Journal of Applied Informatics and Computing (JAIC)
Vol.8, No.2, December 2024, pp. 557~568
e-ISSN: 2548-6861

Comparative Study: Flower Classification using Deep Learning, SMOTE and Fine-Tuning

Vincentius Praskatama [1]*, Guruh Fajar Shidik [2]*, Amanda Prawita Ningrum [3]*
* Teknik Informatika, Universitas Dian Nuswantoro Semarang
111202113456@mhs.dinus.ac.id [1], guruh.fajar@research.dinus.ac.id [2], 111202113646@mhs.dinus.ac.id [3]

Article Info

Article history:

Received 2024-10-23
Revised 2024-11-28
Accepted 2024-11-31

Keyword:

Flower
CNN
Transfer Learning
SMOTE
Fine-Tuning

ABSTRACT

Deep learning is a technology that can be used to classify flowers. In this research, flower type classification using the CNN method with several existing CNN architectures will be discussed. The data consists of 4317 images in .jpg format, covering 5 classes that is sunflower, dandelion, daisy, tulip and rose. The distribution of data for each class is daisy with 764 pictures, dandelion with 1052 pictures, rose with 784 pictures, sunflower with 733 pictures, and tulip with 984 pictures. With total dataset of 4317 pictures is further split to training data with ratio of 60%, validation with ratio of 10%, and testing with ratio of 30% to process with the CNN method and CNN framework. Due to the imbalance data distribution, the SMOTE method is applied to balancing number of samples in each class. This research compares CNN architectures, including CNN, GoogleNet, DenseNet, and MobileNet, where each transfer learning model undergoes fine-tuning to improve performance. At the classification stage, performance will be measured based on model testing accuracy. The accuracy obtained using CNN is 74.61%, using GoogleNet is 87.45%, DenseNet is 93.92%, and MobileNet is 88.34%.

I. INTRODUCTION

Flowers are beautiful plants and can be found all around us [1]. Jia et. al [2], suggested that about 369,000 types of flower species are scattered around the world. Because it consists of many species, some flower species have very similar characteristics, so it can be quite difficult to distinguish [3]. How to distinguish one flower species from another can usually be seen from the shape, the color and the texture from the flower [4]. With the many species of flowers that exist, flowers can be utilized as food and also useful materials for humans and other living things [5]. Hussien [6], explained that with the existence of many herbal plants of the flower type, flowers are also widely used as materials for making medicines and decorations that can support health.

Daisy flowers have a similar structure to sunflowers [7]. The characteristic of the daisy flower is its relatively white and yellow color and the petals are stacked on top of each other. Daisy flowers have a height of about 30-70 cm. Dandelion or Taraxacum Officinale are the flower that widely can be organized. Mostly, dandelion has the bright yellow and unique seed in the heads of flower. Roses is type flower that widely used in flower industry [8]. Mileva et. al [9], suggested that around the world there are at least 1000 known rose genotypes that are grouped based on botanical characteristics. Because they are widely used in both the economic and entertainment industries, almost every day a lot of wilted rose waste is disposed of [10]. The waste can grow because in the cut flower industry, many use roses as gifts [11]. Tulip is flowering plant that belongs to Liliaceae family and typically have the six petals in symmetrical arranged. Sunflower or which has the Latin name Helianthus Annuus [12], is a flower that is quite often found. Just like roses, sunflowers are also widely used for cut flower industry [13]. Nguyen et. al [14], suggested that sunflowers are a type of flower that is widely distributed in the world, which is as many as 23000 species. Besides being used in flower industry, sunflowers are also used in ornamental industry and are widely cultivated [15].

Deep learning is method from field machine learning [16] that implements neural networks like system in computer [17]. Chamier et. al [18], suggested that in the process of deep learning, feature extraction is done automatically. Because of

this advantage, deep learning is widely used to perform complex computations like classification and segmentation of images [19]. Dargan et. al [20], explain the method deep learning is a very effective and efficient method to perform a task. With its advantages, it is also possible to perform learning immadiately by only using input data without preprocessing data first [21]. In the process, deep learning has better capabilities compared to ordinary machine learning methods in performing image processing tasks [22]. Wang et. al [23], explained that deep learning is common use for image analysis process to improve image quality and extract the information from images. Deep learning in its implementation has a complex structure in the form of several hidden layers used to building model [24]. CNN or Convolutional Neural Network is the figure from deep learning, where automatically learns patterns from data and performs data extraction, especially on image data [25]. Because of this convenience, many studies have been conducted using CNN in the image processing domain [26]. Bora et. al [27], suggested that this can happen because in the process, CNN consists of several interconnected neurons where the neurons have weights and biases to be able to learn data.

