

Forest Fire Detection Based on Digital Imagery Using Convolutional Neural Network (CNN) Model

Candra Gudiato^{1✉}, Aditya Pratama², Christian Cahyaningtyas³

^{1,3} Information System, Institut Shanti Bhuna, Indonesia

² Information System, Faculty of Engineering, Universitas Nahdlatul Ulama Kalimantan Barat, Indonesia

✉ **Corresponding Author** : Candra Gudiato (e-mail: candra.gudiato@shantibhuana.ac.id)

Article Information	ABSTRACT
Article History Received : March 10, 2026 Revised : March 20, 2026 Published : April 10, 2026	This study explores the implementation of a Convolutional Neural Network (CNN) for automated forest fire identification using digital image processing. Utilizing the USTC 'Forest Fire' dataset, the research framework involved systematic data preprocessing, including a 70:30 training-validation split and the application of image augmentation techniques to enhance model robustness. The proposed architecture features a sequential design with dual convolution and pooling layers, integrated with ReLU and Sigmoid activations. Although initial training over seven epochs yielded a deceptive validation accuracy of 99%, granular performance analysis exposed critical limitations. Evaluation via a Confusion Matrix revealed that while the model excelled at identifying 'non-fire' scenarios, it struggled significantly with actual fire detection, failing to recognize 301 out of 331 fire instances. These results highlight a severe class imbalance issue, suggesting that standard accuracy metrics are insufficient for this application and emphasizing the need for more balanced sampling or advanced architectural adjustments in future fire detection systems.
Keywords: <i>Detection; Digital Imagery; Convolutional Neural Network (CNN);</i>	

INTRODUCTION

Forest and land fires are one of the environmental issues that frequently occur in Indonesia and even globally (Saharjo & Hasanah, 2023). The National Disaster Management Authority (BNPB) reported that there were 4,938 natural disasters in Indonesia from January 1, 2023, to December 31, 2023. Out of this number, forest and land fires were the most frequent natural disasters, with 1,802 occurrences or 36.5% of the total (Komalasari, 2025). According to data released by the Ministry of Environment and Forestry (KLHK) during the January - August 2023 period, the indicated area of forest and land fires in Indonesia reached 267,935.59 hectares (ha). The three provinces with the largest burned areas include West Kalimantan with 54,402 ha, East Nusa Tenggara with 50,396 ha, and West Nusa Tenggara with 26,453 ha (Ahdiat, 2023).

Forest fires do not only have a negative impact on human health due to the smoke produced. They also cause other losses such as loss of habitat for flora and fauna, reduction in biodiversity, climate change, drought, disruption of the water cycle, and economic losses (Pasai, 2020). Occasionally, the mitigation of forest fire disasters cannot be handled immediately due to the lack of rapid early detection, resulting in information being received several days after the incident (Firmansyah et al., 2019). Therefore, this research aims to develop an effective model for detecting forest fires as early as possible, approaching real-time, so that mitigation steps by relevant authorities, especially the government, can promptly decide on the policies to be taken. The model to be developed in the detection system uses Convolutional Neural Networks (CNN).

A study by Zheng et al. (Zheng et al., 2022) addresses the latency issues inherent in existing detection systems by employing advanced CNN architectures for forest fire smoke identification. Within this framework, CNN functions as the core image processing engine for early detection, designed for seamless integration with on-site monitoring hardware. Their findings demonstrate that specialized algorithms, such as EfficientDet and YOLOv3, outperform traditional methods by achieving 95.7% accuracy and a processing speed of 27 FPS, thereby validating their efficacy for real-time early warning applications

The CNN method is quite popular among deep learning users because it extracts features from image inputs and then reduces the image dimensions without changing its characteristics (Barmpoutis et al., 2020). Within the framework of forest fire detection, Ghosh and Kumar (Ghosh & Kumar, 2022) introduced a hybrid Deep Learning architecture integrating Convolutional Neural Networks (CNN) with Recurrent Neural Networks (RNN). The implementation of CNN is pivotal for its capacity to hierarchically extract complex spatial features—including flame texture, smoke patterns, and color distribution—from low-level to high-level representations.

