Artificial Intelligence and Language Models

ACS Publishing

Article

AI-Generated Image Detection Using Convolutional Neural Network (CNN) Algorithm

Muhamad Shadri[1,*], Anggi Hanafiah[1], Parthasarathy Velusamy[2]

[1] Department of Informatics Engineering, Faculty of Engineering, Universitas Islam Riau, Pekanbaru, Indonesia;
[2] Department of Computer Science and Engineering, Karpagam Academy of Higher Education, Coimbatore, India
* Correspondence: muhamadshadri022@gmail.com;

Abstract: The detection of AI-generated images has become increasingly important for maintaining the authenticity of digital visual content. This study proposes a binary image classification approach to distinguish AI-generated images from non-AI images using four Convolutional Neural Network architectures, namely MobileNet, MobileNetV2, ResNet50, and a custom CNN. The dataset consisted of 16,000 images, equally balanced between AI-generated and non-AI-generated classes, and was divided into training, validation, and testing subsets. Experimental results show that MobileNetV2 achieved the best overall performance among the evaluated models, offering the most favorable balance between classification effectiveness and computational efficiency. In particular, the best MobileNetV2 configuration achieved strong recall for AI-generated images, indicating its suitability for practical content verification tasks. These findings suggest that lightweight transfer learning models are promising solutions for AI-generated image detection in real-world applications.

Keywords: Image detection; AI-generated; Convolutional Neural Network; MobileNetV2; Image classification

Citation: Muhamad Shadri, Anggi Hanafiah, Parthasarathy Velusamy. AI-Generated Image Detection Using Convolutional Neural Network (CNN) Algorithm. *Artificial Intelligence and Language Models* 1–11. https://doi.org/

Received: 16 March 2026
Revised: 22 April 2026
Accepted: 23 April 2026
Published: 23 April 2026

1. Introduction

Over the past few years, rapid advancements in artificial intelligence (AI) have significantly transformed various creative fields, particularly photography. One of the most notable breakthroughs is Generative AI, which enables AI models to generate realistic images from textual descriptions, eliminating the need for human intervention. This technology first gained prominence with OpenAI's DALL-E, introduced in January 2021, which utilized the Generative Pre-trained Transformer 3 (GPT-3) to convert textual prompts into images. Subsequently, models like MidJourney (July 2022) and Stable Diffusion (August 2022) emerged, offering higher visual fidelity and the ability to replicate artistic styles with greater precision. Today, AI-powered tools such as Bing Image Creator can produce images that closely resemble the distinct styles of major entertainment companies, including Pixar and Disney. These advancements are possible because AI systems leverage vast online datasets, allowing them to learn and generate images based on references. Most generative AI programs rely on Generative Adversarial Networks (GANs), which enable the model to analyze and reproduce artistic styles from various sources[1].

Despite its technological sophistication, the rise of AI-generated images has sparked ethical and authenticity concerns, particularly in the realm of photography. As AI becomes increasingly capable of creating photorealistic images, distinguishing genuine photographs from AI-generated visuals has become a significant challenge. This issue recently gained attention in Indonesia when Ariani Dikye, an Indonesian photographer, won the prestigious CEWE Photo Award 2023 with her work titled Warung Kopi. The photograph, which captures the warmth and communal atmosphere of a coffee shop in Bogor, was accused

of being AI-generated. This controversy ignited debates within the global photography community regarding the importance of authenticity in visual art. Although Ariani Dikye ultimately proved the legitimacy of her photograph by presenting evidence of the manual photography process, this case highlighted the growing difficulty in differentiating AI-generated images from authentic ones.

Given these challenges, there is a critical need for a robust detection system to distinguish AI-generated images from real photographs. One of the most effective approaches for addressing this issue is Convolutional Neural Networks (CNNs), which can identify unique patterns commonly found in AI-generated visuals, distinguishing them from natural photographs. Such a detection system is essential not only for preserving artistic authenticity but also for preventing the potential misuse of AI-generated images in competitions, publications, and other creative fields[2].

Despite the growing risk of misuse of AI-generated images, practical evidence comparing lightweight transfer-learning models and a custom CNN for binary AI-image detection remains limited, particularly under a unified experimental setting. This study addresses that gap by systematically comparing MobileNet, MobileNetV2, ResNet50, and a custom CNN on a balanced dataset of AI-generated and non-AI-generated images. The contribution of this work lies in identifying which architecture provides the best trade-off between detection performance and computational efficiency for real-world authenticity verification.

2. Methods

2.1. Population and Sample

The dataset in this study consisted of AI-generated and non-AI-generated images used for binary classification. The final sample contained 16,000 images, evenly distributed between the two classes, with 8,000 AI-generated images and 8,000 non-AI-generated images. This balanced composition was adopted to support unbiased training and evaluation of the classification models.

