Social Science and Everant Journal Humanities Journal

ISSN: 2456-2653
Volume 09 Issue 01 January 2025

https://sshjournal.com/
Impact Factor: 2024: 6.576
2023: 5.731

DOI: https://doi.org/10.18535/sshj.v9i01.1579

Classifying AI-Generated and Original Images Using a Convolutional Neural Network Algorithm

Ario Tua Purba[1,a] | **Hari Suparwito**[2, b *]
[1]Department of Informatics, Sanata Dharma University, Indonesia
[2]Department of Informatics, Sanata Dharma University, Indonesia
[a]ariopurba37@gmail.com, [b]shirsj@jesuits.net
* Corresponding author

***Received** 03-12-2024*
***Revised** 04-12-2024*
***Accepted** 30-12-2024*
***Published** 02-01-2025*

Abstract:

Identifying images generated by artificial intelligence (AI) that are difficult to distinguish from real images has significant implications for the field of security. The use of deep learning, particularly Convolutional Neural Networks (CNNs), as an effective solution to this challenge is highly relevant. CNNs play a crucial role in extracting features from images, enabling the identification of important patterns that are not easily visible. This research applies CNN architecture to differentiate between AI-generated images and real images, supporting fake image detection. The model's training and testing processes were conducted using the TensorFlow library, with parameters optimized through an Early-Stopping callback. This study uses the CIFAKE dataset, consisting of 60,000 synthetic images and 60,000 real images, totaling 120,000 images of 32x32 pixels in RGB format. The test data was sourced from outside the training dataset. The model was trained for 100 epochs using the Adam optimizer with a learning rate of 0.001. The results show that the model, with 4 convolutional layers and 2 fully connected layers, achieved 94% accuracy on the training data. However, accuracy dropped to around 80% when tested with new, previously unseen data. The F1-score was 94%, and the mAP value was 98%. This research makes a significant contribution to the field of fake image detection by demonstrating the effectiveness of CNN architecture in classifying AI-generated and real images. Although the model shows high performance on the training data, the challenge of classifying new data different from the training data requires further attention for future improvements.

Keywords: Artificial Intelligence, Cifake, Convolutional Neural Network, Classification, Fake Image Detection.

Introduction:

In recent years, the development of artificial intelligence (AI) technology has reached impressive heights. One of the most notable aspects of this progress is AI's ability to produce increasingly complex and realistic artificial images [1]. With increasingly sophisticated algorithms and models, AI can create images that are sometimes very difficult to distinguish from genuine human images. This advancement has important implications in various fields, including security, where distinguishing AI-generated images from real ones can have major consequences. Indeed, AI-generated images have now reached a level of quality that can compete with human-created works and even win art competitions [2].

The difficulty of recognizing the difference between increasingly realistic AI-generated images and real images presents a significant challenge. AI's ability to create images that resemble human works has major implications. To address this challenge, it is necessary to develop methods that can accurately distinguish between AI-generated and real human images.

Deep learning is one of the highly recommended methods for overcoming this challenge [3]. A branch of machine learning, deep learning can recognize patterns in complex data [4]. Consequently, it can be applied to identify patterns in both AI-generated and real images. Deep learning plays a crucial role in addressing the problem of recognizing increasingly realistic AI-generated images. Convolutional Neural Networks (CNNs) are a deep learning architecture specifically designed to handle spatial data, such as images [5]. The main advantage of CNNs lies in their ability to automatically extract hierarchical features from image data, such as edges, colors, and textures, which are important for image classification [3]. The CNN architecture consists of convolutional layers that gradually learn a high-level representation of the image, making it well-suited for complex image recognition tasks.

CNN models have demonstrated good performance in image recognition. Several studies have shown the successful application of CNN algorithms in classifying artificial and real images, achieving accuracies of 99% [6] and 97% [7], respectively. This research focuses on using CNN and VGG16 as classification tools to distinguish between AI-generated and real images. It aims to make a significant contribution to the development of fake image detection by comparing CNN with the pre-trained VGG16 algorithm to determine which is more optimized for detecting real or fake images. Various parameters will be applied to both algorithms to obtain optimal results.

The expected contribution of this research is to provide practical guidance for architecture selection in real-world applications such as object detection, image classification, and image segmentation. By comparing the implementation of these models, this research can enhance understanding of how different architectural configurations (e.g., number of layers, kernel size) affect the model's ability to detect specific features in images. The results are expected to offer recommendations for optimal model design for image detection in various contexts.

Materials and methods:

In general, the design and implementation process of this research consists of several stages. The research stages are presented in Figure 1 below.

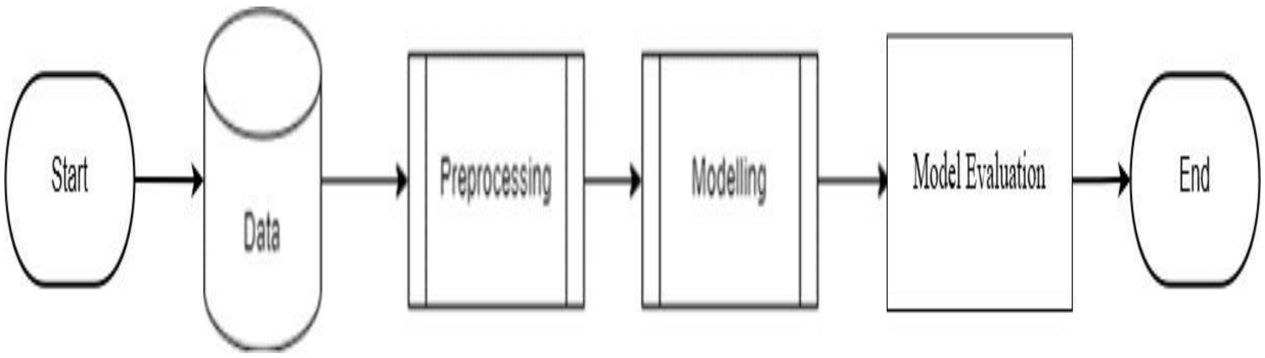


Fig. 1 The steps of the research

The first step is data collection, which involves gathering images for training and testing models using CNN and VGG16 algorithms. In the preprocessing stage, the data is prepared for the subsequent modeling phase. Once data processing is complete, the next step is to build the architectures for both the CNN model and VGG16. The final step is to evaluate the constructed models to assess their accuracy in detecting real or fake images

Data:

The research utilized the CIFAKE dataset [8], which includes 60,000 synthesized images and 60,000 real images, divided into two classes: real and AI-generated. The dataset is split into 100,000 images for training and 20,000 for testing, with a total of 120,000 images, each 32x32 pixels in RGB format. The dataset is accessible at https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images. Additionally, test data outside the training set will be used to evaluate the model.

For the real image data, the CIFAR-10 dataset was employed [9]. This dataset comprises 60,000 images of real subjects, each 32x32 pixels, categorized into ten classes: airplane, car, bird, cat, deer, dog, frog, horse, boat, and truck. These classes will be combined into a single "REAL" class for this research.

