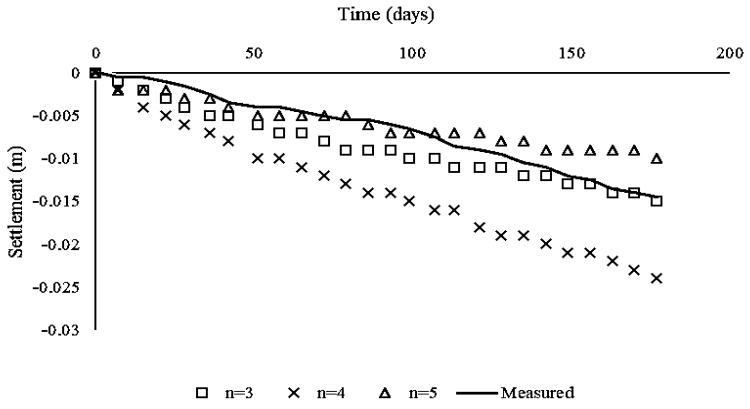


Figure 1. Shows the flowchart of the AI-based models and experimental methods applied



(a)



(b)

Figure 2. The relationship of soil settlement and time, (a) SG1 and (b) SG2

Table 1. The performance of ...

Variable	Speed (rpm)	Power (kW)
x	10	8.6
y	15	12.4
z	20	15.3

3. RESULTS AND DISCUSSION (10 PT)

In this section, it is explained the results of research and at the same time is given the comprehensive discussion. Results can be presented in figures, graphs, tables and others that make the reader understand easily [14], [15]. The discussion can be made in several sub-sections.

3.1. Sub section 1

Equations should be placed at the center of the line and provided consecutively with equation numbers in parentheses flushed to the right margin, as in (1). The use of Microsoft Equation Editor or MathType is preferred.

$$E_v - E = \frac{h}{2.m} (k_x^2 + k_y^2) \quad (1)$$

All symbols that have been used in the equations should be defined in the following text.

3.2. Sub section 2

Proper citation of other works should be made to avoid plagiarism. When referring to a reference item, please use the reference number as in [16] or [17] for multiple references. The use of ”Ref [18]...”

should be employed for any reference citation at the beginning of sentence. For any reference with more than 3 or more authors, only the first author is to be written followed by *et al*. (e.g. in [19]). Examples of reference items of different categories shown in the References section. Each item in the references section should be typed using 9 pt font size [20]–[25].

3.2.1. Subsub section 1

yy

3.2.2. Subsub section 2

zz

4. CONCLUSION (10 PT)

Provide a statement that what is expected, as stated in the "INTRODUCTION" section can ultimately result in "RESULTS AND DISCUSSION" section, so there is compatibility. Moreover, it can also be added the prospect of the development of research results and application prospects of further studies into the next (based on result and discussion).

ACKNOWLEDGEMENTS (10 PT)

Author thanks ... . In most cases, sponsor and financial support acknowledgments.

REFERENCES (10 PT)

The main references are international journals and proceedings. All references should be to the most pertinent, up-to-date sources **and the minimum of references** are **25 entries** (for original research paper) and **50 entries** (for review/survey paper). References are written in **IEEE style**. For more complete guide can be accessed at (http://ipmuonline.com/guide/refstyle.pdf). Use of a tool such as **EndNote**, **Mendeley**, or **Zotero** for reference management and formatting, and choose **IEEE style**. Please use a consistent format for references-see examples (8 pt):

[1] Journal/Periodicals

Basic Format:

J. K. Author, "Title of paper," *Abbrev. Title of Journal/Periodical*, vol. *x*, no. *x*, pp. *xxx-xxx*, Abbrev. Month, year, doi: *xxx*.

Examples:

- M. M. Chiampi and L. L. Zilberti, "Induction of electric field in human bodies moving near MRI: An efficient BEM computational procedure," *IEEE Trans. Biomed. Eng.*, vol. 58, pp. 2787–2793, Oct. 2011, doi: 10.1109/TBME.2011.2158315.
- R. Fardel, M. Nagel, F. Nuesch, T. Lippert, and A. Wokaun, "Fabrication of organic light emitting diode pixels by laser-assisted forward transfer," *Appl. Phys. Lett.*, vol. 91, no. 6, Aug. 2007, Art. no. 061103, doi: 10.1063/1.2759475.

