ILKOM Jurnal Ilmiah Vol. 16, No. 1, April 2024, pp.46-58
Accredited 2nd by RISTEKBRIN No. 200/M/KPT/2020; E-ISSN 2548-7779 | P-ISSN 2087-1716

Research Article Open Access (CC–BY-SA)

Optimizing a Fire and Smoke Detection System Model with Hyperparameter Tuning and Callback on Forest Fire Images Using ConvNet Algorithm

Suryani[a,1,*]; Muhammad Syahlan Natsir[a,2];
[a] *Dipa Makassar University, Jl. Perintis Kemerdekaan No.KM.9, Tamalanrea Indah, Kec. Tamalanrea, Makassar, South Sulawesi 90245, Indonesia*
[1]*suryani187@undipa.ac.id;* [2]*sahlan@undipa.ac.id.*
** Corresponding author*

Article history: Received November 06, 2023; Revised December 18, 2023; Accepted March 24, 2024; Available online April 26, 2024

Abstract
Forest fire is a significant issue, especially for tropical countries like Indonesia. One of the impacts of forest fires is environmental pollution and damage, such as damage to flora and fauna, water, and soil. Fire detection technology is crucial as a preventive measure before the spread or expansion of fire points. Several forest fire detection systems have been developed by various research studies, with detection targets varying. Objects in the form of images are usually detected using the RGB color filtering method, but this method still results in false detections in image processing. Therefore, a classification model is built to detect fire and smoke in images using the Convolutional Neural Network (ConvNet) algorithm. In the development of the ConvNet model, a comparison of models is also conducted to assess the influence of Hyperparameter Tuning and Callbacks in optimizing the model's classification performance. The research results indicate that out of the six comparison scenarios created, the best model is obtained with 90% training data and 10% testing data, which is also optimized with Hyperparameter Tuning and Callbacks, with a Validation Accuracy of 98.18% and Validation Loss of 4.97%. This model is then implemented in the interface system.

Keywords: ConvNet Algorithm; Detection System; Forest Fire; Image; Model Optimization.

Introduction

Forests are a crucial component of the Earth's ecosystem, and everyone is responsible for their protection. Forest fire pose a significant ecological hazard, with smoke being an initial characteristic of such fires [1], [2]. Associated impacts of forest fire include environmental pollution and damage, such as harm to flora and fauna, water and soil degradation, and even posing risks to human life and property. This is particularly pertinent in tropical climate countries like Indonesia. In 2020, according to data from the National Disaster Management Agency (BNPB), at least 300 thousand hectares of forest area burned across various provinces. In Asia, Indonesia stands as one of the countries experiencing the most severe forest fire incidents annually. Forest fires can be caused by various factors, including climate, land cover conditions, soil types, and other biophysical environmental factors [3].

Detection technology for forest fire is highly necessary as a preventive measure before the spread or expansion of fire points. Several forest fire detection systems already exist, such as fire, smoke, and temperature sensors, which have been studied extensively. The detected objects vary, including temperature, fire, smoke, air humidity, and images. Images, another term for visual objects, play a crucial role as visual information components in multimedia. Image objects are typically detected using methods such as Color Filtering (RGB), Gray Level Co-Occurrence Matrices (GLCM), Hue Saturation (HSV), and YcbCr. The primary drawback of using Lab color is that it is time-consuming and involves a lot of manual work [4].

Previous research conducted by [5] focused on segmenting, coloring, and classifying tumor regions to identify abnormalities in medical images. The study aimed to assist medical practitioners in visualizing the shape, size, and orientation of tumors, with only the tumor regions colored in grayscale images. The classification accuracy before and after coloring reached 88.5% and 92.4% respectively, indicating a significant quantitative improvement in the post-coloring stage.

With the advancement of technology, several image classification systems utilize deep learning methods. The most common and straightforward deep learning method used is Convolutional Neural Network (ConvNet) [6]. ConvNet

http://dx.doi.org/10.33096/ilkom.v16i1.1937.46-58

is a method capable of classifying data in the form of images. It has replaced traditional manual feature extraction and has become the primary method for image processing [7]. ConvNet can automatically learn a series of unique features for a given task [8]. As of now, ConvNet is primarily used in image classification [9].

Previous research utilized Back Propagation Artificial Neural Network (BP-ANN) for predicting forest fire using image data as presented in [10]. BP-ANN is a computational algorithm that mimics the functioning of human nerve cells consisting of interconnected neurons in a network. The core of this learning algorithm is to adjust the weight values in response to errors. Weight changes are intended to minimize the network's error count, thereby achieving the desired output. Another study by [11], proposed a model for detecting and classifying new lung nodules using a one-stage detector called "I3DR-Net." The model was designed to prevent misinterpretation, especially in ambiguous anatomical structures resembling lung nodules, such as lymph node enlargement, which can result in decreased sensitivity and accuracy in detecting malignant lung nodules and delays in diagnosis, proven to have fatal consequences for patients.

In the conducted research, an optimization model was developed for a fire and smoke detection system using an images obtained from Kaggle. An image is a representation, similarity, or imitation of an object, which serves as the output of a data recording system that can be optical, such as a photo, analog, video signals like those on a television monitor, or digital, which can be directly stored on a storage medium. The system was built using the ConvNet algorithm and optimized the model with hyperparameter tuning and callbacks to obtain more accurate results.