Subsequently, the architecture incorporates RNN to capture the temporal dependencies of fire progression. This hybrid approach has demonstrated superiority over other state-of-the-art methods, achieving exceptional classification accuracy on both Mivia Lab and Kaggle datasets. These results further validate that CNN-based visual feature extraction is highly robust in identifying the distinct characteristics of fire-related materials. To address the requirements of this study, a custom Sequential CNN architecture is developed. This model is specifically designed with dual convolution and pooling layers, optimized to extract visual indicators such as smoke opacity, flame texture, and color distribution from 64 x 64-pixel imagery. The use of a custom architecture allows for a more streamlined and computationally efficient process, which is critical for achieving rapid early detection in Indonesian forest environments

Research on the impacts of forest fire disasters was presented in a study conducted by Afdhal Amri et al. titled “The Impact of Forest Fire Disasters on the Environment and Mitigation Efforts in Indonesia.” The study describes the impacts of forest fires and post-fire recovery efforts (Amri et al., 2024). Meanwhile, research on early detection systems for forest fires was previously conducted by I’zaaz Irfanianingrum et al. in a study titled “Fuzzy Logic-Based Forest Fire Early Detection System.” The results of the research produced an IoT tool based on fuzzy logic capable of early forest fire detection (Irfanianingrum et al., 2023). Reis and Turk (Reis & Turk, 2023) further advanced the implementation of Convolutional Neural Networks (CNN) for forest fire detection by employing a transfer learning approach with DenseNet121, InceptionV3, and ResNet50V2 architectures. Their study leveraged Unmanned Aerial Vehicle (UAV) imagery to address detection challenges in inaccessible terrains. The findings indicate that utilizing pre-trained models with ImageNet weights yields superior accuracy compared to training from scratch; notably, the DenseNet121 model achieved an optimal accuracy of 99.32%. Building upon the aforementioned background and identifying the existing research gaps, this study proposes a forest fire detection system using an optimized CNN approach. The primary contribution of this research lies in three key aspects: first, the utilization of a specialized forest fire dataset to evaluate detection accuracy across diverse environmental scenarios; second, the development of a lightweight model architecture to ensure computational efficiency; and third, the optimization for real-time deployment on low-power monitoring devices. These enhancements address the limitations of previous high-latency models, providing a more practical solution for immediate fire mitigation in vast forest areas.

RESEARCH METHODS

The research workflow follows the process shown in Figure 1.

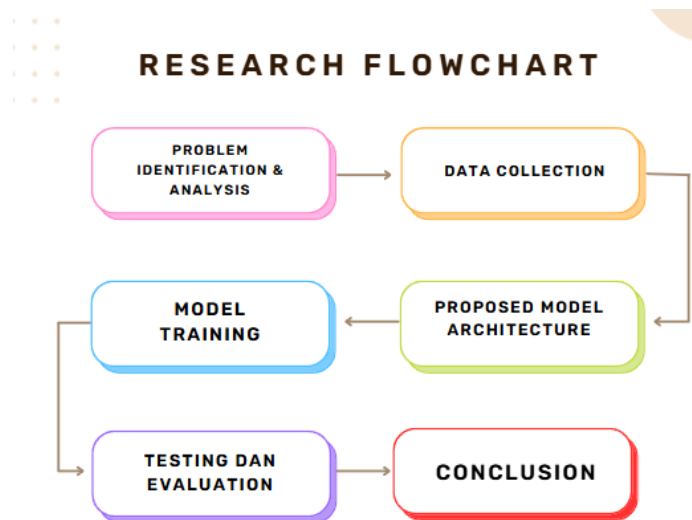


Figure 1. Research Flowchart

Based on Figure 1 above, the research begins with the analysis and formulation of the problems described in the background. Subsequently, data collection supporting the research is conducted, including the preparation of the tools to be used. In this study, the data is sourced from the USTC 'Forest Fire' dataset, which contains 4,121 data records. The tools used are TensorFlow with the Keras Application Programming Interface (API).

TensorFlow is a framework developed by the Google Brain Team in 2015 for numerical computation. Currently, TensorFlow is widely used for developing AI applications, such as word embedding, AI models, and chatbots. TensorFlow provides an interface that can express ML algorithms and applications to execute them, supporting Convolutional Neural Network modeling and others. TensorFlow is based on how tensors flow throughout the network, where a Tensor is a type of multidimensional array (Wiranda et al., 2020). Simply put, the architecture of TensorFlow is shown in Figure 2.



Figure 2. TensorFlow Architecture

Keras is a high-level open-source application programming interface (API) built using the Python language. Keras is specifically designed to build and train deep learning models quickly and easily. Keras was developed by François Chollet in 2015 and is now the official API for TensorFlow (Joseph et al., 2021).

