Jurnal Masyarakat Informatika, 16(1), 2025
DOI:10.14710/jmasif.16.1.73283
ISSN: 2777-0648

Classification of Real and Fake Images using Error Level Analysis Technique and MobileNetV2 Architecture

Muhamad Nur Baihaqi*, Aris Sugiharto, and Henri Tantyoko
Departemen Informatika, Universitas Diponegoro, Semarang
* *Corresponding author: baihaqi2193@gmail.com*

Abstract
Advancements in technology have made image forgery increasingly difficult to detect, raising the risk of misinformation on social media. To address this issue, Error Level Analysis (ELA) feature extraction can be utilized to detect error level variations in lossy-formatted images such as JPEG. This study evaluates the contribution of ELA features in classifying authentic and forged images using the MobileNetV2 model. Two scenarios were tested using the CASIA 2.0 dataset: without ELA and with ELA. Fine-tuning was performed to adapt the model to the new problem. Experimental results show that incorporating ELA improves model accuracy up to 93.1%, compared to only 76.41% in the scenario without ELA. Validation using k-fold cross-validation yielded a high average f1-score of 96.83%, confirming the effectiveness of ELA in enhancing the classification performance of authentic and forged images.

Keywords : *classification, error level analysis, mobilenetv2, fine tuning, k-fold cross validation, pretrained model*

1 Introduction

Currently, images have become one of the most widely used forms of media for conveying information through social media. According to data from Google Search, approximately 14 billion photos are shared on social media every day. This massive distribution of images is supported by various technological advancements, such as more effective compression methods, faster internet connections, and the development of social media platforms [1]. With the increasing number of digital interactions, images have become a crucial aspect of digital communication, as they can provide important context or evidence regarding the interactions that take place in the digital sphere [2]. Unfortunately, this aspect is often misused by certain parties with malicious intent, by altering images to spread false information.

Humans have difficulty detecting visual changes in manipulated images due to a phenomenon known as the *change blindness cognitive effect*, which is the inability to notice changes when they occur gradually or when attention is focused elsewhere. This phenomenon hinders the identification of manipulated elements within an image. A study by Nightingale et al. in 2017 showed that humans were only able to detect 65% of manipulated images, only slightly better than random guessing (50%) which indicates a limited ability to recognize image forgeries [3]. Therefore, the development of a fast and accurate image forgery detection system is highly needed.

To overcome the limitations of human ability in classifying real and fake images, machine learning and deep learning technologies can be utilized. Convolutional Neural Networks (CNNs) are one type of deep learning architecture that can be used for classification tasks. CNN models can be built manually or by using available pretrained models. This approach is known as transfer learning.

Received: 15 May 2025 ; Revised: 26 May 2025; Accepted: 28 May 2025; Published: 31 May 2025

One of the pretrained models that offers simplicity and memory efficiency in its application is MobileNetV2 [4].

To help the model better capture important features in images to be classified, feature extraction methods can be applied. Several feature extraction techniques that can be used in classifying real and fake images include Error Level Analysis (ELA), Speed Up Robust Features (SURF), and YCrCb patch extraction. ELA is one of the simplest and most effective feature extraction techniques. This method works by identifying areas with compression differences across the image's color channels. A study conducted by Choudary showed that ELA combined with a shallow CNN model was quite effective in classifying real and fake images, achieving an accuracy of up to 91.70% [5].

However, most previous studies have primarily utilized shallow or manually built CNN models, which may lack the efficiency and scalability needed for practical deployment. There is limited research exploring the integration of lightweight, pretrained deep learning models such as MobileNetV2 with ELA for image forgery detection tasks. This research aims to evaluate the performance of the MobileNetV2 model combined with ELA feature extraction for classifying real and fake images offering a more efficient yet accurate approach that can be potentially implemented in real-time or resource-constrained environments. The model's performance will be evaluated using accuracy, precision, recall, and F1-score metrics to provide insights into its capability in distinguishing between real and manipulated images.

2 Literature Review

Based on feature extraction approaches, several studies have been conducted on the classification of real and fake images. Rao and Ni [6] used the Spatial Rich Model (SRM) to initialize the weights of a CNN as a feature extractor, combined with a patch sampling technique and a Support Vector Machine (SVM) classifier. Their model achieved accuracies of 98.04% on the CASIA 1.0 dataset, 97.83% on CASIA 2.0. and 96.38% on the COLUMBIA DVMM dataset.

Alsandi et al. [7] implemented the Speed Up Robust Features (SURF) technique to extract Regions of Interest (ROI), which were then classified using Mean-Local Binary Pattern (Mean-LBP) and an SVM model. This approach achieved accuracies of 97.90% on CASIA TIDE V1.0. 98.20% on CASIA TIDE V2.0. and 98.90% on the COLUMBIA dataset.

Choudhary et al. [8] proposed the use of a shallow CNN for fake image classification and segmentation, using VGG U-Net with Error Level Analysis (ELA) features. Their approach achieved an accuracy of 91.70% for classification and an Intersection over Union (IoU) score of 80.80% for segmentation tasks [5]. Hebbar and Kunte (2021) compared the performance of various deep learning models using transfer learning on the CASIA 2.0 dataset. Their results showed that VGG16 achieved an accuracy of 95.90%, VGG19 reached 93.85%, DenseNet121 reached 95.69%, DenseNet169 reached 97.06%, DenseNet201 reached 97.42%, and ResNet50 achieved 97.58%.

Shikalgar et al. [9] evaluated the performance of the lightweight MobileNet model without additional feature extraction for classifying real and fake images, achieving an accuracy of 98.75% on the CASIA 2.0 dataset with 400 validation samples. Meanwhile, Phan-Xuan et al. [10] used MobileNetV2 along with patch-based preprocessing in the YCrCb color space, stride modification, and bottleneck dimension adjustments, achieving an accuracy of 95.15% on the CASIA 2.0 dataset. These studies highlight the promising potential of developing fake image classification models using various architectural approaches and feature extraction techniques.