ConvNet is a layered learning architecture capable of automatically extracting relevant features using hierarchical convolutional layers, without requiring significant human interaction or expert knowledge. These features are then fed into several fully connected layers in a chain-like fashion to accomplish the classification task [12]. Neural Network algorithms are the most suitable and effective for Image Classification cases because they excel in speed and performance while prioritizing accuracy in predictions, with resulting accuracies approaching perfection, at 98.61%. ConvNet also has the advantage of autonomously performing feature learning processes from images, as opposed to feature extraction, which requires obtaining features from images before classification.

Model optimization is carried out with hyperparameters because evaluation results show that the performance of the Neural Network model with hyperparameter optimization is better than the Neural Network model using default hyperparameters. The Neural Network model with default hyperparameters resulted in an accuracy of 81.38% and experienced overfitting. With the continuous advancement of artificial intelligence technology, various hyperparameter optimization algorithms have emerged, promising the potential to build more accurate machine learning models. Hyperparameter tuning is the process of finding the optimal hyperparameter configuration. Meanwhile, callback functions are used to ensure that the stored model is the one with the least validation loss [13].

With a system that detects fire and smoke in forest fire images, it is expected to assist the community in obtaining more accurate information about forest fires based on the acquired images.

Method

ConvNet was first used in the context of processing single digital images with the primary goal of classifying these images, where the considered information is the spatial relationship between pixels [14]. This system is built using the ConvNet algorithm and optimizing the model with hyperparameter tuning and callbacks to obtain more accurate results. Initially, a dataset consisting of 2,200 image data points was collected, comprising 1,100 images labeled as fire and 1,100 images labeled as smoke from forest fires obtained from Kaggle. The image data used in this research is in JPG (Joint Photographic Group) format. The training data is set to 80% of the dataset, comprising a total of 1,760 images, and the testing data is set to 20% of the dataset, comprising a total of 440 images.

After collecting the dataset, preprocessing is performed which includes resizing to standardize the pixels in each image and splitting the data into 80% for training and 20% for validation. Then, Hyperparameter Tuning and Callback are configured, followed by training the model using the ConvNet algorithm. After the training process is complete, accuracy is measured using a confusion matrix to obtain the accuracy results. If the model achieves good accuracy, it will be saved. However, if the model's accuracy is not satisfactory, the training process will be repeated by adjusting the Hyperparameters again.

Next, is applied to a system that can be used for classification. The classification process in the system begins with pre-processing the testing data, which involves resizing. The image processing optimization method with CNN uses the default standard size of 256×256 [15], [16], [17], [18]. After pre-processing, the data will be predicted or classified by the ConvNet model that has been created. Then, the output or final result will be the classification result, indicating whether the image contains fire or smoke. These steps can be seen in **Figure 1**.

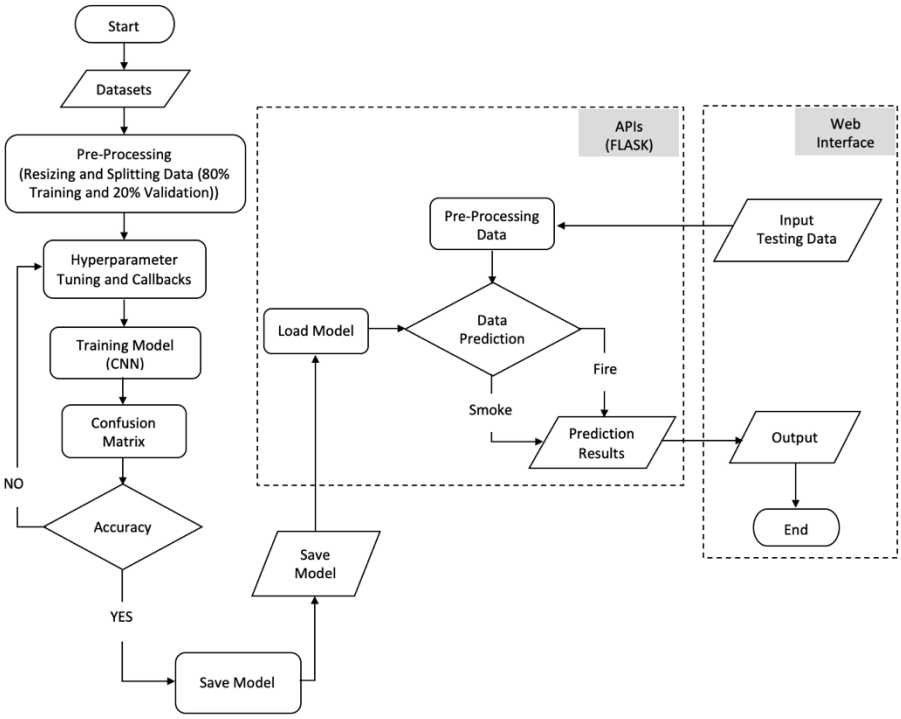


Figure 1. Research Stages

The ConvNet algorithm consists of 2 parts: feature extraction and classification [19]. The first stage of ConvNet is the convolution stage. In the convolution stage, a kernel of a certain size is used for computation. The number of kernels used depends on the number of features produced. Then, it proceeds to the activation function, typically using the ReLU (Rectifier Linear Unit) activation function. After exiting the activation function process, it undergoes the pooling process. This process is repeated several times until an adequate feature map is obtained to proceed to the fully connected neural network, and from the fully connected network, the output class is obtained.

