Comparative analysis of convolutional neural network architecture for post forest fire area classification based on vegetation image

Ahmad Bintang Arif[1], Imas Sukaesih Sitanggang[1], Hari Agung Adrianto[1], Lailan Syaufina[2]
[1]School of Data Science, Mathematics and Informatics, IPB University, Bogor, Indonesia
[2]Department of Silviculture, Faculty of Forestry and Environment, IPB University, Bogor, Indonesia

Article Info

Article history:

Received Aug 26, 2024
Revised Apr 17, 2025
Accepted Jul 3, 2025

Keywords:

Architecture comparison
Convolutional neural network
Field imagery
Forest and land fire

ABSTRACT

This study presents a comparative analysis of 7 Convolutional Neural Network (CNN) architectures—MobileNetV2, VGG16, VGG19, LeNet5, AlexNet, ResNet50, and InceptionV3—for classifying post-forest fire areas using field-based vegetation imagery. A total of 56 models were evaluated through combinations of batch size, input size, and optimizer. The results show that MobileNetV2, VGG16, and VGG19 outperformed other models, with validation accuracies exceeding 88%. MobileNetV2 emerged as the most balanced model, achieving 96% accuracy with the lowest model size and training time, making it ideal for resource-constrained applications. This study highlights the potential of CNN-based classification using mobile field imagery, offering an efficient alternative to costly and condition-dependent satellite or drone data. The findings support real-time, localized identification of burned areas after forest fires, providing actionable insights for prioritizing recovery areas and guiding ecological restoration and land rehabilitation strategies.

Corresponding Author:

Imas Sukaesih Sitanggang
School of Data Science, Mathematics and Informatics, IPB University
Meranti Street, Wing 20 Level 5, IPB Campus Darmaga Bogor 16680, Indonesia
Email: imas.sitanggang@apps.ipb.ac.id

1. INTRODUCTION

Indonesia has a long-standing history of recurring forest fires, which have become an annual occurrence, severely affecting the environment and local communities. Forest fires have been a significant issue as early as the 1980s [1]. These fires have contributed to air pollution, health issues, biodiversity loss, and the degradation of ecosystems [2]. Analyzing post-fire areas is therefore crucial for mitigation, restoration, and sustainable land management. Recent studies have introduced convolutional neural networks (CNNs) as a promising solution for wildfire damage classification due to their powerful feature extraction and image recognition capabilities [3].

Most existing works utilize satellite and drone imagery to classify post-forest fire areas using CNNs. For example, [4] and [5] used drone images with CNN-based models such as VGG-16 and FFireNet, achieving high accuracy levels of 96% and 98.42%, respectively. Other researcher utilized Sentinel-2 satellite images to classify images into five categories (‘field,’ ‘forest,’ ‘smoke,’ ‘urban,’ and ‘burned’) using ResNet and Xception models, with Xception achieving the highest accuracy of 96.7% [6]. Despite their success, these approaches still depend on expensive image acquisition, varying image resolutions, and are limited by spatial or weather-related constraints [7].

To overcome these challenges, field-based or ground-level imagery has emerged as a viable alternative. Field images captured using mobile phones provide detailed vegetation and soil information and are less affected by cloud cover or atmospheric distortions. A study by [8] applied MobileNetV2 on field imagery from Jambi Province, achieving 77.7% accuracy. However, research in this direction remains limited, particularly in optimizing CNN architectures and hyperparameter combinations for such datasets.

This study addresses this gap by systematically evaluating 7 CNN architectures (MobileNetV2, VGG16, VGG19, LeNet5, AlexNet, ResNet50, and InceptionV3) on a dataset of post-fire vegetation images collected in the field. This study applied hyperparameter tuning involving batch size, input size, and optimizer to determine their influence on model performance. The goal of this study is to identify an architecture that balances high classification accuracy with computational efficiency—especially relevant for real-time applications in remote or resource-limited environments. The key contributions of this work include a comparative evaluation of CNN architectures specifically tailored for post-forest fire classification in field conditions, an analysis of how batch size, input size, and optimizer affect performance, and valuable insights into the viability of using mobile-captured field images as a cost-effective and practical alternative to traditional remote sensing methods.

2. METHOD

The methodology of this research comprises five main stages: data collection, preprocessing, data splitting, modeling, and evaluation, as illustrated in Figure 1. These stages are adapted from widely accepted image classification workflows [9], and each is carefully designed to ensure high-quality input data, optimal model performance, and rigorous evaluation. The overall approach integrates field-collected image data with deep learning-based classification to assess post-fire land conditions.