3 Research Methods

The research begins with data collection. The collected data is then filtered to meet specific criteria, ensuring it is suitable for further processing. Preprocessing is applied to the data by implementing Error Level Analysis (ELA) feature extraction, resizing, and pixel normalization. After preprocessing, the data is split into training and validation sets using the k-fold cross-validation mechanism. The model is then evaluated using performance metrics to assess its effectiveness. This series of research method is shown at Figure 1. Furthermore, the details for each research step are described below.

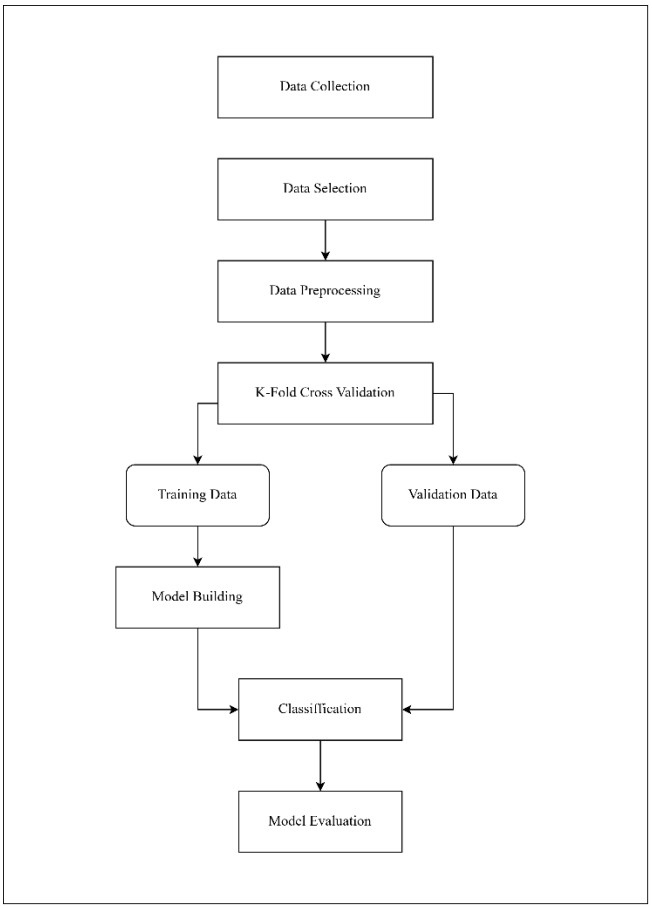

Figure 1 Research Method Flow Chart

3.1 Data Collection

The dataset used in this research is the CASIA Image Tampering Detection Evaluation Database (CASIA ITDE) version 2.0. This dataset was developed by the Chinese Academy of Sciences (CASIA) using Adobe Photoshop CS3 and the Windows XP operating system [15]. CASIA 2.0 contains a total of 12,323 images divided into two classes: *authentic* and *tampered*. The *authentic* class consists of 7,200 original, unmanipulated images, while the *tampered* class includes 5,123 images that have been manipulated. The manipulations were performed using techniques such as copy-move and splicing, with various image editing sizes. The dataset includes files in multiple formats, comprising both

lossless formats such as PNG, BMP, and TIFF, as well as lossy formats such as JPEG. Few of the dataset image can be found at Figure 2.

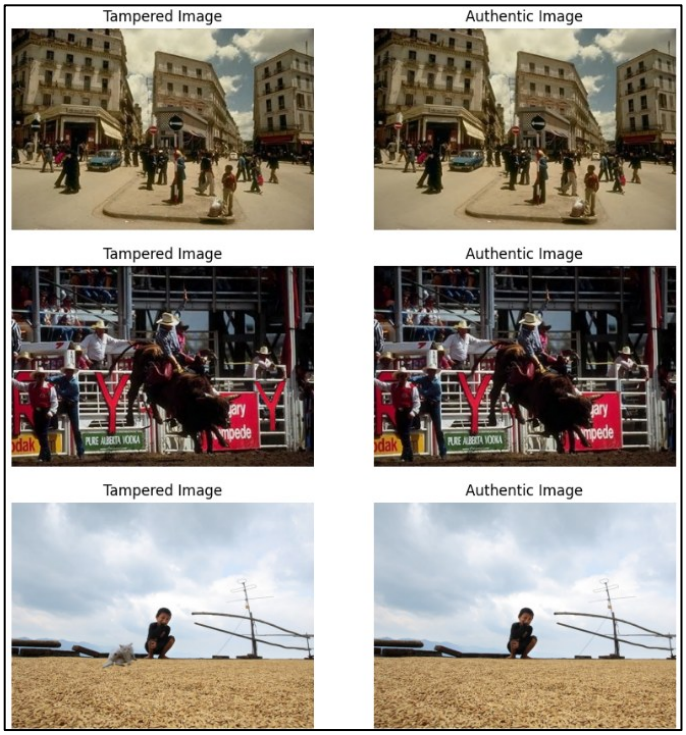

Figure 2 CASIA 2.0 Example Image

3.2 Data Selection

This study uses Error Level Analysis (ELA) as a feature extraction method that analyzes compression errors in images. As a result, ELA can only be applied to images in lossy formats such as JPEG. Therefore, a selection process was carried out so that only JPEG-formatted images were used in this research. This study uses Error Level Analysis (ELA) as a feature extraction method that analyzes compression errors in images. As a result, ELA can only be applied to images in lossy formats such as JPEG. Therefore, a selection process was carried out so that only JPEG-formatted images were used in this research.

Exploration of the dataset revealed that there are 7,437 *authentic* images and 2,064 *tampered* images in JPEG format, indicating an imbalance between the classes. This imbalance can affect the model's performance in predicting the positive class (*tampered*), which has fewer instances compared to the negative class (*authentic*). To address this issue, undersampling was performed on the *authentic* class so that the number of samples is balanced across both classes, with 2,064 images in each class. This undersampling process is illustrated at Figure 3.