2.2. Variables

In this study, the primary variable is the image category, which serves as the dependent variable in the binary classification task. It consists of two classes: AI_GENERATED and NON_AI_GENERATED as shown in Table 1. The AI_GENERATED class refers to images synthetically produced by artificial intelligence models, while the NON_AI_GENERATED class represents authentic images captured or created without AI intervention. These variables are treated as categorical labels and are used to train and evaluate the classification models. No additional independent variables were explicitly defined, as the study focuses on learning discriminative visual features directly from the image data. The clear operational distinction between these two classes ensures consistency in labeling and supports reliable model performance evaluation.

Table 1. Explanation and Operational Definition of the Research Variables.

Variable	Operational Definition
AI_GENERATED	Images generated by AI
NON_AI_GENERATED	Real (non-AI) images

2.3. Data Collection

The dataset used in this study was compiled from the publicly available Kaggle dataset CIFAKE: Real and AI-Generated Synthetic Images[1]. In this dataset, the AI-generated images correspond to the FAKE class, while the non-AI images correspond to the REAL class. The Kaggle data card states that CIFAKE contains 60,000 synthetically generated

[1] https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images

images and 60,000 real images, with the real images collected from the CIFAR-10 dataset. It further specifies that the synthetic images were generated using Stable Diffusion version 1.4, indicating that the AI-generated subset is diffusion-based rather than GAN-based. The dataset is organized into 100,000 training images and 20,000 testing images, with 50,000/10,000 images per class, respectively.

Before model training, all images were screened to remove corrupted, irrelevant, blurry, and extremely low-resolution samples. In the present study, the initial working dataset contained 10,330 AI-generated images and 8,288 non-AI images, which were subsequently reduced to 8,000 images per class after cleaning, resulting in a balanced dataset. The specific threshold for excluding low-resolution images and the exact blur-assessment procedure were not provided in the Kaggle data description and should therefore be reported based on the authors' own preprocessing protocol.

For model input consistency, all images were resized to the input dimensions required by the selected CNN architecture and normalized prior to training. The real-image component of CIFAKE is derived from CIFAR-10, whose images are originally 32 × 32 pixels. Any further resizing, normalization range, and augmentation strategy used in this study constitute task-specific preprocessing decisions. Data augmentation was applied only to the training set.

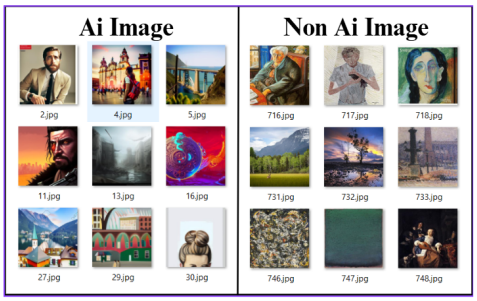

Figure 1. Examples of AI-generated and non-AI images.

The collected images were downloaded in bulk and screened based on their visual quality and relevance to the target categories. Only images that met the requirements for subsequent preprocessing and classification were retained. Figure 1 shows examples of AI-generated and non-AI images.

2.4. Data Cleaning and Preprocessing

Data cleaning is the process of evaluating data quality by modifying or removing information that is deemed irrelevant, incomplete, inaccurate or improperly formatted. The goal is to produce high-quality information[3]. Data preprocessing, on the other hand, involves cleaning, modifying, and adjusting both relevant and irrelevant data to prepare it for analysis or modeling[4]. Once the data set was acquired, a crucial step in the data preparation process was data cleaning. The data cleaning phase involved removing images that did not meet the study standards. Specifically, blurry images, images with very low resolution, or images that were irrelevant to the predefined categories (AI-generated or non-AI-generated) were excluded from the dataset. This step is essential to ensure that only high-quality images are used for model training, which in turn improves the accuracy and reliability of the model. The initial data set contained 10,330 AI-generated images and 8,288 non-AI images. After the data cleaning process, the dataset was reduced to 8,000 AI-generated images and 8,000 non-AI images. The resulting cleaned data set is balanced, with an equal number of images from both categories.

2.5. Data Management and Repository

To facilitate easier access and management of the dataset, all images were uploaded to GitHub. The use of GitHub as a platform for dataset management provides several

benefits, including version control, easy collaboration, and secure access to the dataset. By uploading the dataset to GitHub, it is not only made accessible to the research team but also to other collaborators who may wish to contribute to the model's development.

GitHub was also used to document the research process and keep track of the various stages of the project. This ensures that the dataset and the model development are organized and can be replicated or expanded upon in the future.

2.6. Data Analysis Method

The data analysis for this research was performed using the Google Colab platform, which provides a cloud-based environment for data analysis and machine learning tasks. Google Colab was chosen due to its accessibility, integration with Python libraries, and support for hardware accelerators like GPUs, which significantly speed up the training process.

The Python programming language was used for writing the code and implementing the model. Several key libraries were utilized in this research, including TensorFlow for building and training the Convolutional Neural Network (CNN) model, Keras for defining and compiling the CNN architecture, NumPy for numerical computations and data manipulation, and Matplotlib for visualizing the model training results, such as loss and accuracy curves.