A. Convolution layer

Convolution is a process to obtain pixels based on their values and those of their neighbors by involving a matrix called a kernel, which represents weighting. Convolution operation is an operation on two real-valued argument functions. It is common to combine the Convolution layer with pooling and sampling operations [20]. The convolution operation applies the output function as the Feature Map of the input image. These input and output can be seen as two real-valued arguments. The convolution operation can be expressed as follows:

$$S(t) = (x * t)(t) = \sum x(\alpha) * w(t - \alpha) \infty \alpha = -\infty \quad (1)$$

Where $S(t)$ represents the function resulting from the convolution operation, x is input and w is weight (*kernel*).

B. Pooling

Pooling, or merging, is a process in ConvNet where the input matrix is simplified into a new matrix. In this study, the max-pooling method is used in the pooling process to obtain a new matrix of size 2×2 by taking the highest value from each window. Max pooling layers play a crucial role in combining various low-level features from the surroundings while preserving important information while removing unnecessary details [21].

C. Flattening Layer

The last process is Fully-Connected. The output of the final process of the model within the feature extraction layer is still in the form of a multi-dimensional array, so it needs to be reshaped or "flatten" into a vector to be used as input for the fully-connected layer. Then, the fully-connected layer is added with a dense function, which is a function for adding layers to the fully-connected network.

D. Accuracy

After the training process of the ConvNet algorithm model is completed, accuracy is measured using a confusion matrix to obtain the accuracy results. If the model achieves good accuracy, it will be saved. Next, it will be applied to a system that can be used for classification.

The result of the confusion matrix calculation process has 4 outputs: accuracy, precision, recall, and error rate.

Table 1. Confusion Matrix

Prediction			
Actual		Negative	Positive
	Negative	*A*	*C*
	Positive	*B*	*D*

Results and Discussion

A. Data Collection

Data collection was carried out by taking a dataset from Kaggle of 2,200 images consisting of 1,100 fire images and 1,100 smoke images.

B. Preprocessing

In the preprocessing process, the data resizing, augmentation and splitting stages are carried out. This stage is carried out before the training process by resizing to equalize the pixels in each image, Augmentation to modify the image so that it can recognize different images, and Splitting Data with the first 3 scenarios 70% training data and 30% testing data, second 80% training data and 20% testing data, thirdly 90% training data and 10% testing data.

1) Resize Image

In this stage, the author reduces the size of the images to 256 x 256 pixels. The view of the image resizing process can be seen in **Figure 2**.

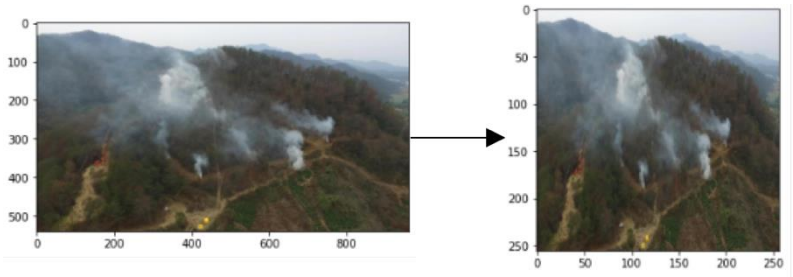


Figure 2. *Resize Image*

In Figure 2, the image on the left is a picture of smoke. This image is before the image resizing process, where the initial size is 800 x 600 pixels. Then, the image on the right is the result of resizing the image to 256 x 256 pixels.

2) Image Augmentation

Augmentation is the process of manipulating or modifying an image, so that the original image in standard form is changed in shape and position. Data augmentation aims to enable the machine to learn and recognize from various different images while also being utilized to increase the amount of data available.

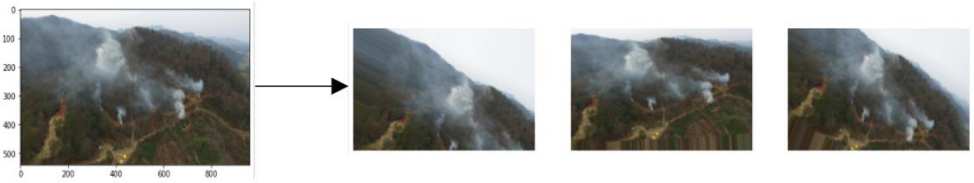


Figure 3. *Augmentasi Image*

In **Figure 3**, the result of image augmentation is shown. The image on the left is the original image before augmentation. In image augmentation, the original image is processed with various transformations and manipulations, resulting in new images with different values but still retaining the same class information. Some image augmentation techniques used include: rotating the image by 25 degrees, then shifting the height by 10% and the width by 10%, followed by enlarging the image by 20%, and finally flipping the image horizontally to create symmetric variations.

C. Model Training Results

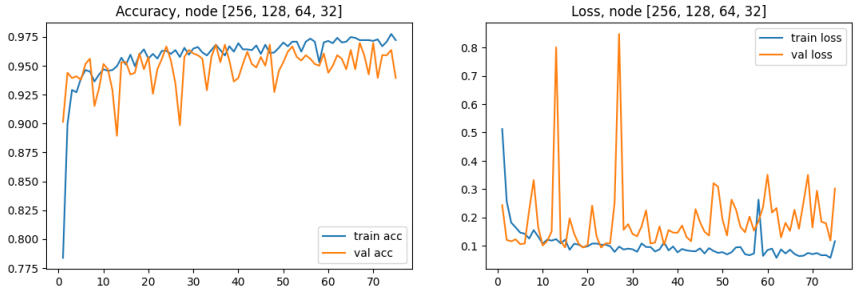


Figure 4. Output of scenario 1 without Hyperparameter Tuning and Callback.