The next stage is the design of the model architecture, which in this study will use a Convolutional Neural Network (CNN). A Convolutional Neural Network (CNN) is a type of neural network specifically designed to handle image processing problems (Subasri & Nithya, 2025). CNN is an effective method for classifying, identifying, and recognizing patterns within digital imagery (Shorten & Khoshgoftaar, 2019). CNN is capable of understanding image details better because it has an architecture that aligns with how the human brain processes visual information (Cahyaningtyas et al., 2025). The data used in CNN is two-dimensional, such as images or sound, and it utilizes convolution operations in matrices as well as weights in four-dimensional forms, which are a set of convolutional kernels. Due to the nature of the convolutional process, CNN can only be used on data with a two-dimensional structure (Anhar & Putra, 2023).

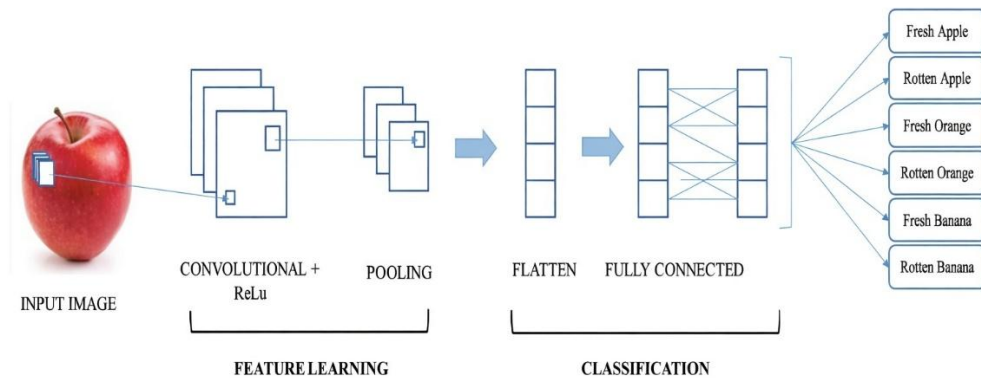


Figure 3. CNN Process

In Figure 3 above, there are essentially two main parts in the CNN process: feature learning and classification. In the feature learning section, three processes are executed:

The CNN process consists of two primary sections: feature learning, involving convolutional layers for extraction, ReLU activation for non-linearity, and pooling layers to reduce spatial dimensions; and classification, which employs a flatten layer to convert feature maps into a vector for the fully connected layer, finalized by a sigmoid or softmax function. Model performance is evaluated using a confusion matrix to categorize results into True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). From these components, four key metrics are derived as shown in equations (1) to (4): Accuracy for overall correct predictions, Precision for relevance, Recall for positive detection capability, and the F1-Score as the harmonic mean, which is critical for assessing models with class imbalances (Paraijun et al., 2022).

Once the model architecture design is complete, the model architecture training will be conducted. The training process is performed using the Adam (Adaptive Moment Estimation) optimization algorithm, chosen for its ability to adaptively adjust the learning rate for each parameter, making the convergence process more stable and efficient (Zuhri et al., 2025). The completed model architecture will then proceed to the testing and model evaluation stage.

		Actual Values	
		1 (Positive)	0 (Negative)
Predicted Values	1 (Positive)	TP (True Positive)	FP (False Positive) <i>Type I Error</i>
	0 (Negative)	FN (False Negative) <i>Type II Error</i>	TN (True Negative)

Figure 4. Confusion Matrix

The performance of a classification model can be evaluated through the use of a confusion matrix, as shown in Figure 4 above. This matrix contains several prediction outcome categories: correct predictions (True Positive/TP), prediction errors for actual positive data (False Negative/FN), incorrect positive predictions (False Positive/FP), and correct predictions for negative data (True Negative/TN).

The evaluation process is conducted using a test set that is distinct from the training and validation data. The primary metric used is accuracy, which represents the percentage of correct predictions relative to the total predictions. Furthermore, the confusion matrix provides a more comprehensive analysis of model performance by displaying the distribution of correct and incorrect predictions for each category. It consists of four main components: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). TP and TN indicate accurate predictions for the fire and non-fire categories (formerly messy and clean), while FP and FN represent classification errors. Through confusion matrix analysis, model error patterns can be identified, for instance, in images that are difficult to classify due to ambiguous visual elements (Mahameru et al., 2025).