All experiments were conducted using the free version of Google Colab with TensorFlow/Keras. The models were trained in a cloud environment with access to a GPU (typically NVIDIA T4, subject to availability) and approximately 12–16 GB of RAM. A batch size of 32 was used as a standard configuration to balance computational efficiency and convergence stability. Each model was trained for 50 and 100 epochs using Adam and RMSprop optimizers with learning rates of 0.001 and 0.0001. Early stopping was applied to prevent overfitting, with a patience of 10 epochs based on validation loss, while model checkpointing was used to retain the best validation weights. Due to the shared and dynamic resource allocation in the free Colab environment, the average training time per model configuration ranged from approximately 15 to 45 minutes, depending on runtime conditions and model complexity.

The dataset, consisting of 8,000 AI-generated images and 8,000 non-AI images, underwent several preprocessing steps. Normalization was applied to scale the input data appropriately, ensuring that the model could learn patterns more efficiently. Augmentation techniques were also implemented to artificially increase the size of the training dataset while introducing variations that enhance the model's robustness. After preprocessing, the dataset was divided into training and validation sets to ensure effective learning and prevent overfitting. The training process involved adjusting hyperparameters such as the learning rate and the number of epochs to optimize the model's performance. The model's evaluation was conducted using key metrics, including accuracy, validation accuracy, loss, and validation loss. These metrics were used to assess the model's effectiveness in distinguishing AI-generated images from non-AI-generated images.

2.7. Data Splitting

After the directories were created, the downloaded and extracted dataset from GitHub was divided into three main parts: training data, validation data, and test data. The data was split with a ratio of 80% for training, 10% for validation, and 10% for testing[5]. The training data was used to train the model in identifying image patterns[6], while the validation data was utilized to monitor and prevent overfitting by testing the model on unseen data[7][8]. After training, the test data was used to evaluate the model's performance[9][10]. This data splitting process is crucial to ensure that the model can recognize patterns in new, unseen data.

2.8. Data Augmentation

Data augmentation was performed to increase the size of the dataset, allowing the trained model to achieve better performance. Therefore, the data augmentation process became an essential step in model training. In this stage, various techniques were applied to enrich the training dataset by generating new variations of existing images[11]. This augmentation process was performed using TensorFlow's Image Data Generator[12][13][14], which included several transformations such as horizontal and vertical shifts (width shift and height shift), image rotation, shear transformation, and zooming. Additionally, images were flipped horizontally to increase variation and reduce the likelihood of the model memorizing specific patterns.

The image brightness was also adjusted within a range of 0.8 to 1.2 to introduce diversity in lighting conditions. These augmentation techniques were implemented to enable the model to recognize a wider range of image patterns, thereby improving its ability to generalize to new, unseen data. Validation data was not subjected to augmentation apart from normalization to ensure accurate model evaluation.

2.9. MobileNet

MobileNet is one of the Convolutional Neural Network (CNN) architectures designed to address high computational demands. MobileNet can be implemented on mobile platforms as well as embedded devices. This architecture was developed by Google researchers based on the need for an efficient CNN design. One of the main differences between MobileNet and other CNN architectures lies in the use of convolutional layers, where the filter depth is adjusted according to the input image depth[15].

MobileNet was selected as a lightweight transfer-learning baseline due to its low computational complexity and suitability for image classification tasks on limited hardware. In this study, the ImageNet-pretrained MobileNet backbone was used without its original top classifier. A GlobalAveragePooling2D layer, a Dense layer with 128 ReLU units, a Dropout layer of 0.5, and a two-unit Softmax output layer were added for binary classification. The pretrained backbone was frozen during training, and the model was optimized using Adam and RMSprop.

By utilizing Depthwise Separable Convolutions, MobileNets are designed to have an efficient architecture and lighter network weights. Depthwise Separable Convolutions serve as the primary layers in the MobileNet model. A 3x3 Depthwise Separable Convolution is applied in this model, which can reduce computational load by eight to nine times compared to standard convolutions, with minimal accuracy loss. Depthwise Separable Convolutions are factorized convolutions that separate standard convolution into depthwise convolution and 1x1 convolution, known as pointwise convolution. Depthwise Convolution in the MobileNet model uses one filter for each input channel, while Pointwise Convolution employs a 1x1 convolution to combine the results from the Depthwise Convolution. Standard convolution performs filtering and combination in a single step, whereas Depthwise Separable Convolutions divide this process into two separate layers: one for filtering and another for combination. This separation contributes to model size reduction and computational efficiency. Although Depthwise Separable Convolution is the core of the MobileNet model, the first layer used is a full convolutional layer. At the final stage, an average pooling process is utilized to reduce the spatial resolution to one before further processing by the fully connected layer. With Depthwise and Pointwise Convolutions calculated separately, the MobileNet model consists of a total of 28 layers[16].