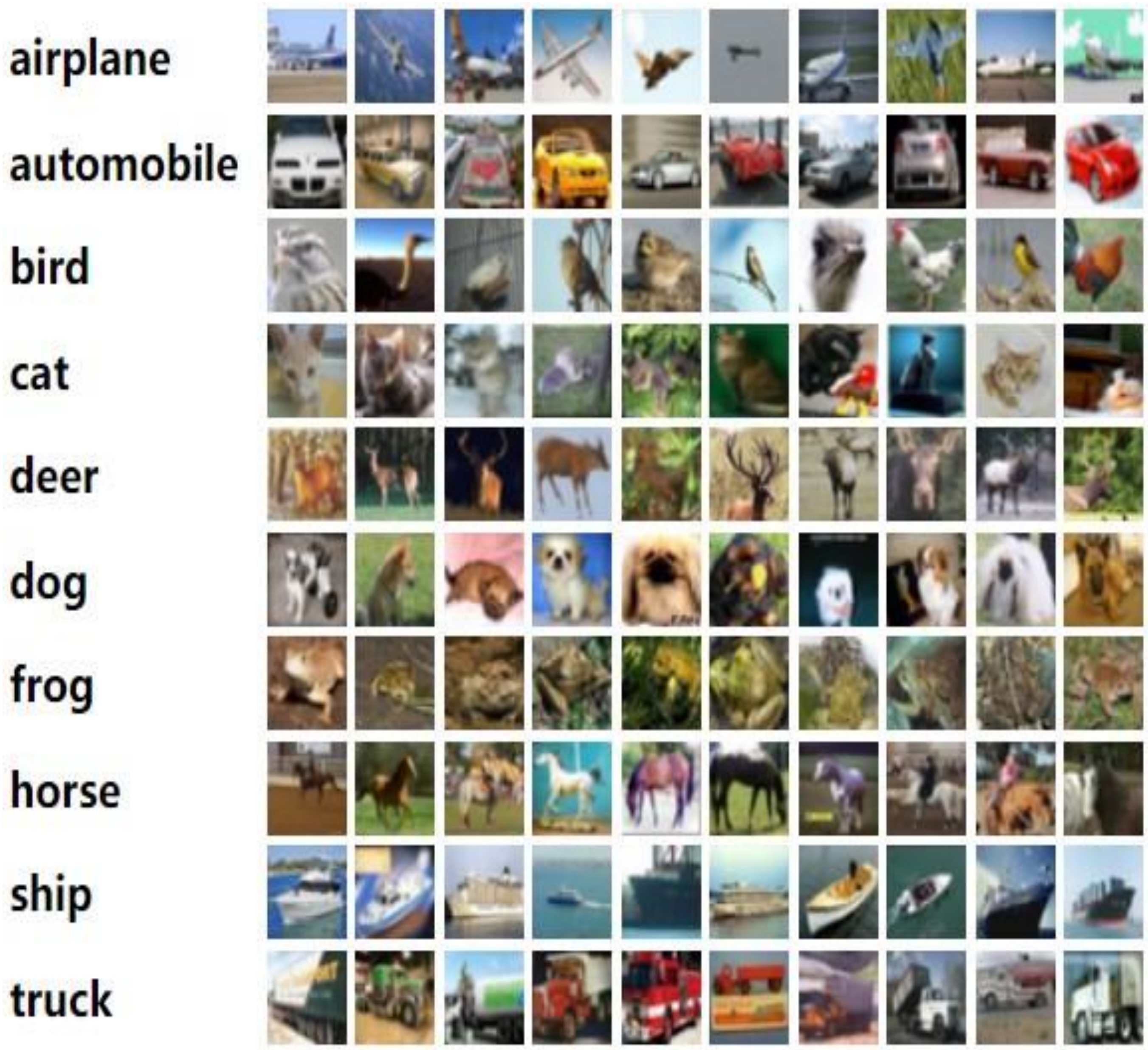


Fig. 2 Samples of the original images

The AI-generated image dataset contains 60,000 images, equivalent to the CIFAR-10 dataset [9], and includes classes such as airplanes, cars, birds, cats, deer, dogs, frogs, horses, boats, and trucks. These classes were combined into a single "FAKE" class. The images were created using Stable Diffusion Model version 1.4, a powerful generative model for synthetic image creation [10]. Sample images are shown in Figure 3 below.

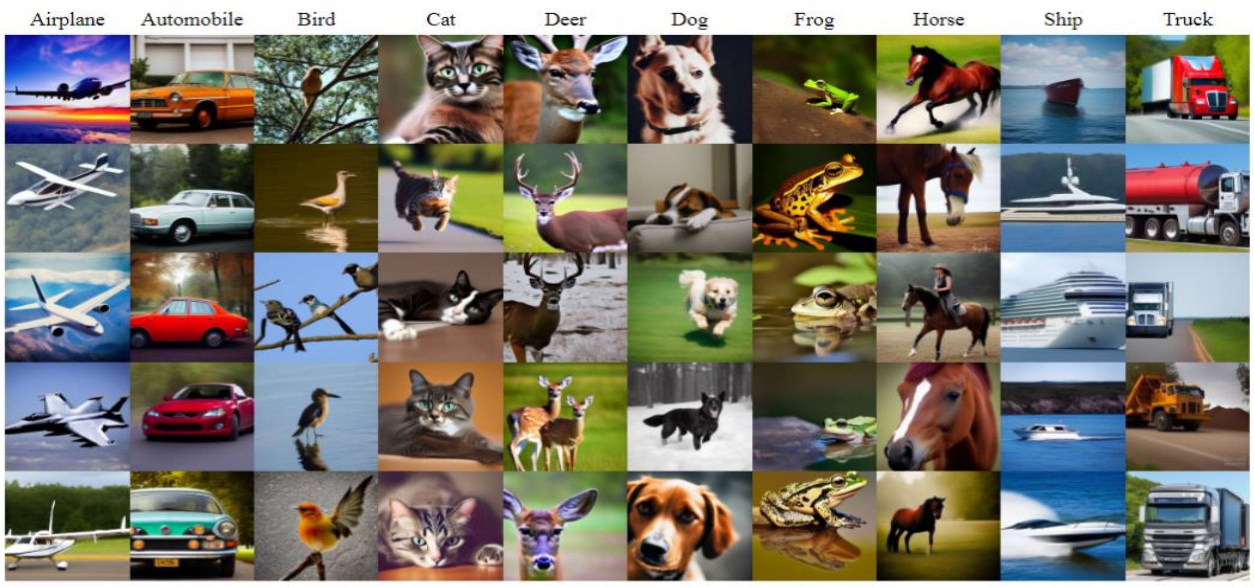


Fig. 3 Samples of images created by AI

The testing dataset differs from the training data and consists of 20 images: 10 labeled "FAKE" and 10 labeled "REAL." The "FAKE" images were generated using Stable Diffusion Model version 1.4, while the "REAL" images were sourced from the Unsplash website. Examples of "FAKE" and "REAL" images are shown in Figures 4 and 5.