[2] Conference Proceedings

Basic Format:

J. K. Author, "Title of paper," in *Abbreviated Name of Conf.*, (location of conference is optional), year, pp. *xxx–xxx*, doi: *xxx*.

Examples:

- G. Veruggio, "The EURON roboethics roadmap," in *Proc. Humanoids '06: 6th IEEE-RAS Int. Conf. Humanoid Robots*, 2006, pp. 612–617, doi: 10.1109/ICHR.2006.321337.
- J. Zhao, G. Sun, G. H. Loh, and Y. Xie, "Energy-efficient GPU design with reconfigurable in-package graphics memory," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2012, pp. 403–408, doi: 10.1145/2333660.2333752.

[3] Book

Basic Format:

J. K. Author, "Title of chapter in the book," in *Title of His Published Book*, X. Editor, Ed., *x*th ed. City of Publisher, State (only U.S.), Country: Abbrev. of Publisher, year, ch. *x*, sec. *x*, pp. *xxx–xxx*.

Examples:

- A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method* in Computational Electrodynamics II, vol. 3, 2nd ed. Norwood, MA, USA: Artech House, 1996.
- R. L. Myer, "Parametric oscillators and nonlinear materials," in *Nonlinear Optics*, vol. 4, P. G. Harper and B. S. Wherret, Eds., San Francisco, CA, USA: Academic, 1977, pp. 47–160.

[4] M. Theses (B.S., M.S.) and Dissertations (Ph.D.)

Basic Format:

J. K. Author, "Title of thesis," M.S. thesis, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.

J. K. Author, "Title of dissertation," Ph.D. dissertation, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.

Examples:

- J. O. Williams, "Narrow-band analyzer," Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, USA, 1993.
- N. Kawasaki, "Parametric study of thermal and chemical nonequilibrium nozzle flow," M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

*In the reference list, however, list all the authors for up to six authors. Use *et al.* only if: 1) The names are not given and 2) List of authors more than 6. *Example*: J. D. Bellamy *et al*., Computer Telephony Integration, New York: Wiley, 2010.

See the examples:

REFERENCES

[1] T. S. Ustun, C. Ozansoy, and A. Zayegh, "Recent developments in microgrids and example cases around the world—A review," *Renew. Sustain. Energy Rev.*, vol. 15, no. 8, pp. 4030–4041, Oct. 2011, doi: 10.1016/j.rser.2011.07.033.

[2] D. Salomonsson, L. Soder, and A. Sannino, "Protection of Low-Voltage DC Microgrids," *IEEE Trans. Power Deliv.*, vol. 24, no. 3, pp. 1045–1053, Jul. 2009, doi: 10.1109/TPWRD.2009.2016622.

[3] S. Chakraborty and M. G. Simoes, "Experimental Evaluation of Active Filtering in a Single-Phase High-Frequency AC Microgrid," *IEEE Trans. Energy Convers.*, vol. 24, no. 3, pp. 673–682, Sep. 2009, doi: 10.1109/TEC.2009.2015998.

[4] S. A. Hosseini, H. A. Abyaneh, S. H. H. Sadeghi, F. Razavi, and A. Nasiri, "An overview of microgrid protection methods and the factors involved," *Renew. Sustain. Energy Rev.*, vol. 64, pp. 174–186, Oct. 2016, doi: 10.1016/j.rser.2016.05.089.

[5] S. Chen, N. Tai, C. Fan, J. Liu, and S. Hong, "Sequence-component-based current differential protection for transmission lines connected with IIGs," *IET Gener. Transm. Distrib.*, vol. 12, no. 12, pp. 3086–3096, Jul. 2018, doi: 10.1049/iet-gtd.2017.1507.

[6] S. Parhizi, H. Lotfi, A. Khodaei, and S. Bahramirad, "State of the Art in Research on Microgrids: A Review," *IEEE Access*, vol. 3, pp. 890–925, 2015, doi: 10.1109/ACCESS.2015.2443119.

[7] S. Chowdhury, S. P. Chowdhury, and P. Crossley, *Microgrids and Active Distribution Networks*. Institution of Engineering and Technology, 2009.

