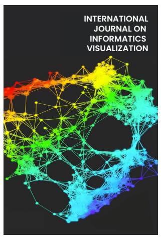
JOIV : Int. J. Inform. Visualization, 9(2) - March 2025 770-778

INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv



A Better Performance of GAN Fake Face Image Detection Using Error Level Analysis-CNN

Maria Ulfah Siregar [a,*], Nurochman Nurochman [a], Anif Hanifa Setianingrum [b], Dwi Larasati [a], William Santoso [b], Meisia Dhea Stefany [a]

[a] *Department of Informatics, UIN Sunan Kalijaga, Jl Marsda Adisucipto, Sleman, Indonesia*
[b] *Department of Informatics, UIN Syarif Hidayatullah, Jl. Ir. H. Djuanda, Tangerang Selatan, Indonesia*
Corresponding author: [*]maria.siregar@uin-suka.ac.id

***Abstract*— The use of face images has been widely established in various fields, including security, finance, education, social security, and others. Meanwhile, modern scientific and technological advances make it easier for individuals to manipulate images, including those of faces. In one of these advancements, the Generative Adversarial Network method creates a fake image similar to the real one. An error-level analysis algorithm and a convolutional neural network are proposed to detect manipulated images generated by generative adversarial networks. There are two scenarios: a stand-alone convolutional neural network and a combination of error-level analysis and a convolutional neural network. Furthermore, the combined scenario has three sub-scenarios regarding the compression levels of the error-level analysis algorithm: 10%, 50%, and 90%. After training the data obtained from a public source, it becomes evident that using a convolutional neural network combined with compression of error level analysis can improve the model's overall performance: accuracy, precision, recall, and other parameters. Based on the evaluation results, it was found that the highest quality convolutional neural network training was obtained when using 50% error level analysis compression because it could achieve 94% accuracy, 93.3% precision, 94.9% recall, 94.1% F1 Score, 98.7% ROC-AUC Score, and 98.8% AP Score. This research is expected to be a reference for implementing image detection processes between real and fake images from generative adversarial networks.**

***Keywords*— Compression level; evaluation; manipulated images; real image; training.**

Manuscript received 30 Apr. 2024; revised 24 Aug. 2024; accepted 1 Oct. 2024. Date of publication 31 Mar. 2025.

I. Introduction

The use of facial images has been widely established in various fields, including security, finance, education, social security, and others [1], [2]. Meanwhile, modern scientific and technological advances make it easier for individuals to manipulate images, including those of faces. Problems relating to images have been considered for many years, such as in [2] or as said by Krawetz cited in [3]. Thus, the capability to recognize phony pictures cannot be avoided, and it is necessary in this era [4], either the phony picture is obtained after being affected by a global or local perturbation [5]. On the other hand, synthetic images are sometimes inevitably needed because the real image is difficult to get [6] or is it too costly to get [7].

In one of the technological advancements, the Generative Adversarial Network (GAN) method was used to create a fake image that is very similar to the original image. GAN belongs to deep learning techniques [8] which makes it hard for humans to identify fake or real images. It is because the phone pictures are so real. Thus, GAN can produce realistic fake images or deepfakes, which can be formed even when the original images are not accessed, as said by Hitaj in [9]. GAN can also be used to produce synthetic data in the case of rare data, such as to produce synthetic chest X-ray (CXR) images during the last COVID [6] . Moreover, GAN is a powerful help in augmenting data compared to classical data augmentation [6], and optimizing problems [10]. In previous studies, several methods have been proposed for the detection of fake images generated by GANs [11], [12]. As part of the proposed study, Error Level Analysis (ELA) and Convolutional Neural Network (CNN) algorithms are proposed for detecting manipulated images generated by GANs. As the availability of applications that could produce deepfake images is growing massively, it is challenging for researchers to contribute to this issue. On the other hand, the use of images in social content is increasing rapidly and facing new/first-time challenges [8], [13], [14].

Using the GAN method, it is possible to identify manipulated face images produced by the GAN [11]. This technique is used to remove GAN fingerprints from the fake image. The results indicate that additional efforts are still required to develop a reliable system for detecting image manipulation. According to another research, face features were encrypted using a GAN [1]. As a one-way process, encryption effectively protects face features. A certain study attempted to solve the problem of blurred images [15], [16]. GAN in [16] restored blurred face and body images. GANs have also helped to increase the performance of CNN in medical image classification, as said by Frid-Adar et al. in [17]. GAN is used to generate medical images, and then they are used for synthetic data augmentation. The generated or synthetic data should have high quality [18]. Other research has also experienced generating data, which is then treated as unlabeled samples, as said by Xin and Huang in [19]. Other research regarding GAN proposes that GAN bridges the gap in person re-identification (ReID), as said by Wei in [20]. In that research, the proposed method is Person Transfer GAN (PTGAN). Social GAN is challenging research relating to GAN, which predicts human motion, as said by Gupta et al. in [21], predicts a future path for an agent, as said by Sadeghian et al. in [14], etc. The GAN, which is built here, is based on an encoder-decoder architecture. GANs, which is a relatively new framework used to estimate generative models through an adversarial process, as said by Goodfellow et al in [22], involves simultaneously training two models: a generative model *G* that represents the distribution of data, and a discriminative model *D* that estimates the probability that one sample comes from a different training set than the one produced by *G*. A training procedure for *G* is designed to maximize the chances that it will make an error. GANs have been integrated with CNNs, resulting in a class of CNNs, namely, Deep Convolutional Generative Adversarial Networks (DCGAN). This class has architectural limitations and is claimed to be a strong candidate for unsupervised learning, as said by Radford in [23].

It has been found that other researchers have conducted research that combined ELA and Deep Learning (DL). One of them is a study, as said by Gunawan et al. in [24]. In that study, 1771 images with tampered labels and 2940 images with real labels were taken from the CASIA dataset. ELA is a forensic method for identifying parts of an image based on different compression levels [25]. In other words, the error level will be computed, as said by Jeronymo et al. in [26], where an image will usually be divided into 8x8 small image blocks and compressed using JPEG to 95% accuracy. Each block will provide the same level of compression quality. If there are blocks with different compression qualities, this indicates that manipulations have occurred [4] , [27], in which the manipulated ones have higher error potential compared to the unmanipulated part of images, as said by Gunawan et al. in [28]. The ELA process, which is a multimodal data analysis technique [29]. This can be accomplished by saving the image using a specified level of compression quality, calculating and observing the difference between these levels of compression quality, which is to extract the noise from the image [5] [30]. ELA is one of the advanced image analysis, as said by Krawetz in [3], but it suffers from image noise, which is too intense [26]. There are other studies which use ELA to improve detection accuracy, one example is presented in [5]. There, ELA is conducted after JPEG compression. Features are extracted using SIFT in classification. They used datasets from ImageNet and Caltech-256. The accuracies outperform the state-of-the-art methods.