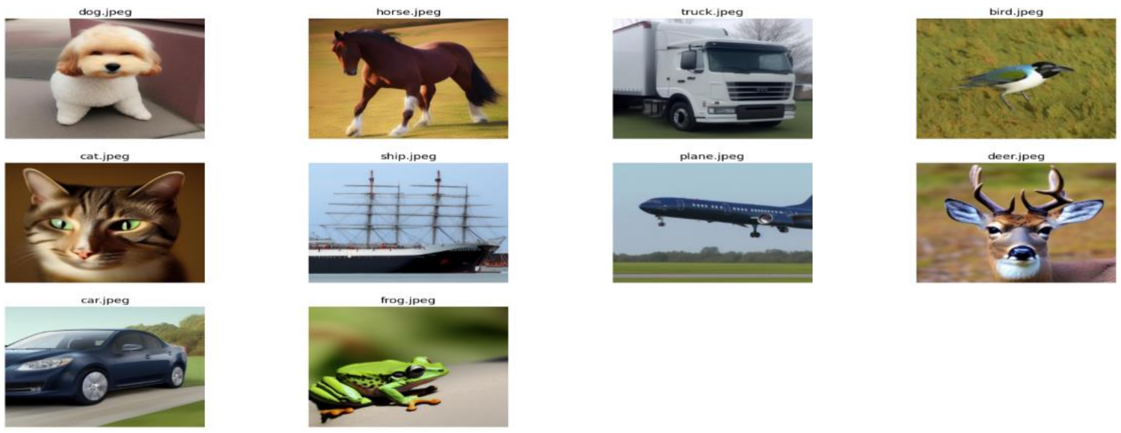


Fig. 4 FAKE data samples

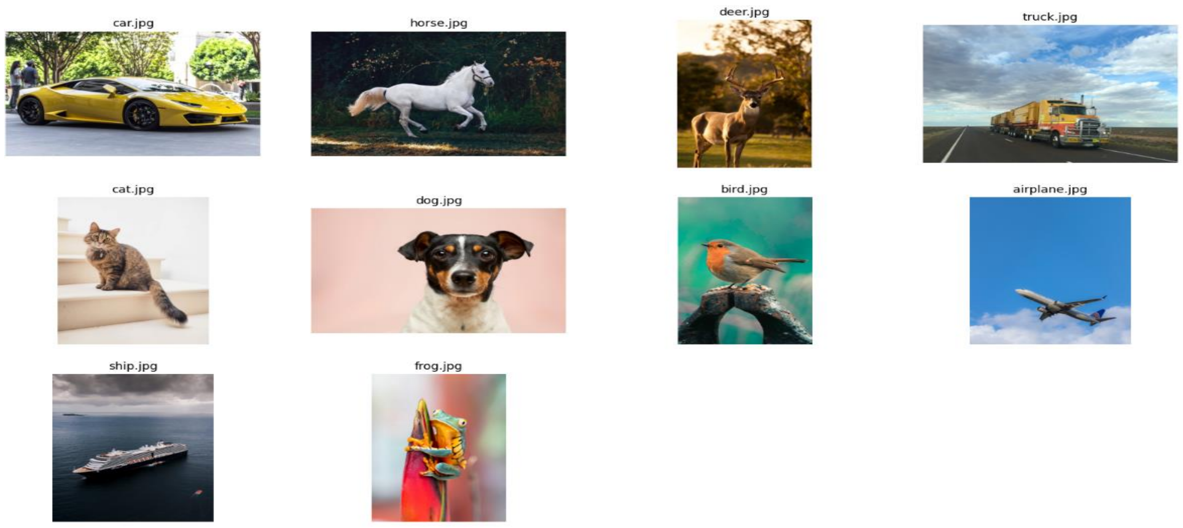


Fig. 5 REAL data samples

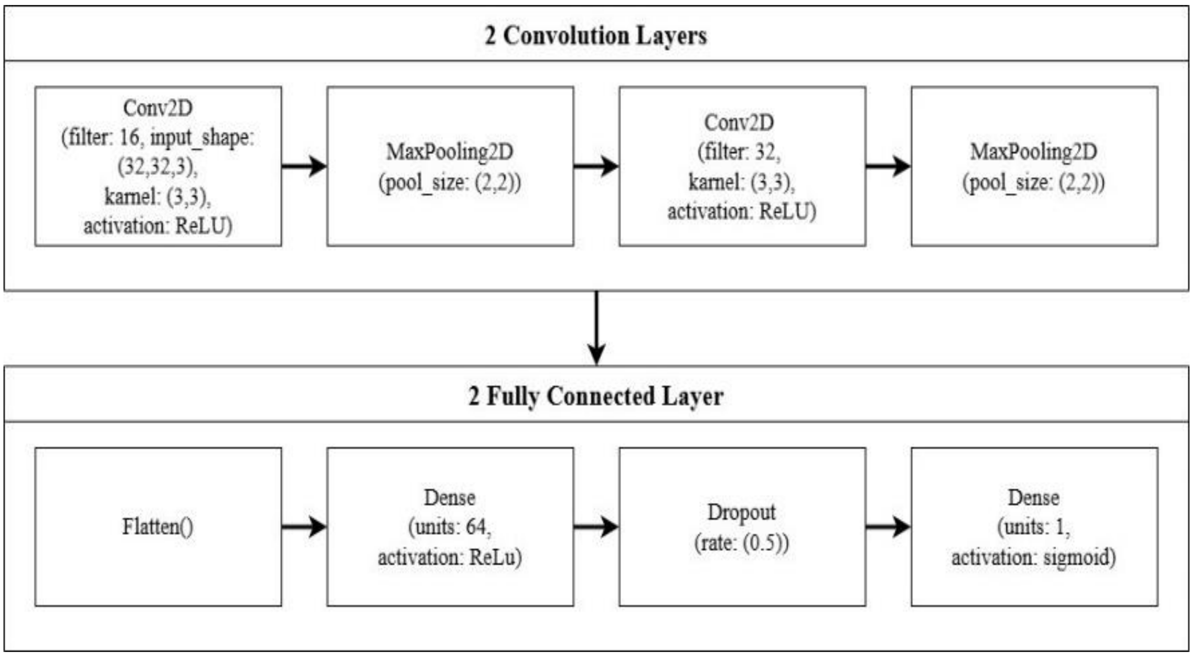
Methodologies:

Pre-processing:

In this stage, data is prepared for training and testing with the CNN and VGG16 classification models. Pixel normalization is applied to scale all pixel values to a range of 0-1, which helps the model learn faster. This normalization is achieved using the ImageDataGenerator class from the TensorFlow library. Specifically, pixel values are rescaled from [0, 255] to [0, 1] by dividing each pixel by 255. The training dataset consists of 100,000 samples, and the testing dataset includes 20,000 samples. Two ImageDataGenerator objects, train_data_gen and valid_data_gen, are created for the training and validation datasets, respectively, with the parameter rescale=1.0/255 to ensure proper normalization. This process often improves model performance.

Modelling:

The CNN architecture used in this research includes Convolutional, Max-Pooling, Flatten, Dropout, and Fully Connected layers. Four different architectures with varying numbers of layers were applied. The first CNN model, shown in Figure 6, features two convolutional layers with 16 and 32 filters, respectively, both using the ReLU activation function and a 3x3 kernel. MaxPooling2D is applied after each convolutional layer with a 2x2 kernel. The input_shape parameter specifies that the model processes images of size 32x32. This architecture is followed by a fully connected layer with two units, 64 and 1. The first unit uses the ReLU activation function, while the final unit uses the sigmoid activation function to classify images as real or AI-generated. A dropout layer with a rate of 0.5 is also included.



