Jurnal Teknik Informatika Vol. 17 No. 2, Oktober 2024 (139-152)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v17i2.40453

JURNAL TEKNIK INFORMATIKA

Homepage : http://journal.uinjkt.ac.id/index.php/ti

Performance Analysis of Transfer Learning Models for Identifying AI-Generated and Real Images

Arini[1] , Muhamad Azhari[2], Isnaieni Ijtima' Amna Fitri[3], Feri Fahrianto[4]

[1,2,3,4]Department of Informatics Engineering, Faculty of Science and Technology, State Islamic University Syarif Hidayatullah Jakarta
[1,2,3,4]Jl.Ir H. Juanda Street Number 95, Ciputat, West Tangerang, Banten, Indonesia

Article:
Accepted: May 30, 2024
Revised: April 10, 2024
Issued: October 29, 2024

***Correspondence Address:**
feri.fahrianto@uinjkt.ac.id

ABSTRACT

This study aims to analyze and compare the performance of three transfer learning methods, namely InceptionV3, VGG16, and DenseNet121, in detecting AI-generated and real images. The background of this research is the unknown performance of transfer learning methods for detecting AI-generated and real images. This study introduces innovation by conducting 54 experiments involving three types of transfer learning, three dataset split ratios (60:40, 70:30, and 80:20), three optimizers (Adam, SGD, and RMSprop), two numbers of epochs (20 and 50), and the addition of dense and flatten layers during fine tuning. Performance evaluation was conducted using binary cross entropy loss and confusion matrix. This research provides significant benefits in determining the most effective transfer learning model for detecting AI-generated and real images and offers practical guidance for further development. The results show that the InceptionV3 model with the Adam optimizer, an 80:20 split ratio, and 20 epochs achieved the highest accuracy of 84.26%, with a loss of 39.54%, precision of 81.33%, recall of 82.43%, and an F1-Score of 81.88%.

Keywords : *AI generated image; deep learning, transfer learning; inceptionV3; VGG16; DenseNet121;*

Jurnal Teknik Informatika Vol. 17 No. 2, Oktober 2024 (139-152)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v17i2.40453

1. INTRODUCTION

The rapid advancement of artificial intelligence (AI) has led to the proliferation of AI-generated content, including images that are often indistinguishable from real ones. This raises significant challenges in various domains, from digital forensics to social media platforms, where the ability to distinguish between AI-generated and real images is crucial [1]. As a result, the application of machine learning techniques, particularly Transfer Learning, has become an essential area of research to address these challenges [2].

Transfer Learning has proven effective in scenarios where limited labeled data is available, as it leverages pre-trained models to transfer knowledge from one task to another [3]. In the context of image recognition, Transfer Learning allows models to generalize from previously learned features, making it highly suitable for tasks such as detecting AI-generated images [4]. Despite its potential, there is still a gap in the literature regarding the comparative performance of different Transfer Learning models in this specific application [5].

This study aims to fill that gap by evaluating and comparing the performance of three widely used Transfer Learning models: InceptionV3, VGG16, and DenseNet121. These models have been selected for their proven effectiveness in image classification tasks [6]. The choice of these models is motivated by their distinct architectural features, which offer varying strengths in handling image data, particularly in complex scenarios like distinguishing AI-generated images from real ones [7].

InceptionV3, for example, is known for its efficient use of computational resources and its ability to maintain high accuracy even with fewer parameters. On the other hand, VGG16's depth allows it to capture more complex features, while DenseNet121's dense connectivity promotes richer feature representations, reducing the risk of overfitting [8]. By analyzing these models under different conditions, this research seeks to identify the most effective approach for this critical task.

The findings of this study will not only contribute to the academic understanding of Transfer Learning's application in AI-generated image detection but also provide practical insights for developing more robust image recognition systems. This is particularly relevant in an era where the line between real and synthetic media is increasingly blurred, and the need for reliable detection methods is more urgent than ever.

2. METHODS

In reviewing the basic theory of the methods to be used, the following is an explanation of the related theories that will be addressed in the development of several transfer learning models and the analysis of their performance.

2.1. Deep Learning

Deep learning is a branch of artificial intelligence (AI) that leverages deep neural networks to model and understand complex data. This technology enables systems to autonomously learn from large and intricate datasets by hierarchically extracting high-level features. Deep learning is renowned for its ability to handle challenging tasks such as image and voice recognition, natural language processing, and solving problems that require deep contextual understanding. Deep neural networks allow for more abstract data representation, facilitating machines to learn and adapt with increasing levels of intelligence. [9]

2.2. CNN

Convolutional Neural Networks (CNN) are widely used in deep learning for tasks such as image classification, segmentation, object detection, video processing, natural language processing, and voice recognition. CNNs typically include convolutional layers, pooling layers, fully connected layers, and non-linear layers. They use kernel filters to extract fundamental features from input images and popular activation functions such as Sigmoid, Tanh, ReLU, Leaky ReLU, Noisy ReLU, and Parametric Linear Units. This architecture, influenced by the visual cortex, mimics the neural connections of the human brain. Examples include LeNet, AlexNet, and VGGNet. [10]

Jurnal Teknik Informatika Vol. 17 No. 2, Oktober 2024 (139-152)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v17i2.40453

2.3. Transfer Learning

Transfer Learning (TL) aims to enhance the understanding of the current task by linking it to related source domain tasks. TL improves learning by associating previous tasks with the target task, providing faster and better solutions. TL facilitates efficient learning and communication between the source tasks and the target task. However, it can lead to negative transfer if the testing and training samples are incorrectly transferred, thus reducing the performance of the target task. [11]

2.4. Inception V3

Inception V3 is a deep learning technique for classification. Its advantage lies in the complexity of its network structure, which includes input, output, and classification stages. The process involves extraction and various hidden layers such as convolution, pooling, activation (ReLU), softmax, and fully connected layers. With its structured approach, Inception V3 efficiently organizes object information by using convolutional outputs for the next convolutional step. [12]