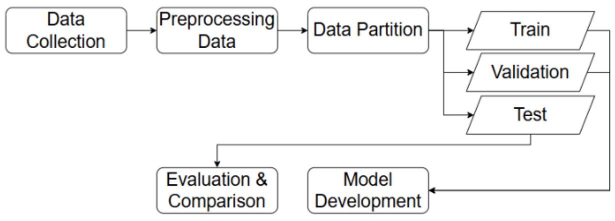

Figure 1. Research stages

2.1. Data collection

This study utilized field images captured using mobile phones from 4 post-fire locations in Jambi Province, namely Pematang Rahim, Pematang Lumut, Pelayangan, and Tenam. Field imagery was chosen due to its high resolution, clearer visual details, immunity to atmospheric disturbances, and lower cost compared to satellite or drone imagery. The collected images were initially categorized into three factors: area, soil, and vegetation. However, only vegetation-related images were used for the classification model, as vegetation plays a significant role in assessing post-fire areas [10]. In total, 239 images were used, divided into two classes: burned area (139 images) and unburned area (100 images). Sample images from each class are presented in Figures 2 and 3.

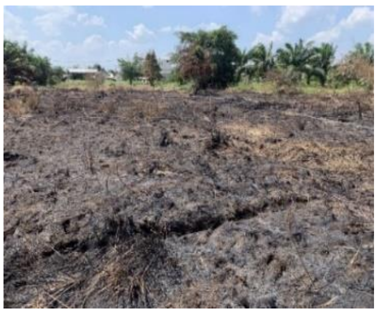

Figure 2. Burned area image

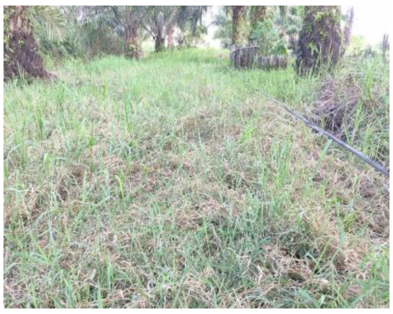

Figure 3. Unburned area image

2.2. Preprocessing data

Preprocessing is a critical step in deep learning pipelines to improve model learning efficiency and output quality [11]. In this study, three key preprocessing operations were applied. First, resizing was performed by standardizing all images to dimensions of either 192×192 or 224×224 pixels. This not only reduced computational cost and memory usage but also ensured consistent input dimensions across models. While resizing can introduce information loss, this was mitigated by selecting relatively high target resolutions and preserving aspect ratios where possible [12]. Second, normalization was applied by scaling pixel values to the [0, 1] range, which facilitated faster and more stable convergence during training, particularly when using gradient-based optimizers [13]. Third, data augmentation techniques such as random rotation, flipping, and zooming were employed to increase data variability, reduce overfitting, and improve model generalization [14].

2.3. Data partition

Before training, the dataset was divided into three subsets: training, validation, and test sets. The training set was used to fit the model, the validation set was used to tune the model and prevent overfitting during training, and the test set was reserved for final performance evaluation on unseen data [15]. This partitioning ensures that the model has sufficient data to learn effectively while also being properly validated and tested. The dataset was split with the following proportions: 80% for training (111 burned images and 80 unburned images), 10% for validation (14 burned and 10 unburned), and 10% for testing (14 burned and 10 unburned).

2.4. Model development

This study explored two approaches to CNN model development: transfer learning using pretrained models and training from scratch. The pretrained models included MobileNetV2, VGG-16, VGG-19, ResNet-50, and Inception, all of which were pretrained on the ImageNet dataset. Leveraging transfer learning allows for improved performance on small datasets and faster convergence due to the reuse of learned feature representations [16]. In parallel, two CNN models—LeNet-5 and AlexNet—were trained from scratch. These architectures serve as baseline models and enable comparison of shallow versus deep feature extractors, particularly in the context of forest fire classification, which lacks dedicated pretrained models.

To optimize model performance, hyperparameter tuning was conducted on three primary parameters: batch size (16 and 32), input size (192×192 and 224×224), and optimizer (Adam and RMSprop). Batch size determines how many samples are processed at each training step [17], while input size defines the image dimensions provided to the CNN model. Optimizers, which help find the optimal model parameters by minimizing the loss function through gradient computation, significantly impact training speed and convergence [18]. Each optimizer also affects the learning speed and convergence of a model [19]. These hyperparameter combinations resulted in 8 unique scheme as shown in Table 1. Hyperparameter tuning is critical to finding the most effective configuration for achieving accurate and efficient classification of post-fire areas [20].