2 Convolution Layers
Conv2D (filter: 16, input_shape: (32,32,3), karnel: (3,3), activation: ReLU)
MaxPooling2D (pool_size: (2,2))
Conv2D (filter: 32, karnel: (3,3), activation: ReLU)
MaxPooling2D (pool_size: (2,2))
2 Fully Connected Layer
Flatten()
Dense (units: 64, activation: ReLu)
Dropout (rate: (0.5))
Dense (units: 1, activation: sigmoid)

Fig. 6 Two convolution layers

The second CNN model architecture consists of three convolutional layers. The first two layers are the same as those in the previous model, and the third layer has 64 filters, a 3x3 kernel, and uses the ReLU activation function. MaxPooling2D with a 2x2 kernel is applied after each convolutional layer. This architecture is followed by a fully connected layer for classification.

The third CNN model, shown in Figure 8, features four convolutional layers. It includes the first three layers from the previous model and adds a fourth layer with 128 filters, a 3x3 kernel, and ReLU activation. MaxPooling2D with a 2x2 kernel is applied after each convolutional layer. This model also uses a fully connected layer for classification.

The fourth CNN model utilizes the VGG16 architecture, known for its effectiveness in object recognition [11]. It includes thirteen convolutional layers and three dense layers, as shown in Figure 7.

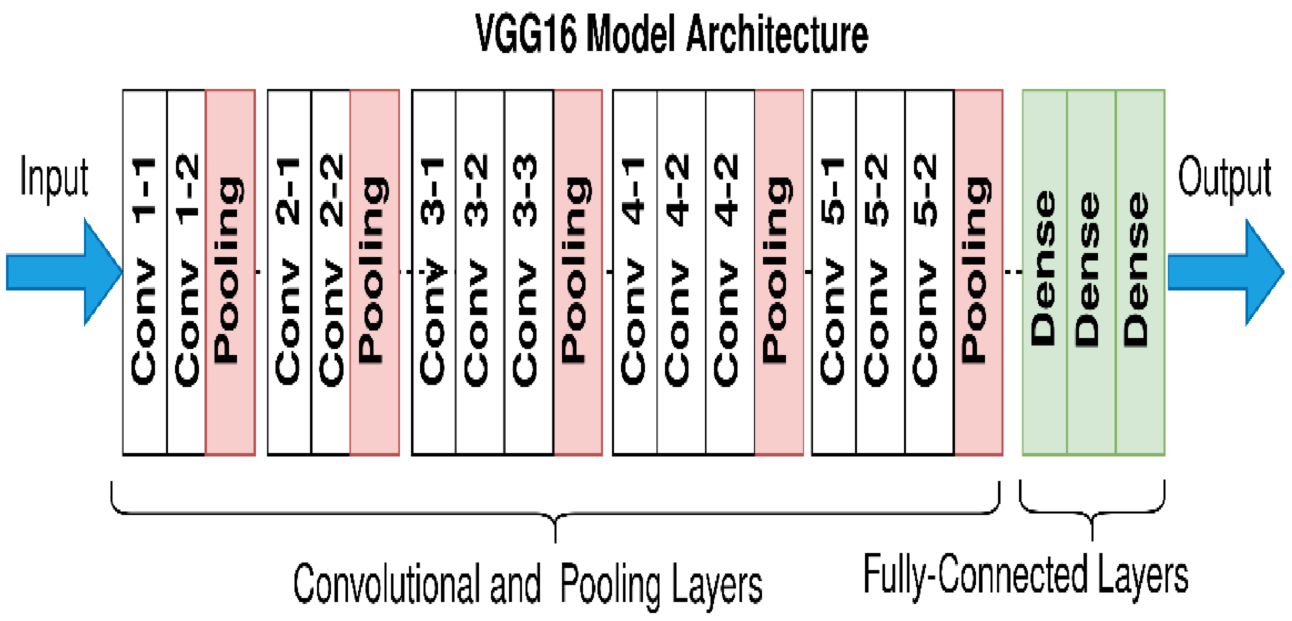


Fig. 7 VGG16 architecture

(Source: https://commons.wikimedia.org/wiki/File:VGG16.png)

The Adam optimizer, known for accelerating convergence and improving optimization in CNN models [12], is used in this research. A learning rate of 0.001 is applied across all model architectures [13]. Each model is trained for 100 epochs to minimize computation time while ensuring convergence, preventing overfitting, and enabling a thorough performance evaluation [14].

A comparison of each model architecture is listed in Table 1 below.

Table 1. The model architectures

Model	Number of convolution layers	Number of Fully Connected layers	Number of Parameters
CNN #1	2	2	78,945
CNN #2	3	2	89,633
CNN #3	4	2	360,609
VGG16	13	3	14,879,041

Training the model:

The CNN and VGG16 models will each be trained for 100 epochs using the Adam optimizer with a learning rate of 0.001. The training dataset consists of 100,000 samples, and the testing dataset has 20,000 samples. An Early-Stopping callback will monitor the validation loss, stopping training after 2 consecutive epochs without improvement to prevent overfitting. Training time will also be used to compare model efficiency. The parameter descriptions for model training are shown in Table 3 below:

Table 2. Parameters of the training model

Parameters	Values
Optimizer	Adam
Learning Rate	0.001
Epochs	100
Callbacks	EarlyStopping

Results and discussion:

The following table, table 4, described the results on the validation dataset

Tabel 3. Results of the model based on the experiment scenario

Model	Number of Layers	F1 Score	Accuracy	mAP	Time
CNN #1	2 Convolutional, 2 Fully Connected	0.93	0.92	0.98	1401.48
CNN #2	3 Convolutional, 2 Fully Connected	0.93	0.92	0.98	718.77
CNN #3	4 Convolutional, 2 Fully Connected	0.94	0.94	0.98	501.43
VGG16	13 Convolutional, 3 Fully Connected	0.93	0.93	0.98	638.72

CNN #3, which uses 4 convolutional layers and 2 fully connected layers, achieved the highest performance with an F1 score and accuracy of 0.94 (94%). This model also produced a mean average precision (mAP) of 0.98 (98%). Adding convolutional layers up to a certain point improves model performance in terms of accuracy and F1 score. However, beyond a certain number of layers, as seen in the VGG16 model, further increases do not yield significant improvements. The fully connected layer is crucial for combining extracted features for final classification.

Figure 10 shows the accuracy changes for CNN architecture 3. The Y-axis represents accuracy (0 to 1, where 1 equal 100%), and the X-axis represents the number of epochs. The blue line indicates training accuracy, and the orange line indicates validation accuracy. The graph shows that training accuracy increases from 82% to around 94%, with the training process stopping at the 5th epoch.