The CNN, which is in the deep learning area, was developed initially to recognize handwriting, which then proved capable of solving the problems of image recognition, detection, and segmentation, as said by Sudiatmika et al. in [31]. Since using deep learning techniques, CNN could be used to differentiate between real and fake images [8]. CNN has a remarkable ability to classify large-scale images. This capability is influenced by the arrangement of a CNN, which consists of three layers: the convolutional layer, the pooling layer, and the fully connected layer. On the other hand, the use of deep learning has privacy implications, as a result of being trained in a centralized technique, as said by Hitaj in [9].

This study aims to compare the accuracy and other metrics of the identification of face images. The identification is done using the CNN method. Before this, ELA compression will be implemented to enhance accuracy. Thus, the contribution of this study is to devise a better method to detect fake faces by combining ELA and CNN and using the GAN face dataset.

This proposed study differs from the study in [2], [32], such as in the dataset that is used. Those studies use datasets from CASIA 2.0 [2], [32] and MICC F200 [2]. Ref. [2] uses 90% ELA. Ref. [32] outperforms existing training time and efficiency of the state-of-the-art deep learning models. The same with [2]. The system is better than the existing methods.

This paper is organized as follows: Section 1 discusses the background, related works, and objectives. Section 2 explains the method for conducting this research. Section 3 gives the results and discusses them. Section 4 concludes the paper.

II. Materials and Method

The research begins with collecting image data from public datasets. It is then followed by conducting a literature review. The next step involves image pre-processing, edge detection with the Canny Edge Detector/image compression with ELA, detection by CNN, and accuracy analysis. To detect real and fake images, two scenarios are used: one in which the image is compressed with ELA (ELA-CNN), and one in which the image is not compressed with ELA (non-ELA-CNN).

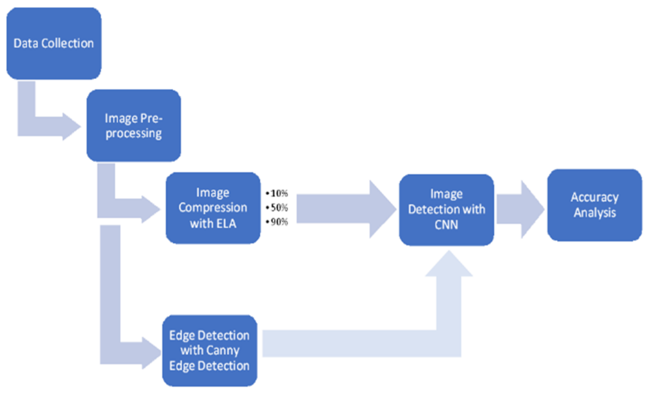


Fig. 1 Method of the research

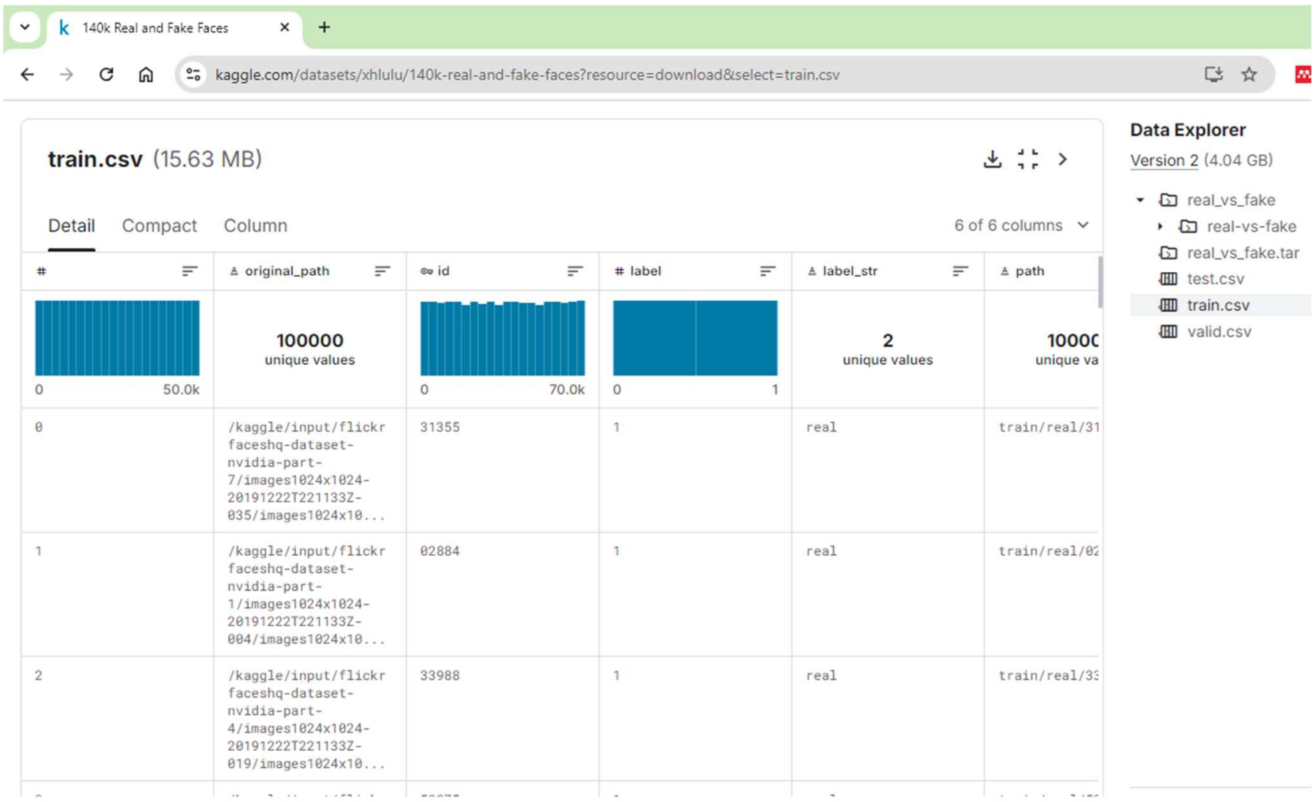
For the ELA-CNN scenarios, there are three sub-scenarios, namely 10%, 50%, and 90% compression levels. Before the CNN occurs, the Canny Edge Detector will be implemented for the non-ELA-CNN scenario. A Canny Edge Detector is one edge detection method that incorporates several stages of edge detection in an image [33]. This operator was developed by John F. Canny in 1986. Fig. 1 shows the method used in this research.