In **Figure 4**, which represents the output of the first scenario without using Hyperparameter Tuning and Callback, with a data split of 70% for training and 30% for testing, it can be observed that the graph does not experience overfitting, where overfitting is a condition where the validation accuracy is higher than the training accuracy. In this scenario, the training accuracy is 95.58% and the validation accuracy is 93.94%.

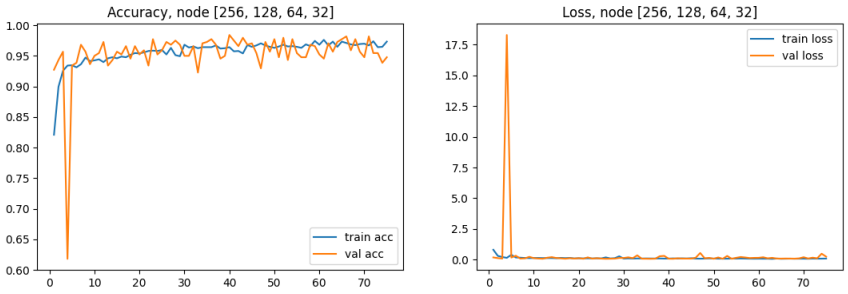


Figure 5. Output of scenario 2 without Hyperparameter Tuning and Callback.

Figure 5 represents the output of the second scenario without using Hyperparameter Tuning and Callback, with a data split of 80% for training and 20% for testing, it can be observed that the graph does not experience overfitting. In this scenario, the training accuracy is 96.25% and the validation accuracy is 94.77%

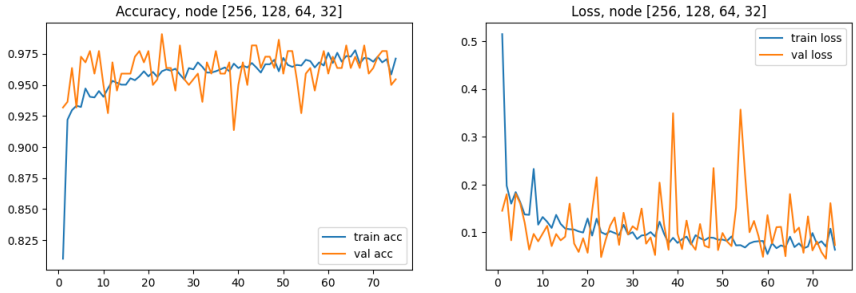


Figure 6. Output of scenario 3 without Hyperparameter Tuning and Callback.

Figure 6 represents the output of the third scenario without using Hyperparameter Tuning and Callback, with a data split of 90% for training and 10% for testing, it can be observed that the graph does not experience overfitting. In this scenario, the training accuracy is 96.87% and the validation accuracy is 95.45%.

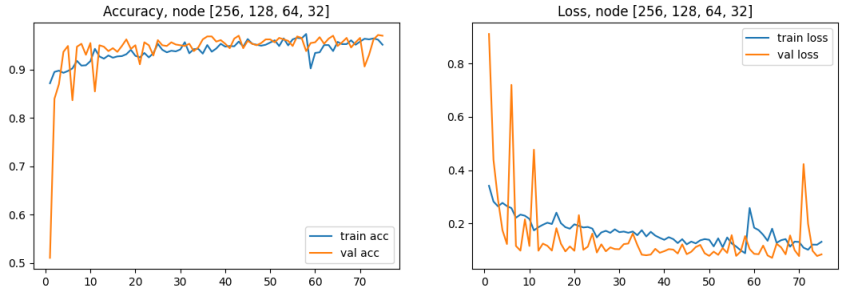


Figure 7. Output of scenario 1 with Hyperparameter Tuning and Callback.

Figure 7 represents the output of the first scenario with Hyperparameter Tuning and Callback, with a data split of 70% for training and 30% for testing, it can be observed that the graph does not experience overfitting. In this scenario, the training accuracy is 97.99% and the validation accuracy is 97.88%.

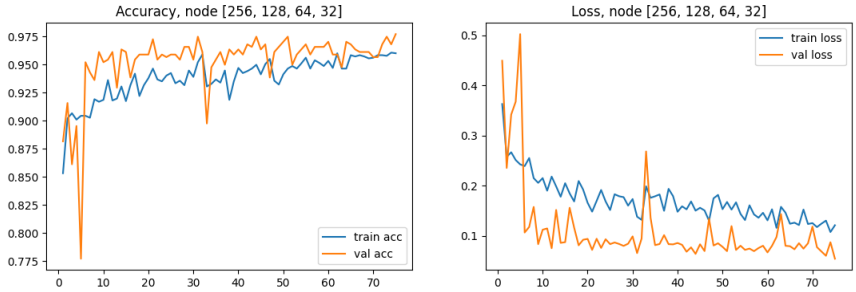


Figure 8. Output of scenario 2 with Hyperparameter Tuning and Callback

Figure 8 represents the output of the second scenario with Hyperparameter Tuning and Callback, with a data split of 80% for training and 20% for testing, it can be observed that the graph does not experience overfitting. In this scenario, the training accuracy is 98.35% and the validation accuracy is 97.73%.

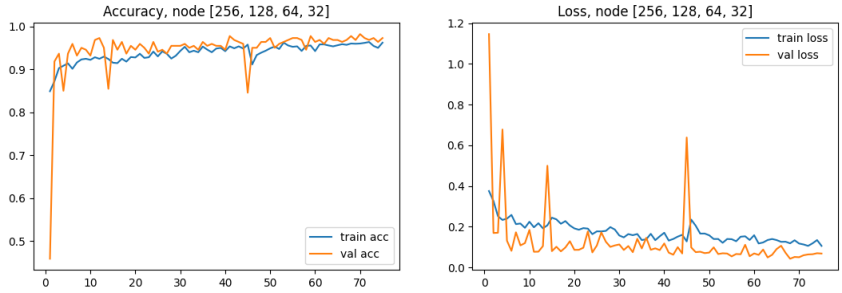


Figure 9. Output of scenario 3 with Hyperparameter Tuning and Callback.