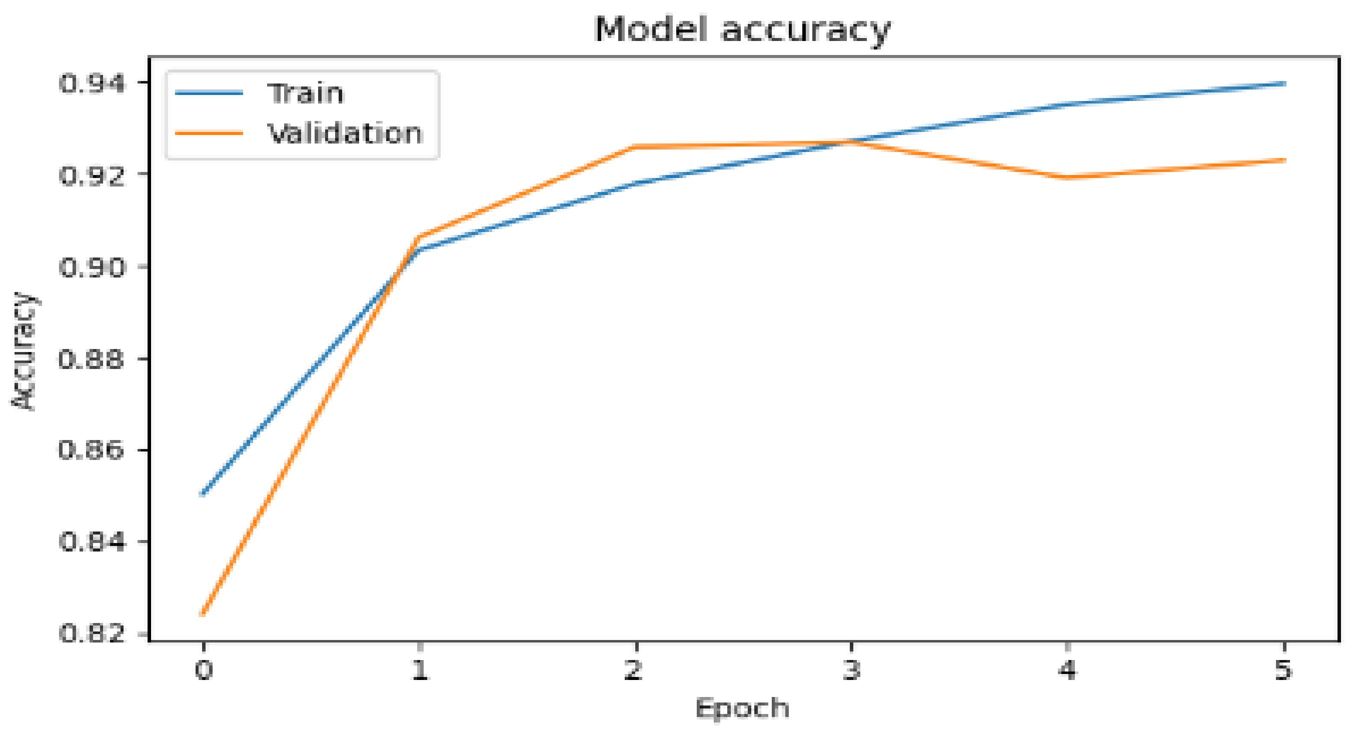


Fig. 10 Accuracy on CNN #3

The loss value graph in Figure 11 shows a decrease from 0.45 to around 0.15. The Y-axis represents the loss value, with lower values indicating smaller prediction errors. The X-axis represents the number of epochs, similar to the accuracy graph. The blue line indicates training loss, while the orange line represents validation loss

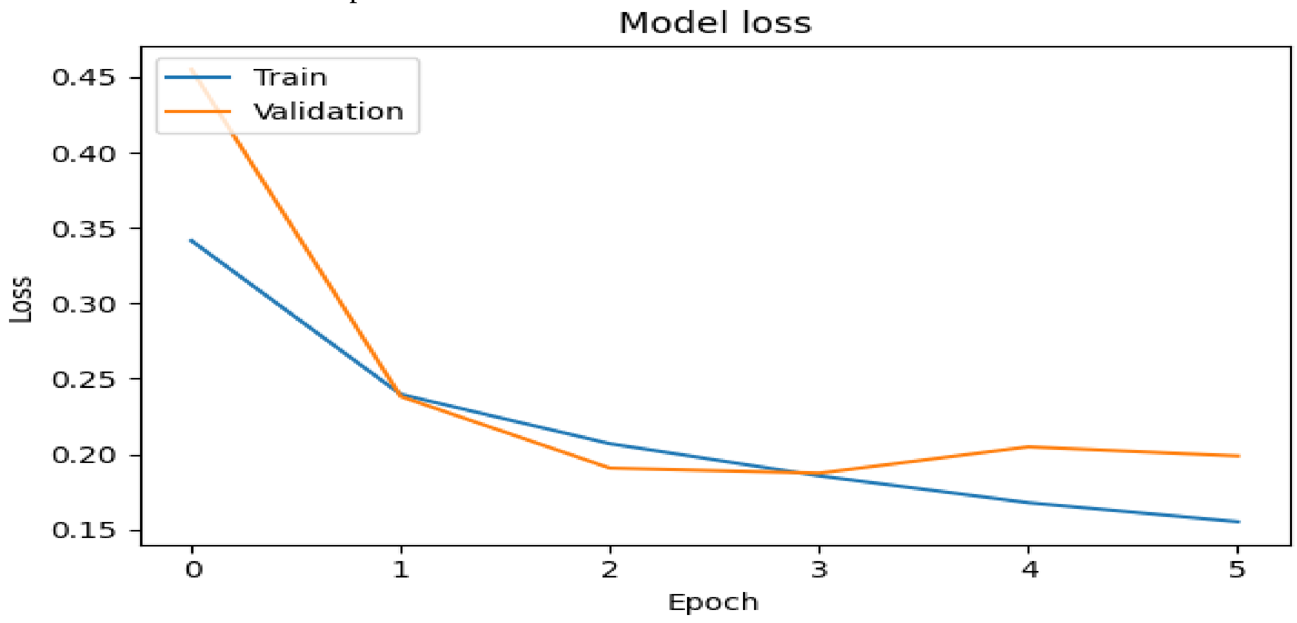


Fig. 11 Model loss on CNN #3

The VGG16 architecture, despite having more layers, does not necessarily perform better in classifying test data, achieving an F1 score and accuracy of 0.93 (93%). However, the mean average precision (mAP) for both models is 98%.

Figure 12 visualizes accuracy changes in the VGG16 architecture. Training accuracy increased from 86% to 94% but dropped during the sixth iteration. The training process stopped at the seventh iteration. The loss value graph in Figure 13 shows a decrease from 0.325 to around 0.150. However, there is no significant difference in accuracy performance between the CNN architecture 3 and the VGG16 architecture.