A. Data Collection

This study used a public dataset from Kaggle, namely 140k real and fake faces (xhlulu) [34]. The dataset consists of 70k real faces from the Flickr dataset, which Nvidia collected, and 70k fake faces sampled from 1 million fake faces generated by StyleGAN. We chose the dataset because it provides users with GAN-generated fake images. Thus, this dataset suits the aim of this research.

The images are 256 pixels and divided into three folders: train, validation, and test. The training dataset contains 100,000 faces, divided into two. Thus, there will be 50,000 real face images and 50,000 fake face images. The validation dataset contains 20,000 faces with 10,000 real face images and 10,000 fake face images. Dataset testing has the same explanation as dataset validation.

Each folder consists of six columns: [blank] with type numeric for numbering the rows automatically, *original_path* with type String, *id* is a primary key with type numeric, *label* which has two values: 0 or 1, *label_str* is a String: real or fake, and *path* with type String. Label 1 is for real images, otherwise, 0 is for fake images. Fig. 2 depicts the screenshot of these columns from the folder train; the size of the file is 15.63 MB.


140k Real and Fake Faces

kaggle.com/datasets/xhlulu/140k-real-and-fake-faces?resource=download&select=train.csv

train.csv (15.63 MB)

Detail Compact Column

6 of 6 columns

#	original_path	id	label	label_str	path
0 – 50.0k	100000 unique values	0 – 70.0k	0 – 1	2 unique values	1000C unique va
0	/kaggle/input/flickrfaceshq-dataset-nvidia-part-7/images1024x1024-20191222T221133Z-035/images1024x10...	31355	1	real	train/real/31
1	/kaggle/input/flickrfaceshq-dataset-nvidia-part-1/images1024x1024-20191222T221133Z-004/images1024x10...	02884	1	real	train/real/02
2	/kaggle/input/flickrfaceshq-dataset-nvidia-part-4/images1024x1024-20191222T221133Z-019/images1024x10...	33988	1	real	train/real/33

Data Explorer

Version 2 (4.04 GB)

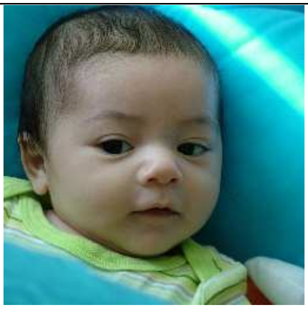
- real_vs_fake
  - real-vs-fake
- real_vs_fake.tar
- test.csv
- train.csv
- valid.csv

Fig. 2 Columns of the dataset ([34])

Some of the images are given in Table 1. These images are taken from the train folder.

TABLE I

SOME IMAGES FROM THE TRAIN FOLDER

Real	Fake
 00000.jpg	0AVBZYCGE9.jpg
00009.jpg	0A52ADV5M5.jpg

B. Image Pre-processing

The size of the images obtained from a public dataset makes it difficult for the system to run on Google Colab. Thus, the size should be managed and normalized so that it suits the requirements of [35] Google Collab. The online running option allows the GPU to speed up the running process, which reduces the image size to 150 pixels at the pre-processing stage.

C. Edge Detection with Canny Edge Detection

For detection scenarios without ELA, image pre-processing is continued by applying the Canny Edge Detector. The steps in this detection are:

- Convert RGB to YCbCr
- Contrast adjustment
- Convert YCbCr to Grayscale
- Apply the Canny Edge Detector on the gray image

D. Image Detection by CNN

As can be seen in Table 2, the parameters for CNNs used in this study are given. Meanwhile, Fig. 3 shows the CNN architecture used in this study. The CNN architecture is designed as follows. The input on this layer is BatchNormalization. The purpose of this layer is to normalize the inputs to the network, which then helps to stabilize and accelerate training by reducing internal covariate shift. It ensures that the input to each layer has a consistent distribution, which can improve convergence rates and performance.

TABLE II
THE CNN'S PARAMETERS

Layer	Volume	Parameter
batch_normalization_input (InputLayer)	150x150x1	0
batch_normalization (BatchNormalization)	150x150x1	4
conv2d (Conv2D)	150x150x64	640
max_pooling2d (MaxPooling2D)	75x75x64	0
batch_normalization_1 (BatchNormalization)	75x75x64	256
conv2d_1 (Conv2D)	75x75x64	36928
max_pooling2d_1 (MaxPooling2D)	37x37x64	0
batch_normalization_2 (BatchNormalization)	37x37x64	256
dropout (Dropout)	37x37x64	0
conv2d_2 (Conv2D)	37x37x128	73856
max_pooling2d_2 (MaxPooling2D)	18x18x128	0
batch_normalization_3 (BatchNormalization)	18x18x128	512
dropout_1 (Dropout)	18x18x128	0
conv2d_3 (Conv2D)	18x18x256	295168
max_pooling2d_3 (MaxPooling2D)	9x9x256	0
batch_normalization_4 (BatchNormalization)	9x9x256	1024
dropout_2 (Dropout)	9x9x256	0
conv2d_4 (Conv2D)	9x9x512	1180160
max_pooling2d_4 (MaxPooling2D)	4x4x512	0
batch_normalization_5 (BatchNormalization)	4x4x512	2048
dropout_3 (Dropout)	4x4x512	0
conv2d_5 (Conv2D)	4x4x512	2359808
max_pooling2d_5 (MaxPooling2D)	2x2x512	0
batch_normalization_6 (BatchNormalization)	2x2x512	2048
dropout_4 (Dropout)	2x2x512	0
global_average_pooling (GlobalAveragePooling2D)	512	0
flatten (Flatten)	512	0
dense (Dense)	2048	1050624
dropout_5 (Dropout)	2048	0
dense_1 (Dense)	1024	2098176
dropout_6 (Dropout)	1024	0
dense_2 (Dense)	1	1025
Total params: 7,102,533		
Trainable params: 7,099,459		
Non-trainable params: 3,074		

Six layers of convolution come next. The purpose of this layer is to apply convolution operations to extract features from the input images. Each convolutional layer detects unique features such as edges, textures, and more complex patterns in the images. Convolutional layers are the core of a CNN, allowing the model to learn spatial hierarchies of features. Stacking this layer helps in capturing more complex and abstract features.