Figure 9 represents the output of the third scenario with Hyperparameter Tuning and Callback, with a data split of 90% for training and 10% for testing, it can be observed that the graph does not experience overfitting. In this scenario, the training accuracy is 98.59% and the validation accuracy is 98.18%.

Table 2. Results of the model without using Hyperparameter Tuning and Callback

Training/Testing	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
70 30	95.58	93.94	17.23	30.24
80 20	96.25	94.77	12.49	23.88
90 10	96.87	95.45	6.55	7.34

Table 3. Results of the model using Hyperparameter Tuning and Callback

Training/Testing	*Training Accuracy*	*Validation Accuracy*	*Training Loss*	*Validation Loss*
70 30	97,99	97,88	5,14	5,42
80 20	98,35	97,73	4.67	5,46

90 10	**98,59**	**98,18**	**3,93**	**4,97**

Based on **Tables 2** and **3**, it is evident that learning using Hyperparameter Tuning and Callback is better compared to those that do not use them. It can be seen in scenario 1 with a 70% training data and 30% testing data split, where the Validation Accuracy using Hyperparameter Tuning and Callback is higher compared to those that do not use them. From the discussion above, it can also be concluded that among the six scenarios, the best scenario is using Hyperparameter Tuning and Callback in scenario 3 because the Validation Accuracy reaches 98.18%, which is the highest among all scenarios.

Table 4. Results of all experiments

Scenario Number:	*Hyperparameter*	*Training*	*Testing*	*Training Accuracy*	*Validation Accuracy*	*Training Loss*	*Validation Loss*
1	• *Epoch* = 75 • *Batch Size* = 32 (default) • *Optimizer* = RMSprop • *Callback* = there is none	70%	30%	95.58	93.94	17.23	30.24
2	• *Epoch* = 75 • *Batch Size* = 32 (default) • *Optimizer* = RMSprop • *Callback* = there is none	80%	20%	96.25	94.77	12.49	23.88
3	• *Epoch* = 75 • *Batch Size* = 32 (default) • *Optimizer* = RMSprop • *Callback* = there is none	90%	10%	96.87	95.45	6.55	7.34
4	• *Epoch* = 75 • *Batch Size* = 8 • *Optimizer = Adam* • *Callback = ModelCheckpoint*	70%	30%	97.99	97.88	5.14	5.42
5	• *Epoch* = 75 • *Batch Size* = 8 • *Optimizer = Adam* • *Callback = ModelCheckpoint*	80%	20%	98.35	97.73	4.67	5.46
6	• *Epoch* = 75 • *Batch Size* = 8 • *Optimizer = Adam* • *Callback = ModelCheckpoint*	90%	10%	**98.59**	**98.18**	**3.93**	**4.97**

D. Confusion Matrix

The results of accuracy validation using the Confusion Matrix can be seen in **Figure 10**.

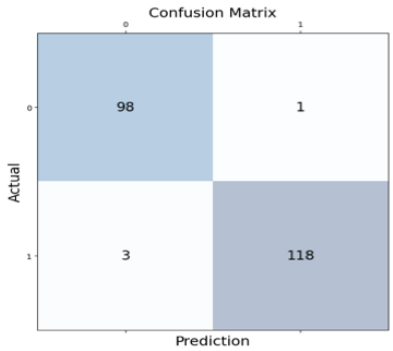


Figure 10. *Confusion Matrix*

In **Figure 10**, there are 2 categories, 0 for fire and 1 for smoke. From the figure, it can be seen that there are 98 correct predictions of smoke and 118 correct predictions of fire. Then, out of 99 smoke predictions, there is 1 false detection of smoke, and out of 121 fire predictions, there are 3 false detections of fire (*false detection*).

The validation metrics that can be summarized from the output of the Confusion Matrix, along with their explanations, are as follows:

1) *Accuracy*

Accuracy measures the level of prediction accuracy of the overall model. The percentage value indicates how many correct predictions there are from all classes (fire and smoke).

$Accuracy = (TP + TN)/Total = (118+98)/(220) = 98.1818\%$

2) *Recall*

Recall measures the ability of the model to classify positive class instances out of all actual positive class instances. The percentage value indicates how many of all positive class instances (smoke) are correctly predicted as smoke by the model.

$$Recall = \frac{TP}{TP+TN} = \frac{118}{118+3} = 97.5206\%$$

3) *Precision*

Precision measures the accuracy of positive predictions made by the model. The percentage value indicates how many of all predicted positive instances (smoke) by the model are actually positive instances (smoke).

$$Precision = \frac{TP}{TP+FP} = \frac{118}{118+1} = 99.1596\%$$

4) *F1-score*

F1-score helps measure Precision and Recall simultaneously using Harmonic Mean as a replacement for Arithmetic Mean. F1-score is often used to obtain a better overview of the model's performance, especially if we have an unbalanced number of data in each class.

$$\text{F1 Score} = \frac{2(Recall \times Precision)}{Recall+Precision} = \frac{19338,216}{196,67} = 98.3282\%$$