All models were trained for 50 epochs and other training parameters, such as learning rate and dropout rate, were kept at their default values to isolate the impact of the selected hyperparameters. Model training and evaluation were conducted on a local machine equipped with an AMD Ryzen 5 5600X CPU, 64 GB RAM, and a 512 GB SSD. The software environment included Python 3.8 along with essential libraries such as TensorFlow 2.13, Scikit-learn, Pillow, NumPy, Seaborn, Pandas, and Matplotlib.

Table 1. Hyperparameter scheme in building model

Scheme	Batch size	Input size	Optimizer	Scheme	Batch size	Input size	Optimizer
1	32	192×192	Adam	4	32	192×192	Adam
2	32	224×224	Adam	6	16	224×224	RMSProp
3	16	192×192	Adam	7	32	192×192	RMSProp
4	16	224×224	Adam	8	32	224×224	RMSProp

2.5. Evaluation and comparison

The performance of each model was evaluated through both quantitative metrics, training metrics, and visual assessment techniques. Quantitative metrics involved the calculation of accuracy, precision, recall, and F1-score based on the confusion matrix. These metrics provide a comprehensive assessment of classification performance, especially in scenarios involving class imbalance [21], [22]. Accuracy, in particular, was computed using in (1):

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively. To assess training dynamics and model generalization, training metrics such as training and validation accuracy, training time (in seconds), and model size (in megabytes) were also recorded. Visual analysis was conducted using confusion matrices to observe misclassification trends, along with accuracy and loss plots throughout training epochs to evaluate learning progress and detect potential overfitting or underfitting. The overall evaluation aimed to identify not only the best-performing CNN architecture but also the optimal combination of hyperparameters by considering both predictive performance and computational efficiency.

3. RESULTS AND DISCUSSION

3.1. Model training results

Based on the training process, a total of 56 CNN models were generated from 8 experiments conducted across 7 different architectures. The performance summary of each architecture was presented in Table 2, which includes metrics such as training accuracy, validation accuracy, training time, model size, and the corresponding hyperparameter combinations. Each architecture exhibited varying performance depending on the hyperparameter combinations applied.

Table 2. Summary of model performance

Architecture	Train acc	Val acc	Train loss	Val loss	Train Time (second)	Model size (mb)	Scheme
MobileNetV2	0.96	0.96	0.13	0.16	563	9	6
VGG16	0.98	0.88	0.07	0.34	569	129.7	2
VGG19	0.97	0.96	0.06	0.17	567	150.0	4
LeNet5	0.94	0.88	0.18	0.82	550	29.8	5
AlexNet	0.58	0.58	0.68	0.68	546	29.8	7
ResNet50	0.66	0.63	0.65	0.68	560	90.3	6
InceptionV3	0.94	0.92	0.23	0.32	552	83.7	7

From Table 2, it can be observed that VGG16, VGG19, and MobileNetV2 demonstrated the best performance among the seven CNN architectures evaluated. MobileNetV2 achieved the highest validation accuracy of 96%, with relatively fast training time (563 seconds) and the smallest model size (9 MB), making it highly efficient for deployment on resource-constrained devices. Meanwhile, VGG16 recorded the highest training accuracy at 98%, although its validation accuracy (88%) was lower than that of MobileNetV2 and VGG19. VGG19 also performed well, with a validation accuracy of 96% and a low validation loss value (0.17), despite its relatively large model size (150 MB). In contrast, architectures such as AlexNet and ResNet50 showed poor performance, with low validation accuracies of 58% and 63%. These results suggest that despite their architectural depth, these models (AlexNet and ResNet50) may be less effective in learning from the limited, ground-level imagery used in this study. Considering validation accuracy, training time efficiency, and model size, MobileNetV2 is the most balanced model, while VGG19 and VGG16 stand out in terms of accuracy. To further analyze model performance, training history plots and confusion matrices of the three models are presented. The performance of MobileNetV2, VGG19, and VGG16 is illustrated in Figures 5, 6, and 7.