When doing process of developing the deep learning model, it important to having quality data is very essential to keep ensure the model learns optimally. In this research, the process of balancing each class in data is using Synthetic Minority Over-sampling Technique. SMOTE is technique that can widely used to addressing the data imbalance problem in classification datasets by generating the syntetic dataset. In current study, our objective is developing an accurate flower classification model that can help automate process of identification of the flower species, particularly for flower industry, where in the real industry, manual classification is took more time-consuming and also cost. The complexity of this task is from the fact that, usually flowers had such as similar structure, colors, and textures, making it so difficult to distinguish between species through the manual inspection. By automating in this process, our reseach is aim for minimize errors in classification prrocess then can safe time and cost instead manual flower classification. The challenge for developing classification model is dataset that used had imbalanced class data distribution, where the certain flower species are much presented while others are under. This imbalance data could result in the poor model performance, model may struggle for recognize patterns for less frequent flower species. To overcome this limitation, we employ SMOTE process, which improves the model learning data by enhancing the representation of minority classes, allowing for a more balanced and accurate data.

This study is investigates performance of the various CNN architectures, specifically GoogleNet, DenseNet, and MobileNet, that all model is recognized for their effectiveness in image classification task. Reason from choosing this method is because CNN method is the particularly well-suited method this research because can automating process of feature extraction, then can reducing data that need for process of feature engineering manually. GoogleNet's Inception modules is design for capturing multi-scale of features, which can be essential for distinguishing flowers that had similar structures with the minor differences. DenseNet method itself excels in preserving information throughout network by the maintaining dense connections between layer, which can help prevent vanishing gradient problem in classification task and ensures critical data retained from each layer. Meanwhile, MobileNet is selected in this reseach for its lightweight architecture that can prioritizes the efficiency, balancing for speed but can get still optimal accuracy, making it ideal for environments that categorize low. To further improve performance, the transfer learning and the fine-tuning process are employed. This combination enhances the model's ability to identify specific patterns unique to different flower species, leading to better classification accuracy. A key part of this research is knowing how the models that build can perform on real-world data, that mostly often includes the data noise, complex of backgrounds, and the occlusions. By analyzing how each model can handles these real-world data challenges, our study provides insights their robustness and applicability in the real world industry. This approach ensures is that the models not only well performed in the ideal conditions but can also remain effectiveness in more challenging, such as data in the real-life scenarios.

In previous research by Narvekar et. al [28], discuss about process classification of flower types with CNN in the agricultural world. Purpose from these study aiming to building an model deep learning that performed to classify flower types using the CNN method and CNN transfer learning, to see which method is more effective and efficient. The results that can obtained in this study are that using machine learning methods provides benefits through automation and can be used in the marketing process of agricultural products. Research conducting by bozkurt et. al [29], discusses classification of flowers using transfer learning method. Purpose from these study is for building model that extracts using the deep of CNN process and classification using transfer learning method such as the VGG16, the VGG19, the SqueezeNet, the DenseNet-121, the DenseNet-201, and the InceptionResNetV2. Results that obtain from these study are the classification efficient more if using InceptionResNetV2 model because the model achieve acceptable performance rates while the highest when compared to other algorithms. Based on previous studies on flower classification, it's evident that the commonly used approach is basic transfer learning models. In study [28], CNN models with transfer learning architectures like VGG16, MobileNetV2, and ResNet50 were implemented. Meanwhile, study [29] explored other architectures, including the VGG16, the VGG-19, Squeeze-Net, the DenseNet121, the DenseNet-201, and the InceptionResNetV2. The best results achieved were 92.12% accuracy with VGG16 in study [28] and 92.25% with InceptionResNetV2 in study [29]. While these results are promising, there is still room for improvement. Previous models largely relied on transfer learning and ensemble

learning without deep data processing, which can make them less effective at learning complex patterns and prone to bias. To enhance model performance, this study introduces a new contribution by deepening data processing, allowing the architectures to learn more efficiently and reducing bias. Additionally, this research intensively applies fine-tuning techniques to maximize the potential of models, aiming to significantly improve accuracy and overall performance compared to the previous studies before.

II. METHOD

This research, would developing deep learning model to assist in the classification and identification of flower species, thus simplifying and accelerating the sorting process in the flower industry. The research flow from our research given with Figure 1.