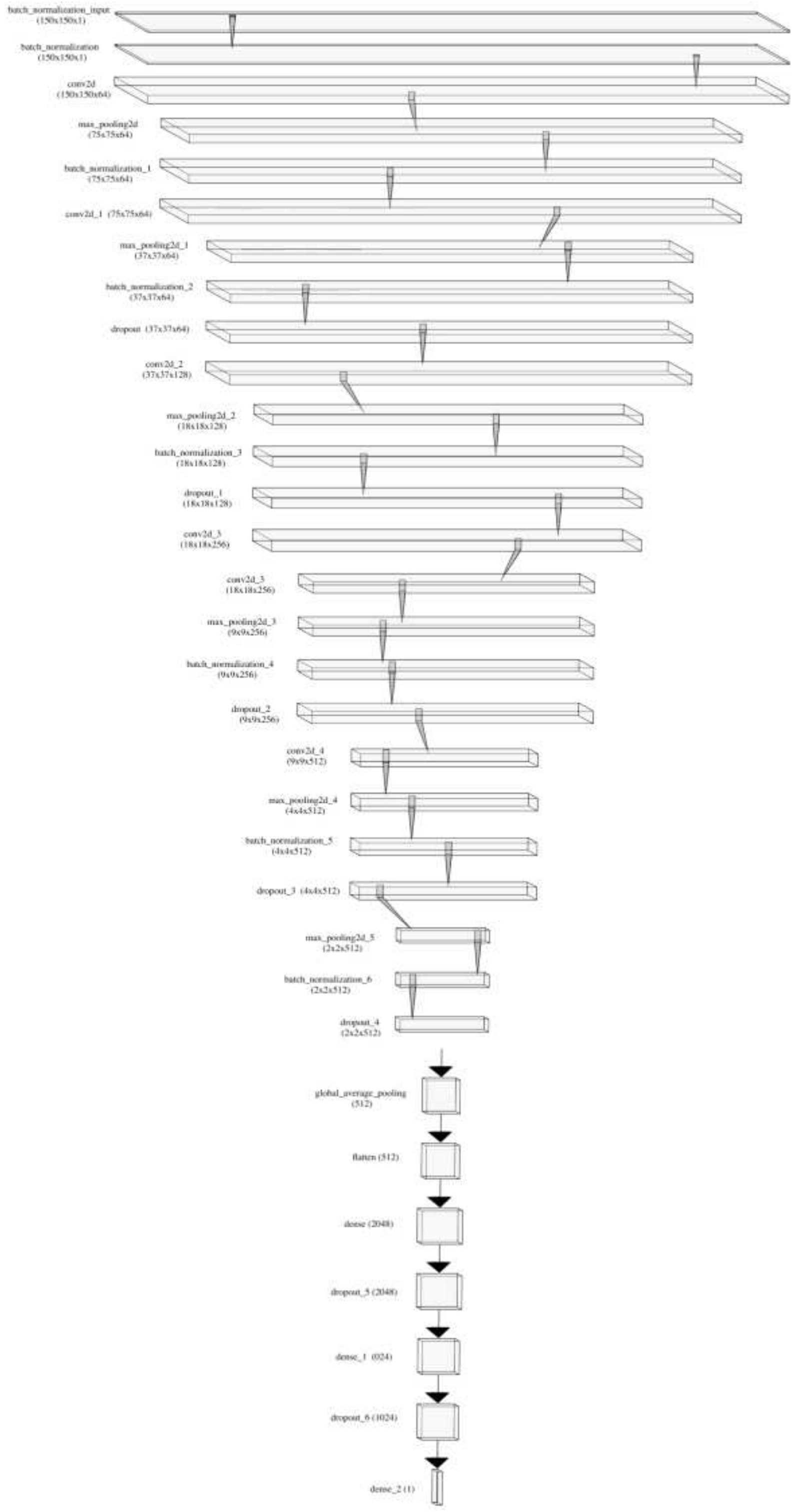
The convolutional layers include MaxPooling, BatchNormalization, and Dropout. The number of neurons on each layer is 64, 64, 128, 256, 512, and 512. Max Pooling layers down-sample the input by reducing its spatial dimensions, reducing the computational complexity and helping achieve spatial invariance. These layers reduce the dimensionality of the feature maps. It helps to prevent overfitting, reduces the computational load, and summarises the most important features. Batch normalization or intermediate layers have the same purpose as the initial batch normalization, which normalizes the activations of the previous layers. These layers help maintain the benefits of normalization throughout the network, ensuring that each layer receives appropriately scaled inputs. Dropout layers randomly set a fraction of input units to 0 at each update during training to prevent overfitting. Dropout is a regularization technique that helps make the model more robust by preventing it from being dependent on any individual neurons. This improves the generalization of unseen data.

Then, GlobalAveragePooling and Flatten layers are used. Global Average Pooling layers compute the average output of each feature map. It reduces the spatial dimensions of the feature maps to a single value per feature map. It helps prevent overfitting and is more interpretable. It also significantly reduces the number of parameters in the model. The flatten layer converts the 2D feature maps into a 1d vector. It is a necessary stage before passing the data into fully connected (dense) layers, which require a 1d input. These layers equalise the size of the convolution.

Dense layers are fully connected layers that learn the non-linear combinations of the features extracted by the convolutional layers. Dense layers at the end of the network combine the features into a higher-level representation for classification. The final dense layer with a sigmoid activation outputs a probability for the binary classification task (real or fake). In the classification, three dense layers (ANN Perceptron) are used, which are 2048 and 1024 hidden layers and one output layer. The activation function is Sigmoid. This function is chosen because the label is a binary classification [36]. The ReLU function is also used, which helps in preventing the vanishing gradient problem, making training faster and more effective. The activation function introduces non-linearities into the model, enabling it to learn more complex patterns.