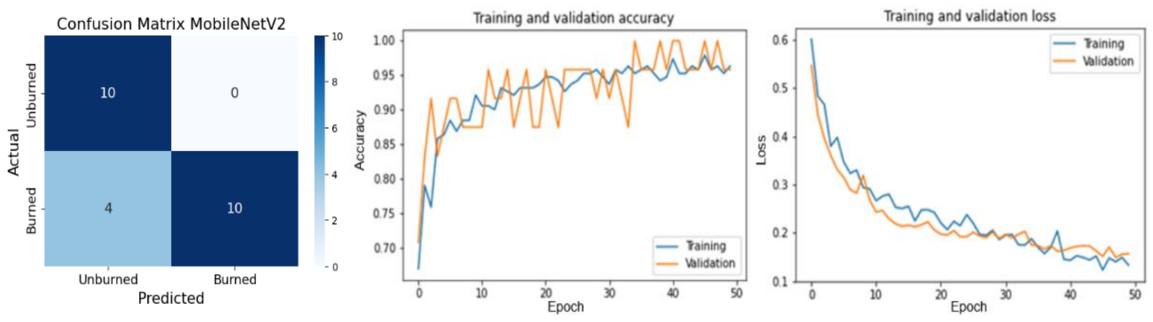

Figure 5. MobileNet V2 performance result

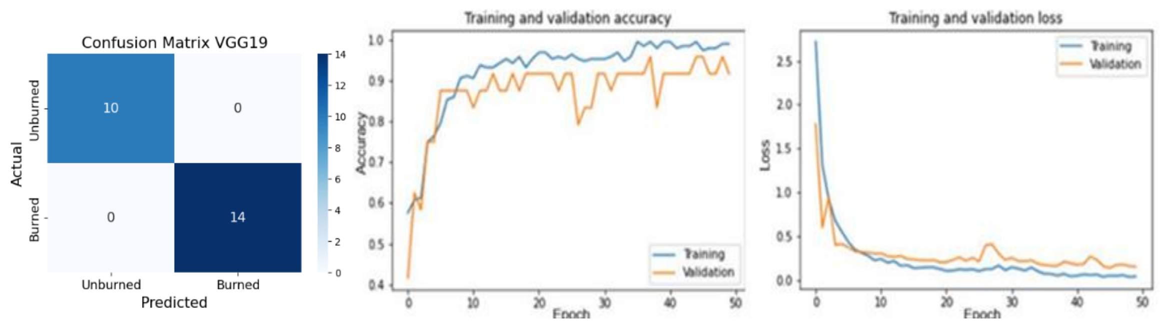

Figure 6. VGG-19 performance result

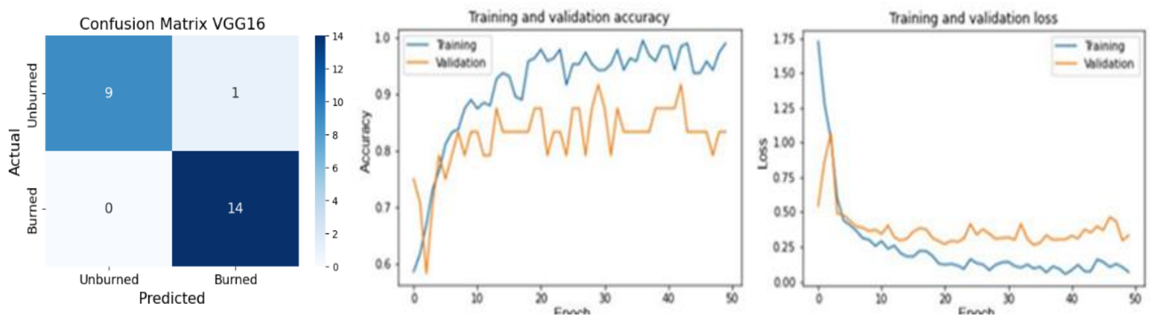

Figure 7. VGG-16 performance result

Figures 5, 6, and 7 present the training history and confusion matrices of MobileNetV2, VGG16, and VGG19. These visualizations indicate that all three models exhibited steadily increasing accuracy over epochs and decreasing loss curves, confirming stable learning behavior. However, MobileNetV2 shows more fluctuations during training, suggesting that its performance, although efficient, may be more sensitive to data variations or due to insufficient data. In summary, VGG19 produced the most accurate predictions, while MobileNetV2 offered the best trade-off between performance and efficiency. This reinforces prior findings [23] which state that MobileNetV2 was designed to balance accuracy and computational demands through hyperparameter flexibility, making it ideal for field-based, low-resource scenarios. These results indicate that deep learning models—especially lightweight architectures like MobileNetV2—can effectively distinguish between burned and unburned areas based on field imagery, supporting the studies of CNN applicability in post-fire assessment.

3.2. The effect of hyperparameters on model performance

In this study, several hyperparameters were tested on CNN models to evaluate their impact on performance. Each architecture was assigned a numerical label for ease of reference (1=MobileNetV2, 2=VGG16, 3=VGG19, 4=LeNet5, 5=AlexNet, 6=ResNet50, and 7=InceptionV3). The hyperparameters tested include batch size, input size, and optimizer. The batch sizes tested were 16 and 32, input sizes were 192×192 and 224×224, and the optimizers used were Adam and RMSProp. The following section discusses each hyperparameter in detail.