E. ConvNet Architecture

The input image to the ConvNet model is a $256 \times 256 \times 3$ image. The number 3 represents an image with 3 channels, including Red, Green, and Blue (RGB). The input image will first undergo convolution and pooling processes in the feature learning stage. The design includes three convolution layers. Each convolution has a different number of filters and kernel sizes. Then, the feature map from the pooling layer is flattened into a vector. This process is commonly referred to as the fully connected layer stage. There are two stages in the ConvNet architecture: Feature Learning and Classification. In the Feature Learning stage, features are extracted from the input image data, involving processes such as Convolution, followed by activation function filtering like ReLU, and then continuing to the Pooling process using Max Pooling. Then, in the Classification stage, information from the features learned during the Feature Learning process will be used to determine the classification of the data into a corresponding category or class. The Classification stage starts with the Flattening process from the end of the Feature Learning stage, then continues to

the Fully Connected Layer, which is also connected to the Output Layer, where the classification process takes place by applying an Activation Function. In this model, the Output Layer uses Sigmoid as its Activation Function to classify into one of two classes, fire or smoke. The interpretation of each stage can be seen in **Figure 11**.

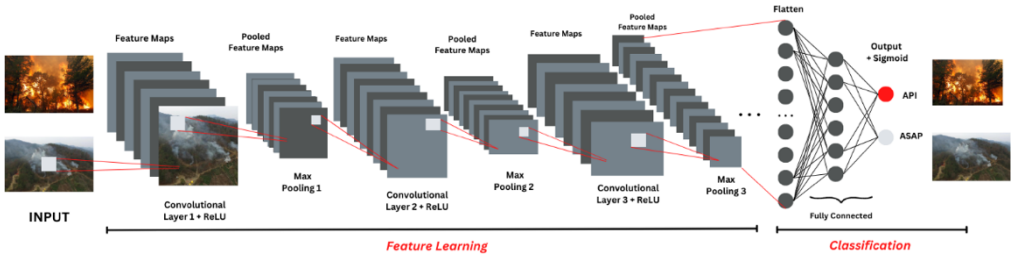


Figure 11. *ConvNet Architecture*

In **Figure 11**, the ConvNet architecture consists of:

a. The first convolutional process uses a kernel size of 3×3 and 3 depths. The convolution process is a combination of two different matrices resulting in a new matrix value. **Figure 12** shows the convolution process with a 3 × 3 kernel size and a stride of 1, where the stride is the displacement of the kernel over the input matrix.

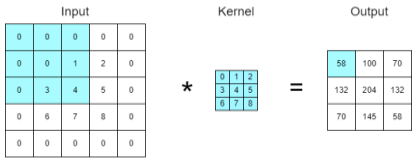


Figure 12. Convolution Calculation Process

After the convolution process, a Rectified Linear Unit (ReLU) activation function is added. This activation function aims to transform negative values into zero. The result of this convolution has the same size, which is 256x256, because a padding value of 0 was used during the convolution process.

b. The pooling process involves reducing the size of the matrix using pooling operations. This study employs max-pooling to obtain a new matrix size of 2 × 2 by taking the maximum value from each window. The output of this process contains the maximum values taken from the convolutional feature map matrix, as depicted in **Figure 13**.

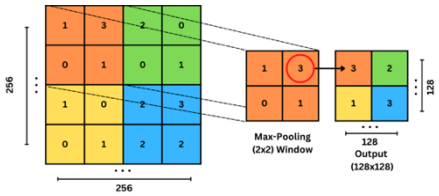


Figure 13. *Pooling Process*

c. The second convolutional process continues the results from the first pooling process, with an input matrix of size 128 × 128 and a kernel size of 3×3. This second convolutional process also employs the ReLU activation function.

d. The next process is the second pooling process. This process is almost identical to the first pooling process, with the difference being that the final output matrix size is 64 × 64.

e. Next, the third convolutional process continues the results from the second pooling process, with an input matrix of size 64 × 64 and a kernel size of 3×3. This third convolutional process also employs the ReLU activation function

f. Next, in the third pooling process, it is nearly identical to the first and second pooling processes, with the difference being that the final output matrix size is 32 × 32.

g. The result of the convolution process yields output feature maps in the form of a multidimensional array. The subsequent process involves flattening, which converts the feature maps into a single vector, allowing them to be inputted into the fully connected layer for classification purposes.

h. Next is the fully connected stage, where a single hidden layer in the Multi-Layer Perceptron (MLP) network is employed. In **Figure 14**, the Flatten operation converts the output from the pooling layer into a vector. Before proceeding to the classification process (predicting images), a training process is conducted using dropout values. Dropout is a regularization technique in neural networks aimed at randomly selecting and temporarily dropping out some neurons during the training process, meaning these neurons are not utilized during training. The purpose of this process is to reduce overfitting during training.

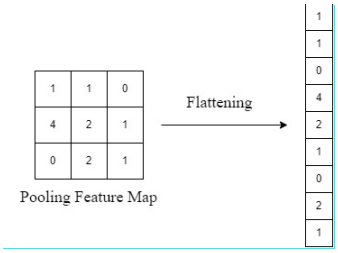


Figure 14. Flatten process

i. The classification process involves utilizing the sigmoid activation function to classify the input into the respective targets, including fire and smoke.