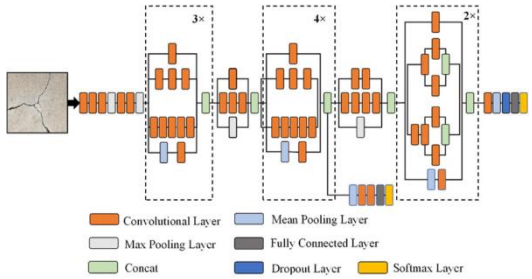

Figure 1. *Arsitektur inceptionV3*

2.5. DenseNet121

DenseNet121 is a unique Convolutional Neural Network (CNN) architecture with dense inter-layer connections. Unlike traditional CNNs, each layer in DenseNet121 is connected not only to its neighboring layers but also to every preceding layer. This dense connectivity improves information flow throughout the network, promotes richer feature representations, and efficiently reduces the risk of overfitting by utilizing outputs from previous layers as inputs.[13]

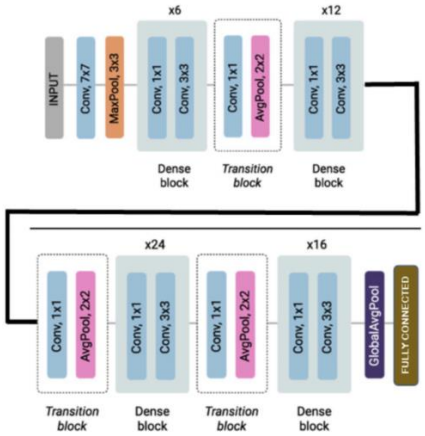

Figure 2. *Arsitektur DenseNet121*

2.6. VGG16

VGG16 is a Deep Convolutional Neural Network (DCNN) model proposed by Simonyan and Zisserman. The model achieved a top-5 test accuracy of 92.7% on the ImageNet dataset and won the Large-Scale Visual Recognition Challenge (ILSVRC) held by the Oxford Visual Geometry Group. The increased depth in the VGG model helps the kernels learn more complex features. In research on the effectiveness of transfer learning, the pre-trained and fine-tuned VGG16 achieves significantly higher accuracy compared to networks trained from scratch.[14]

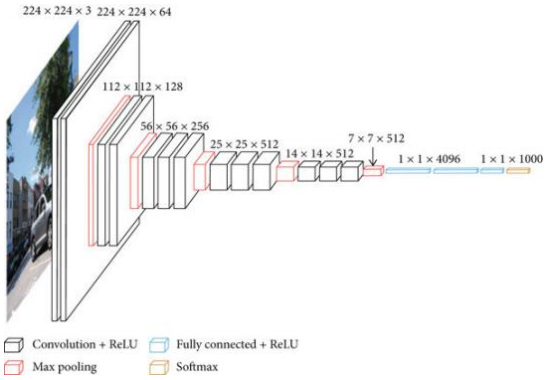

Figure 3. *Arsitektur VGG16*

2.7. Evaluation Method

2.7.1. Binary Cross Entropy

Binary cross entropy is a loss function used in binary classification [15]. The calculation of binary cross entropy can be done using the following formula:

$$L = -\frac{1}{m \times n}\left[\sum_{i=1}^{m}\sum_{j=1}^{n}\left(y_{ij}log(p_{ij}) + (1 - y_{ij})\log\left(1 - p_{ij}\right)\right)\right] \quad (1)$$

ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v17i2.40453

2.7.2. Confusion Matrix

The confusion matrix is a widely used tool in machine learning to assess the performance of a classifier on a dataset where the true labels are known. This tool helps clarify basic performance metrics in binary classification, including true positives, false positives, true negatives, and false negatives, which are often used as the basis for other evaluation metrics. Recently, with the growing interest in explaining and visualizing artificial intelligence, the confusion matrix has also been used as a fundamental explanatory component and a core visual element in many Explainable AI (XAI) toolkits and tutorials [16].

2.8. Data Preprocessing and Data Augmentation

2.8.1. Resize and Rescale

Resizing images in transfer learning ensures a consistent input size for the model, improving computational efficiency by adjusting the image resolution to fit device constraints such as memory limits [17]. Rescaling involves normalizing pixel scales to accelerate model convergence, reduce numerical issues, and enhance learning stability, as well as adapting the model to the diverse characteristics of the data [18].

2.8.2. Split Ratio

The choice of dataset split ratio in machine learning is crucial for optimizing model performance and evaluation. The proportion allocated to training and testing subsets directly affects the model's ability to generalize. A low training proportion can prevent the model from understanding complex data patterns, potentially leading to overfitting, where the model fits too closely to the training data but fails to generalize to new data. Conversely, a high training proportion can make the model less flexible and prone to underfitting, where the model fails to capture significant variations in the data.[19].

2.8.3. Rotation and Flip

Rotation and flipping are common preprocessing techniques in machine learning for image datasets, especially in computer vision tasks. Rotation diversifies the dataset by rotating images at angles such as 90 degrees, 180 degrees, or more complex angles, helping the model handle object orientation variations, prevent overfitting, and improve recognition from different viewpoints [20]. Meanwhile, horizontal or vertical flipping creates additional variations in the dataset by mirroring images, enhancing the model's generalization across various orientations in the test data. Balancing the use of rotation and flipping is crucial to preserving the original meaning of the images and avoiding significant information loss during preprocessing. [21].

2.8.4. Tools

OpenCV (Open Source Computer Vision) is an essential open-source library for image processing and computer vision applications. Released in 1999, the library supports basic image operations such as reading, writing, color conversion, cropping, and resizing. OpenCV also includes a range of computer vision algorithms, including face and object detection, tracking, feature matching, and segmentation, enabling the development of advanced machine vision systems [22].
Keras provides a tool called ImageDataGenerator that performs real-time augmentation of tensor image datasets. This function continuously generates batches of data [23].

2.9. Fine Tuning

Fine-tuning in the context of deep learning involves adjusting a pre-trained neural network model for a more specific task. First, a model pre-trained on a large dataset such as ImageNet is selected. Then, the model is adapted for the specific task by modifying the last layers and adding new layers to fit the desired output. During fine-tuning, some layers, especially the early ones that capture general features, may be kept unchanged (frozen), while the new layers are retrained for the specific task. [24].

2.10. Method of Collecting Data

2.10.1. Literature Review

The collection of literature review data is a systematic approach to gathering information from various relevant sources such as academic databases and digital libraries. This process involves identifying research questions, searching for scholarly articles and books, evaluating their relevance and quality, and analyzing them to understand relevant concepts, theories, and methodologies [25].