[8] R. Ndou, J. I. Fadiran, S. Chowdhury, and S. P. Chowdhury, "Performance comparison of voltage and frequency based loss of grid protection schemes for microgrids," in *2013 IEEE Power & Energy Society General Meeting*, 2013, pp. 1–5, doi: 10.1109/PESMG.2013.6672788.

[9] S. Liu, T. Bi, A. Xue, and Q. Yang, "Fault analysis of different kinds of distributed generators," in *2011 IEEE Power and Energy Society General Meeting*, Jul. 2011, pp. 1–6, doi: 10.1109/PES.2011.6039596.

[10] K. Jennett, F. Coffele, and C. Booth, "Comprehensive and quantitative analysis of protection problems associated with increasing penetration of inverter-interfaced DG," in *11th IET International Conference on Developments in Power Systems Protection (DPSP 2012)*, 2012, pp. P31–P31, doi: 10.1049/cp.2012.0091.

[11] P. T. Manditereza and R. Bansal, "Renewable distributed generation: The hidden challenges – A review from the protection perspective," *Renew. Sustain. Energy Rev.*, vol. 58, pp. 1457–1465, May 2016, doi: 10.1016/j.rser.2015.12.276.

[12] D. M. Bui, S.-L. Chen, K.-Y. Lien, Y.-R. Chang, Y.-D. Lee, and J.-L. Jiang, "Investigation on transient behaviours of a uni-grounded low-voltage AC microgrid and evaluation on its available fault protection methods: Review and proposals," *Renew. Sustain. Energy Rev.*, vol. 75, pp. 1417–1452, Aug. 2017, doi: 10.1016/j.rser.2016.11.134.

[13] T. N. Boutsika and S. A. Papathanassiou, "Short-circuit calculations in networks with distributed generation," *Electr. Power Syst. Res.*, vol. 78, no. 7, pp. 1181–1191, Jul. 2008, doi: 10.1016/j.epsr.2007.10.003.

[14] H. Margossian, G. Deconinck, and J. Sachau, "Distribution network protection considering grid code requirements for distributed generation," *IET Gener. Transm. Distrib.*, vol. 9, no. 12, pp. 1377–1381, Sep. 2015, doi: 10.1049/iet-gtd.2014.0987.

[15] O. Núñez-Mata, R. Palma-Behnke, F. Valencia, A. Urrutia-Molina, P. Mendoza-Araya, and G. Jiménez-Estévez, "Coupling an adaptive protection system with an energy management system for microgrids," *Electr. J.*, vol. 32, no. 10, p. 106675, Dec. 2019, doi: 10.1016/j.tej.2019.106675.

[16] M. Brucoli and T. C. Green, "Fault behaviour in islanded microgrids," in *Proceedings of the 19th international conference on electricity distribution, CIRED*, 2007, pp. 0548-(1-4).

[17] I. K. Tarsi, A. Sheikholeslami, T. Barforoushi, and S. M. B. Sadati, "Investigating impacts of distributed generation on distribution networks reliability: A mathematical model," in *Proceedings of the 2010 Electric Power Quality and Supply Reliability Conference*, Jun. 2010, pp. 117–124, doi: 10.1109/PQ.2010.5550010.

[18] L. K. Kumpulainen and K. T. Kauhaniemi, "Analysis of the impact of distributed generation on automatic reclosing," in *IEEE PES Power Systems Conference and Exposition, 2004.*, pp. 1152–1157, doi: 10.1109/PSCE.2004.1397623.

[19] A. A. Memon and K. Kauhaniemi, "A critical review of AC Microgrid protection issues and available solutions," *Electr. Power Syst. Res.*, vol. 129, pp. 23–31, Dec. 2015, doi: 10.1016/j.epsr.2015.07.006.

[20] H. A. Abdel-Ghany, A. M. Azmy, N. I. Elkalashy, and E. M. Rashad, "Optimizing DG penetration in distribution networks concerning protection schemes and technical impact," *Electr. Power Syst. Res.*, vol. 128, pp. 113–122, Nov. 2015, doi: 10.1016/j.epsr.2015.07.005.

[21] S. Chaitusaney and A. Yokoyama, "An Appropriate Distributed Generation Sizing Considering Recloser-Fuse Coordination," in *2005 IEEE/PES Transmission & Distribution Conference & Exposition: Asia and Pacific*, pp. 1–6, doi: 10.1109/TDC.2005.1546838.