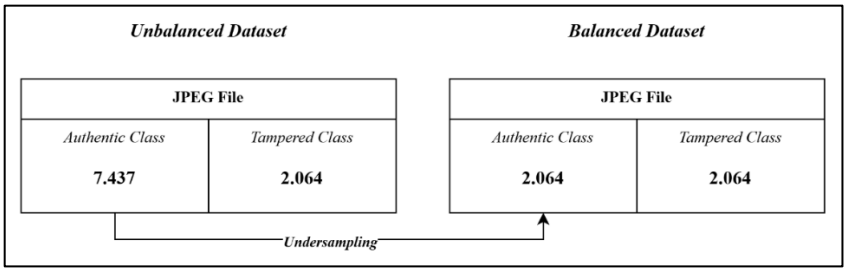

Figure 3 Undersampling Process of the Dataset

3.3 Dataset Preprocessing

Preprocessing was carried out to prepare the data before it is used by the model during the training process. Several preprocessing steps were performed in this study, including resaving, ELA feature extraction, resizing, and normalization through scaling. Figure 4 shows the image preprocessing steps of this research.

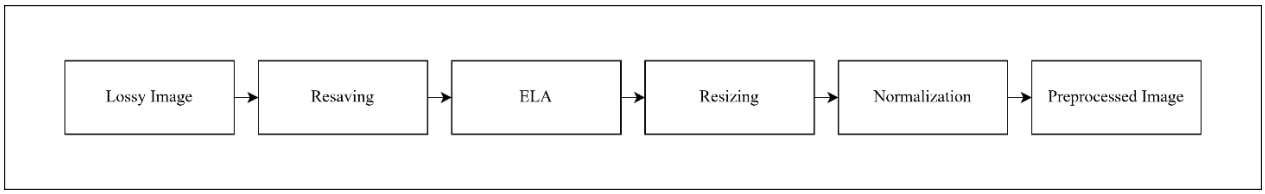

Figure 4 Image Preprocessing Steps

3.3.1 Resaving

To perform ELA, lossy-formatted images are first resaved to produce compressed images with a specific quality level. The purpose of resaving is to generate a compressed version of the image, which can then be compared with the original image to create the ELA image. This resaving process involves converting the color space from RGB to YCrCb (YUV). Additionally, a Discrete Cosine Transform (DCT) is applied to convert the image from the spatial domain to the frequency domain. The formula for the DCT is shown in Equation (1).

$$F(i,j) = \frac{2}{N} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y)\, cos\left[\frac{(2x+1)i\pi}{2N}\right] cos\left[\frac{(2x+1)j\pi}{2N}\right]$$

$$C(i), C(j) = \begin{cases} \frac{1}{\sqrt{2}}\ if\ i,j = 0 \\ 1\ if\ i,j > 0 \end{cases} \quad (1)$$

On the equation above, *F(i, j)* represents the DCT coefficient at position *(i, j)*. *N* denotes the block size. *C(i)* and *C(j)* are normalization factors for the row and column, respectively. *P(x, y)* refers to the pixel intensity at coordinate (x, y) in the original image block. The resulting frequency domain is then quantized using a quantization matrix, producing an image with compressed pixel values. Pixel values with higher frequency components are compressed more aggressively, while those with lower frequency components are compressed more minimally in order to preserve more important information.

3.3.2 ELA Calculation

Data preprocessing continues with the calculation of ELA. The resaved image is compared with the original image to calculate the error level between the two. This comparison is performed on 8 × 8 blocks across each color channel. The ELA calculation is shown in Equation (2).

$$DIFF(i,j) = f(i,j) - f_{comp}(i,j) \quad (2)$$

DIFF(i, j) represents the pixel difference value at position (i, j). *f(i, j)* refers to the pixel value at position (i, j) in the original image, while $f_{comp}(i, j)$ denotes the pixel value at the same position in the compressed image.

By performing this calculation, a visualization of the compression error in the image is obtained, which can help in identifying manipulated areas more easily. Brightness enhancement is applied to make the intensity of the error level more visible. Figure 5 visualizes the image produced after ELA calculation. This image then scaled using scaling technique.

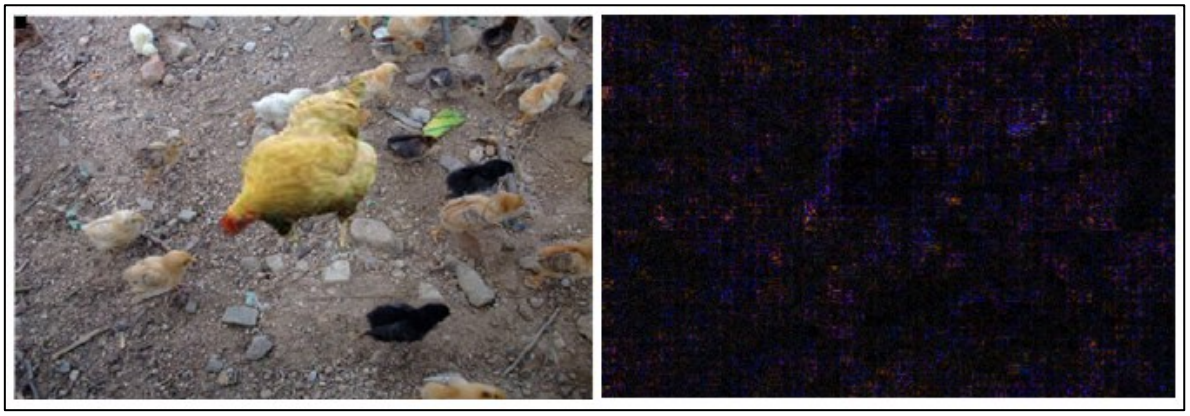

Figure 5 Example of ELA Produced Image

To apply the scaling, a scaling factor is needed, which is determined by first finding the maximum difference value (*maxdiff*). This *maxdiff* value is then used to compute the scaling factor by dividing it by the maximum color intensity value, which is 255. The formula for determining the scaling factor is shown in Equation (3). From the equation below, *scale* refers to the new pixel intensity value after brightness enhancement. *max_diff* represents the maximum pixel difference value obtained from the comparison between the original and compressed images.

$$scale = \frac{255.0}{\max\_diff} \tag{3}$$

To further support the explanation of ELA values, Figure 6 presents a detailed visualization of an 8 × 8 pixel block extracted from the green (G) channel of the ELA image shown in Figure 6.