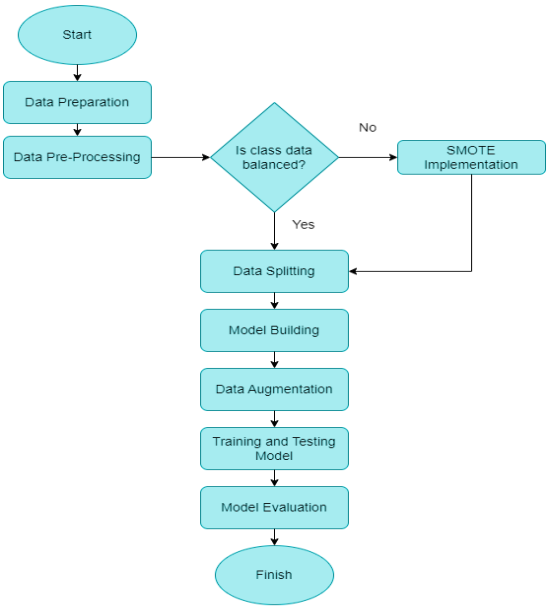

Figure 1. Research Workflow

Figure 1 shows the flow from our research process. As seen in Figure 1, first step that doing is prepare data. Once the data is ready, it undergoes preprocessing, which includes reshaping the data to a size of 150 x 150 that will make images pixel value around 0 – 255 so can standardize images to fit the input size of model and normalizing so the image pixel size is same that is 150 x 150, but the pixel value range is around 0 – 1, so that pixel values scaled by dividing by 255, make the model can better generalize the data. This makes the data more general and allows for faster processing without losing essential information. Next, a check is conducted to determine whether the data distribution is balanced. Since the data in this study is imbalanced, SMOTE is implemented to balance the data classes. After processing data, would build deep learning model. Once the model is constructed, data augmentation is performed to increase data variety, allowing model can learn more patterns from data and avoid the overfitting. After dataset and model prepared, the training and the testing processes is doing, then can followed by performance evaluation from model to analyze the model's effectiveness.

A. Dataset and Data Preprocessing

In this study, a flower classification model will be developing using CNN method. Dataset that used consists total of the 4.317 image data in .jpg format. The data is sourced from kaggle.com website, which is the data used is public data that can accessible via link https://www.kaggle.com/datasets/alxmamaev/flowers-recognition. The data was gathered through web scraping from sites like Flickr, Google Images, and Yandex Images and then compiled into a public dataset. The dataset contains 5 main classes, specifically sunflower, dandelion, daisy, tulip and rose. The distribution of data for each class includes 764 images of daisies, 1,052 pictures of dandelions, 784 pictures of roses, 733 pictures of sunflowers, and 984 pictures from tulips. Dataset would been divided with the distribution ratio form 60% used for train data, then 10% used for valid data, and then 30% used for test data. This results in 2,590 pictures for training, 1,295 pictures for testing, and 431 pictures for validation. The purpose of this data division is to ensure effective processing using the CNN method and the architectures that will be utilized. The purpose of using this dataset is to test the performance of a CNN model, specifically due to the unique challenges in flower classification. Flowers often have visual similarities across different types, both in shape and color, which can impact the accuracy of the model being developed. A visualization of the data for each class is provided in Figure 2.

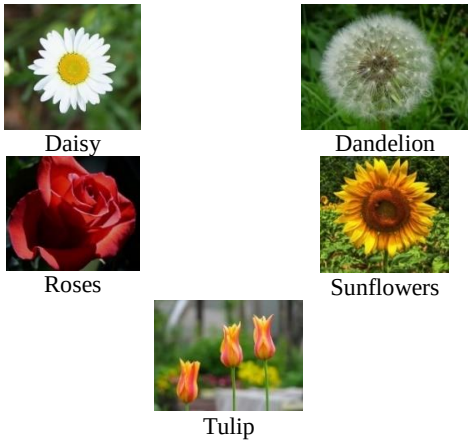

Figure 2. Dataset Visualization

Figure 2 illustrates the visualization of images from each class used in this study. To increase variety of data then can allow the model to learn patterns more effectively, this research also incorporates method of data augmentation. Data augmentation using for expand and enrich the training data by modifying existing data, such as rotating, flipping, or adjusting the lighting of images. This technique done for enhance model generalization by making it resilient to data variations that may occur in real-world scenarios. Parameters

for augmentation process that used in this research given in Table 1 below.