In total, there are seven million parameters. The dropout rate on Convolution Layers is 0.1, and the Dropout on Dense Layers is 0.5 to avoid overfitting. The dropout rate was set to 0.1, which means that 10% of neurons will be dropped out. This number is relatively small. It helps to normalize the network without losing so much information during training. It can reduce overfitting by randomly setting input units to 0 during training. The Dropout is important in the dense layers to ensure the network generalizes well and does not overfit the training data.

The other hyperparameter is padding. In this study, it is initialized to "Same". This padding ensures that the output feature map has the same spatial dimensions as the input. It helps to maintain the spatial resolution of the input throughout the convolutional layers, which can be beneficial for capturing features at different scales and positions in the image.



batch_normalization_input (150x150x1)
batch_normalization (150x150x1)
conv2d (150x150x64)
max_pooling2d (75x75x64)
batch_normalization_1 (75x75x64)
conv2d_1 (75x75x64)
max_pooling2d_1 (37x37x64)
batch_normalization_2 (37x37x64)
dropout (37x37x64)
conv2d_2 (37x37x128)
max_pooling2d_2 (18x18x128)
batch_normalization_3 (18x18x128)
dropout_1 (18x18x128)
conv2d_3 (18x18x256)
conv2d_3 (18x18x256)
max_pooling2d_3 (9x9x256)
batch_normalization_4 (9x9x256)
dropout_2 (9x9x256)
conv2d_4 (9x9x512)
max_pooling2d_4 (4x4x512)
batch_normalization_5 (4x4x512)
dropout_3 (4x4x512)
max_pooling2d_5 (2x2x512)
batch_normalization_6 (2x2x512)
dropout_4 (2x2x512)
global_average_pooling (512)
flatten (512)
dense (2048)
dropout_5 (2048)
dense_1 (024)
dropout_6 (1024)
dense_2 (1)

Fig. 3 Architecture of CNN

Epsilon in Batch Normalization has a value of 0.001. This number is a common choice which ensures numerical stability without significantly affecting the normalized values. Then, Binary Crossentropy is the value of the loss function. This loss function is suitable for binary classification problems. It measures the difference between the predicted probabilities and the actual class labels, penalizing incorrect predictions more heavily. Next, Adam is used for the optimizer. This value is chosen for its efficiency and adaptive learning rate properties. It combines the advantages of two other popular optimizers: AdaGrad (which works well with sparse gradients) and RMSProp (which works well in non-stationary settings). Adam adjusts the learning rate for each parameter, which helps in faster convergence. The learning rate was set to 0.001 as it is a common starting point for Adam. It allows the model to converge reasonably quickly without making large updates that could overshoot the optimal solution. The learning rate decay is 1E-6. It helps reduce the learning rate gradually over time. It is useful for fine-tuning the model towards the end of training, ensuring that it does not oscillate around the minimum and instead settles into the optimal weights.

The batch size was chosen at 150 because this number compromises the computational efficiency and stability of the gradient updates. A batch size is the number of training examples utilized in one iteration. Larger batch sizes can lead to more stable gradients, whereas smaller ones can provide more frequent updates. For epochs, it was set at 10 as this number is often a starting point to observe how the model performs and to ensure that it is not overfitting or underfitting. An epoch determines how frequently the training dataset passes through the network. Bigger epochs can be added later if the model still shows improvement.

In this study, validation data is used. The separation of the validation dataset helps to monitor the model's performance on unseen data during training. It will then provide an unbiased evaluation of the model's ability to generalize, helping in early stopping or hyperparameter tuning if it is necessary. The training data is also shuffled to ensure that the model does not learn any unintended patterns due to the order of the data. Shuffling promotes better generalization and prevents the model from becoming dependent on the order of training examples. Another hyperparameter is Verbose, which is set to 1. This parameter controls the verbosity of the output during training. Setting it to 1 means that the progress of training (including loss and accuracy metrics) will be displayed for each epoch, helping to monitor the training process.

Overall, the effectiveness of the CNN architecture lies in its ability to gradually extract more complex and abstract features from the input images through multiple layers of convolutions, pooling, and normalization. This architecture is designed to be deep and complex, making it capable of learning the intricate details necessary to distinguish between real and fake images in the DeepFake function.

III. Results and Discussion

In this section, results and discussion are presented. It begins with results obtained from the training stage.

A. Training Result

Table 3 shows the results of running the system based on accuracy, loss, and time at the training and validation stages. Four scenarios were used in this research. The results indicate that the accuracy of the learning process in the detection system appears to be very good. In the same way, the results of the validation of the training process are positive. The highest accuracy and least loss are achieved when ELA with a 50% compression level is applied before CNN detection is conducted. On the other hand, cases without ELA resulted in the lowest number of results among the other three scenarios.

However, as seen in Table 3, the differences among the four scenarios are less significant. The differences in values do not reach 0.1. The same as time measured in seconds, the difference is less than 60 seconds between the longest and the shortest time.

Fig. 4 depicts the accuracy of CNN combined with ELA at 50% at the training and validation stages. Meanwhile, the model loss is given in Fig. 4 using the same percentage. From Fig. 3, overfitting is assumed not to happen because the accuracy in the training stage does not get worse when the accuracy in the validation stage is getting better. Based on the results seen in Table 4, using that percentage, the accuracy is the best among the three percentages. Thus, using this model, the overfitting can be prevented. On the other hand, it is concluded that ELA 10% and 90% tend to overfit (on validation loss), whereas ELA 50% stays stable on convergence. This indicates that ELA 50% will be the best result based on training and validation performance.

B. Testing Result

The confusion matrix is given in Fig. 5. The Confusion Matrix measures the model's performance. From the matrix, true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) can be calculated.