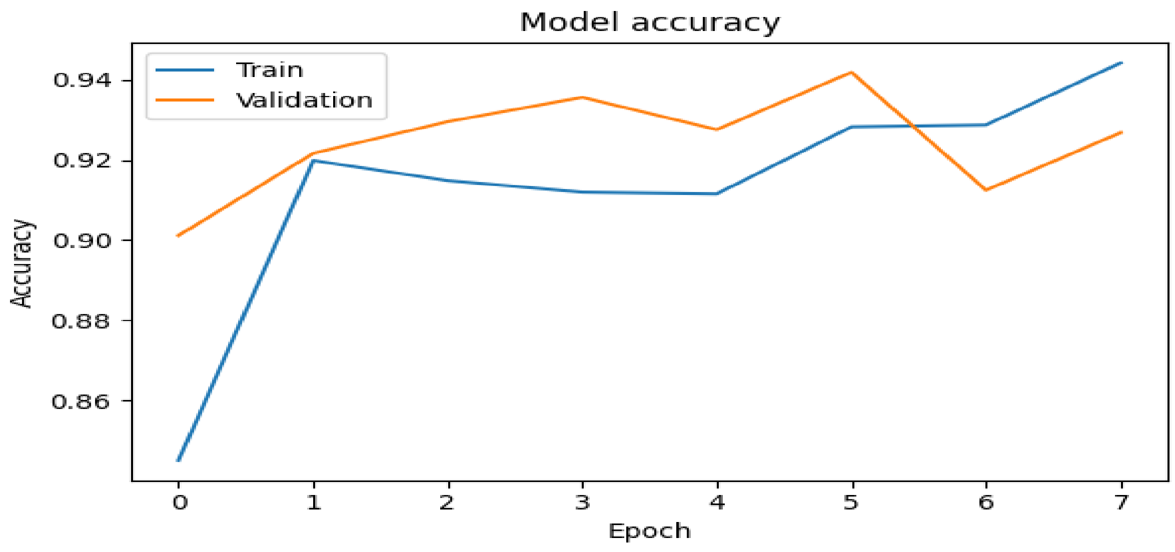


Fig. 12 VGG16 model accuracy

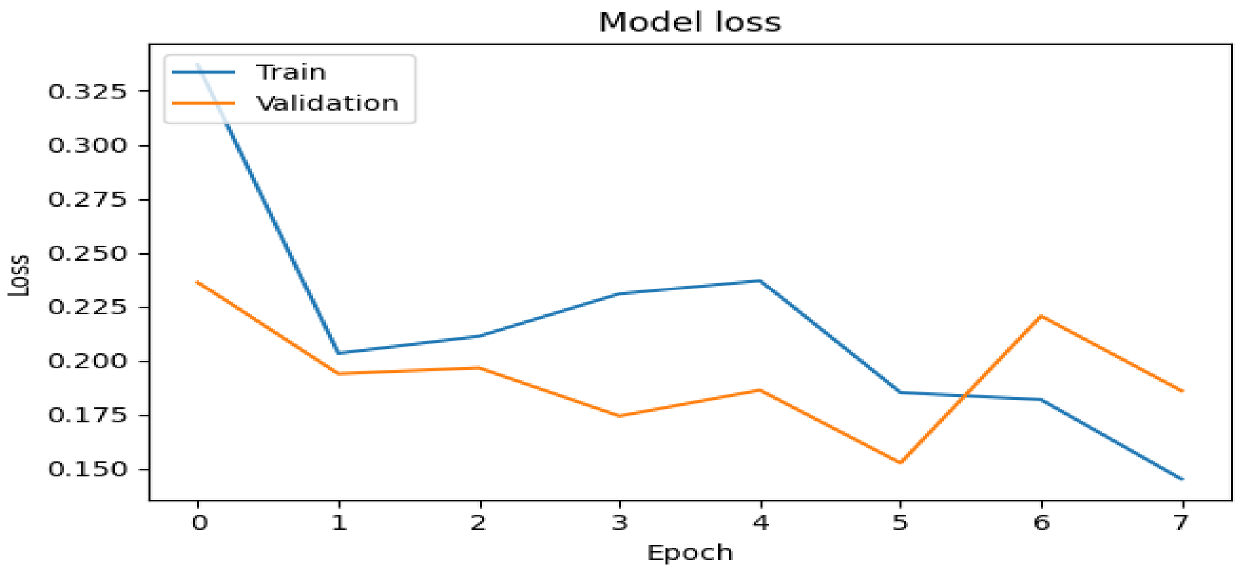


Fig. 13 VGG16 model loss

Figure 14 shows the prediction results on the test data from the CNN #3 architecture model. The model misclassifies 613 'REAL' data as not 'REAL' and 436 'FAKE' data as not 'FAKE'.

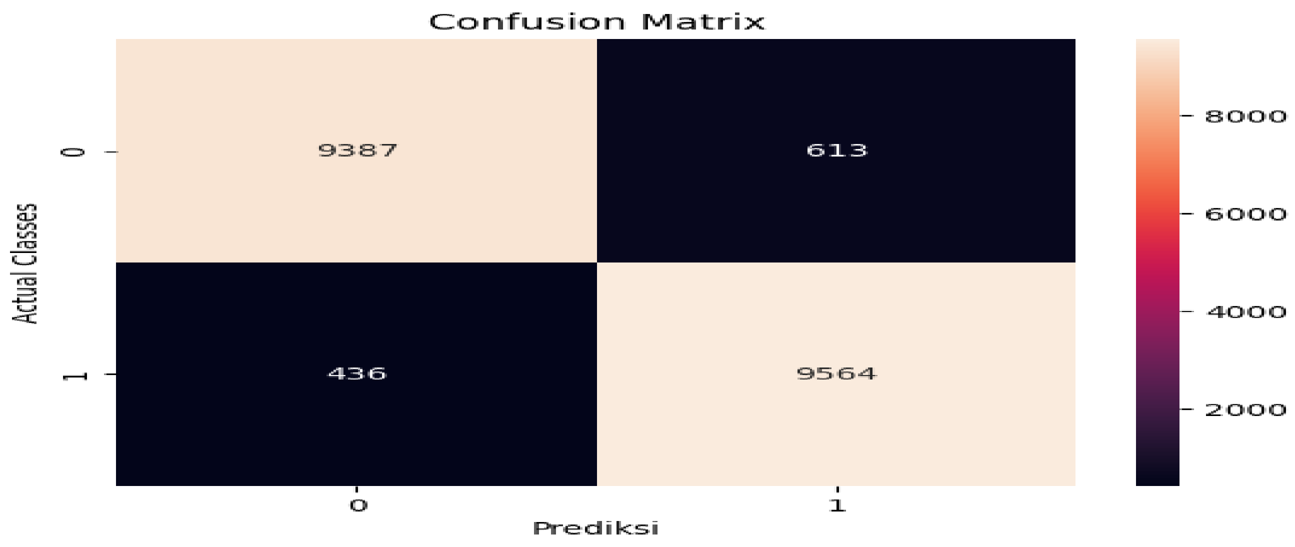


Fig. 14 confusion matrix of CNN #3

In comparison, Figure 15 displays the predictions from the VGG16 architecture model, where 1,269 'REAL' data are misclassified, but only 196 'FAKE' data are incorrectly predicted