In the model development stage, a transfer learning architecture was utilized by implementing the MobileNet model that had been pretrained on the ImageNet dataset[17][18]. This transfer learning approach allowed the model to leverage the knowledge previously acquired by MobileNet, which had been trained on millions of images across various categories. The main goal was to accelerate the training process and improve accuracy, especially when the available dataset was limited.

In the proposed model, the convolutional base of MobileNet was retained, while the top layers were removed to adapt to the specific classification task. The pretrained layers of MobileNet were frozen to prevent their weights from being updated during training. Subsequently, additional layers were added to complete the classification task. A GlobalAveragePooling2D layer was used to reduce the output dimension of MobileNet to a single value per feature. Next, the data was passed through a Dense layer with 128 neurons using the ReLU activation function. To mitigate overfitting, a Dropout layer with a rate of 0.5 was applied, randomly deactivating 50% of neurons during training.

For the output layer, a Dense layer with two neurons was added, utilizing the Softmax activation function to generate classification probabilities between two classes: AI_GENERATED and NON_AI_GENERATED.

The model was compiled using the Adam and RMSprop optimizers to enhance convergence speed. Categorical crossentropy was employed as the loss function for multi-class classification, while accuracy was used as the primary metric to evaluate model performance during training. By leveraging transfer learning, the model was expected to achieve high accuracy even with a limited dataset.

2.10. MobileNetV2

MobileNetV2 is an improvement over the previous MobileNets[19][20][21]. Its core layers utilize a more detailed separation method, primarily replacing full convolutional operators with a factored version. This approach splits the convolution into two layers. The first layer, known as depthwise convolution or 1x1 convolution, performs lightweight filtering by applying a single convolution filter to each input channel. The second layer, called pointwise convolution, constructs new features by computing a linear combination of the input channels.

Each layer L_i in a Deep Neural Network has an activation tensor with dimensions $h_i \times w_i \times d_i$. Every set of input images forms a "manifold of interest," defined by the set of activations generated for each layer L_i. It is assumed that the low-dimensional structure of this "manifold of interest" can be preserved by integrating a Linear Bottleneck layer into the convolutional block. Applying linear layers is crucial to prevent significant data loss.

MobileNetV2 consists of 32 filters in the initial full convolutional layer, followed by 19 remaining bottleneck layers. Since ReLU is well-suited for low-precision computation, it is used as the activation function. In contemporary networks, the kernel size is always 3x3, which has become the standard. Additionally, dropout and batch normalization techniques are applied during the training process to enhance model generalization and stability[16].

The model was built using a transfer learning architecture with MobileNetV2, which was pre-trained on the ImageNet dataset. The use of transfer learning aims to enhance training efficiency and accuracy, particularly when data is limited, by leveraging the knowledge MobileNetV2 has already acquired from millions of images. This model utilizes the convolutional base of MobileNetV2 while modifying its top layers to suit the specific classification task. The weights in the MobileNetV2 layers were frozen to maintain consistency during training. Several new layers were added to complement the classification architecture. A GlobalAveragePooling2D layer was employed to reduce the output dimensions of MobileNetV2, followed by a Dense layer with 128 neurons and a ReLU activation function. To prevent overfitting, a Dropout layer with a rate of 0.5 was applied. Finally, a Dense layer with two neurons and a Softmax activation function was used to generate probability scores for the two target classes: `AI_GENERATED` and `NON_AI_GENERATED`.

The model was compiled using the Adam and RMSprop optimizers to ensure faster convergence, categorical cross-entropy as the loss function for multi-class classification, and accuracy as the evaluation metric. This transfer learning approach is expected to produce an accurate model even with a limited dataset.

2.11. ResNet50

Residual Networks, often referred to as ResNet50, are built upon the fundamental concept of block-based processing within convolutional layers with shortcut connections. ResNet50 adhere to two key design principles that enhance their efficiency. The first principle states that each layer must have a sufficient number of filters to maintain the same output size. The second principle asserts that if the spatial size of the feature map is reduced by half, the number of filters in each layer should be doubled[16]. The model utilizes a transfer learning architecture based on ResNet50, which has been pre-trained on the ImageNet dataset. The use of ResNet50 and transfer learning aims to accelerate the training process and enhance accuracy, particularly when the available dataset is limited, by leveraging the knowledge that ResNet50 has acquired from millions of images.

The convolutional base of ResNet50 is utilized, while the top layers are modified to fit the specific classification task. The weights of the ResNet50 layers are frozen to prevent updates during training. Several additional layers are incorporated to complete the classification architecture.

A `GlobalAveragePooling2D` layer is applied to reduce the dimensionality of ResNet50's output. The next step involves a `Dense` layer with 128 neurons and a ReLU activation function. To mitigate overfitting, a `Dropout` layer with a rate of 0.5 is applied, randomly deactivating some neurons during training. Finally, a `Dense` layer with two neurons and a Softmax activation function is added to produce probability distributions for the two target classes: `AI_GENERATED` and `NON_AI_GENERATED`.