TABLE III
THE ACCURACY AND LOSS OF THE TRAINING AND VALIDATION STAGES

	Without ELA	**With ELA 10%**	**With ELA 50%**	**With ELA 90%**
Accuracy	0.9806	0.9836	0.9869	0.9844
Loss	0.0525	0.0458	0.0362	0.0431
Validation Accuracy	0.8906	0.9082	0.9373	0.9119
Validation Loss	0.4729	0.3780	0.2316	0.4602
Test Accuracy	0.89155	0.91054	0.93589	0.91289
Time (s)	102	161	161	157

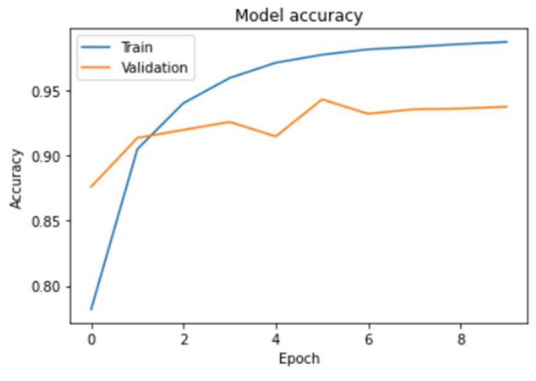


Fig. 4 Model accuracy on training and validation of CNN + ELA 50%

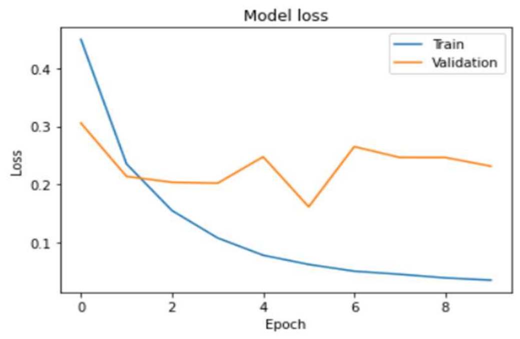


Fig. 5 Model loss on training and validation of CNN + ELA 50%

Table 4 shows the metrics of testing. It is an evaluation of the model's performance against test data. As shown in Table 4, the performance of the real and fake image detection processes is improved when the CNN process is preceded by ELA compression. Based on the evaluation of the model, it is evident that using ELA can improve the overall performance of the model that has been made. The best performance was achieved by CNN ELA training at 50% with 93.59% accuracy and 96.54% precision ($\frac{TP}{TP+FP}$), 90.42% recall ($\frac{TP}{TP+FN}$), 93.38% F1 score ($\frac{2*precision*recall}{precision+recall}$), ROC-AUC score of 98.59 %, and AP score of 98.7%. However, this scenario's recall score is not the best. The best recall score was obtained on ELA with a compression of 90%.

TABLE IV
THE ACCURACY AND LOSS OF THE TESTING STAGE

	Without ELA	With ELA 10%	With ELA 50%	With ELA 90%
Accuracy Score	0.89155	0.9105	0.9359	0.9129
Precision Score	0.86245	0.9636	0.9654	0.8644
Recall Score	0.9317	0.8533	0.9042	0.9763
F1 Score	0.89574	0.90512	0.9338	0.91809
ROC-AUC Score	0.96333	0.97756	0.9859	0.9827
AP Score	0.96671	0.9792	0.987	0.9805

Next, a comparison of the tests among three related studies is given. For this study, the test was done with ELA 50%. This is because the compression level of ELA provides the best accuracy. Table 5 shows the results in the testing stage. Although the accuracy of this study is fairly good, it is less than the one obtained from a study, such as in [32] and it is roughly the same as a study in [2].

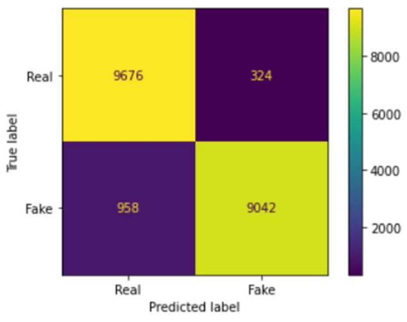


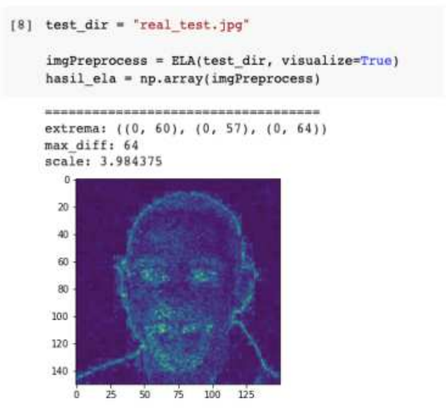
Fig. 6 Confusion matrix with CNN + ELA 50%

The former study's accuracy is 95.19%. It uses ELA, which is combined with CNN. The dataset is from CASIA 2.0, which is also a public dataset. The latter has an accuracy of 93%, and the dataset used is from CASIA 2.0 and MICC F200. The Precision metric of this study is better than the one that resulted from the study in [32]. However, for the rest, the study in [32] has beaten this study. Table 5 summarizes some of the metric comparisons of those three studies.

TABLE V
COMPARISON OF SOME METRICS OF RELATED RESEARCH

Research	Accuracy	Precision	Recall	F1
[2]	93%	-	-	-
[32]	95.19%	0.93365	0.93278	0.93278
This study	93.59%	0.9654	0.9042	0.9338

Fig. 7 (a) shows one real face image from the testing folder. This image is taken from Kaggle (xhlulu). The face result after ELA is shown in Fig. 7 (b). The testing indicates that the face is real, and it is a correct analyzing. The other testing is on a fake face image from the testing folder. Fig. 8 (a) depicts the face. The result is shown in Fig. 8 (b). The system successfully identifies the face as fake. However, some faces are identified wrongly. One example face is shown in Fig. 9 (a). ELA's result is shown in Fig. 9 (b). The model predicts that the image is fake compared to the originality of the real image. Fig. 10 (a) and (b) show another wrongly identified example. This face image is taken from fake faces in the testing folder, and the model predicts that it is real.