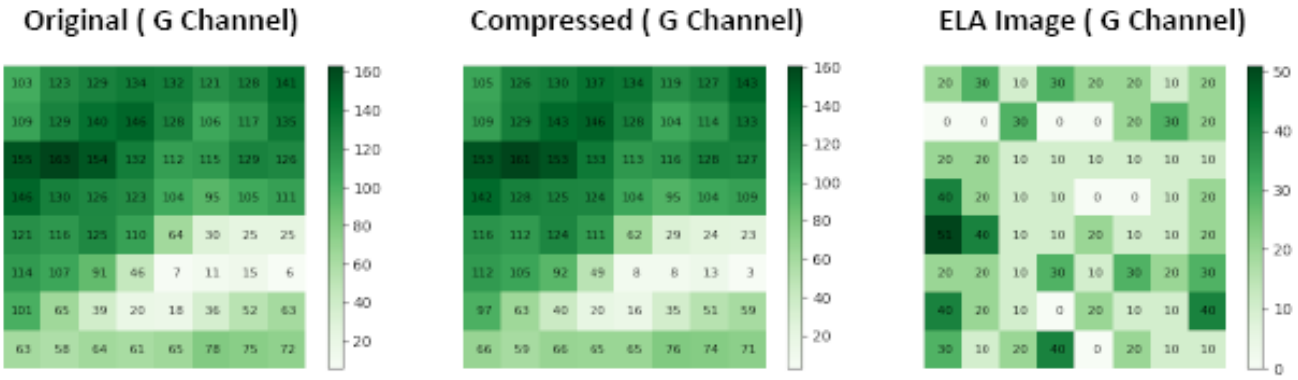

Figure 6 ELA Visualization of an 8 × 8 Block (G Channel)

Figure 6 provides a comparative visualization of an 8 × 8 pixel block from the green (G) channel, illustrating the original image, the resaved (compressed) version, and the resulting ELA image. The

leftmost matrix shows the pixel intensity values of the original image, while the middle matrix presents the same region after JPEG compression. The rightmost matrix displays the ELA image, which is obtained by calculating the absolute difference between the original and compressed images, then multiplying it with scaling factor. This error map highlights the compression artifacts, where brighter values indicate greater differences and thus potential signs of manipulation. As observed in the ELA image, several pixel positions exhibit notably high error values (e.g., 40, 30), suggesting inconsistencies in compression typical of altered regions. This visualization effectively demonstrates how localized error level differences, especially when viewed at the channel level, can serve as strong indicators of image tampering.

3.3.3 Resizing

The Error Level Analysis (ELA) images are resized to a resolution of 224 × 224 pixels to align with the standard input size required by the MobileNetV2 convolutional neural network. MobileNetV2 is specifically designed to accept input images of this fixed dimension, and adhering to this requirement is essential for the model to function correctly. Resizing the ELA images ensures that each image maintains a consistent format suitable for feature extraction and classification. This step also helps preserve the spatial and structural integrity of the visual content, minimizing distortion or the loss of subtle compression artifacts that are crucial for effective analysis. By standardizing the input size, the model can leverage its pre-trained weights more effectively, leading to better performance during training and inference.

3.3.4 Normalization

After resizing, the ELA images undergo a normalization process to prepare the pixel values for input into the MobileNetV2 model. This normalization involves scaling the original pixel values, which typically range from 0 to 255, down to a standardized range. Specifically, each pixel value is divided by 255, resulting in values that fall within the [0, 1] range.

However, the MobileNetV2 architecture, when implemented using the Keras library, expects input values to be in the range of [-1, 1] for optimal performance, particularly when using pre-trained weights. To meet this requirement, an additional step is often applied to convert the scaled [0, 1] values to the [-1, 1] range. This is usually achieved by multiplying the [0, 1] values by 2 and subtracting 1 from each result. This transformation helps ensure that the input data distribution is consistent with the conditions under which the original model was trained, thereby improving convergence, reducing training time, and enhancing model accuracy.

3.4 K-Fold Cross Validation

A k-fold cross-validation approach was used for training the model and splitting the dataset into training and testing sets. In this approach, the dataset is divided into k parts, referred to as *folds*. The training and testing processes are repeated k times, with each part of the dataset taking turns as the testing set while the remaining parts are used for training. The proportion of training and testing data is determined by the value of k. In this study, k was set to 5, resulting in an 80% training and 20% testing data split. With $k = 5$ and a total of 4,128 images, each fold consists of 826 test samples and 3,302 training samples. This step is visualized at Figure 7 below.