Accuracy indicates how often the model generates correct predictions relative to the total tested data. Accuracy is the ratio of correct predictions (positive and negative) to the total data. The accuracy equation is shown in equation (1).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

The Precision metric describes the extent to which the positive predictions provided by the model are truly relevant. A high precision value indicates that the model rarely gives false positive predictions. This metric is crucial in cases where false positive errors have a significant impact. Precision is the ratio of true positive predictions compared to the total predicted positive results. The precision equation is shown in equation (2).

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

The Recall metric assesses the extent to which the model is able to detect all positive data without omission. A high recall value signifies that the model successfully identifies most of the positive data. This metric is highly relevant when missing a single positive data point is considered a major error. Recall measures how well the model identifies all samples that are actually positive. The recall equation is shown in equation (3).

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

The F1-Score metric functions to evaluate the overall model performance by considering two main aspects: precision and completeness. A high F1-Score indicates that the model is able to maintain a balance between detection capability and prediction accuracy. This metric is very useful when there is a class imbalance, as it considers both false positives and false negatives. F1-Score is the harmonic mean of precision and recall, providing a balance between these two matrices. The F1-Score equation is shown in equation (4).

$$F1score = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (4)$$

In evaluating CNN models, the selection of metrics must be adapted to the task objectives and data characteristics. For image classification, a combination of metrics such as accuracy, precision, recall, and F1-Score is frequently used. Each type of metric has different strengths and weaknesses according to the context of the data used. Proper analysis of evaluation results can help significantly improve model performance [13]. Once the model evaluation is complete, the process continues with drawing conclusions from the research findings.

RESULTS AND DISCUSSION

A. Model Architecture

The Convolutional Neural Network (CNN) model used in this study was developed utilizing the TensorFlow framework and Keras API. The model was designed using a Sequential architecture, ensuring that each layer is arranged linearly according to the data processing flow.

This architecture is configured to receive an input in the form of 64×64 pixel images with three color channels (RGB), serving as the standard input throughout the training and testing processes.

The feature extraction stage begins with the first convolutional layer, which applies 32 filters of size 7×7 with the ReLU activation function. This layer functions to identify basic visual features such as edges, textures, and other local patterns. The output from the convolutional layer is then reduced through a 2×2 max pooling layer with a stride of 2. This pooling process significantly reduces the spatial dimensions of the feature representation, thereby lowering computational complexity while increasing the model's robustness against minor translational variations in the images.

The extraction process continues with a second convolutional layer that also utilizes 32 filters, but with a smaller kernel size of 3×3. This more compact kernel size allows the model to detect more complex patterns based on the features generated in the previous layer. This layer is again followed by 2×2 max pooling to maintain computational efficiency and simplify the feature representation before entering the classification stage.

The output from the feature extraction process is then transformed into a one-dimensional vector through a Flatten layer. This vector is processed in a dense layer with 256 neuron units and the ReLU activation function, which plays a role in learning the relationships and abstract patterns relevant to the prediction process. The final stage of the model consists of a single output neuron with a sigmoid activation function. The use of sigmoid activation aligns with the characteristics of binary classification problems, as it allows the model to produce probability values between 0 and 1, representing the likelihood of fire presence in the input image.

The model was compiled using the Adam optimization algorithm, known for its effectiveness in performing adaptive weight adjustments during the training process. The loss function used is binary cross-entropy, consistent with the binary classification objective, while accuracy serves as the primary metric to evaluate model performance. A complete summary regarding the parameters of each layer, the number of trainable parameters, and the overall structure of the model is presented in Figure 5.

```
print('Output untuk Tabel 4.1 (cnn.summary()):')
cnn.summary()
```

Output untuk Tabel 4.1 (cnn.summary()):
Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 56, 56, 32)	4,736
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 32)	0
conv2d_3 (Conv2D)	(None, 27, 27, 32)	9,248
max_pooling2d_3 (MaxPooling2D)	(None, 13, 13, 32)	0
flatten_1 (Flatten)	(None, 5408)	0
dense_2 (Dense)	(None, 256)	1,384,704
dense_3 (Dense)	(None, 1)	257

Total params: 4,196,837 (16.01 MB)
Trainable params: 1,384,945 (5.34 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2,797,892 (10.67 MB)

Figure 5. Model Structure

The data preparation stage is a crucial step in ensuring that the CNN model obtains structured, representative, and suitable data for the training and evaluation process. In this research, data was obtained from the Forest Fire dataset available through Kagglehub. The entire data preparation process includes downloading, directory organization, dataset splitting, as well as the application of pre-processing and augmentation.