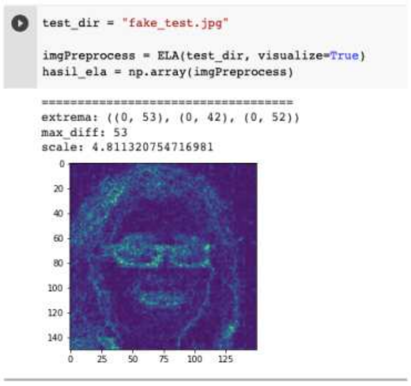


```
[8] test_dir = "real_test.jpg"

    imgPreprocess = ELA(test_dir, visualize=True)
    hasil_ela = np.array(imgPreprocess)

    ====================================
    extrema: ((0, 60), (0, 57), (0, 64))
    max_diff: 64
    scale: 3.984375
```

Fig. 7 Correct identification of a real face image: a) Testing on a real face image, b) Result of ELA on a real face image

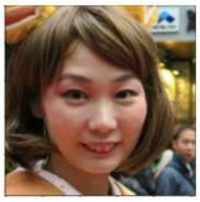
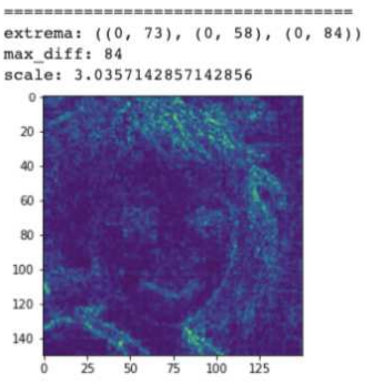


```
test_dir = "fake_test.jpg"

imgPreprocess = ELA(test_dir, visualize=True)
hasil_ela = np.array(imgPreprocess)
```

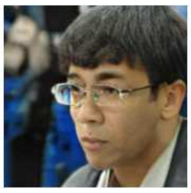
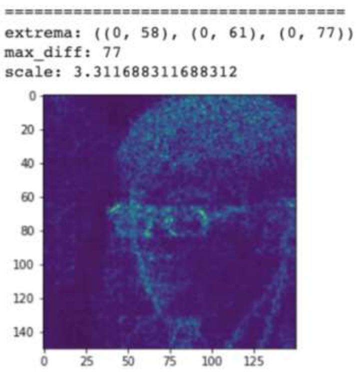
```
===================================
extrema: ((0, 53), (0, 42), (0, 52))
max_diff: 53
scale: 4.811320754716981
```

Fig. 8 Correctly identified a fake face image: a) Testing on a fake face image, b) Result of ELA on a fake face image

```
===================================
extrema: ((0, 73), (0, 58), (0, 84))
max_diff: 84
scale: 3.0357142857142856
```

Fig. 9 Incorrectly identified a real face image: a) Testing on a real face image, b) Result of ELA on a real face image

```
===================================
extrema: ((0, 58), (0, 61), (0, 77))
max_diff: 77
scale: 3.311688311688312
```

Fig. 10 Incorrectly identified a fake face image: a) Testing on a real face image, b) Result of ELA on a fake face image

IV. CONCLUSION

Generative Adversarial Networks are algorithms used to generate deep fake images through AI. The combination of the CNN algorithm with the ELA compression method can be used to determine whether an image obtained using a GAN is real or fake. From training the data, CNN with ELA compression could improve the overall performance, such as accuracy, precision, recall, or other parameters. Based on the evaluation results, it was found that the most effective CNN training was obtained when using 50% ELA compression because it can achieve 98.6% accuracy. This research is expected to be a reference for performing image detection processes between real and GAN images.

A suggestion for future research is to incorporate genetic algorithms to yield a better compression level. Another suggestion is to use a denoising technique to accompany ELA. Yet another one is to analyze fake facial images other than those produced by GAN. Furthermore, this study is planned to implement facial expression recognition.

REFERENCES

[1] C. Wu, B. Ju, Y. Wu, N. N. Xiong, and S. Zhang, "WGAN-E: A generative adversarial networks for facial feature security," *Electronics (Switzerland)*, vol. 9, no. 3, 2020, doi: 10.3390/electronics9030486.

[2] M. Patel, K. Rane, N. Jain, P. Mhatre, and S. Jaswal, "Image Forgery Detection using CNN," in *2023 3rd International Conference on Intelligent Technologies (CONIT)*, 2023, pp. 1–4. doi:10.1109/CONIT59222.2023.10205377.

[3] I. Amerini, G. Baldini, and F. Leotta, *Image and video forensics*, vol. 7, no. 11. 2021. doi: 10.3390/jimaging7110242.

[4] A. Vadrevu, R. Rajeshwari, L. Pabbathi, S. Sirimalla, and D. Vodnala, "Image Forgery Detection Using Metadata Analysis and ELA Processor BT - Innovations in Computer Science and Engineering," H. S. Saini, R. Sayal, A. Govardhan, and R. Buyya, Eds., Singapore: Springer Singapore, 2022, pp. 579–586.

[5] S. Huang, S. Wang, J. Chen, G. Li, and W. Wang, "Adversarial Examples Detection Based on Error Level Analysis and Space Mapping," in *ICASSP, IEEE International Conference on Acoustics,*

Speech and Signal Processing - Proceedings, 2022, pp. 4213–4217. doi:10.1109/ICASSP43922.2022.9747171.

[6] A. Waheed, M. Goyal, D. Gupta, A. Khanna, F. Al-Turjman, and P. R. Pinheiro, "CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection," *IEEE Access*, vol. 8, pp. 91916–91923, 2020, doi: 10.1109/ACCESS.2020.2994762.

[7] J. Phan, M. Sarmad, L. Ruspini, G. Kiss, and F. Lindseth, "Generating 3D images of material microstructures from a single 2D image: a denoising diffusion approach," *Sci Rep*, vol. 14, no. 1, pp. 1–12, 2024, doi: 10.1038/s41598-024-56910-9.

[8] A. M. Almars, "Deepfakes Detection Techniques Using Deep Learning: A Survey," *Journal of Computer and Communications*, vol. 09, no. 05, pp. 20–35, 2021, doi: 10.4236/jcc.2021.95003.

[9] B. Zeng, S. Luo, F. Yu, G. Yang, K. Zhao, and L. Wang, "GAN-based data reconstruction attacks in split learning," *Neural Networks*, vol. 185, p. 107150, 2025, doi: 10.1016/j.neunet.2025.107150.

[10] B. Zhang, Y. Yao, H. K. Kan, and W. Luo, "A GAN-based genetic algorithm for solving the 3D bin packing problem," *Sci Rep*, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-56699-7.

[11] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proença, and J. Fierrez, "GANprintR: Improved Fakes and Evaluation of the State of the Art in Face Manipulation Detection," *IEEE Journal on Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 1038–1048, 2020, doi:10.1109/JSTSP.2020.3007250.

[12] S. Venkatesh, H. Zhang, R. Ramachandra, K. Raja, N. Damer, and C. Busch, "Can GAN Generated Morphs Threaten Face Recognition Systems Equally as Landmark Based Morphs? - Vulnerability and Detection," in *IWBF 2020*, 2020.