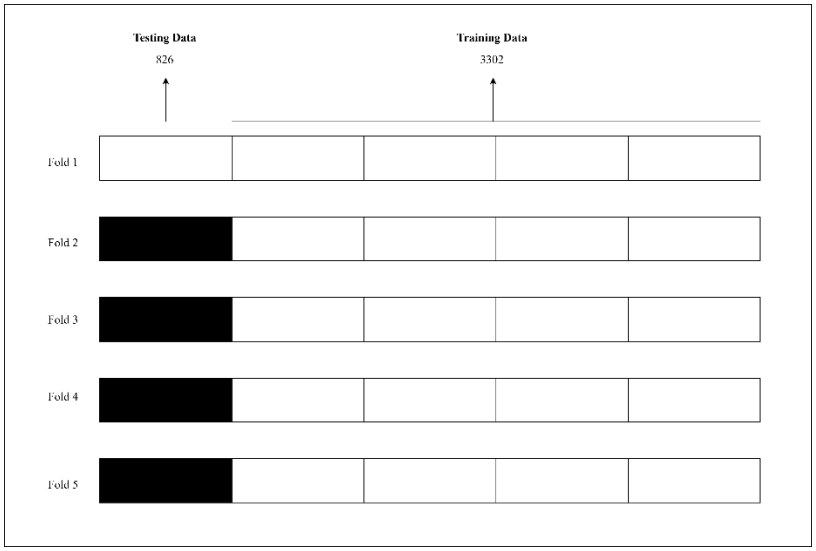

Figure 7 K-Fold Cross Validation Mechanism

3.5 Model Building

In this study, the pre-trained MobileNetV2 model from the Tensorflow/Keras version 2.18.0 was used. To build this model, the pre-trained network was loaded without the default top layer. A new top layer was constructed by adding several layers, including a global average pooling layer, a dense (fully connected) layer with 512 neurons, and a dropout layer with values varied according to the testing scenarios. The final dense layer uses an activation function to map the output values into the range of 0 to 1. The output from this last layer is interpreted as the classification result for the binary classification task, using a threshold of 0.5. A detailed visualization of the complete model architecture, including all added layers, is presented in Figure 8.

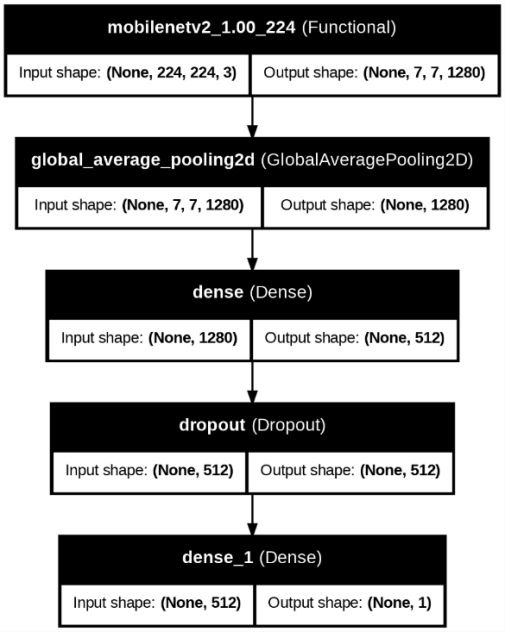

Figure 8 Model Architecture

As part of the testing scenarios, the unfreeze method was also employed. This method allows some of the frozen layers in the pre-trained model to be retrained. This process, called fine-tuning, is used to maximize the model's performance.

3.6 Model Evaluation

The final stage involves evaluating the model using several performance metrics derived from the confusion matrix. The metrics used include accuracy, precision, recall, and F1-score. The formulas for these metrics are shown in Equations (4) through (7).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4}$$

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

4 Results and Discussion

4.1 Research Scenario

Table 1 Research Scenario

No	Scenario	Feature Extraction	Hyperparameter		
			Unfreeze Start	Dropout	Optimizer
1	Scenario 1	-	0. 100	0.3, 0.5	Adam, SGD, RMSprop
2	Scenario 2	ELA			

The training scenarios consist of two main setups as shown in Table 1: without ELA feature extraction and with ELA feature extraction. Both scenarios were tested with combinations of hyperparameters including the unfreeze value, dropout rate, and optimizer. The unfreeze value controls how many pre-trained layers are retrained, with variations of 100 (only the final layers retrained) and 0 (all layers retrained). The dropout rate was varied between 0.3 and 0.5 to prevent overfitting. Three types of optimizers were used: RMSprop, Adam, and SGD. As a control variable, an adaptive learning rate was applied so that the model can automatically adjust the learning speed. This was intended to reduce the number of hyperparameters that need manual tuning, allowing the experiments to focus more on architecture design and feature extraction techniques.

4.2 Scenario 1 Result (Without ELA)

The test results of MobileNetV2 without ELA feature extraction showed that the model's performance varied greatly depending on the combination of hyperparameters used. Overall, the RMSprop and Adam optimizers tended to yield better results compared to the SGD optimizer. The best result was observed in experiment number 1, which combined the RMSprop optimizer, a dropout

rate of 0.3, and unfreezing all layers. This experiment achieved an accuracy of 0.7641 and an F1-score of 0.7693. Table 2 shows the entire evaluation result of scenario 1.

Table 2 Scenario 1 Evaluation Results

No	Unfreeze Start	Dropout	Optimizer	Evaluation Metrics			
				Accuracy	Precision	Recall	F1-Score
1	**0**	**0.3**	**RMSprop**	**0.7641**	**0.7546**	**0.7839**	**0.7682**
2			Adam	0.7597	0.7559	0.7670	0.7609
3			SGD	0.5421	0.5133	0.5280	0.4560
4			RMSprop	0.7597	0.7559	0.7670	0.7609
5		0.5	Adam	0.7641	0.7522	0.7889	0.7693
6			SGD	0.5196	0.5224	0.4898	0.5041
7			RMSprop	0.7556	0.7468	0.7727	0.7594
8	100	0.3	Adam	0.7490	0.7393	0.7698	0.7538
9			SGD	0.5196	0.5224	0.4898	0.5041
10			RMSprop	0.7459	0.7392	0.7623	0.7497
11		0.5	Adam	0.7510	0.7384	0.7780	0.7527
12			SGD	0.5380	0.5223	0.4891	0.4320

With the same dropout and unfreeze values, the Adam optimizer produced results slightly below the best performance mentioned earlier. On the other hand, the SGD optimizer was unable to optimize the model effectively, resulting in poorer performance compared to both Adam and RMSprop optimizers. Based on the dropout value analysis, it was found that a dropout rate of 0.5 was more effective when used with the Adam optimizer.