Jurnal Teknik Informatika Vol. 17 No. 2, Oktober 2024 (139-152)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v17i2.40453

2.10.2. Field Study

Field studies use observational data collection, which involves systematic and direct observation of objects, behaviors, or phenomena in their natural context. Researchers observe events directly without significant interference, using methods such as participant or non-participant observation. This approach collects data on behavior, social interactions, and contextual situations, providing in-depth insights into human behavior, group dynamics, or environmental characteristics in qualitative research [26].

2.11. Development Method CRISP-DM

The CRISP-DM (Cross-Industry Standard Process for Data Mining) framework is utilized in this study to guide the systematic approach to analyzing the performance of Transfer Learning models in detecting AI-generated and real images. This framework is composed of several phases, including Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. The selected Transfer Learning models InceptionV3, VGG16, and DenseNet121 are integrated into the Data Preparation, Modeling, and Evaluation phases of this framework, each contributing uniquely to the overall process.

Data Preparation: In the Data Preparation phase, the dataset is preprocessed to ensure compatibility with the chosen models. Each model requires specific image input sizes and data normalization procedures. InceptionV3, for example, requires images to be resized to 299x299 pixels, while VGG16 and DenseNet121 work with 224x224 pixels. The preprocessing steps also include data augmentation techniques, such as rotation, flipping, and scaling, which are applied to enhance the robustness of the models.

Modeling: During the Modeling phase, the three Transfer Learning models are fine-tuned to adapt to the specific characteristics of the AI-generated and real images in the dataset. Fine-tuning involves adjusting the pre-trained weights of the models on the new dataset, freezing some layers to retain the general features learned from the original training on large datasets like ImageNet, while retraining the final layers to improve specificity in distinguishing AI-generated images. The choice of optimizer, learning rate, and number of epochs are carefully selected for each model to maximize performance.

Evaluation: The Evaluation phase focuses on assessing the models' performance using metrics confusion matrix and binary cross-entropy loss. A confusion matrix is generated to visualize the performance of each model in correctly identifying AI-generated and real images. InceptionV3, with its efficient architecture, shows strengths in scenarios with complex image patterns, while VGG16 and DenseNet121 offer deeper feature extraction capabilities that are beneficial in cases where subtle differences between AI-generated and real images need to be detected [27].

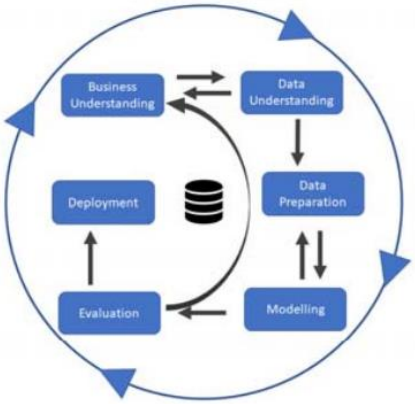

Figure 4. *Phase of CRISP-DM method*

3. RESULTS AND DISCUSSION

3.1. Research Plan

3.1.1. Business Understanding

The goal is to compare the performance of three transfer learning (TL) algorithms InceptionV3, VGG16, and DenseNet121 due to the lack of existing research in this area, in order to identify the optimal performance in detecting AI-generated images compared to real images.

3.1.2. Data Understanding

a. Dataset Search

1. The researchers began their search for datasets on Kaggle, a well-known platform offering a wide variety of datasets and research projects for data scientists and machine learning enthusiasts.
2. After registering and logging into Kaggle, the researchers utilized the search feature by entering keywords such as "AI Image and Real," "AI Image

Generated," and "AI vs Real Image" to identify relevant datasets.
3. Kaggle displayed a list of search results matching the entered keywords, allowing the researchers to review and assess potential datasets that fit their research needs.
4. The researchers evaluated the identified datasets based on criteria such as image consistency, the number of images, class distribution, and image clarity, ensuring that the dataset met the requirements for training with transfer learning methods.
5. After selecting a suitable dataset, the researchers proceeded to download it from Kaggle for further processing and analysis in their research efforts.

b. Data Visualization

Visualization plays a key role in understanding the distribution and characteristics of the dataset. In this phase, various visualizations are generated to provide insights into the dataset composition and the impact of preprocessing steps:

1. Class Distribution Visualization: Bar charts are used to visualize the distribution of AI-generated and real images across training, validation, and test sets. This helps ensure that the dataset is balanced and that the models are exposed to a diverse range of images during training.
2. Class Distribution Visualization: Bar charts are used to visualize the distribution of AI-generated and real images across training, validation, and test sets. This helps ensure that the dataset is balanced and that the models are exposed to a diverse range of images during training.

Here are the steps to create a bar chart visualization of the dataset split ratio:

1. Import the necessary modules: `os` for system operations, `pandas` for data manipulation, `seaborn` and `matplotlib.pyplot` for visualization. Use the code `import os`, `import pandas as pd`, `import seaborn as sns`, and `import matplotlib.pyplot as plt.
2. Define the directory where the data is located using `directory = "/content/data/train_split"`.
3. Count the number of images for each class (real and AI) in each part of the dataset (train, test, and validation) using list comprehensions and the `len()` function. Example code: `train_real = len([os.path.join(directory+'/train/real', filename) for filename in os.listdir(directory+'/train/real')])`.
4. Create a DataFrame from the previously counted data. This DataFrame contains information about the number of images for each class in each part of the dataset.
5. "Melt" the DataFrame for visualization. Melting changes the DataFrame structure from wide format to long format, which is more suitable for visualization. Example code: `melted_df = df.melt(id_vars=['Dataset'], value_vars=['real', 'ai'], var_name='Class', value_name='Count')`.
6. Finally, create a plot using seaborn to display the dataset distribution for each part (train, test, and validation). This plot shows the number of images on the y-axis and the dataset type (real or AI) on the x-axis, with dataset parts (train, test, or validation) represented by color. Use the code `sns.barplot(data=melted_df, x='Dataset', y='Count', hue='Class')`.

Here are the steps to create a visualization of training, testing, and validation sample sets with an output of 10 images:

1. First, code randomly selects 5 samples from the "real" and "AI" classes in the training dataset using `random.sample`.
2. Next, these samples are plotted using the `plot_samples` function.
3. Then, the plot title is set to "Training/Test/Validation Set Samples" with a font size of 30.
4. These three steps are repeated three times for samples from the training, testing, and validation sets.