The model is compiled using the Adam and RMSprop optimizers to achieve efficient convergence. The categorical crossentropy loss function is employed for multi-class classification, and accuracy is used as the evaluation metric. This transfer learning strategy is expected to yield a highly accurate model despite the limited dataset.

2.12. Custom Convolutional Neural Network (CNN) Built from Scratch

A Convolutional Neural Network (CNN) is designed from scratch without utilizing transfer learning. This CNN architecture is specifically designed for image classification tasks and consists of multiple convolutional layers, pooling layers, and fully connected layers.

The convolutional layers extract essential features from input images through convolution operations using a set of learnable filters. After each convolutional layer, pooling layers such as MaxPooling are applied to reduce the spatial dimensions of the features, thereby accelerating computations and improving robustness against minor variations in images.

After passing through several convolutional and pooling blocks, the extracted features are flattened into a one-dimensional vector and fed into one or more fully connected (Dense) layers. These layers learn non-linear combinations of the extracted features to perform classification. The ReLU activation function is applied in fully connected layers to introduce non-linearity and enhance the model's representational capacity. Dropout layers can also be added to mitigate overfitting.

The model's output layer is a fully connected layer with a number of neurons corresponding to the target classes (in this case, two classes: `AI_GENERATED` and `NON_AI_GENERATED`). The Softmax activation function is applied in the output layer to generate a probability distribution over these classes.

The CNN model is compiled using optimizers such as Adam or RMSprop. For multi-class classification, categorical crossentropy is used as the loss function, with accuracy as the evaluation metric. The specific model architecture, including the number of layers, the number of filters, kernel sizes, and other parameters, can be adjusted based on the complexity of the dataset and the classification task requirements.

3. Result

To detect whether an image is AI-generated or non-AI, the user simply uploads the image file to be tested. The system will then process the image and may take a few moments to complete the analysis. The detection results will appear once the process is finished, indicating whether the image is AI-generated or an original non-AI image. This process uses a CNN algorithm capable of recognizing patterns and distinctive features of AI-generated images, comparing them with real-world images taken from the natural world.

To obtain the best detection results, thorough testing is required for each model used. Each model, such as MobileNet, MobileNetV2, ResNet50, and Custom CNN Built From Scratch (Without Transfer Learning), has its own advantages and characteristics in detecting AI-generated images and original (non-AI) images. Therefore, testing is carried out with a systematic approach to evaluate the performance of each model under various conditions, including the number of epochs, optimizer choices, and learning rate values. Through structured experiments on each model, the model that provides the best accuracy and detection speed can be identified, leading to a more accurate and efficient detection system. The detection results of the tests can be seen in Table 2.

Table 2. Model Training Results with Different Optimizers, Learning Rates, and Epochs

Model (Optimizer, Learning Rate, Epoch)	Accuracy	Val Accuracy	Loss	Val Loss
MobileNet (Adam, 0.001, 50)	0.9219	0.8994	0.2369	0.2326
MobileNet (Adam, 0.001, 100)	0.9233	0.8906	0.2135	0.2385
MobileNet (RMSprop, 0.0001, 50)	0.9042	0.8794	0.2413	0.2904
MobileNet (RMSprop, 0.0001, 100)	0.9157	0.8875	0.2471	0.2617
MobileNetV2 (Adam, 0.001, 50)	0.9062	0.8794	0.2354	0.2656
MobileNetV2 (Adam, 0.001, 100)	0.9287	0.8900	0.2230	0.2607
MobileNetV2 (RMSprop, 0.0001, 50)	0.8771	0.8831	0.2743	0.2906
MobileNetV2 (RMSprop, 0.0001, 100)	0.9021	0.8800	0.2630	0.2746
ResNet50 (Adam, 0.001, 50)	0.6917	0.6875	0.6076	0.6000
ResNet50 (Adam, 0.001, 100)	0.7021	0.7031	0.5858	0.5975
ResNet50 (RMSprop, 0.0001, 50)	0.6917	0.6875	0.5948	0.5964
ResNet50 (RMSprop, 0.0001, 100)	0.6875	0.6862	0.5992	0.6036
Custom CNN (Adam, 0.001, 50)	0.7708	0.7644	0.4995	0.5102
Custom CNN (Adam, 0.001, 100)	0.8042	0.7700	0.4717	0.5002
Custom CNN (RMSprop, 0.0001, 50)	0.7500	0.7569	0.4850	0.5181
Custom CNN (RMSprop, 0.0001, 100)	0.7958	0.7631	0.4501	0.4874

In the MobileNet model, although the accuracy is already quite good, its performance can be improved by using the MobileNetV2 model. MobileNetV2 has a more efficient architecture, allowing for deeper and more accurate feature processing. In the tests, the MobileNetV2 model showed higher accuracy compared to MobileNet.