3.2.1. Batch size

Batch size affects several aspects of training, including convergence time, training stability, and the model's ability to generalize to unseen data. For instance, smaller batch sizes often allow faster computations but may require more iterations to converge compared to larger batch sizes [24]. To further explore the effect of batch size on model performance, the results of the experiments are presented in Table 3.

Based on Table 3, the use of different batch sizes had a significant impact on both validation accuracy and training time across various CNN architectures. With a batch size of 16, models such as MobileNetV2 (1) performed exceptionally well, achieving validation accuracies up to 1.00 in certain experiments. However, other models like VGG16 (2) and VGG19 (3) showed more variability, with validation accuracies tending to be lower (around 0.79). Conversely, a batch size of 32 generally produced more consistent validation accuracy. For instance, InceptionV3 (7) achieved near-perfect validation accuracy in several experiments. However, VGG16 and VGG19 showed a decrease in accuracy when using the larger batch size. In terms of training time, batch size 32 generally reduced the overall training time compared to batch size 16, as the model could process more data in a single iteration, thus requiring fewer iterations to complete the training process.

Nevertheless, smaller batch sizes (*e.g.*, 16) tended to result in higher validation accuracy, the cost of longer training time. For architectures like AlexNet and ResNet50, validation accuracy remained relatively low under both batch size settings, suggesting that batch size had less influence on these models within the context of this dataset. In summary, smaller batch sizes yielded better accuracy for certain models, whereas larger batch sizes were more efficient in terms of training time but could potentially compromise accuracy. This trend is consistent with findings from previous research [25], where smaller batch sizes (*e.g.*, 8) yielded higher accuracy compared to larger batch sizes (*e.g.*, 128).

Table 3. Model performance results for each batch size

Batch size	Scheme	Val accuracy							Training time (second)						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
16	3	0.88	0.83	0.83	0.75	0.58	0.63	0.96	547	553	544	545	549	553	550
	4	0.92	0.88	0.96	0.79	0.58	0.63	0.88	568	568	567	562	561	567	558
	5	0.96	0.92	0.88	0.88	0.58	0.58	0.88	547	549	549	550	548	554	554
	6	0.96	0.88	0.92	0.71	0.58	0.63	0.96	563	568	568	562	559	560	571
32	1	0.88	0.79	0.92	0.75	0.58	0.58	0.88	561	560	550	546	543	561	561
	2	1.00	0.88	0.92	0.79	0.58	0.63	0.96	548	569	553	534	543	551	546
	7	0.92	0.79	0.79	0.88	0.58	0.58	0.92	557	558	555	547	546	557	552
	8	1.00	0.79	0.67	0.79	0.58	0.58	0.96	556	560	546	539	526	542	541

3.2.2. Input size

Input size, or image resolution, influences the extent to which spatial features within an image can be effectively extracted by a model. To examine the impact of input size variations on model performance, experiments were conducted using input sizes of 192×192 and 224×224 pixels. This allowed for an assessment of how input size affects both validation accuracy and training time across different CNN architectures. The experimental results are summarized in Table 4. Overall, Table 4 shows that an input size of 224×224 tends to produce higher and more consistent validation accuracy compared to 192×192, particularly in architectures such as MobileNetV2, VGG16, and VGG19. For example, in MobileNetV2 (model 1), the highest validation accuracy of 1.00 was achieved with a 224×224 input size during experiment 8, whereas with 192×192 input size, the highest accuracy reached only 0.96 in experiment 5. This suggests that increasing image resolution enables the model to better recognize patterns and extract features. This is in line with findings from studies [26], which state that larger image resolutions tend to improve model performance, though they also increase computational time and resource consumption, leading to a trade-off between computational efficiency and recognition accuracy.

However, other studies have shown that increasing image size does not necessarily improve deep learning model performance, as it highly depends on the complexity of the images and the problem being solved. In general, increasing input size does not guarantee better accuracy; in some cases, smaller input sizes can yield better performance and vice versa. This is because each dataset may have an optimal input size that yields the best results, and accuracy can even decrease if image size exceeds a certain threshold [27]. In terms of training time, larger input sizes tend to increase training duration. This is observable across several architectures, where models trained on 224×224 inputs required more time than those trained on 192×192. This finding was consistent with [28], which notes that larger input sizes invariably lead to longer training times.