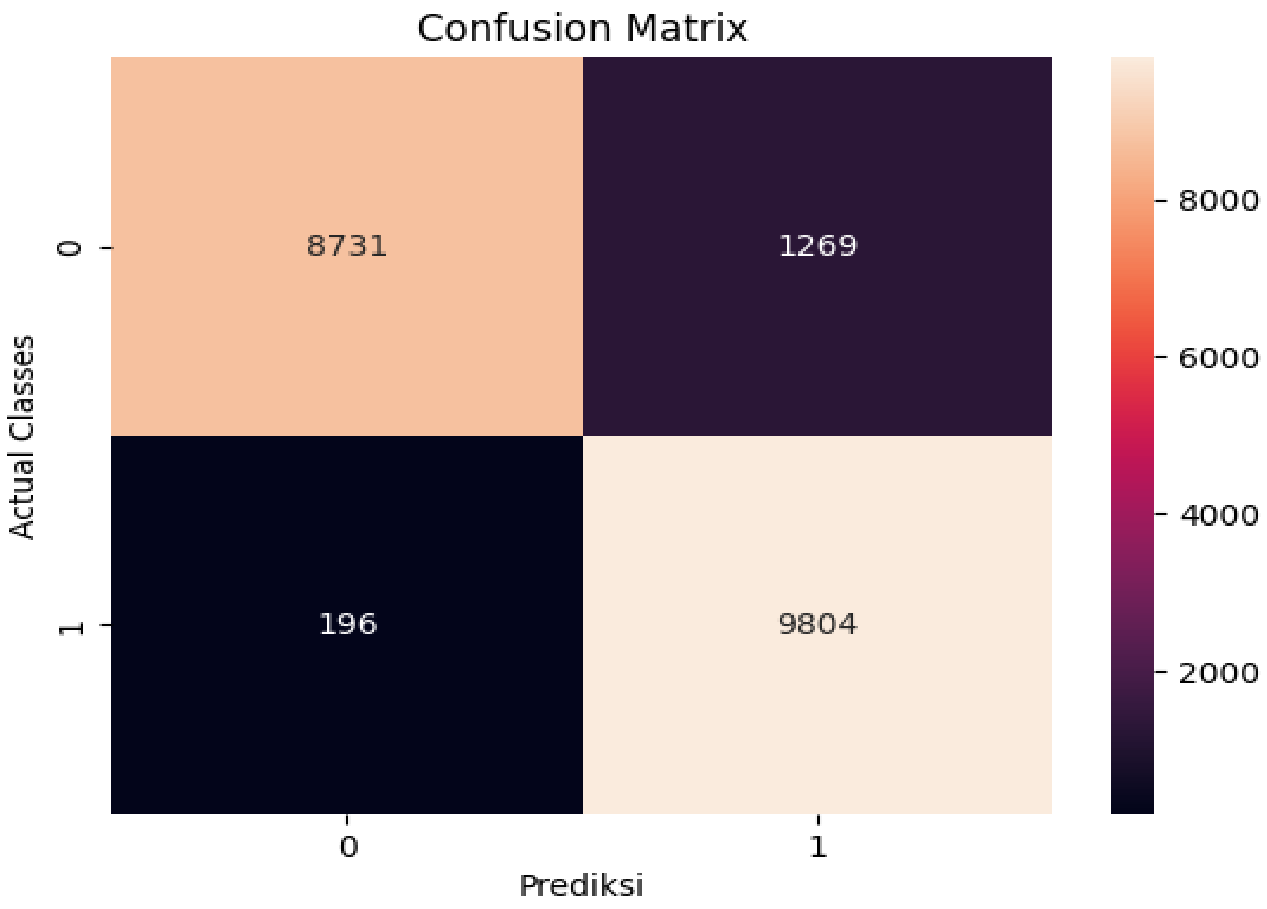


Fig. 15 Confusion Matrix of VGG16

The time column in Table 4 shows the duration required to train each model in seconds. CNN Architecture 1 took 1,401.48 seconds, CNN Architecture 2 took 718.77 seconds, and CNN Architecture 3 was the fastest at 501.43 seconds. The VGG16 architecture required 638.72 seconds for training. These times reveal significant variations between the different neural network architectures. Although CNN Architecture 1 has fewer layers (2 convolutional and 2 fully connected), its training time is the longest, possibly due to processing efficiency or data complexity. CNN 2 and CNN 3, despite having more convolutional layers, show improved efficiency with shorter training times. This suggests that adding layers up to a certain limit can enhance training efficiency. The VGG16 architecture, despite having many more layers (13 convolutional and 3 fully connected), has a relatively short training time, highlighting that complex architectures with efficient designs can offset the high number of layers, resulting in faster training times than some simpler architectures.

Overall, this analysis indicates that training efficiency is influenced not just by the number of layers but also by the design and optimization of the architecture. All models performed well in predicting test data, with CNN 3 and VGG16 achieving the highest performance. The models were further tested on a new dataset, which included data entirely separate from the training and validation sets

Table 4. Results on testing dataset

Model	Number of layers	F1 Score	Accuracy	mAP
CNN #1	2 Convolutional, 2 Fully Connected	0.53	0.65	0.65
CNN #2	3 Convolutional, 2 Fully Connected	0.75	0.70	0.83
CNN #3	4 Convolutional, 2 Fully Connected	0.80	0.83	0.80
VGG16	13 Convolutional, 3 Fully Connected	0.80	0.80	0.81

The test results show that the CNN #3 model achieved the highest performance with an F1-score of 0.80 (80%) and an accuracy of 0.83 (83%). However, its mAP value is 0.80 (80%), which is lower than the CNN #2 model and the VGG16 model, which have mAP values of 0.83 and 0.81, respectively. The VGG16 architecture also performed well in classifying new data, with an F1-score and accuracy of 0.80 (80%).

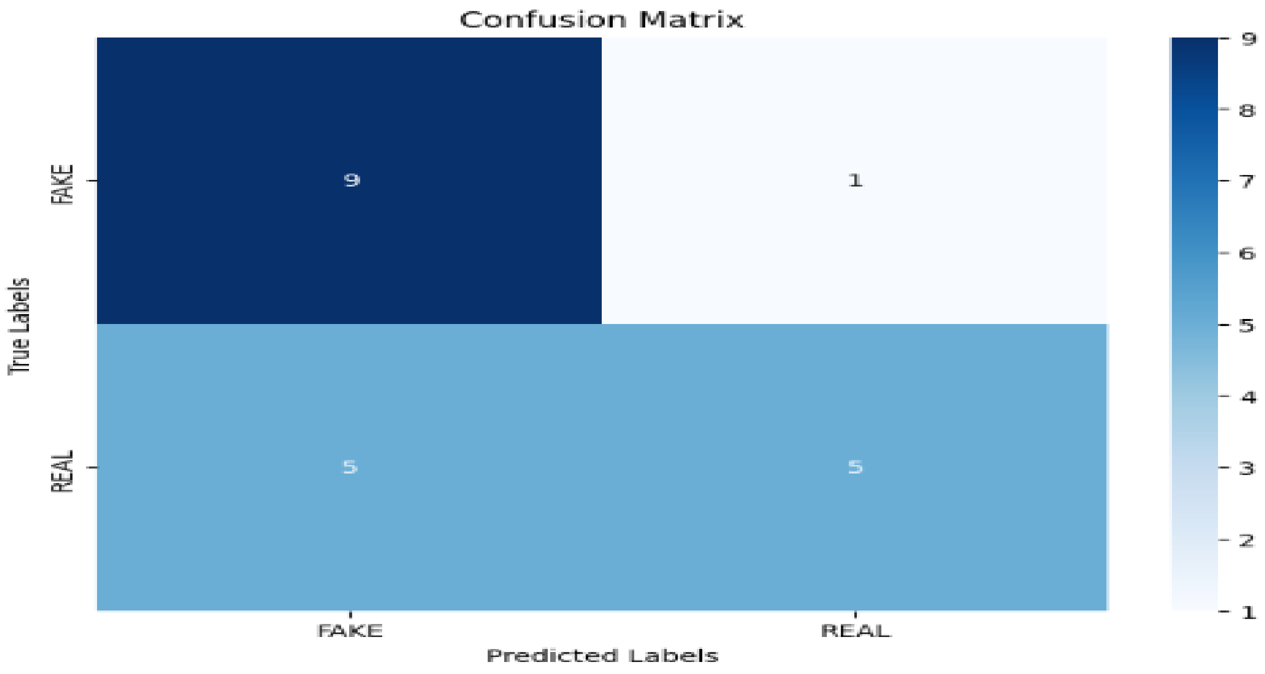


Fig. 16 Confusion matrix of VGG16 models

To identify the data causing model mispredictions, refer to the confusion matrix visualization for the VGG16 architecture in Figure 16. It shows that 'REAL' data was correctly predicted 5 times and incorrectly predicted 5 times, while 'FAKE' data was correctly predicted 9 times and incorrectly predicted once. In contrast, the confusion matrix for the CNN #3 architecture in Figure 17 shows that 'REAL' data was correctly predicted 10 times with no errors, while 'FAKE' data was correctly predicted 4 times and incorrectly predicted 6 times.