This is evident from the model performance results using Adam at different dropout values. In this scenario, the Adam optimizer requires stronger regularization to prevent overfitting. From the tests, the model without ELA requires more fine-tuning. Models trained by unfreezing all layers achieved better performance. By training more neurons, the model has the opportunity to adjust weights across all layers, which can lead to improved performance. To examine the model's performance in greater detail, an analysis was conducted on the best-performing model in this scenario using loss and accuracy graphs per epoch.

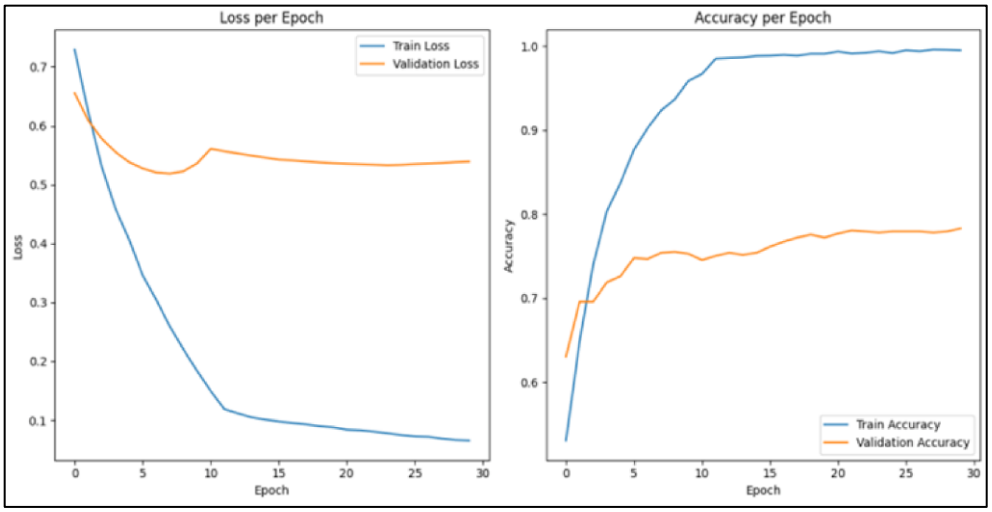

Figure 9 Loss Per Epoch and Accuracy Per Epoch Graph (Scenario 1)

Based on the loss and accuracy per epoch graphs shown in Figure 9, the training loss decreases rapidly before eventually leveling off. However, the validation loss does not decrease significantly. This indicates the presence of overfitting in the model, meaning it is unable to generalize well to the validation data.

4.3 Scenario 2 Result (With ELA)

Table 3 Scenario 2 Evaluation Results

No	*Unfreeze Start*	*Dropout*	*Optimizer*	Evaluation Metrics			
				Accuracy	*Precision*	*Recall*	*F1-Score*
1	0	0.3	RMSprop	0.9259	0.8931	0.9674	0.9287
2			Adam	0.9293	0.8995	0.9664	0.9317
3			SGD	0.6565	0.6860	0.5816	0.6265
4		0.5	RMSProp	0.9261	0.8938	0.9669	0.9289
5			Adam	0.9273	0.8964	0.9666	0.9301
6			SGD	0.6235	0.6255	0.7270	0.6595
7	**100**	**0.3**	**RMSprop**	**0.9310**	**0.9008**	**0.9683**	**0.9334**
8			Adam	0.9297	0.8980	0.9699	0.9325
9			SGD	0.7219	0.7682	0.6560	0.6974
10		0.5	RMSProp	0.9235	0.8884	0.9684	0.9267
11			Adam	0.9273	0.8961	0.9670	0.9301
12			SGD	0.7185	0.7680	0.6414	0.6863

The testing of MobileNetV2 using the ELA technique in scenario 2, shown in Table 3, yielded consistently high model performance across all experiments. The best-performing model achieved an accuracy of 0.9310 and an F1-score of 0.9334. These results were obtained in experiment number 7, with a model configured using the RMSprop optimizer, a dropout rate of 0.3, and an unfreeze start value of 100.

In this scenario, the model using the SGD optimizer performed the worst among all models. Additionally, within this scenario, using a smaller dropout rate resulted in better model performance.

For example, with the same unfreeze start value, experiments 7, 8, and 9 achieved higher performance compared to experiments 10. 11, and 12.

Based on the fine-tuning analysis, it was found that the model's performance improves when only a small number of neurons in the final layer are retrained. It can be seen that with the same dropout and optimizer values, the SGD model with the lowest accuracy of 0.6235 was able to reach an accuracy of 0.7219 when trained with minimal fine-tuning.

This scenario also shows that the model only requires a small dropout rate to achieve optimal performance. The model does not need much regularization because the data patterns are already effectively captured. Further analysis of the best-performing model in this scenario is presented below.