3.1.3. Data Preparation

a. Data Preprocessing

1. Split Ratio
   a) The code first collects all image files from the 'AI' and 'real' directories within the train directory.
   b) Using `train_test_split` from scikit-learn, the dataset for each

class (AI and real) is divided into training, testing, and validation sets. The proportions are set using `random_state` to ensure consistency in the splits each time the code is run.

c) Next, the code creates a new directory structure to store the divided dataset: train, test, and validation.

d) Each of these directories contains subdirectories 'AI' and 'real' representing the dataset classes.

e) Image files from the training, testing, and validation sets for each class are moved to their respective directories based on the previous split using `shutil.copy`.

f) After this process, there are three new directories containing images divided into training, testing, and validation sets. This directory structure is ready to be used as input for the model.

2. Rescaling: This is done using the parameter `rescale = 1/255.`, which adjusts pixel values to be between 0 and 1.
3. Resizing: This is done using the parameter `target_size = (224, 224)`, which resizes the images to dimensions of 224x224 pixels.

b. Data Augmentation

To enhance the robustness of the models, data augmentation techniques such as rotation, flipping, zooming, and shifting are applied. These techniques artificially increase the size of the training dataset and help the models generalize better to unseen data. Here is a further explanation:

1. Horizontal Flip: Applied using `horizontal_flip = True` to perform horizontal flipping of the images.
2. Vertical Flip: Applied using `vertical_flip = True` to perform vertical flipping of the images.
3. Rotation Range: Rotates images within a range of 0.3 degrees using `rotation_range = 0.3`.
4. Width Shift Range: Shifts images horizontally by 0.25 of the image width using `width_shift_range = 0.25`.
5. Height Shift Range: Shifts images vertically by 0.25 of the image height using `height_shift_range = 0.25`.
6. Channel Shift Range: Shifts color channels within a range of 0.35 using `channel_shift_range = 0.35`.
7. Shear Range: Applies shear transformation to images within a range of 0.2 using `shear_range = 0.2`.
8. Zoom Range: Applies zoom to images within a range of 0.4 using `zoom_range = 0.4`.
9. ZCA Whitening: Uses `zca_whitening = True` to reduce pixel correlation in images.

3.1.4. Modelling

Next is the modeling phase. For this study, the researchers chose to use three transfer learning models: InceptionV3, VGG16, and DenseNet121. Additionally, there are 3 optimizers (Adam, SGD, and RMSprop), 3 dataset split ratios (60:40, 70:30, and 80:20), and 2 epoch types (20 and 50). The base layers of each model are frozen to retain the general features learned from ImageNet, while the top layers are re-trained on the new dataset. This allows the models to adapt to the specific nuances of AI-generated and real images. Different optimizers (Adam, SGD, RMSprop) and lratio dataset are tested to find the optimal training configuration for each model. The number of epochs is varied to determine the point at which each model achieves the best balance between accuracy and overfitting. Here is a further explanation:

a. Import the `Sequential` class from the `models` module in Keras. `Sequential` is a Keras model that allows for adding layers sequentially.
b. Import the `Flatten` and `Dense` classes from the `layers` module in Keras. `Flatten` is used to flatten the output from the previous layer into a single dimension, while `Dense` is used to add fully connected layers.
c. Import the chosen transfer learning models from the `applications` module in Keras.
d. Import the `ImageDataGenerator` class from the `preprocessing.image` module in Keras. `ImageDataGenerator` is used for data augmentation on images during model training.

e. Create each of the chosen transfer learning models using pre-trained weights from the ImageNet dataset (`weights='imagenet'`), without the top layer (`include_top=False`). Then specify the expected input shape for each model.
f. Add a `Flatten` layer to flatten the output from each transfer learning model into a single dimension. Add several `Dense` layers with ReLU activation, followed by a final `Dense` layer with a single neuron and sigmoid activation. These layers can be trained, while the previous transfer learning layers are frozen.

3.1.5. Evaluation

The "evaluation" phase in the CRISP-DM methodology is a crucial step in comparative research on the performance of InceptionV3, VGG16, and DenseNet121 models in detecting AI-generated images versus real images. During training, models are evaluated using metrics such as confusion matrix and binary cross-entropy loss. These metrics are tracked across different configurations to identify the best-performing model under various conditions. Here is an explanation of the binary cross-entropy loss function:

a. Using `evaluate_generator` to assess the model with test data (`test_set`). This means the model will be evaluated using data that was not used during the training or validation processes.
b. Printing the evaluation results as percentages. `model.metrics_names` is an attribute that contains a list of metric names used in the model. `model.metrics_names[0]` refers to the name of the first metric, which in this context is the loss, and `model.metrics_names[1]` refers to the name of the second metric, which is accuracy.
c. The model is re-evaluated using the `evaluate` method, this time with `test_set`. The evaluation results, namely loss and accuracy, are stored in the variables `test_loss` and `test_accuracy`. Finally, these results are printed as percentages to illustrate the model's performance on the test data.

Here is an explanation of the confusion matrix:

a. The text "-----CONFUSION MATRIX-----" is printed to mark the visualization of the confusion matrix.
b. Predictions are made on the test data (`test_data`) using the trained model, and the results are stored in the `predictions` variable.
c. The confusion matrix (`conf_m`) is computed using the `confusion_matrix` function from sklearn with `test_labels` (the true labels of the test data) and `np.round(predictions)` (the labels predicted by the model).
d. Accuracy (`acc`) is calculated using the `accuracy_score` function from sklearn with `test_labels` and `np.round(predictions)`, then multiplied by 100 to convert it to a percentage.
e. The number of true negatives (tn), false positives (fp), false negatives (fn), and true positives (tp) is extracted from the confusion matrix using the `ravel()` method.
f. The confusion matrix is visualized by calling the `plot_confusion_matrix` function with arguments `conf_mat=conf_m` (the confusion matrix), `figsize=(6, 6)` (image size), and `cmap=matplotlib.pyplot.cm.Reds` (color map for visualization).
g. After visualization, the text "-----CLASSIFICATION REPORT-----" is printed to mark the presentation of the classification report.
h. Precision (`precision`) is calculated using the formula (tp / (tp + fp) * 100), where TP is true positive and FP is false positive.
i. Recall (`recall`) is calculated using the formula (tp / (tp + fn) * 100), where FN is false negative.
j. Accuracy, precision, recall, and F1-score are printed as percentages along with their respective values. The F1-score is calculated using the harmonic mean formula (2 * precision * recall / (precision + recall)).