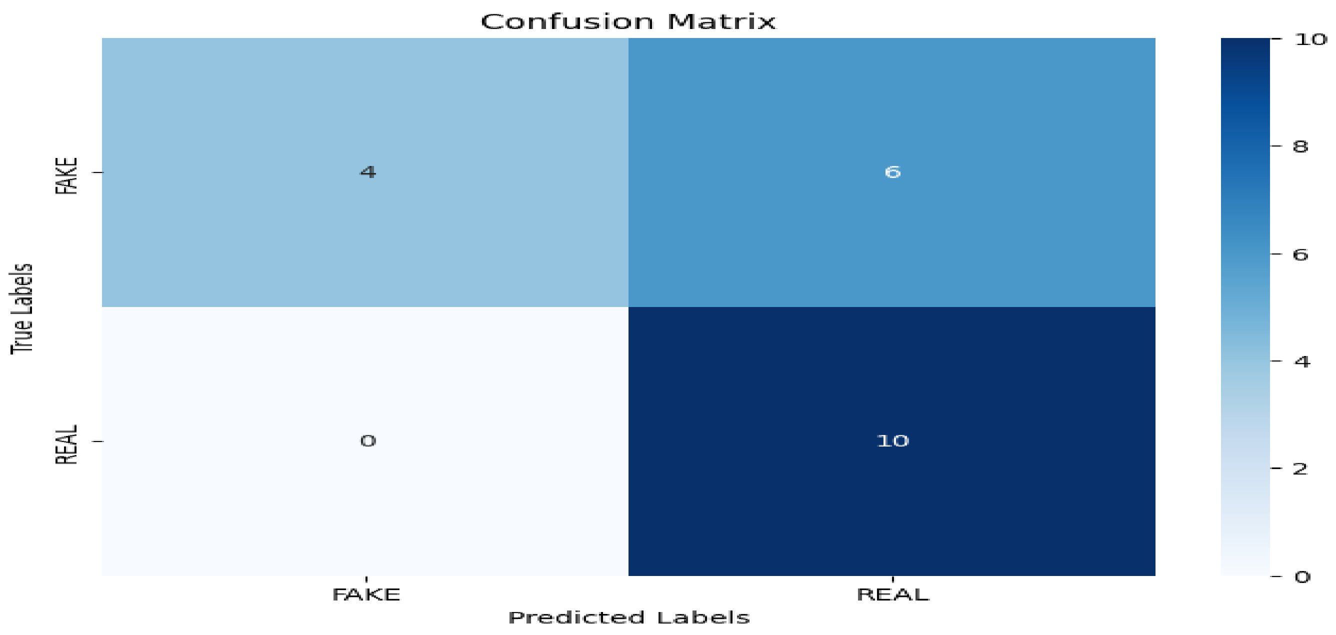


Fig. 17 Confusion Matrix of CNN 3 models

Conclusion:

This study underscores the crucial role of architectural design in CNN performance for image classification, particularly in distinguishing AI-generated from real images. CNN #3, featuring four convolutional layers and two fully connected layers, emerged as the top performer, achieving the highest accuracy (94%) and F1-score (94%) on training data. However, its performance dropped on new, unseen data, with an F1-score of 80% and accuracy of 83%. Interestingly, CNN #3's mean average precision (mAP) on new data was slightly lower (80%) compared to CNN #2 (83%) and the VGG16 architecture (81%). This indicates that while CNN #3 is highly effective in training, its robustness may diminish with entirely new datasets.

The study highlights that increasing the number of layers, as in the VGG16 model, does not necessarily enhance performance. Despite its complexity, VGG16 did not significantly outperform CNN #3. Training efficiency also varied, with CNN #3 showing faster training times despite having more layers, emphasizing the importance of architecture optimization. Misclassifications, particularly in the VGG16 model, point to areas needing improvement, especially with 'REAL' data. While CNN #3 performed slightly better overall, future research should focus on optimizing models for better accuracy and robustness across diverse datasets.

Acknowledgement:

We would like to thank the Intelligent System laboratory of the Informatics department of Sanata Dharma University for allowing us to use the lab to do this research.

References:

[1] Mo, S., Lu, P., & Liu, X. AI-Generated Face Image Identification with Different Color Space Channel Combinations. *Sensors*, *22*(21). (2022) https://doi.org/10.3390/s22218228

[2] Kevin Roose Sept, B. (n.d.). “I won, and I didn’t break any rules,” the artwork’s creator says. https://www.nytimes.com/2022/09/02/technology/ai-artificial-intelligence-artists.html

[3] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. Recent advances in convolutional neural networks. Pattern Recognition, *77*, (2018) 354–377. https://doi.org/10.1016/j.patcog.2017.10.013

[4] LeCun, Y., Bengio, Y., & Hinton, G. Deep learning. Nature, *521*(7553), (2015) 436–444. https://doi.org/10.1038/nature14539

[5] Bayat, O., Aljawarneh, S., Carlak, H. F., International Association of Researchers, Institute of Electrical and Electronics Engineers, & Akdeniz Üniversitesi. (2017). Proceedings of 2017 International Conference on Engineering & Technology (ICET'2017) : Akdeniz University, Antalya, Turkey, 21-23 August, 2017.

[6] Maher Salman, F., & Abu-Naser, S. S. Classification of Real and Fake Human Faces Using Deep Learning. In International Journal of Academic Engineering Research Vol. 6. (2022) www.ijeais.org/ijaer

[7] Alshariah, N. M., Khader, A., & Saudagar, J. Detecting Fake Images on Social Media using Machine Learning. In IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 10, Issue 12. (2019) www.ijacsa.thesai.org

[8] Bird, J. J., & Lotfi, A. CIFAKE: Image Classification and Explainable Identification of AI-Generated Synthetic Images. (2023) http://arxiv.org/abs/2303.14126

[9] Krizhevsky, A. Learning Multiple Layers of Features from Tiny Image. (2009)

[10] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. High-Resolution Image Synthesis with Latent Diffusion Models. (2021) http://arxiv.org/abs/2112.10752

[11] Mascarenhas, S., & Agarwal, M. A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification. International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON*), 1*, 2021,96–99. https://doi.org/10.1109/CENTCON52345.2021.9687944

[12] Prince, S. J. D., *Understanding Deep Learning* (2023) https://mitpress.mit.edu,

[13] Tato, A., & Nkambou, R., Improving Adam Optimizer(2018). http://yann.lecun.com/exdb/mnist/

[14] You, Y., Zhang, Z., Hsieh, C.-J., Demmel, J., & Keutzer, K., Imagenet Training in minutes (2017). http://arxiv.org/abs/1709.05011