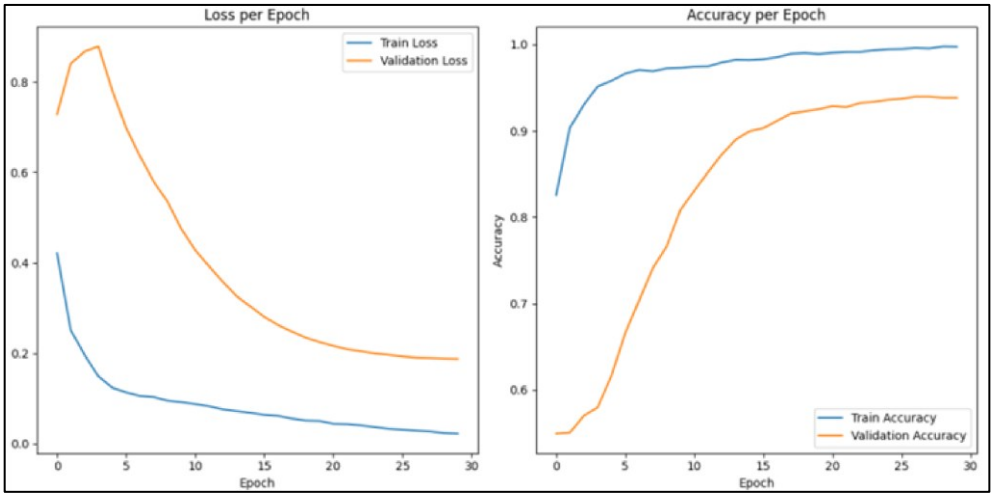

Figure 10 Loss Per Epoch and Accuracy Per Epoch Graph (Scenario 2)

The loss graph in Figure 10 shows a stable decrease in training loss values approaching zero. This indicates that the model training successfully learned the patterns contained in the data. For the validation loss, there is an increase during the initial epochs. However, the validation loss then continues to decrease until reaching a stable point. The accuracy graph in Figure 10 shows that despite the initial increase in loss, accuracy also improves, indicating that the learning process proceeds well without overfitting. This trend, along with the steadily decreasing validation loss, confirms that the model is not underfitting. Underfitting typically occurs when both training and validation performance remain poor due to the model's inability to capture underlying patterns. In contrast, the results presented in figure 8 show that the model still achieves high accuracy and low loss, both in training and validation sets, indicating a well trained model.

4.4 Experiment Results Comparison

Table 4 Best Result from Each Scenario

Scenario	Feature Extractor	Unfreeze Start	Dropout	Optimizer	Evaluation Metrics			
					Accuracy	Precision	Recall	F1-Score
Scenario 1 Number 1	-	0	0.3	RMSprop	0.7641	0.7546	0.7839	0.7682
Scenario 2 Number 7	**ELA**	**100**	**0.3**	**RMSprop**	**0.9310**	**0.9008**	**0.9683**	**0.9334**

The experiment results comparison on Table 4 shown that scenario 2 with ELA feature extraction performs significantly better than scenario 1 without ELA. In scenario 2, the model achieved an accuracy of 93.10% and an F1-score of 93.34%, whereas scenario 1 only reached an accuracy of 76.41% and an F1-score of 76.82%. This difference indicates that ELA helps the model capture important patterns in the form of error levels that are not present in the original images. The results also demonstrate that MobileNetV2 combined with ELA has superior capability in classifying authentic and tampered images, with a high recall value (96.83%) and balanced overall performance. Furthermore, the experiments show that the RMSprop optimizer is the most suitable for MobileNetV2 in this classification task, with an optimal dropout rate of 0.3 to reduce the risk of overfitting.

4.5 Qualitative Prediction Results Using ELA Technique

In this research, the Error Level Analysis (ELA) technique was employed as a feature extractor during the data preprocessing stage. Based on the approach, authentic images, which are uniformly compressed, tend to have consistent error levels throughout visually. In contrast, tampered or fake images typically exhibit uneven error levels, especially in manipulated regions which appear either brighter or darker than the rest of the image under ELA visualization. However, visually analyzing ELA images can be challenging, particularly in cases where the manipulation is subtle or the image quality is low. The interpretation of ELA results is inherently subjective and may vary depending on the observer's experience. Therefore, in this study, ELA images were further processed and fed into a classification model to automate and enhance the accuracy of forgery detection.

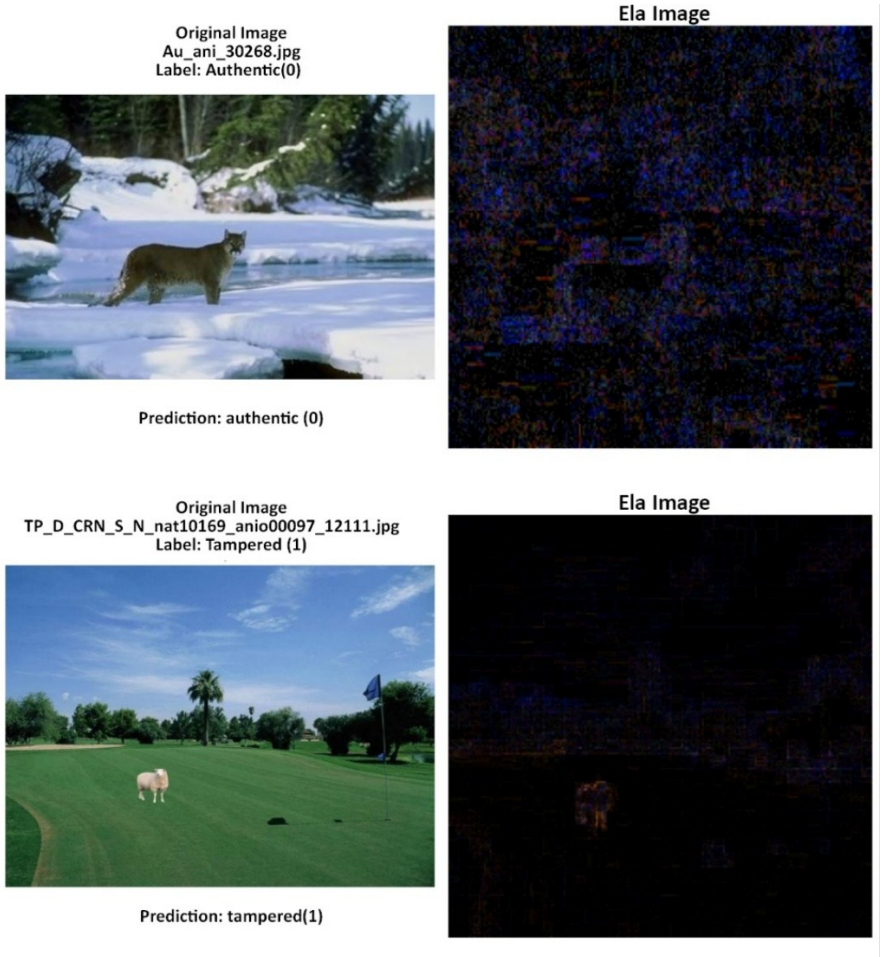

Figure 11 Visualization of ELA of Real and Fake Images

In this experiment, image samples were analyzed using ELA as feature extraction technique and MobileNetV2 model. The prediction results, as illustrated in Figure 11, show a difference between original and tampered images. For instance, in an original image, the ELA result displayed uniform brightness levels across the entire image. Conversely, in a tampered image, the manipulated object exhibited noticeably different error levels, thereby serving as visual proof of alteration.