In the ResNet50 model, the test results showed still low accuracy, leading to errors in image detection. This low accuracy resulted in the model often misclassifying images, such as identifying images that should be classified as AI-generated as non-AI, or vice versa.

In the Custom CNN model without transfer learning, although the overall accuracy is relatively high, there is a deficiency in the ability to detect images that are clearly AI-generated. While this model can achieve good accuracy under general conditions, it shows suboptimal results when detecting clearer AI-generated images. This may be due to the lack of a deeper understanding of the complex features present in AI-generated images, which require more extensive training and more advanced feature processing. Without transfer learning, the CNN model relies solely on training from scratch, which may not be sufficient to capture the finer patterns or distinctive features of AI-generated images. Therefore, while accuracy in detecting non-AI images is good, further improvement through techniques such as transfer learning could help enhance the model's ability to detect AI-generated images more accurately.

3.1. Evaluation Using Confusion Matrix

In this testing phase, the model evaluates the performance of image detection by using a confusion matrix to provide a better understanding of accuracy and classification errors made by the model. A confusion matrix is a table used to measure the accuracy of a classification model by displaying the number of correct and incorrect predictions for each class based on actual data and model predictions. Various evaluation metrics can then be calculated to assess the performance of the image detection model. Table 3 presents the results based on the accuracy, precision, recall, and F1 score for each model.

Table 3. Performance metrics comparison of various models for AI-generated image detection based on confusion matrix analysis.

Model (Optimizer, Learning Rate, Epoch)	Accuracy	Precision (AI/Non-AI)	Recall (AI/Non-AI)	F1-score (AI/Non-AI)
MobileNet (Adam, 0.001, 50)	0.91	0.89 / 0.93	0.93 / 0.88	0.91 / 0.90
MobileNet (Adam, 0.001, 100)	0.89	0.83 / 0.98	0.98 / 0.79	0.90 / 0.87
MobileNet (RMSprop, 0.0001, 50)	0.89	0.85 / 0.94	0.95 / 0.83	0.90 / 0.89
MobileNet (RMSprop, 0.0001, 100)	0.89	0.86 / 0.93	0.94 / 0.84	0.90 / 0.89
MobileNetV2 (Adam, 0.001, 50)	0.86	0.79 / 0.98	0.98 / 0.74	0.88 / 0.84
MobileNetV2 (Adam, 0.001, 100)	0.87	0.81 / 0.95	0.96 / 0.77	0.88 / 0.85
MobileNetV2 (RMSprop, 0.0001, 50)	0.85	0.79 / 0.96	0.97 / 0.74	0.87 / 0.83
MobileNetV2 (RMSprop, 0.0001, 100)	0.87	0.81 / 0.96	0.97 / 0.78	0.88 / 0.86
ResNet50 (Adam, 0.001, 50)	0.66	0.71 / 0.63	0.55 / 0.78	0.62 / 0.70
ResNet50 (Adam, 0.001, 100)	0.70	0.69 / 0.71	0.71 / 0.69	0.70 / 0.70
ResNet50 (RMSprop, 0.0001, 50)	0.64	0.62 / 0.67	0.73 / 0.56	0.67 / 0.61
ResNet50 (RMSprop, 0.0001, 100)	0.68	0.67 / 0.68	0.69 / 0.67	0.68 / 0.68
CNN Costum (Adam, 0.001, 50)	0.74	0.85 / 0.68	0.58 / 0.90	0.69 / 0.78
CNN Costum (Adam, 0.001, 100)	0.76	0.87 / 0.70	0.60 / 0.91	0.71 / 0.79
CNN Costum (RMSprop, 0.0001, 50)	0.74	0.70 / 0.81	0.85 / 0.63	0.76 / 0.71
CNN Costum (RMSprop, 0.0001, 100)	0.80	0.82 / 0.78	0.77 / 0.82	0.79 / 0.80

Based on the evaluation results of various CNN models, the performance of the tested models showed significant variations in accuracy, precision, recall, and F1-score. The models tested included MobileNet, MobileNetV2, ResNet50, and a Custom CNN without transfer learning. These models were optimized using Adam and RMSprop with two different learning rates (0.001 and 0.0001) and trained for two different numbers of epochs (50 and 100). The evaluation results show that MobileNetV2 with the Adam optimizer (learning rate 0.001, 100 epochs) was the most optimal model in this study. This model achieved an accuracy of 87%, with a precision of 0.81 for the AI_GENERATED class and 0.95 for the NON_AI_GENERATED class. Additionally, the recall for the AI_GENERATED class reached 0.96, indicating that this model had an excellent ability to detect AI-generated images. The F1-score for both classes also showed a good balance, with 0.88 for AI_GENERATED and 0.85 for NON_AI_GENERATED. In addition to MobileNetV2, the MobileNet model with Adam and RMSprop optimizers also showed quite good performance, with a maximum accuracy of 91%. However, despite MobileNet having higher accuracy, MobileNetV2 was more stable during the training process, especially with higher epochs. This indicates that MobileNetV2 is more efficient and has better generalization compared to MobileNet.