Table 4. Model performance results for each input size

Input size	Scheme	Val accuracy							Training time (second)						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
192×192	1	0.88	0.79	0.92	0.75	0.58	0.58	0.88	561	560	550	546	543	561	561
	3	0.88	0.83	0.83	0.75	0.58	0.63	0.96	547	553	544	545	549	553	550
	5	0.96	0.92	0.88	0.88	0.58	0.58	0.88	547	549	549	550	548	554	554
	7	0.92	0.79	0.79	0.88	0.58	0.58	0.92	557	558	555	547	546	557	552
224×224	2	1.00	0.88	0.92	0.79	0.58	0.63	0.96	548	569	553	534	543	551	546
	4	0.92	0.88	0.96	0.79	0.58	0.63	0.88	568	568	567	562	561	567	558
	6	0.96	0.88	0.92	0.71	0.58	0.63	0.96	563	568	568	562	559	560	571
	8	1.00	0.79	0.67	0.79	0.58	0.58	0.96	556	560	546	539	526	542	541

3.2.3. Optimizer

In this study, two types of optimizers were employed: Adam and RMSprop. Both are widely used in deep learning model training due to their adaptive learning rate capabilities. The impact of each optimizer on validation accuracy across different CNN architectures was examined to assess the consistency and optimization effectiveness during training. The performance results based on the optimizer used are presented in Table 5.

According to Table 5, RMSProp generally provided higher validation accuracy across several models, most notably MobileNetV2 and InceptionV3. For instance, MobileNetV2 reached a perfect accuracy of 1.0 under RMSProp in one training scheme, while it only reached up to 0.92 under Adam. InceptionV3 also performed consistently well with both optimizers, often achieving validation accuracy as high as 0.96. On the other hand, other architectures such as ResNet50 and AlexNet delivered lower performance, with validation accuracy typically ranging between 0.58 and 0.63, indicating their limitations in handling the complex textures and features present in burned field imagery.

In addition to accuracy, training time was also analyzed. Both optimizers showed comparable training durations, though RMSProp occasionally offered slightly shorter training times in models like ResNet50 and InceptionV3. Despite the small differences, these time savings could be beneficial when scaling up to large datasets or deploying models in resource-constrained environments. Among all schemes, the combination of RMSProp and MobileNetV2 proved to be the most effective, achieving perfect classification accuracy in just 556 seconds of training, suggesting an ideal balance of performance and efficiency. This aligns with findings from study [29] which highlights RMSProp as one of the best default optimizers due to its use of decay and momentum variables to optimize image classification accuracy.

Table 5. Model performance results for each optimizer

Optimzer	Scheme	Val accuracy							Training time (second)						
		1	2	3	4	5	6	7	1	2	3	4	5	6	7
Adam	1	0.88	0.79	0.92	0.75	0.58	0.58	0.88	561	560	550	546	543	561	561
	2	1.00	0.88	0.92	0.79	0.58	0.63	0.96	548	569	553	534	543	551	546
	3	0.88	0.83	0.83	0.75	0.58	0.63	0.96	547	553	544	545	549	553	550
	4	0.92	0.88	0.96	0.79	0.58	0.63	0.88	568	568	567	562	561	567	558
RMS	5	0.96	0.92	0.88	0.88	0.58	0.58	0.88	547	549	549	550	548	554	554
Prop	6	0.96	0.88	0.92	0.71	0.58	0.63	0.96	563	568	568	562	559	560	571
	7	0.92	0.79	0.79	0.88	0.58	0.58	0.92	557	558	555	547	546	557	552
	8	1.00	0.79	0.67	0.79	0.58	0.58	0.96	556	560	546	539	526	542	541

3.3. Discussion

The findings highlight that lightweight CNNs can be highly effective for image classification in constrained environments, such as post-fire field settings. Unlike satellite imagery, field images captured with mobile devices are flexible and cost-effective, yet remain underutilized in wildfire damage assessment. This study demonstrates that, with proper preprocessing and model selection, CNNs can achieve competitive results even with such ground-level data. Notably, our best models rivaled or exceeded reported performances from previous drone-based studies. For instance, VGG19's performance (96%) is comparable to VGG16 in [4], which used drone imagery with similar tasks. More importantly, MobileNetV2 model achieved 96% val accuracy—significantly outperforming the 77.7% reported in [8], which also used MobileNetV2 on field imagery from Jambi Province. This improvement may be attributed to the use of enhanced preprocessing, optimized hyperparameters, and more systematic training schemes. These findings reinforce the practical value of using mobile imagery for post-fire classification and demonstrate that careful model tuning can yield competitive, even superior, results compared to prior approaches using the same model architecture.

4. CONCLUSION

This study compared seven CNN architectures for classifying post-forest fire areas using mobile-captured field imagery and found that MobileNetV2, VGG16, and VGG19 achieved the best performance, with MobileNetV2 offering the best balance between accuracy, training time, and model size. The results highlight the potential of field imagery as a low-cost and flexible alternative to satellite or drone data for wildfire damage assessment. The study's novelty lies in its focus on mobile imagery and systematic evaluation of hyperparameters across multiple architectures. These findings support the development of lightweight, real-time tools for post-fire assessment and suggest future research should explore larger datasets, more hyperparameter tuning, and on-device deployment to enhance practical applications.