[13] Y. Lyu, Y. Jiang, Z. He, B. Peng, Y. Liu, and J. Dong, "3D-Aware Adversarial Makeup Generation for Facial Privacy Protection," *IEEE Trans Pattern Anal Mach Intell*, vol. 45, no. 11, pp. 13438–13453, 2023, doi: 10.1109/TPAMI.2023.3290175.

[14] H. Yang *et al.*, "SoPerModel: Leveraging Social Perception for Multi-Agent Trajectory Prediction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 63, pp. 1–13, 2025, doi:10.1109/TGRS.2025.3543661.

[15] H. Dong, H. Liu, M. Li, F. Ren, and F. Xie, "An Algorithm for the Recognition of Motion-Blurred QR Codes Based on Generative Adversarial Networks and Attention Mechanisms," *International Journal of Computational Intelligence Systems*, vol. 17, no. 1, 2024, doi:10.1007/s44196-024-00450-7.

[16] J. H. Koo, S. W. Cho, N. R. Baek, and K. R. Park, "Face and Body-Based Human Recognition by GAN-Based Blur Restoration," *sensors*, vol. 20, 2020.

[17] S. K. Zhou *et al.*, "A Review of Deep Learning in Medical Imaging: Imaging Traits, Technology Trends, Case Studies With Progress Highlights, and Future Promises," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 820–838, 2021, doi: 10.1109/JPROC.2021.3054390.

[18] K. El Emam, L. Mosquera, X. Fang, and A. El-Hussuna, "An evaluation of the replicability of analyses using synthetic health data," *Sci Rep*, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-57207-7.

[19] X. Xin and W. Huang, "Unsupervised Person Re-identification using Adversarial Attack Examples and Multi-view Clustering," in *IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society*, 2024, pp. 1–7. doi: 10.1109/IECON55916.2024.10905119.

[20] T. Mamedov, A. Konushin, and V. Konushin, "ReMix: Training Generalized Person Re-Identification on a Mixture of Data," in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025, pp. 8186–8196. doi:10.1109/WACV61041.2025.00794.

[21] H. Damirchi, A. Etemad, and M. Greenspan, "Socially-Informed Reconstruction for Pedestrian Trajectory Forecasting," in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025, pp. 7460–7469. doi:10.1109/WACV61041.2025.00725.

[22] O. Li, J. Cai, Y. Hao, X. Jiang, Y. Hu, and F. Feng, *Improving Synthetic Image Detection Towards Generalization: An Image Transformation Perspective*, vol. 1, no. 1. Association for Computing Machinery, 2024. doi: 10.1145/3690624.3709392.

[23] M. Stephen Lui, F. Abdurrachman Bachtiar, and N. Yudistira, "Penerapan Deep Convolutional Generative Adversarial Network Untuk Menciptakan Data Sintesis Perilaku Pengemudi Dalam Berkendara," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 10, no. 5, pp. 963–972, 2023, doi: 10.25126/jtiik.20231056978.

[24] S. Chakraborty, K. Chatterjee, and P. Dey, "Discovering Tampered Image in Social Media Using ELA and Deep Learning," *SN Comput Sci*, vol. 3, Jul. 2022, doi: 10.1007/s42979-022-01311-w.

[25] D. A. Faroek, R. Umar, and I. Riadi, "Deteksi Keaslian Citra Menggunakan Metode Error Level Analysis (ELA) dan Principal Component Analysis (PCA)," *Format : Jurnal Ilmiah Teknik Informatika*, vol. 8, no. 2, p. 132, 2020, doi:10.22441/format.2019.v8.i2.006.

[26] W. Zhang, C. Zhao, and Y. Li, "A Novel Counterfeit Feature Extraction Technique for Exposing Face-Swap Images Based on Deep Learning and Error Level Analysis," 2020. doi: 10.3390/e22020249.

[27] 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 (Switzerland)*, vol. 11, no. 9, 2022, doi: 10.3390/electronics11091468.

[28] K. B. Mohammed, I. Agrawal, M. Kandimalla, P. Govathoti, C. Prakash, and P. Singh, "Advancing Digital Image Forensics: Enhancing Image Forgery Detection Through Error Level Analysis and Convolutional Neural Networks," 2024, pp. 325–340. doi: 10.1007/978-981-97-3292-0_23.

[29] P. Meel and D. K. Vishwakarma, "HAN, image captioning, and forensics ensemble multimodal fake news detection," *Inf Sci (N Y)*, vol. 567, pp. 23–41, Aug. 2021, doi: 10.1016/J.INS.2021.03.037.

[30] B. Shah, D. Shah, S. Thakar, S. Shah, and S. Dhage, "Image Manipulation Detection Using Error Level Analysis," in *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, 2023, pp. 1–6. doi: 10.1109/asiancon58793.2023.10270614.

[31] S. More, "Enhancing Image Forgery Detection with Convolutional Neural Networks and Error Level Analysis," *Journal of Information Systems Engineering and Management*, vol. 10, pp. 980–993, Mar. 2025, doi: 10.52783/jisem.v10i27s.4757.

[32] S. Patankar, A. Joshi, G. Durge, A. Jaid, K. Kalambe, and H. Dhale, "Image Forgery Detection using ELA and CNN," in *2023 2nd International Conference on Futuristic Technologies, INCOFT 2023*, 2023. doi: 10.1109/incoft60753.2023.10425116.

[33] S. Sahir, "Canny edge detection step by step in Python—Computer vision," *Towards Data Science*, 2019.

[34] Xhlulu, "140k Real and Fake Faces," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces.

[35] Rr. H. P. Sejati, R. Mardhiyyah, Z. Zulkhairi, N. Istiqomah, and R. I. B. Prasetya, "Real-time Smartphone Usage Surveillance System Based on YOLOv5," *IJID (International Journal on Informatics for Development)*, vol. 11, no. 2 SE-Articles, pp. 242–251, Feb. 2023, doi:10.14421/ijid.2022.3766.

[36] Q. A. Fitroh and S. 'Uyun, "Deep Transfer Learning to Improve Classification Accuracy in Dermoscopic Images of Skin Cancer," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 12, no. 2 SE-Articles, May 2023, doi: 10.22146/jnteti.v12i2.6502.