On the other hand, ResNet50 had the lowest performance in this study, with a maximum accuracy of only 70%. The precision and recall of this model were unbalanced, with precision of 0.69 for AI_GENERATED and 0.71 for NON_AI_GENERATED, as well as recall of 0.71 for AI_GENERATED and 0.69 for NON_AI_GENERATED. This shows that ResNet50 was less optimal in distinguishing AI-generated and non-AI-generated images on the used dataset.

For the Custom CNN model without transfer learning, the performance of this model was relatively lower compared to MobileNet and MobileNetV2 but still better than ResNet50. With the RMSprop optimizer and learning rate of 0.0001 at 100 epochs, this model achieved an accuracy of 80%, with fairly balanced precision and recall. However, compared to MobileNetV2, this model still lagged in terms of efficiency and training stability.

From a practical perspective, false negatives for the AI-generated class are especially critical because they represent AI-generated images that are incorrectly accepted as authentic. The strong recall of MobileNetV2 for the AI-generated class indicates that the model is effective at minimizing this error type. However, its lower recall for the non-AI class suggests that some authentic images may still be incorrectly flagged as AI-generated, which may be acceptable in screening-oriented applications but should be considered in real-world deployment.

Based on these evaluation results, MobileNetV2 with the Adam optimizer (learning rate 0.001, 100 epochs) was selected as the best model in this study because it achieved a balance between high accuracy, training efficiency, and stability in the training process. This model can be implemented for detecting AI-generated images with reliable and optimal results.

4. Discussion

The results of this study demonstrate that MobileNetV2, optimized with Adam (lr = 0.001) over 100 epochs, achieved the highest training accuracy of 92.87%. This model outperformed MobileNet, ResNet50, and a custom CNN, confirming its efficiency in feature extraction and classification. The precision-recall analysis indicates a high recall (0.96) and F1-score (0.88) for AI-generated images, while non-AI images achieved a precision of 0.95 and an F1-score of 0.85. These results highlight the model's sensitivity to AI-generated content, which is crucial for real-world applications in content verification.

Despite these promising outcomes, certain limitations exist. The dataset, sourced from Kaggle, may not fully represent all AI image generation techniques, potentially affecting model generalization. Performance may also vary when detecting images generated by newer AI models not included in the dataset. Future research should explore transfer learning with larger datasets and real-time detection improvements.

Further development should focus on extending the system for video-based detection, enabling real-time AI-generated content classification. This enhancement would be valuable in dynamic applications such as surveillance and video content verification. Additionally, implementing the system in an Android application would improve accessibility, allowing users to detect AI-generated images directly on mobile devices. This would make the system more practical and widely applicable for everyday use. These advancements would further strengthen AI-generated content detection, ensuring better robustness and usability across various domains.

5. Conclusion

This study compared four CNN-based approaches for distinguishing AI-generated images from non-AI images. Among the evaluated models, MobileNetV2 provided the best overall balance between detection performance and computational efficiency, making it the most suitable candidate for practical deployment. The findings also indicate that transfer learning is more effective than training a custom CNN from scratch for this task. However, the generalizability of the results depends on the diversity of the source datasets and the range of AI image generators represented. Future work should evaluate broader datasets, additional metrics, and more robust deployment-oriented settings.

References

1. Fajrin, F.A.; Sugiastuti, N.Y. The Ability of Indonesian Lawyers in Assisting the Development of Copyright Protection Law in Indonesia for Comic Books Whose Illustrations are Created with the Aid of Artificial Intelligence. *JURNAL LEX PHAENOMENON* **2024**, *1*, 74–93.
2. Yusuf, A. VIRAL Karya Fotografer Indonesia Juara di Eropa Dituduh Hasil AI, Warganet Miris Negara Penuduh. *https://kalteng.tribunnews.com/2023/10/19/viral-karya-fotografer-indonesia-juara-di-eropa-dituduh-hasil-ai-warganet-miris-negara-penuduh* **2023**. Accessed on October 20, 2024.
3. Raharja, A.R.; Jayadi.; Pramudianto, A.; Muchsam, Y. Penerapan Algoritma Decision Tree dalam Klasifikasi Data "Framingham" Untuk Menunjukkan Risiko Seseorang Terkena Penyakit Jantung dalam 10 Tahun Mendatang. *Technologia Journal* **2024**, *1*.