The initial step involved downloading the original dataset and storing it in a temporary directory. To facilitate processing, the entire contents of the source directory were then copied to

a new local directory (./data). Before copying, the destination directory was cleared if it already existed to ensure a consistent workspace. At this stage, the test_big and test_small directories from the original dataset were excluded because this study only required one primary data set to be re-divided into training and validation subsets.

Once the data was available in the local directory, the dataset splitting process was conducted using the splitfolders library with a 70:30 ratio for training and validation data. The results were stored in the ./splitted_data directory, consisting of two subdirectories: train and val. Based on this process, the training set contains 9,612 images divided into two classes: train_smoke and train_fire. Meanwhile, the validation set contains 4,121 images with the same class division. This split ensures that the model obtains adequate data variation during the training phase and can be representatively evaluated through validation data.

The next stage is the pre-processing and data augmentation process. Pre-processing was performed to normalize pixel values with a $1/255$ scale, so all images fall within the $[0,1]$ range. This normalization is a standard step in neural network training to accelerate convergence and improve the stability of the optimization process. To enhance the model's generalization capability and reduce the risk of overfitting, data augmentation was applied to the training set. The augmentation techniques used include shear transformation, zoom scale changes, and horizontal flipping. All images were then resized to 64×64 pixels and processed in batches of 32 with a binary class mode, according to the characteristics of two-class classification.

Conversely, for the validation set, only the normalization process was applied without augmentation. This approach was taken to ensure that the model evaluation reflects performance on unmodified data, making the evaluation results closer to real-world scenarios.

B. Model Training Process

Once the CNN model architecture was established, the model was trained using the training dataset and evaluated in parallel on the validation dataset. The training process utilized the Adam optimization algorithm, chosen for its capability to adaptively adjust the learning rate for each parameter, resulting in a more stable and efficient convergence process. The loss function employed was binary cross-entropy, consistent with the characteristics of the binary classification problem in detecting the presence of fire. Accuracy metrics were used throughout the training process to monitor the proportion of predictions matching the actual labels.

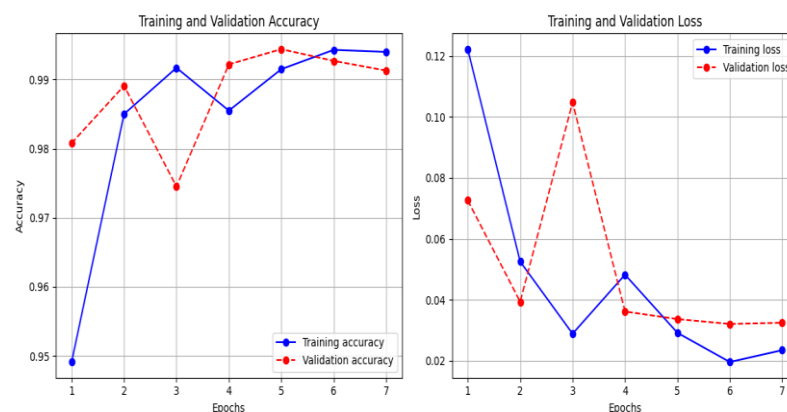


Figure 6. Accuracy and Loss Progression During CNN Model Training

The model was trained for seven epochs. In each epoch, the model performed forward and backpropagation across the entire training dataset, subsequently evaluating its performance on the validation data. This procedure enabled the early identification of potential overfitting or underfitting symptoms by observing the patterns of accuracy and the resulting loss values. The training summary indicates that the model achieved consistently high performance on both datasets, with training accuracy increasing from 0.9491 in the first epoch to 0.9940 in the seventh epoch. Meanwhile, the validation accuracy also showed stable values in the range of 0.98 to 0.99,

with the highest value of 0.9944 occurring in the fifth epoch. The loss values for both sets tended to decrease, although a slight increase in validation loss to 0.1048 was observed in the third epoch, indicating minor fluctuations during the learning process. Nevertheless, these fluctuations did not persist, and performance stabilized in subsequent epochs.

Visually, the progression of accuracy and loss values during the training process is displayed in Figure 6. The accuracy graph shows a consistent upward trend for both training and validation data, while the loss graph displays a relatively stable downward pattern. The combination of these two graphs indicates that the model was able to learn the data patterns effectively without showing significant symptoms of overfitting.