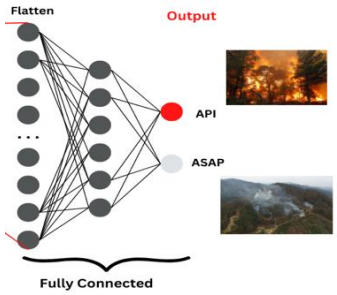


Figure 15. Process Classification

E. Analysis and Data Validation

The following is the data analysis conducted by the researchers and the form of data validation used as testing for the analysis produced by the authors in this study:

1) Hyperparameter Tuning and Callback

In this process, Google Colaboratory is used to create the ConvNet model. The dataset, which has been pre-processed, is then trained using the following hyperparameter tuning:

Epoch = 75
Batch Size = 8
Activation Function = *ReLu* dan *Sigmoid*
LossFunction = *binary_crossentropy*
Optimizer = Adam (0.001 *Iniatiate Learning Rate*)
ModelCheckpoint (*val_accuracy*, *mode* = *max*)

In the model summary, there are 4 convolutional layers followed by pooling layers. After that, the flatten process is conducted, which involves learning processes (dense/hidden layers). Following the learning processes, classification is performed using the activation function (sigmoid) to obtain the best model.

2) Training & Testing Model

At this stage, there are 6 scenarios, including 3 scenarios without using Hyperparameter Tuning and Callback, and 3 scenarios that use Hyperparameter Tuning and Callback. The purpose of this stage is to compare the accuracy results between those using Hyperparameter Tuning and Callback with those that do not.

Table 5. Scenarios without Hyperparameter Tuning and Callbacks

Scenario number	Training	Testing	Batch Size	Callback	Optimizer
1	70%	30%	32 *(default)*	Tidak Ada	*RMSprop*

2	80%	20%	32 *(default)*	Tidak Ada	*RMSprop*
3	90%	10%	32 *(default)*	Tidak Ada	*RMSprop*

Table 6. Scenario with *Hyperparameter Tuning* and *Callback*

Scenario Number	Training	Testing	Batch Size	Callback	Optimizer
1	70%	30%	8	*Model CheckPoint*	*Adam*
2	80%	20%	8	*Model CheckPoint*	*Adam*
3	90%	10%	8	*Model CheckPoint*	*Adam*

After the training model process and testing were conducted, a curve displaying the training and validation accuracy, as well as training and validation loss, was obtained. The architecture of the model used in the training process scenario above can be seen in **Table 7**.

Table 7. Model *ConvNet*

No.	Name	Size	Parameter
0	*Input*	(256, 256, 3)	0
1	Conv2d_1	(256, 256, 256)	7168
2	Batch_normalization_1	(256, 256, 256)	1024
3	MaxPool_1	(128, 128, 256)	0
4	Conv2d_2	(128, 128, 128)	295040
5	Batch_normalization_2	(128, 128, 128)	512
6	MaxPool_2	(64, 64, 128)	0
7	Conv2d_3	(64, 64, 64)	73792
8	Batch_normalization_3	(64, 64, 64)	256
9	MaxPool_3	(32, 32, 64)	0
10	*Flatten*	65536	0
11	*Dense*	32	2097184
12	Batch_normalization_4	32	128
13	*Dropout*	32	0
14	*Output*	1	33
Total			2.475.137

Validation testing of the system was conducted on 10 input data images of fire and 10 input data images of smoke. **Figure 16** represents one of the validation test results for one input data image of fire and one input data image of smoke.

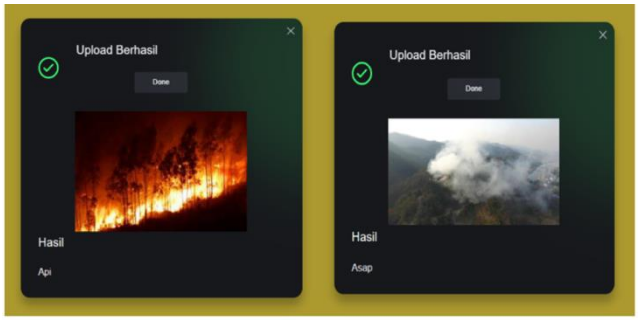


Figure 16. Validation Test Input

The output results from the fire and smoke model that have been created are deemed true or valid, meaning that each input image yields results consistent with images of fire and smoke.

Conclusion

Based on the conducted research, several model creation scenarios were carried out to compare the effects of Hyperparameter Tuning and Callback on optimizing the model's ability in the classification process. Out of the six scenarios, consisting of three scenarios without optimization and three scenarios with optimization, it is evident that in the three scenarios where Hyperparameter Tuning and Callback optimization were applied, there was an increase in the model's accuracy or a decrease in the model's error/loss value, indicating that the optimization had a positive impact on the model.

From the performance measurement of the model using the confusion matrix, the following values were obtained: accuracy of 98.18%, recall of 97.52%, precision of 99.15%, and F1-Score of 98.32%. Among the six scenarios, the third scenario, which utilized Hyperparameter Tuning and Callback optimization, emerged as the model with the best accuracy rate, with a Validation Accuracy value of 98.18% and Validation Loss of 4.97%.

For further research, it is expected that the addition of various forest fire images will support the classification model to perform even better, considering the varying conditions and visual appearances of forests in the field.

References

[1] Y. Hu *et al.*, "Fast forest fire smoke detection using MVMNet," *Knowl Based Syst*, vol. 241, pp. 1–20, Apr. 2022, doi: 10.1016/j.knosys.2022.108219.

[2] Y. Hu *et al.*, "Fast forest fire smoke detection using MVMNet," *Knowl Based Syst*, vol. 241, p. 108219, 2022, doi: 10.1016/j.knosys.2022.108219.