Jurnal Teknik Informatika Vol. 17 No. 2, Oktober 2024 (139-152)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v17i2.40453

3.2. Results and Discussion

3.2.1. Research Results

a. Dataset search results

Based on the data obtained from the Kaggle website, the dataset was downloaded and extracted from the ZIP file into PNG files. The dataset contains a total of 975 images, with 539 images generated by AI and 436 real images. This dataset was obtained from Kaggle and created by Bekhzod Olimov.

b. Data visualization results

There are two data visualizations: the first is the visualization for the dataset split ratio using a bar chart, and the second is the visualization for random sample images.

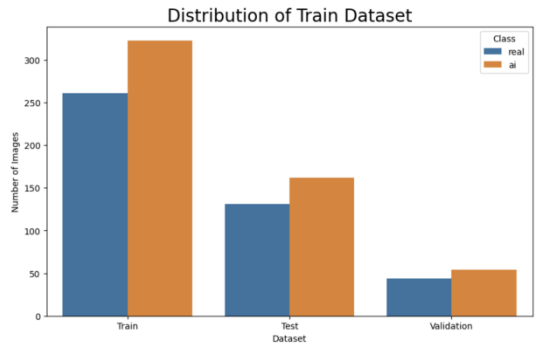

Figure 5. *One of data visualization split ratio dataset*

Figure 6. *One of data visualization random samples image*

c. Data preparation results

```
Found 776 images belonging to 2 classes.
Found 340 images belonging to 2 classes.
Found 152 images belonging to 2 classes.
CPU times: user 59.4 ms, sys: 6.07 ms, total: 65.4 ms
Wall time: 141 ms
```

Figure 7. *One of ImageDataGenerator Output*

Figure 7 shows the results of the ImageDataGenerator process. The training set contains 776 images across 2 classes, while the validation and test sets have 340 and 152 images, respectively. "CPU times" and "Wall time" measure execution duration and real-time duration. Data augmentation is consistently applied across all experiments.

d. Evaluation results

1. Split ratio (60:40) and Optimizer (Adam)

Tabel 1. *Result split ratio (60:40) and optimizer (adam)*

Model	Classification Report					TrainingTime (minute)	Confusion Matrix			
	Accuracy	Precision	Recall	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
InceptionV3-20	75.09%	69.86%	77.86 %	73.65 %	56.22 %	43	118	102	44	29
VGG16-20	81.23%	77.14%	82.44 %	79.70 %	48.71 %	144	130	108	32	23
DenseNet121-20	78.16%	71.90%	83.97 %	77.46 %	59.63 %	43	119	110	43	21
InceptionV3-50	75.09%	72.31%	71.76 %	72.03 %	54.92 %	99	126	94	36	37
VGG16-50	76.79%	74.42%	73.28 %	73.85 %	46.70 %	172	129	96	33	35
DenseNet121-50	78.50%	80.36%	68.70 %	74.07 %	45.48 %	63	140	90	22	41

2. Split ratio (60:40) and Optimizer (SGD)

Tabel 2. *Result split ratio (60:40) and optimizer (SGD)*

Model	Classification Report					TrainingTime (minute)	Confusion Matrix			
	Accuracy	Precision	Recall	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
InceptionV3-20	74.06%	71.32%	70.23 %	70.77 %	51.23 %	45	125	92	37	39
VGG16-20	73.72%	85.53%	49.62 %	62.80 %	53.96 %	151	151	65	11	66
DenseNet121-20	75.77%	81.91%	58.78 %	68.44 %	49.96 %	42	145	77	17	54
InceptionV3-50	75.77%	71.43%	76.34 %	73.80 %	50.67 %	69	122	100	40	31
VGG16-50	77.13%	69.05%	88.55 %	77.59 %	49.14 %	374	110	116	52	15
DenseNet121-50	79.18%	75.74%	78.63 %	77.15 %	44.24 %	77	129	103	33	28

3. Split ratio (60:40) and Optimizer (RMSprop)

Tabel 3. *Result split ratio (60:40) and optimizer (RMSprop)*

Model	Classification Report					TrainingTime (minute)	Confusion Matrix			
	Accuracy	Precision	Recall	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
InceptionV3-20	77.82%	77.50%	70.99 %	74.11 %	50.93 %	44	135	93	27	38
VGG16-20	75.09%	75.00%	66.41 %	70.45 %	47.25 %	144	133	87	29	44
DenseNet121-20	72.01%	73.79%	58.02 %	64.96 %	61.07 %	45	135	76	27	55
InceptionV3-50	75.09%	75%	66.41 %	70.45 %	50.24 %	86	133	87	29	44
VGG16-50	78.50%	76.15%	75.57 %	75.86 %	60.63 %	296	131	99	31	32
DenseNet121-50	74.40%	71.54%	70.99 %	71.26 %	52.48 %	77	125	93	37	38

Jurnal Teknik Informatika Vol. 17 No. 2, Oktober 2024 (139-152)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v17i2.40453

4. Split ratio (70:30) and Optimizer (Adam)

Tabel 4. *Result split ratio (70:30) and optimizer (adam)*

Model	Classification Report					Training Time (minute)	Confusion Matrix			
	Accuracy	Precision	Recall	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
InceptionV3-20	81.65%	76.26%	80.92 %	78.52 %	41.68 %	43	152	106	33	25
VGG16-20	75.32%	81.93%	51.91 %	63.55 %	49.94 %	158	170	68	15	63
DenseNet121-20	77.22%	67.88%	85.50 %	75.67 %	58.26 %	49	132	112	53	19
InceptionV3-50	75.32%	81.93%	51.91 %	63.55 %	57.99 %	48	170	68	15	63
VGG16-50	76.19%	81.98%	69.47 %	75.20 %	49.85 %	216	101	91	20	40
DenseNet121-50	80.70%	78.23%	74.05 %	76.08 %	40.85 %	78	158	97	27	34

5. Split ratio (70:30) and Optimizer (SGD)

Tabel 5. *Result split ratio (70:30) and optimizer (SGD)*

Model	Classification Report					Training Time (minute)	Confusion Matrix			
	Accuracy	Precision	Recall	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
InceptionV3-20	73.42%	63.91%	82.44 %	72.00 %	49.73 %	44	124	108	61	23
VGG16-20	74.21%	68.97%	91.60 %	78.69 %	56.31 %	165	67	120	54	11
DenseNet121-20	78.16%	84.44%	58.02 %	68.78 %	50.55 %	49	171	76	14	55
InceptionV3-50	79.43%	79.46%	67.94 %	73.25 %	44.26 %	45	162	89	23	42
VGG16-50	75.79%	78.69%	73.28 %	75.89 %	50.96 %	410	95	96	26	35
DenseNet121-50	79.37%	89.11%	68.70 %	77.59 %	50.88 %	133	110	90	11	41