C. Model Evaluation Results

Upon completion of the training process, the next step was to evaluate the model's performance to assess its generalization capability on data not used during training. In this study, the evaluation was conducted using the validation dataset, which also served as the testing data, given that the split process only yielded two subsets: train and val. The model evaluation was performed using several metrics commonly used in binary classification problems to obtain a comprehensive overview of prediction performance.

The primary metric used was accuracy, representing the proportion of predictions matching the actual labels. While accuracy provides a general understanding of model performance, a more in-depth analysis is required to evaluate the detailed relationship between predictions and actual labels. Therefore, a confusion matrix was used to display the distribution of model predictions into four categories: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). This matrix provides richer information regarding the model's prediction error patterns and serves as the basis for calculating other derivative metrics.

Additionally, a classification report is presented to calculate the precision, recall, and F1-score for each class. Precision measures the proportion of positive predictions that are truly positive, while recall measures the proportion of actual positive cases successfully identified. The F1-score, as the harmonic mean of these two metrics, provides an indication of balanced performance, particularly in conditions with uneven class distribution. These three metrics help assess whether the model is biased toward a specific class or possesses uniform classification capabilities.

```
Output untuk Tabel 4.2 (Matriks Konfusi Model CNN):
[[3515  275]
 [ 301   30]]
```

Figure 7. Model Confusion Matrix

Based on the evaluation results, the model demonstrated a validation accuracy of 0.9940, indicating that it can classify fire and non-fire images with a very high level of precision. Detailed information regarding the prediction distribution can be seen in the Confusion Matrix presented in Figure 7, while the precision, recall, and F1-score metrics for each class are shown in Figure 8.

```
Output untuk Tabel 4.3 (Laporan Klasifikasi Model CNN):
precision    recall  f1-score   support

train-smoke      0.92      0.93      0.92      3790
train_fire       0.10      0.09      0.09        331

accuracy          0.92      0.92      0.92      4121
macro avg         0.51      0.51      0.51      4121
weighted avg      0.86      0.86      0.86      4121
```

Figure 8. CNN Model Classification Report

Furthermore, to obtain additional perspective on the model's ability to probabilistically distinguish between the two classes, the ROC (Receiver Operating Characteristic) curve and AUC

(Area Under the Curve) value were also analyzed. The ROC curve illustrates the relationship between the true positive rate and false positive rate at various decision thresholds, while the AUC provides a quantitative measure of the quality of separation between classes. The closer the value is to 1, the better the model's ability to consistently distinguish between positive and negative classes. The visualization of the ROC curve and AUC value can be seen in Figure 9 below.

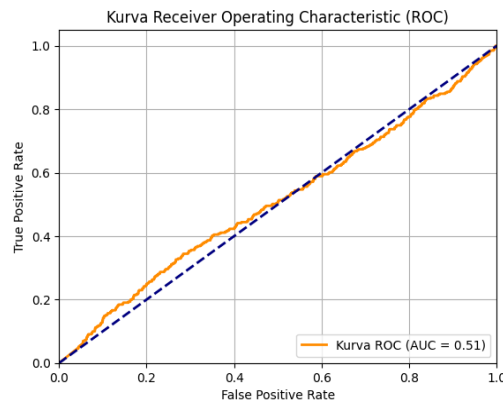


Figure 9. ROC Curve and AUC Value

D. Discussion

Based on the results presented in the previous section, the developed Convolutional Neural Network (CNN) model shows performance that still requires significant improvement, particularly in the task of identifying forest fire images. Although the overall accuracy appears high, a more in-depth evaluation reveals that the model has not been able to effectively generalize visual patterns of fire. This finding is crucial to understand comprehensively, as it directly impacts the feasibility of implementing the model in a real-world fire detection system.

At the training stage, the model demonstrated stable performance. Training and validation accuracy consistently increased, while the loss value decreased without showing signs of overfitting. The validation results in the final epoch reached an accuracy of approximately 0.9913 with a loss value of 0.0324, which, at first glance, indicates the model's ability to learn the patterns within the dataset. The training graphs further reinforce that the learning process was controlled and balanced between the training and validation data.