[3] E. Iwantri Goma *et al.*, "Analysis Of Forest And Land Fire In Samarinda," *Jurnal Sains Informasi Geografi [J SIG]* , vol. 4, no. 2, pp. 99–104, 2021, doi: 10.31314/jsig.v4i2.1082.

[4] K. Saastamoinen and S. Penttinen, "Visual seabed classification using k-means clustering, CIELAB colors and Gabor-filters," in *Procedia Computer Science*, Elsevier B.V., 2021, pp. 2471–2478. doi: 10.1016/j.procs.2021.09.016.

[5] M. Mehmood *et al.*, "Improved colorization and classification of intracranial tumor expanse in MRI images via hybrid scheme of Pix2Pix-cGANs and NASNet-large," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 4358–4374, Jul. 2022, doi: 10.1016/j.jksuci.2022.05.015.

[6] J. Florentin, T. Dutoit, and O. Verlinden, "Detection and identification of European woodpeckers with deep convolutional neural networks," *Ecol Inform*, vol. 55, p. 101023, 2020, doi: 10.1016/j.ecoinf.2019.101023.

[7] G. Sun, Y. Wen, and Y. Li, "Instance segmentation using semi-supervised learning for fire recognition," *Heliyon*, vol. 8, no. 12, Dec. 2022, doi: 10.1016/j.heliyon.2022.e12375.

[8] Z. Zhang, Q. Jin, L. Wang, and Z. Liu, "Video-based Fire Smoke Detection Using Temporal-spatial Saliency Features," in *Procedia Computer Science*, Elsevier B.V., 2021, pp. 493–498. doi: 10.1016/j.procs.2021.12.275.

[9] M. Despotovic, D. Koch, S. Thaler, E. Stumpe, W. Brunauer, and M. Zeppelzauer, "Linking repeated subjective judgments and ConvNets for multimodal assessment of the immediate living environment ☆ Method details," *MethodsX*, vol. 12, p. 102556, 2024, doi: 10.1108/JERER-11-2022-0036.

[10] J. LeBien *et al.*, "A pipeline for identification of bird and frog species in tropical soundscape recordings using a convolutional neural network," *Ecol Inform*, vol. 59, p. 101113, 2020, doi: 10.1016/j.ecoinf.2020.101113.

[11] I. W. Harsono, S. Liawatimena, and T. W. Cenggoro, "Lung nodule detection and classification from Thorax CT-scan using RetinaNet with transfer learning," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 3, pp. 567–577, Mar. 2022, doi: 10.1016/j.jksuci.2020.03.013.

[12] J. M. Bueno-Barrachina, Y. Ye-Lin, F. Nieto-del-Amor, and V. Fuster-Roig, "Inception 1D-convolutional neural network for accurate prediction of electrical insulator leakage current from environmental data during its normal operation using long-term recording," *Eng Appl Artif Intell*, vol. 119, Mar. 2023, doi: 10.1016/j.engappai.2022.105799.

[13] J. Omar, N. Husna Shabrina, A. N. Bhakti, and A. Patria, "Emotion Recognition using Convolutional Neural Network on Virtual Meeting Image," *Ultima Computing: Jurnal Sistem Komputer*, vol. 13, no. 1, 2021, doi: 10.31937/sk.v13i1.2108.

[14] H. K. Zhang, D. P. Roy, and D. Luo, "Demonstration of large area land cover classification with a one-dimensional convolutional neural network applied to single pixel temporal metric percentiles," *Remote Sens Environ*, vol. 295, Sep. 2023, doi: 10.1016/j.rse.2023.113653.

[15] S. Saifullah and R. Dreżewski, "Modified Histogram Equalization for Improved CNN Medical Image Segmentation," *Procedia Comput Sci*, vol. 225, pp. 3021–3030, 2023, doi: 10.1016/j.procs.2023.10.295.

[16] S. A. Güven and M. F. Talu, "Brain MRI high resolution image creation and segmentation with the new GAN method," *Biomed Signal Process Control*, vol. 80, p. 104246, 2023 doi: 10.1016/j.bspc.2022.104246.

[17] M. Siddharth and R. Aarthi, "Blended multi-class text to image synthesis GANs with RoBerTa and Mask R-CNN," *Procedia Comput Sci*, vol. 218, pp. 845–857, 2023, doi: 10.1016/j.procs.2023.01.065.

[18] A. Nechyporenko, M. Frohme, V. Alekseeva, V. Gargin, D. Sytnikov, and M. Hubarenko, "Deep learning based image segmentation for detection of odontogenic maxillary sinusitis," in *2022 IEEE 41st International Conference on Electronics and Nanotechnology (ELNANO)*, IEEE, 2022, pp. 339–342, doi: 10.1109/ELNANO54667.2022.9927086.

[19] X. Wang, V. Liesaputra, Z. Liu, Y. Wang, and Z. Huang, "An in-depth survey on Deep Learning-based Motor Imagery Electroencephalogram (EEG) classification," *Artificial Intelligence in Medicine*, vol. 147. Elsevier B.V., Jan. 01, 2024. doi: 10.1016/j.artmed.2023.102738.

[20] A. Jardines *et al.*, "Thunderstorm prediction during pre-tactical air-traffic-flow management using convolutional neural networks," *Expert Syst Appl*, vol. 241, May 2024, doi: 10.1016/j.eswa.2023.122466.

[21] A. Chouiekh and E. H. I. El Haj, "ConvNets for fraud detection analysis," in *Procedia Computer Science*, Elsevier B.V., 2018, pp. 133–138. doi: 10.1016/j.procs.2018.01.107.