6. Split ratio (70:30) and Optimizer (RMSprop)

Tabel 6. *Result split ratio (70:30) and optimizer (RMSprop)*

Model	Classification Report					Training Time (minute)	Confusion Matrix			
	Accuracy	Precision	Recall	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
InceptionV3-20	78.72%	70.27%	87.84 %	78.08 %	46.05 %	57	140	130	55	18
VGG16-20	78.97%	81.45%	77.10 %	79.22 %	47.59 %	160	98	101	23	30
DenseNet121-20	78.80%	80.19%	64.89 %	71.73 %	47.01 %	51	164	85	21	46
InceptionV3-50	75.63%	90.91%	45.80 %	60.91 %	52.53 %	62	179	60	6	71

Table 6 continued…

Model	Classification Report					Training Time (minute)	Confusion Matrix			
	Accuracy	Precision	Recal	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
VGG16-50	74.21%	84.38%	61.83 %	71.37 %	53.02 %	199	106	81	15	50
DenseNet121-50	78.97%	87.50%	69.47 %	77.45 %	54.56 %	68	108	91	13	40

7. Split ratio (80:20) and Optimizer (Adam)

Tabel 7. *Result split ratio (80:20) and optimizer (adam)*

Model	Classification Report					Training Time (minute)	Confusion Matrix			
	Accuracy	Precision	Recall	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
InceptionV3-20	84.26%	81.33%	82.43 %	81.88 %	39.54 %	62	167	122	28	26
VGG16-20	72.79%	82.50%	50.00 %	62.26 %	51.90 %	161	74	33	7	33
DenseNet121-20	73.47%	86.49%	48.48 %	62.14 %	61.80 %	58	76	32	5	34
InceptionV3-50	79.88%	89.11%	60.81 %	72.29 %	43.61 %	142	184	90	11	58
VGG16-50	81.79%	84.67%	78.38 %	81.40 %	42.36 %	319	122	116	21	32
DenseNet121-50	80.27%	75.34%	83.33 %	79.14 %	42.22 %	93	63	55	18	11

8. Split ratio (80:20) and Optimizer (SGD)

Tabel 8. *Result split ratio (80:20) and optimizer (SGD)*

Model	Classification Report					Training Time (minute)	Confusion Matrix			
	Accuracy	Precision	Recall	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
InceptionV3-20	83.38%	84.73%	75.00 %	79.57 %	38.70 %	57	175	111	20	37
VGG16-20	61.22%	73.68%	21.21 %	32.94 %	60.95 %	160	76	14	5	52
DenseNet121-20	75.51%	89.47%	51.52 %	65.38 %	53.99 %	56	77	34	4	32
InceptionV3-50	82.31%	83.33%	75.76 %	79.36 %	38.68 %	62	71	50	10	16
VGG16-50	76.87%	71.05%	81.82 %	76.06 %	47.59 %	471	59	54	22	12
DenseNet121-50	74.83%	70.42%	75.76 %	72.99 %	48.86 %	75	60	50	21	16

Jurnal Teknik Informatika Vol. 17 No. 2, Oktober 2024 (139-152)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v17i2.40453

9. Split ratio (80:20) and Optimizer (RMSprop)

Tabel 9. *Result split ratio (80:20) and optimizer (RMSprop)*

Model	Classification Report					Training Time (minute)	Confusion Matrix			
	Accuracy	Precision	Recall	F1-Score	Binary Cross Entropy		TP	TN	FP	FN
InceptionV3-20	78.72%	89.47%	57.43 %	69.96 %	45.59 %	61	185	85	10	63
VGG16-20	77.55%	80.00%	66.67 %	72.73 %	55.59 %	113	70	44	11	22
DenseNet121-20	80.95%	83.93%	71.21 %	77.05 %	44.05 %	58	72	47	9	19
InceptionV3-50	81.63%	79.10%	80.30 %	79.70 %	40.70 %	95	67	53	14	13
VGG16-50	75.51%	71.43%	75.76 %	73.53 %	48.86 %	210	61	50	20	16
DenseNet121-50	78.23%	75.00%	77.27 %	76.12 %	43.63 %	70	64	51	17	15

3.2.2. The Influence of Aspects on Best Performance Results

Tabel 10. *The influence of aspects on best performance results*

Pengaruh Aspek-Aspek Pada Hasil Performa Terbaik			
TL / Parameter	**Nama**	**Akurasi**	**Keterangan**
TL	InceptionV3	81.65%	TL: InceptionV3, Split ratio: 70:30, Optimizer: Adam, Epoch: 20
Split Ratio	70:30	81.65%	TL: InceptionV3, Optimizer: Adam, Epoch: 20
Optimizer	Adam	81.65%	TL: InceptionV3, Split ratio: 70:30, Epoch: 20
Epoch	50	81.63%	TL: InceptionV3, Split ratio: 70:30, Optimizer: RMSprop

a. Transfer Learning

Based on performance analysis using various evaluation metrics, the transfer learning model that consistently achieved the highest accuracy percentage is InceptionV3.

InceptionV3:

1. Using the Adam optimizer, this model achieved an accuracy of 81.65% with a 70:30 split ratio and 20 epochs.
2. With the RMSprop optimizer, the accuracy reached 81.63% with the same 70:30 split ratio but with 50 epochs.
3. Using the SGD optimizer, this model achieved an accuracy of 79.43% with a 70:30 split ratio and 50 epochs.

InceptionV3 consistently showed high accuracy across various optimizers and with the 70:30 data split setting. This indicates the model's reliability and effectiveness in handling different dataset partitions and achieving high classification performance.

b. Split Ratio Dataset

Based on performance analysis using various evaluation metrics, the 70:30 split ratio often results in the best performance in terms of accuracy percentage. The experiments that showed promising results are as follows:

DenseNet121:

1. With the Adam optimizer, accuracy reached 79.18% at 50 epochs.
2. With the SGD optimizer, accuracy reached 79.37% at 50 epochs.
3. With the RMSprop optimizer, the model showed stability with an accuracy of 78.97% at 50 epochs.