However, the interpretation of the evaluation results on the validation set provides a different perspective. The confusion matrix indicates that the model encountered significant difficulty in detecting images containing fire. Out of 331 actual fire images, only 30 were successfully identified as fire, while the other 301 images were misclassified as non-fire. This type of error is categorized as critical because False Negatives in the context of fire detection can lead to undetected fire incidents. Additionally, 275 non-fire images were incorrectly predicted as fire, resulting in relatively high False Positives. This imbalance suggests that the model more easily recognizes patterns associated with the absence of fire but fails to consistently capture the visual characteristics representing the presence of fire.

The classification report further confirms the performance disparity between classes. The non-fire image class showed excellent precision and recall, approaching 0.92 and 0.93, respectively. Conversely, performance for the fire image class was very low, with a precision of only 0.10 and a recall of 0.09. The extreme difference between these two classes indicates that the model is heavily biased toward the majority class (non-fire), which dominates the number of samples in the dataset. The overall accuracy of 0.86 is actually misleading as it is inflated by the model's performance on the majority class. Furthermore, the Area Under the Curve (AUC) value of 0.51 indicates that the model's ability to distinguish between the two classes is nearly equivalent to a random guess, showing a lack of adequate discriminative capability.

The model's primary strength lies in its ability to recognize images that do not contain fire. The model appears capable of learning patterns and features from this class effectively, resulting

in high precision and recall. Additionally, the training process was stable and synchronized between training and validation, indicating that the model did not experience overfitting despite the relatively few training epochs. Nevertheless, these strengths cannot compensate for the model's primary weakness: its failure to detect fire-containing images. The substantial class imbalance is a dominant factor hindering the model from learning patterns within the minority class. The model's limited ability to distinguish visual fire patterns is also reflected in the low AUC value, suggesting that the model has not yet grasped the key features related to the presence of fire.

The implications of these findings are significant. In a real-world context, such a model cannot be relied upon for forest fire detection systems, primarily due to the high number of False Negatives that could lead to missed fire events. This demonstrates that accuracy alone is an insufficient metric for model evaluation, especially when the dataset has an imbalanced class distribution. Consequently, metrics such as precision, recall, F1-score, and AUC must be the primary references, as they provide a more realistic assessment of the model's capabilities.

CONCLUSION

Although the developed CNN model achieved a high validation accuracy of approximately 0.99, an in-depth evaluation reveals significant weaknesses in detecting 'fire' images due to severe class imbalance. With a very low F1-score (0.10) and an AUC value of 0.51, the model exhibits a critical False Negative rate—detecting only 30 out of 331 fire cases—making it impractical for real-world forest fire detection. Future improvements must prioritize addressing this imbalance through techniques such as oversampling, class-weighted loss, or implementing Transfer Learning with advanced architectures like ResNet or EfficientNet to enhance minority class recognition. Additionally, collecting a more balanced and representative dataset is essential to improve the model's generalization and reliability for image-based fire detection systems.

ACKNOWLEDGEMENTS

The author would like to express gratitude to all parties involved who cannot be mentioned individually for providing both material and non-material support towards the completion of this research.

REFERENCES

- Ahdiat, A. (2023). *No Title*. <https://databoks.katadata.co.id/datapublish/2023/09/20/luas-kebakaran-hutan-indonesia-capai-267-ribu-hektare-sampai-agustus-2023>
- Amri, A., Akhirman, Zaharani, A., Rizki, C., Harianja, L. M., Prameswari, N., Vauzia, & Putri, W. N. (2024). Dampak bencana kebakaran hutan terhadap lingkungan dan upaya penanggulangan di indonesia. *JRTI (Jurnal Riset Tindakan Indonesia)*, 9(2), 159–166.
- Anhar, & Putra, R. A. (2023). Perancangan dan Implementasi Self-Checkout System pada Toko Ritel menggunakan Convolutional Neural Network (CNN). *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi & Teknik Elektronika*, 11(2), 466–478.
- Azmi, K., Defit, S., & Sumijan. (2023). Implementasi Convolutional Neural Network (CNN) Untuk Klasifikasi Batik Tanah Liat Sumatera Barat. *Jurnal Unitek*, 16(1), 2580–2582.
- Barmpoutis, P., Papaioannou, P., Dimitropoulos, K., & Grammalidis, N. (2020). A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing. *Sensors*, 20(6442), 1–26.
- Cahyaningtyas, C., Mira, Gudiato, C., & Sari, M. (2025). Deteksi Ekspresi Wajah Manusia Menggunakan Metode Convolutional Neural Network. *Jurnal FASILKOM*, 15(1), 138–145.
- Firmansyah, F., Budianto, E., Yulianto, A., Sudrajat, B., & Wigandi, D. P. (2019). Rancang Bangun Sistem Informasi Bank Sampah (SIBAS) Berbasis Desktop Dengan Metode Waterfall. *REMIK (Riset Dan E-Jurnal Manajemen Informatika Komputer)*, 4(1), 44.