5 Conclusion

Based on the results of this study, it can be concluded that the use of the ELA feature extraction technique on the MobileNetV2 architecture for authentic and tampered image classification performs better compared to the model without ELA feature extraction. The model with ELA feature extraction achieved an accuracy of 93.10% and an F1-score of 93.34%. The best model was obtained by setting the unfreeze hyperparameter starting from the 100th layer, using a dropout rate of 0.3, and employing the RMSprop optimizer. Overall, the combination of the MobileNetV2 model with the ELA feature extraction technique provides good results in the task of authentic and tampered image classification.

Bibliography

[1] A. Passarella, "A survey on content-centric technologies for the current Internet: CDN and P2P solutions," *Comput. Commun.*, vol. 35, no. 1, pp. 1–32, Jan. 2012, doi: 10.1016/j.comcom.2011.10.005.

[2] M. Zanardelli, F. Guerrini, R. Leonardi, and N. Adami, "Image forgery detection: a survey of recent deep-learning approaches," *Multimed. Tools Appl.*, vol. 82, no. 12, pp. 17521–17566, May 2023, doi: 10.1007/s11042-022-13797-w.

[3] S. J. Nightingale, K. A. Wade, and D. G. Watson, "Can people identify original and manipulated photos of real-world scenes?," *Cogn. Res. Princ. Implic.*, vol. 2, no. 1, p. 30, Dec. 2017, doi: 10.1186/s41235-017-0067-2.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Mar. 21, 2019, *arXiv*: arXiv:1801.04381. doi: 10.48550/arXiv.1801.04381.

[5] R. R. Choudhary, S. Paliwal, and G. Meena, "Image Forgery Detection System using VGG16 UNET Model," *Procedia Comput. Sci.*, vol. 235, pp. 735–744, 2024, doi: 10.1016/j.procs.2024.04.070.

[6] Y. Rao and J. Ni, "A deep learning approach to detection of splicing and copy-move forgeries in images," in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, Abu Dhabi, United Arab Emirates: IEEE, Dec. 2016, pp. 1–6. doi: 10.1109/WIFS.2016.7823911.

[7] N. S. A. Alsandi, "Image Splicing Detection Scheme Using Surf and Mean-LBP Based Morphological Operations," *Sci. J. Univ. Zakho*, vol. 9, no. 4, pp. 178–183, Dec. 2021, doi: 10.25271/sjuoz.2021.9.4.866.

[8] Thadomal Shahani Engineering College, India, N. K Hebbar, A. S Kunte, and Thadomal Shahani Engineering College, India, "TRANSFER LEARNING APPROACH FOR SPLICING AND COPY-MOVE IMAGE TAMPERING DETECTION," *ICTACT J. Image Video Process.*, vol. 11, no. 4, pp. 2447–2452, May 2021, doi: 10.21917/ijivp.2021.0348.

[9] S. Shikalgar, R. K. Yadav, and P. N. Mahalle, "Lightweight MobileNet Model for Image Tempering Detection," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 11, no. 5, pp. 55–69, May 2023, doi: 10.17762/ijritcc.v11i5.6524.

[10] H. Phan-Xuan, T. Le-Tien, T. Nguyen-Chinh, T. Do-Tieu, Q. Nguyen-Van, and T. Nguyen-Thanh, "Preserving Spatial Information to Enhance Performance of Image Forgery

Classification," in *2019 International Conference on Advanced Technologies for Communications (ATC)*, Hanoi, Vietnam: IEEE, Oct. 2019, pp. 50–55. doi: 10.1109/ATC.2019.8924504.

[11] L. Zheng, Y. Zhang, and V. L. L. Thing, "A survey on image tampering and its detection in real-world photos," *J. Vis. Commun. Image Represent.*, vol. 58, pp. 380–399, Jan. 2019, doi: 10.1016/j.jvcir.2018.12.022.

[12] Tian-Tsong Ng, Shih-Fu Chang, and Qibin Sun, "Blind detection of photomontage using higher order statistics," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, Vancouver, BC, Canada: IEEE, 2004, p. V-688-V–691. doi: 10.1109/ISCAS.2004.1329901.

[13] C. Jia, M. Long, and Y. Liu, "Enhanced face morphing attack detection using error-level analysis and efficient selective kernel network," *Comput. Secur.*, vol. 137, p. 103640, Feb. 2024, doi: 10.1016/j.cose.2023.103640.

[14] N. A. N. Azhan, R. A. Ikuesan, S. A. Razak, and V. R. Kebande, "Error Level Analysis Technique for Identifying JPEG Block Unique Signature for Digital Forensic Analysis," *Electronics*, vol. 11, no. 9, p. 1468, May 2022, doi: 10.3390/electronics11091468.

[15] J. Dong, W. Wang, and T. Tan, "CASIA Image Tampering Detection Evaluation Database," in *2013 IEEE China Summit and International Conference on Signal and Information Processing*, Beijing, China: IEEE, Jul. 2013, pp. 422–426. doi: 10.1109/ChinaSIP.2013.6625374.