InceptionV3:

1. With the Adam optimizer, accuracy reached 81.65% at 20 epochs.
2. With the SGD optimizer, accuracy reached 79.43% at 50 epochs.
3. With the RMSprop optimizer, accuracy reached 81.63% at 50 epochs.

c. Optimizer

Based on the analysis using model evaluation methods, the Adam optimizer often yields the best performance in terms of accuracy percentage.

1. DenseNet121 with Adam Optimizer, at a 70:30 split ratio and 50 epochs, achieved an accuracy of 79.18%.
2. InceptionV3 with Adam Optimizer, at a 70:30 split ratio and 20 epochs, achieved an accuracy of 81.65%.

Optimizer Adam generally provides good performance due to its efficiency in handling gradients and accelerating model convergence. This effectiveness is demonstrated in image classification tasks using DenseNet121 and InceptionV3.

d. Epoch

Based on the performance analysis of the models using various evaluation metrics, the best accuracy results for the DenseNet121 and InceptionV3 models tend to occur at epoch 50 in the experiments conducted.

1. DenseNet121, with a 70:30 split ratio and 50 epochs, achieved an accuracy of 79.37% with the SGD optimizer.
2. InceptionV3, with a 70:30 split ratio and 50 epochs, achieved an accuracy of 81.63% with the RMSprop optimizer.

Although there were higher accuracy results at epoch 20 for DenseNet121 (79.37% with SGD optimizer) and InceptionV3 (81.65% with Adam optimizer), the choice of epoch 50 as the optimal point is based on the better consistency of performance across various experiments. Higher results at epoch 20 in specific cases indicate that the model may have reached an optimal convergence point with the validation data at that stage. However, overall, epoch 50 provides a good balance between sufficient learning time and the model's ability to achieve optimal generalization.

3.2.3. Comparasion of Results with Previous Research

In the study conducted by Bekhzod Olimov using the same dataset he created, titled "AI-Generated Vs Real Images Classifier," several elements are compared with previous research using the same dataset:

a. Previous research used only one transfer learning model, ResNet150, while this study compares three models: InceptionV3, VGG16, and DenseNet121.
b. The dataset split ratio in previous research was 90:5:5, whereas this study experiments with split ratios of 60:40, 70:30, and 80:20.
c. Previous research used only one optimizer, Adam, while this study employs three optimizers: Adam, SGD, and RMSprop, although Adam is used in both studies.
d. Previous research used 8 epochs, whereas this study experiments with 20 and 50 epochs.
e. The highest accuracy achieved in previous research was 98.9%, while in this study, the highest accuracy achieved is 84.26% with InceptionV3 using Adam optimizer, 80:20 split ratio, and 20 epochs.
f. The lowest loss in previous research was 27%, whereas the lowest loss in this study for the InceptionV3 model is 38.68% with the RMSprop optimizer, 80:20 split ratio, and 50 epochs.

CONCLUSION

Configuration of split ratio, optimizer, and epoch significantly affects model performance. Here are the best and worst parameter compositions based on average accuracy and training time for each model experiment:

a. The parameter composition that achieved the best accuracy performance at 20 epochs is the RMSprop optimizer with an 80-20 dataset split ratio, reaching an average accuracy of 79.70%.
b. The parameter composition that resulted in the fastest training time at 20 epochs is the Adam optimizer with a 60-40 dataset split ratio, reaching an average training time of 76.66 minutes.
c. The parameter composition that achieved the best accuracy performance at 50 epochs is the Adam optimizer with an 80-20 dataset split ratio, reaching an average accuracy of 80.65%.
d. The parameter composition that resulted in the fastest training time at 50 epochs is the RMSprop optimizer with a 70-30 dataset split ratio, reaching an average training time of 109.66 minutes.

Based on the performance evaluation results discussed in the previous chapter, here are the best evaluation results for each model based on the evaluation methods applied:

a. Based on binary cross-entropy loss, the lowest loss for InceptionV3 is 38.68% using an 80-20 split ratio, RMSprop optimizer, and 50 epochs. For VGG16, the lowest loss is 46.70% using a 60-40 split ratio, Adam optimizer, and 50 epochs. For DenseNet121, the lowest loss is 38.68% using a 70-30 split ratio, Adam optimizer, and 50 epochs.
b. Based on the confusion matrix for InceptionV3, the highest accuracy achieved is 84.26%, with a precision of 81.33%, recall of 82.43%, F1-Score of 81.88, 167 True Positives (TP), 122 True Negatives (TN), 28 False Positives (FP), and 26 False Negatives (FN) using the Adam optimizer, an 80-20 split ratio, and 20 epochs.
c. Based on the confusion matrix for VGG16, the highest accuracy achieved is 81.79%, with a precision of 84.67%, recall of 78.38%, F1-Score of 81.40, 122 True Positives (TP), 116 True Negatives (TN), 21 False Positives (FP), and 32 False Negatives (FN) using the Adam optimizer, an 80-20 split ratio, and 50 epochs.

d. Based on the confusion matrix for DenseNet121, the highest accuracy achieved is 80.95%, with a precision of 83.93%, recall of 71.21%, F1-Score of 77.05, 72 True Positives (TP), 47 True Negatives (TN), 9 False Positives (FP), and 19 False Negatives (FN) using the Adam optimizer, an 80-20 split ratio, and 20 epochs.

In this study, the researcher acknowledges several limitations and shortcomings. Based on the findings, the following recommendations are proposed to enhance future research: (1) Expanding from the current binary classification to multi-class classification could advance the study. (2) Utilizing diverse datasets with varying characteristics is crucial to assess the consistency of results across different conditions and to improve model generalization. (3) Further fine-tuning of the model by adding new layers, such as convolutional or dense layers, may enhance performance. (4) Experimenting with different parameters like learning rate, batch size, optimizer, and number of epochs, and considering hyperparameter tuning or grid search, could help identify the optimal combination for better model performance.