- <https://doi.org/10.33395/remik.v4i1.10229>
- Ghosh, R., & Kumar, A. (2022). A hybrid deep learning model by combining convolutional neural network and recurrent neural network to detect forest fire. *Multimedia Tools and Applications*, 81(27), 38643–38660. <https://doi.org/10.1007/s11042-022-13068-8>
- Irfanianingrum, I., Chaidir, A. rizal, Sumardi, Rahardi, G. A., & Herdiyanto, D. W. (2023). Sistem Pendeteksi Dini Kebakaran hutan Berbasis Logika Fuzzy. *MITOR: Jurnal Teknik Elektro*, 23(2), 120–127. <https://doi.org/10.23917/emiton.v22i2.22019>
- Joseph, F. J. J., Nonsiri, S., & Monsakul, A. (2021). Keras and Tensorflow: A Hands On Experience. In *Advanced Deep Learning for Engineers and Scientist* (pp. 85–111).
- Komalasari, T. D. (2025). *No Title*. <https://katadata.co.id/ekonomi-hijau/ekonomi-sirkular/659ca314b6936/kebakaran-hutan-jadi-bencana-terbanyak-pada-2023-capai-1802-kejadian>
- Mahameru, I. A., Ravly Dwi Septian, M., Roliawati, D., Yusuf, A., & Ridwan, M. (2025). Implementasi Algoritma Convolutional Neural Network (CNN) Untuk Klasifikasi Kerapihan Ruangan. *Jurnal Komputer Teknologi Informasi Sistem Komputer (JUKTISI)*, 4(2), 1264–1271.
- Paraijun, F., Aziza, R. N., & Kuswardani, D. (2022). Implementasi Algoritma Convolutional Neural Network Dalam Mengklasifikasi Kesegaran Buah Berdasarkan Citra Buah. *KILAT*, 11(1), 1–9.
- Pasai, M. (2020). Dampak Kebakaran Hutan Dan Penegakan Hukum. *Jurnal Pahlawan*, 3(1), 36–46. <http://190.119.145.154/handle/20.500.12773/11756>
- Reis, H. C., & Turk, V. (2023). Detection of forest fire using deep convolutional neural networks with transfer learning approach. *Applied Soft Computing*, 143, 110362. <https://doi.org/https://doi.org/10.1016/j.asoc.2023.110362>
- Saharjo, B. H., & Hasanah, U. (2023). Analisis Faktor Penyebab Terjadinya Kebakaran Hutan dan Lahan di Kabupaten Pulang Pisau, Kalimantan Tengan. *Journal of Tropical Silviculture*, 14(01), 25–29. <https://doi.org/10.29244/j-siltrop.14.01.25-29>
- Setiawan, D., Putra, A. D., Stefani, K., & Felisa, J. (2021). Implementasi Covolutional Neural Network Untuk Facial Recognition. *Media Informatika*, 20(2), 66–79.
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(60), 1–48. <https://doi.org/10.1186/s40537-019-0197-0>
- Subasri, P., & Nithya, B. (2025). Forest Fire Detection Using Convolutional Neural Networks (CNN). *International Journal For Multidisciplinary Research (IJFMR)*, 7(4), 1–8.
- Wiranda, N., Purba, H. S., & Sukmawati, R. A. (2020). Survei Penggunaan Tensorflow pada Machine Learning untuk Identifikasi Ikan Kawasan Lahan Basah. *Indonesian Journal of Electronics and Instrumentation System (IJEIS)*, 10(2), 179–188.
- Zheng, X., Chen, F., Lou, L., Cheng, P., & Huang, Y. (2022). Real-Time Detection of Full-Scale Forest Fire Smoke Based on Deep Convolution Neural Network. *Remote Sensing*, 14(536), 1–15.
- Zuhri, A. S. M., Nafiiyah, N., & Budi, A. S. (2025). Identification of Road Damage Using the Convolutional Neural Network (CNN) Method. *Jurnal Ilmiah Sistem Informasi (JUSI)*, 4(3), 802–817.