REFERENCES

[1] Aminah, C. Krah, and Perdinan, "Forest fires and management efforts in Indonesia (a review)," in *IOP Conference Series: Earth and Environmental Science*, vol. 504, no. 1, p. 012013, May 2020, doi: 10.1088/1755-1315/504/1/012013.

[2] E. Nuradi *et al.*, "Development of a mobile application for forest and land fire patrol using prototype method," in *IOP Conference Series: Earth and Environmental Science*, vol. 1177, no. 1, p. 012005, May 2023, doi: 10.1088/1755-1315/1177/1/012005.

[3] G. Zhang, M. Wang, and K. Liu, "Forest fire susceptibility modeling using a convolutional neural network for Yunnan Province of China," *International Journal of Disaster Risk Science*, vol. 10, no. 3, pp. 386–403, Sep. 2019, doi: 10.1007/s13753-019-00233-1.

[4] D. Hindarto, "Comparison accuracy of CNN and VGG16 in forest fire identification: a case study," *Journal of Computer Networks, Architecture and High Performance Computing*, vol. 6, no. 1, pp. 137–148, Dec. 2023, doi: 10.47709/cnahpc.v6i1.3371.

[5] S. Khan and A. Khan, "FFireNet: deep learning based forest fire classification and detection in smart cities," *Symmetry*, vol. 14, no. 10, p. 2155, Oct. 2022, doi: 10.3390/sym14102155.

[6] V. Hnatushenko, V. Hnatushenko, and D. Soldatenko, "Neural network-based analysis of forest fire aftermath in class-imbalanced remote sensing earth image classification," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLVIII-3–2, pp. 223–229, Nov. 2024, doi: 10.5194/isprs-archives-XLVIII-3-2024-223-2024.

[7] S. Ghouzlane, "Wildfire remote sensing applications," in *6th International Students Science Congress Proceedings Book*, Izmir International Guest Student Association, Sep. 2022, doi: 10.52460/issc.2022.027.

[8] A. Hidayat, I. S. Sitanggang, and L. Syaufina, "Classification of forest and land fire severity levels using convolutional neural network," in *BIO Web of Conferences*, Jakaria, Ed., Aug. 2024, p. 01030, doi: 10.1051/bioconf/202412301030.

[9] Y. Huang and A. Renaud, "The application of CNN-based image classification to wildfire early detection," *Journal of Student Research*, vol. 12, no. 4, Nov. 2023, doi: 10.47611/jsrhs.v12i4.5865.

[10] L. Syaufina and A. Abi Hamzah, "Changes of tree species diversity in peatland impacted by moderate fire severity at Teluk Meranti, Pelalawan, Riau Province, Indonesia," *Biodiversitas Journal of Biological Diversity*, vol. 22, no. 5, May 2021, doi: 10.13057/biodiv/d220555.

[11] K. Zhou, S.-K. Oh, W. Pedrycz, and J. Qiu, "Data preprocessing strategy in constructing convolutional neural network classifier based on constrained particle swarm optimization with fuzzy penalty function," *Engineering Applications of Artificial Intelligence*, vol. 117, p. 105580, Jan. 2023, doi: 10.1016/j.engappai.2022.105580.

[12] P. R. Togatorop, Y. Pratama, A. Monica Sianturi, M. Sari Pasaribu, and P. Sangmajadi Sinaga, "Image preprocessing and hyperparameter optimization on pretrained model MobileNetV2 in white blood cell image classification," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 12, no. 3, p. 1210, Sep. 2023, doi: 10.11591/ijai.v12.i3.pp1210-1223.

[13] J. H. Yousif and H. A. Kazem, "Prediction and evaluation of photovoltaic-thermal energy systems production using artificial neural network and experimental dataset," *Case Studies in Thermal Engineering*, vol. 27, p. 101297, Oct. 2021, doi: 10.1016/j.csite.2021.101297.

[14] M. A. I. Fahim and S. A. Tumpa, "Image augmentation techniques: enhancing deep learning performance," *ResearchGate*, pp. 1–7, 2023.

[15] R. A. de Oliveira and M. H. J. Bollen, "Deep learning for power quality," *Electric Power Systems Research*, vol. 214, p. 108887, Jan. 2023, doi: 10.1016/j.epsr.2022.108887.