REFERENCES

[1] D. Epstein, S. Jain, S. Wang, and Z. Zhang, "Online Detection of AI-Generated Images," in Proc. ICCVW, 2023. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023W/DFAD/papers/Epstein_Online_Detection_of_AI-Generated_Images__ICCVW_2023_paper.pdf

[2] S. S. Barahheem and T. V. Nguyen, "AI vs. AI: Can AI Detect AI-Generated Images?" [Online]. Available: https://www.researchgate.net/publication/374269358_AI_vs_AI_Can_AI_Detect_AI-Generated_Images

[3] J. Gu et al., "AI-enabled image fraud in scientific publications," J. Big Data, vol. 10, no. 1, pp. 1-12, 2022. [Online]. Available: https://www-sciencedirect-com.translate.goog/science/article/pii/S2666389922001039?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=tc

[4] A. Peryanto, A. Yudhana, and R. Umar, "Rancang Bangun Klasifikasi Citra Dengan Teknologi Deep Learning Berbasis Metode Convolutional Neural Network," in Proc. FORMAT, 2019, vol. 8, pp. 10-20. [Online]. Available: https://publikasi.mercubuana.ac.id/index.php/format/article/view/7849

[5] Y. Zhou, X. Zhang, Y. Wang, and B. Zhang, "Transfer Learning and Its Application Research," J. Phys. Conf. Ser., vol. 1920, no. 1, p. 012058, 2021. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1920/1/012058/pdf

[6] M. Ahmed et al., "An inception V3 approach for malware classification using machine learning and transfer learning," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 8, pp. 1-12, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666603022000252

[7] A. M. Ibrahim et al., "Skin Cancer Classification Using Transfer Learning by VGG16 Architecture (Case Study on Kaggle Dataset)," Open Access Library Journal, vol. 10, pp. 1-12, 2023. [Online]. Available: https://www.scirp.org/journal/paperinformation?paperid=126855

[8] J. Pardede and D. A. L. Putra, "Implementasi DenseNet Untuk Mengidentifikasi Kanker Kulit Melanoma," JUTISI, vol. 8, no. 1, pp. 10-20, 2020. [Online]. Available: https://journal.maranatha.edu/index.php/jutisi/article/download/2814/1708/10108

[9] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," Bus. Inf. Syst. Eng., vol. 63, no. 3, pp. 303–314, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s12525-021-00475-2

[10] P. Purwono et al., "Understanding of Convolutional Neural Network (CNN): A Review," IJRCS, vol. 1, no. 1, pp. 1-8, 2022. [Online]. Available: https://www.pubs2.ascee.org/index.php/IJRCS/article/view/888

[11] A. Hosna et al., "Transfer learning: a friendly introduction," J. Big Data, vol. 9, no. 1, pp. 1-21, 2022. [Online]. Available:

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00652-w

[12] U. Ungkawa and G. A. Hakim, "Klasifikasi Warna pada Kematangan Buah Kopi Kuning menggunakan Metode CNN Inception V3," ELKOMIKA, vol. 11, no. 1, pp. 12-20, 2023. [Online]. Available: https://ejurnal.itenas.ac.id/index.php/elkomika/article/view/8899

[13] W. W. Kusuma, R. R. Isnanto, and A. Fauzi, "Analisis Perbandingan Model CNN VGG16 Dan DenseNet121 Menggunakan Kerangka Kerja Tensorflow Untuk DeteksiI Jenis Hewan," J. Teknol. Komput., vol. 8, no. 1, pp. 20-30, 2023. [Online]. Available: https://ejournal3.undip.ac.id/index.php/jtk/article/view/37009/28851

[14] S. Montaha et al., "BreastNet18: A High Accuracy Fine-Tuned VGG16 Model Evaluated Using Ablation Study for Diagnosing Breast Cancer from Enhanced Mammography Images," Biology, vol. 10, no. 12, p. 1347, 2021. [Online]. Available: https://www.mdpi.com/2079-7737/10/12/1347

[15] M. A. Djohar et al., "Liver Segmentation Using Convolutional Neural Network Method with U-Net Architecture," J. Inf. Technol. Electr., vol. 5, no. 2, pp. 1-8, 2022. [Online]. Available: https://ojs.uma.ac.id/index.php/jite/article/view/6751/4088

[16] H. Shen et al., "Designing Alternative Representations of Confusion Matrices to Support Non-Expert Public Understanding of Algorithm Performance," in Proc. 23rd ACM Conf. Comput. Interact. Sci. Pract., 2020, pp. 1-11. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3415224

[17] S. Saponara and A. Elhanashi, "Impact of Image Resizing on Deep Learning Detectors for Training Time and Model Performance," in Lecture Notes in Comput. Sci., vol. 12345, pp. 1-10, 2022. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-95498-7_2

[18] Z. Pan et al., "Towards Bidirectional Arbitrary Image Rescaling: Joint Optimization and Cycle Idempotence," in Proc. CVPR, 2022. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Pan_Towards_Bidirectional_Arbitrary_Image_Rescaling_Joint_Optimization_and_Cycle_Idempotence_CVPR_2022_paper.html

[19] A. Racz, D. Bajusz, and K. Heberger, "Effect of Dataset Size and Train/Test Split Ratios in QSAR/QSPR Multiclass Classification," Molecules, vol. 26, no. 4, p. 1111, 2021. [Online]. Available: https://www.mdpi.com/1420-3049/26/4/1111

[20] K. Zhang, Z. Cao, and J. Wu, "Circular Shift: An Effective Data Augmentation Method For Convolutional Neural Network On Image Classification," in Proc. 25th Int. Conf. Pattern Recognit., 2020, pp. 1-8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9191303

[21] A. Abdelhamed et al., "NTIRE 2020 Challenge on Real Image Denoising: Dataset, Methods and Results," in Proc. CVPRW, 2020, pp. 1-12. [Online]. Available: https://openaccess.thecvf.com/content_CVPRW_2020/html/w31/Abdelhamed_NTIRE_2020_Challenge_on_Real_Image_Denoising_Dataset_Methods_and_CVPRW_2020_paper.html

[22] J. Sigut et al., "OpenCV Basics: A Mobile Application to Support the Teaching of Computer Vision Concepts," in Proc. 2020 IEEE Global Eng. Educ. Conf., 2020, pp. 1-8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9103956

[23] B. Raharjo, "Deep Learning dengan Python," Yayasan Prima Agus Teknik, 2022. [Online]. Available: https://digilib.stekom.ac.id/assets/dokumen/ebook/feb_eab5a7c1f295129ba69a76fee4dff22266879314_1643796893.pdf

[24] A. K. Reyes, J. C. Caicedo, and J. Camargo, "Fine-tuning Deep Convolutional Networks for Plant Recognition," in Proc. 2019 IEEE Conf