[22] H. H. Zeineldin, Y. A.-R. I. Mohamed, V. Khadkikar, and V. R. Pandi, "A Protection Coordination Index for Evaluating Distributed Generation Impacts on Protection for Meshed Distribution Systems," *IEEE Trans. Smart Grid*, vol. 4, no. 3, pp. 1523–1532, Sep. 2013, doi: 10.1109/TSG.2013.2263745.

[23] D. Eltigani and S. Masri, "Challenges of integrating renewable energy sources to smart grids: A review," *Renew. Sustain. Energy Rev.*, vol. 52, pp. 770–780, Dec. 2015, doi: 10.1016/j.rser.2015.07.140.

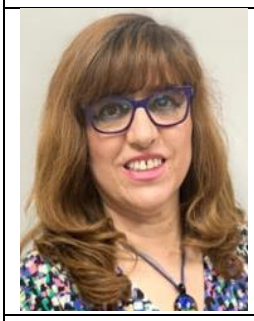
[24] M. M. Eissa (SIEEE), "Protection techniques with renewable resources and smart grids—A survey," *Renew. Sustain. Energy Rev.*, vol. 52, pp. 1645–1667, Dec. 2015, doi: 10.1016/j.rser.2015.08.031.

[25] A. Oudalov *et al.*, "Novel Protection Systems for Microgrids," 2009. [Online]. Available: http://www.microgrids.eu/documents/688.pdf.

BIOGRAPHIES OF AUTHORS (10 PT)

The recommended number of authors is at least 2. One of them as a corresponding author.

Please attach clear photo (3x4 cm) and vita. Example of biographies of authors:

	**Abdel-Rahman Hedar** holds a Doctor of Informatics degree from Kyoto University, Japan in 2004. He also received his B.Sc. and M.Sc. (Mathematics) from Assiut University, Egypt in 1993 and 1997, respectively. He is currently an associate professor at Computer Science Department in Jamoum, Umm Al-Qura University, Makkah, Saudi Arabia. He is also an associate professor of artifcial intelligence in Assiut University since January 2012. His research includes meta-heuristics, global optimization, machine learning, data mining, bioinformatics, graph theory and parallel programming. He has published over 70 papers in international journals and conferences. From July 2005 to July 2007, he was a JSPS research fellow in Kyoto University, Japan. He can be contacted at email: ahahmed@uqu.edu.sa or hedar@aun.edu.eg.
	**Patricia Melin** received the D.Sc. degree (Doctor Habilitatus D.Sc.) in computer science from the Polish Academy of Sciences, Warsaw, Poland, with the Dissertation "Hybrid Intelligent Systems for Pattern Recognition using Soft Computing". She is a Professor of Computer Science in the Graduate Division, Tijuana Institute of Technology, Tijuana, Mexico since 1998. In addition, she is serving as Director of Graduate Studies in computer science and Head of the research group on Computational Intelligence (2000–present). Her research interests are in Type-2 Fuzzy Logic, Modular Neural Networks, Pattern Recognition, Neuro-Fuzzy and Genetic-Fuzzy hybrid approaches., She is currently the President of Hispanic American Fuzzy Systems Association (HAFSA) and is the founding Chair of the Mexican Chapter of the IEEE Computational Intelligence Society. She can be contacted at email: pmelin@tectijuana.mx.
	**Dr. Kennedy Okokpujie** holds a Bachelor of Engineering (B.Eng.) in Electrical and Electronics Engineering, Master of Science (M.Sc.) in Electrical and Electronics Engineering, Master of Engineering (M.Eng.) in Electronics and Telecommunication Engineering and Master of Business Administration (MBA), Ph.D in Information and Communication Engineering, besides several professional certificates and skills. He is currently lecturing with the department of Electrical and Information Engineering at Covenant University, Ota, Ogun State, Nigeria. He is a member of the Nigeria Society of Engineers and the Institute of Electrical and Electronics Engineers (IEEE). His research areas of interest include Biometrics, Artificial Intelligent, and Digital signal Processing. He can be contacted at email: kennedy.okokpujie@covenantuniversity.edu.ng.