[16] M. Imad, O. Doukhi, and D.-J. Lee, "Transfer learning based semantic segmentation for 3D object detection from point cloud," *Sensors*, vol. 21, no. 12, p. 3964, Jun. 2021, doi: 10.3390/s21123964.

[17] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, no. 4, pp. 312–315, Dec. 2020, doi: 10.1016/j.icte.2020.04.010.

[18] R. Zaheer and H. Shaziya, "A Study of the optimization algorithms in deep learning," in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, IEEE, Jan. 2019, pp. 536–539, doi: 10.1109/ICISC44355.2019.9036442.

[19] S. H. Haji and A. M. Abdulazeez, "Comparison of optimization techniques based on gradient descent algorithm: a review," *PalArch's Journal of Archaeology of Egypt / Egyptology*, vol. 18, no. 4, pp. 2715–2743, 2021.

[20] F. Ucar and D. Korkmaz, "COVIDiagnosis-Net: deep Bayes-SqueezeNet based diagnosis of the coronavirus disease 2019 (COVID-19) from X-ray images," *Medical Hypotheses*, vol. 140, p. 109761, Jul. 2020, doi: 10.1016/j.mehy.2020.109761.

[21] S. Yang and G. Berdine, "Confusion matrix," *The Southwest Respiratory and Critical Care Chronicles*, vol. 12, no. 53, pp. 75–79, Oct. 2024, doi: 10.12746/swrccc.v12i53.1391.

[22] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[23] K. Dey, M. M. Hassan, M. M. Rana, and M. H. Hena, "Bangladeshi indigenous fish classification using convolutional neural networks," in *2021 International Conference on Information Technology (ICIT), IEEE*, Jul. 2021, pp. 899–904, doi: 10.1109/ICIT52682.2021.9491681.

[24] P. M. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," *Information Technology and Management Science*, vol. 20, no. 1, Jan. 2017, doi: 10.1515/itms-2017-0003.

[25] K. Kurniawan, A. Perdana Windarto, and S. Solikhun, "Refining CNN architecture for forest fire detection: improving accuracy through efficient hyperparameter tuning," *Bulletin of Electrical Engineering and Informatics*, vol. 14, no. 2, pp. 1202–1211, Apr. 2025, doi: 10.11591/eei.v14i2.8805.

[26] S. Tang *et al.*, "The effect of image resolution on convolutional neural networks in breast ultrasound," *Heliyon*, vol. 9, no. 8, p. e19253, Aug. 2023, doi: 10.1016/j.heliyon.2023.e19253.

[27] M. Balaji and R. Joseph, "Impact of image size on accuracy and generalization of convolutional neural networks," *International Journal of Research and Analytical Reviews*, vol. 6, no. 1, pp. 70–80, 2019.

[28] S. Saponara and A. Elhanashi, "Impact of image resizing on deep learning detectors for training time and model performance," 2022, pp. 10–17, doi: 10.1007/978-3-030-95498-7_2.

[29] P. Verma, V. Tripathi, and B. Pant, "Comparison of different optimizers implemented on the deep learning architectures for COVID-19 classification," *Materials Today: Proceedings*, vol. 46, pp. 11098–11102, 2021, doi: 10.1016/j.matpr.2021.02.244.

BIOGRAPHIES OF AUTHORS

Ahmad Bintang Arif is currently pursuing his master's degree in computer science at IPB University, building upon his bachelor's degree from the same institution. With a focus on data mining and machine learning, he is actively engaged as a lab assistant for the data mining course, contributing to both his own learning and that of his peers. He can be contacted at email: ahmadbintangarif@apps.ipb.ac.id.

Imas Sukaesih Sitanggang received a Ph.D. degree in computer science from the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, in 2013. She is a lecturer in School of Data Science, Mathematics and Informatics, IPB University, Indonesia. Her main research interests include spatial data mining and smart agriculture. She can be contacted at email: imas.sitanggang@apps.ipb.ac.id.

Hari Agung Adrianto is a professor at IPB University. He holds a Ph.D. student in School of Earth and Environment at University of Leeds, a master's degree in computer science at the IPB University and a bachelor's degree in computer science at IPB University. His research areas are in the fields of data mining, data warehousing and spatial data processing. He can be contacted at email: agung@apps.ipb.ac.id.

Lailan Syaufina is a professor in forest protection with forest fire as her major at Department of Silviculture Faculty of Forestry and Environment IPB. Her undergraduate from Bogor Agricultural University (IPB), master degree from Georg August University Germany and Ph.D. from Universiti Putra Malaysia. Her field of interest including forest fire in the aspects of fire severity assessment, fire management, peatland fire, fire-biodiversity, fire-climate, and fire-emission. She can be contacted at email: lailans@apps.ipb.ac.id.