4. Pujiono, S.; Astuti, R.; Basysyar, F.M. Implementasi Data Mining Untuk Menentukan Pola Penjualan Produk Menggunakan Algoritma K-Means Clustering. *JATI (Jurnal Mahasiswa Teknik Informatika)* **2024**, *8*, 615–620.
5. Joseph, V.R. Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal* **2022**, *15*, 531–538.
6. Rizki, F.; Putra, M.P.K.; Assuja, M.A.; Ariany, F. Implementasi Deep Leraning Lenet Dengan Augmentasi Data Pada Identifikasi Anggrek. *Jurnal Informatika dan Rekayasa Perangkat Lunak* **2023**, *4*, 357–366.
7. Muhartini, S.; Sunyoto, A.; Muhammad, A.H. Implementasi Metode Deep Learning CNN Dalam Klasifikasi Tajong (Sarung) Samarinda. *Jurnal SENOPATI: Sustainability, Ergonomics, Optimization, and Application of Industrial Engineering* **2024**, *6*, 28–41.
8. Utomo, M.W.S.; Murti, H.W.; Sujatmoko, A.W.I.; Sari, A.P. DETEKSI SPAM EMAIL MENGGUNAKAN METODE LSTM (LONG SHORT TERM MEMORY). *JATI (Jurnal Mahasiswa Teknik Informatika)* **2024**, *8*, 11406–11411.
9. Bintang, Y.K.; Imaduddin, H. Pengembangan Model Deep learning untuk Deteksi Retinopati Diabetik Menggunakan Metode Transfer Learning. *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)* **2024**, *9*, 1442–1455.
10. Fadhillah, I.R.; Al Haromainy, M.M.; Maulana, H. Implementasi Model Transfer Learning EfficientNet untuk Pendeteksian Bahasa Isyarat Indonesia (BISINDO) pada Perangkat Android. *JATI (Jurnal Mahasiswa Teknik Informatika)* **2024**, *8*, 7816–7822.
11. Agustina, A. KLASIFIKASI PENYAKIT TANAMAN PADI MENGGUNAKAN METODE CNN ARSITEKTUR DENSENET121 DAN AUGMENTASI DATA. *KLASIFIKASI PENYAKIT TANAMAN PADI MENGGUNAKAN METODE CNN ARSITEKTUR DENSENET121 DAN AUGMENTASI DATA* **2024**, *8*.
12. Syaharani, M.A.; Budianto, T.A.C.; Adam, R.I. KLASIFIKASI BUAH SEGAR DAN BUSUK MENGGUNAKAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK (CNN). *JATI (Jurnal Mahasiswa Teknik Informatika)* **2024**, *8*, 10823–10827.
13. Abel, S.; Pranidana, A.M.; Qasos, L.; Ula, M.; et al. Perbandingan Akurasi Metode Convolutional Neural Network (CNN) dan Sobel untuk Klasifikasi Buah Rambutan melalui Pengolahan Citra. In Proceedings of the Prosiding Seminar Nasional Teknologi dan Teknik Informatika (SENASTIKA), 2024, Vol. 1.
14. Khumaidi, A.; Nurpadilah, A. Klasifikasi Molting Kepiting Soka Menggunakan Algoritma Convolusional Neural Network. *Komputa: Jurnal Ilmiah Komputer dan Informatika* **2024**, *13*, 43–50.
15. Apendi, S.; Setianingsih, C.; Paryasto, M.W. Deteksi Bahasa Isyarat Sistem Isyarat Bahasa Indonesia Menggunakan Metode Single Shot Multibox Detector. *eProceedings of Engineering* **2023**, *10*.
16. Haksoro, Elok Iedfitra Setiawan, A. Pengenalan Jamur Yang Dapat Dikonsumsi Menggunakan Metode Transfer Learning Pada Convolutional Neural Network. *Jurnal ELTIKOM* **2021**, *5*, 81–91.
17. Fadlun, M.H.; Martanto, M.; Hayati, U. Klasifikasi Tumor Otak menggunakan Convolutional Neural Network dan Transfer Learning. *Jurnal Informatika dan Rekayasa Perangkat Lunak* **2024**, *6*, 289–295.
18. Prilianti, K.R.; Oktariyanto, V.V.; Setiawan, H. ORNAMENTAL PLANT IDENTIFICATION SYSTEM USING TRANSFER LEARNING ON CONVOLUTIONAL NEURAL NETWORK. *Jurnal Teknik Informatika (Jutif)* **2024**, *5*, 1015–1023.
19. Ae, N.H.; Zul, M.I.; et al. Aplikasi Penerjemah Bahasa Isyarat Indonesia menjadi Suara berbasis Android menggunakan Tensorflow. *Jurnal Komputer Terapan* **2021**, *7*, 74–83.
20. Firmansah, R.A.; Santoso, H.; Anwar, A. Transfer Learning Implementation on Image Recognition of Indonesian Traditional Houses. *Jurnal Teknik Informatika (Jutif)* **2023**, *4*, 1469–1478.
21. Maulana, S.A.; Batubara, S.H.; Pasaribu, Y.P.P.; Syahputra, H.; Ramadhani, F. DETEKSI BURUNG MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK (CNN) DENGAN MODEL ARSITEKTUR MOBILENETV2. *JATI (Jurnal Mahasiswa Teknik Informatika)* **2024**, *8*, 6108–6114.