TABLE I.
DATA AUGMENTATION PARAMETER

Parameter	Value	Pixel Size
Rotate	40	150 x 150
Width shifting	0.25	112 - 150
Height shifting	0.2	120 - 150
Shearing	0.2	150 x 150
Zooming	0.1	135 - 150
Horizontal flipping	True	150 x 150
Fill mode	nearest	-

Table 1 shows the parameters used in the data augmentation process for add more data. As shown in Table 1, the Rotate parameter is set to 40 degrees, allowing images to rotate within a ±40 degree range. Despite this rotation, the overall image dimensions remain standardized at 150 x 150 pixels, maintaining consistency in size for model input. The Width Shifting parameter, set at 0.25, shifts the image's content horizontally by up to ±25%. This results in an effective content width ranging from approximately 112 to 150 pixels, where portions of the image may shift outside the visible frame, depending on the degree of shift. The Height Shifting parameter similarly adjusts the image vertically by up to ±20%, with the visible height ranging between 120 and 150 pixels, which maintains the standardized 150-pixel height by filling any areas outside the frame. The Shearing parameter, with a 0.2 setting, introduces a slight tilt or skew to the image. This keeps the image at 150 x 150 pixels, but alters the perspective, which can help the model generalize better. The Zooming parameter of 0.1 enables a zoom-in or zoom-out effect, effectively reducing or enlarging the visible content within a range of 135 to 150 pixels without changing the overall dimensions. Horizontal Flipping, set to true, mirrors the image horizontally without changing the 150 x 150 pixel size, helping the model learn from variations in object orientation. Lastly, Fill Mode, set to nearest, ensures that any empty areas created by shifting, rotation, or zooming are filled with the nearest pixel values, maintaining visual consistency in the augmented images.

B. Syntetic Minority Over Sampling

SMOTE is the method can address the class imbalance in datasets by generating synthetic examples for minority class [30]. This technique works by projecting minority data points into the existing feature space and creating new examples that lie between existing data points [31]. Although SMOTE is typically used for tabular data, it can also be adapted for image datasets by treating the images as feature vectors. In this approach, SMOTE generates synthetic samples based on the pixel or feature values of neighboring images, which can help balance class distributions even in image-based datasets. The mathematical formula for SMOTE is provided in section 1.

$$x_{new} = x_i + \lambda * (x_i^{neighbor} - x_i) \quad (1)$$

Point 1 presents the mathematical formula for SMOTE to generate new data. x_{new} is the new synthetic data generated, where x_i represents an existing minority data point. $x_i^{neighbor}$ is the nearest neighbor of x_i, and λ is the random parameter distributed range of the 0 to 1. In this way, SMOTE increases amount of data in minority class and helps the model learn more representative patterns from the entire dataset.

C. Convolutional Neural Network

CNN is technique where the process of find pattern and extract features from image is done automatically [25]. These advantages obtained because the CNN method composed several interconnecting neurons where this neurons have the weights and the bias so make to able for learn the pattern of data [26]. In the implementation of CNN, using layers, where usually used the layer input, convolution, pool, fully-connected, then the output [32]. Input layer is layer that perform to enter data into the model [33]. The convolution layer designed to extracting features from input images. From the each convolution layer use filter or kernels that can sliding across input image to performing the convolution process [34]. The result from this operation is the feature map that can highlighting specific patterns the characteristics from image, such as edges, the corner, or the textures. Convolution layer formula given in point 2.

$$(I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) * K(m,j) \quad (2)$$

Point 2 presents the mathematical formula for the convolutional layer. From this, we can see that *I(I,j)* represents the image input, *K(m,n)* is the kernel or feature, and *m,n* indicate the size of the kernel used. This process allows the network to capture local features such the edges or repeating patterns. Pooling is layer that useful for reduces the dimensions of feature of map while retaining essential information, like maximum value from the pooling window [35]. Formula for calculating pooling layer given in point 3.

$$P_{(i,j)} = \max(S(m,n)) \quad (3)$$

Point 3 presents the mathematical formula for the pooling layer. Based on Point 3, $P_{(i,j)}$ represents the pooling value at position i,j and $S(m,n)$ refers to the feature map values from the previous layer from which the maximum value will be selected. In max pooling, only the highest value within the pooling window (e.g., 2 x 2) is retained, which helps in preserving the dominant features while reducing the data dimensions. The fully-connected used for connects all neurons from previous layer to performing classification based on extracted features before [36]. The mathematical formula fully connect layer is provided in Point 4.

$$z_j = \sum_i w_{ij} x_i + b_j \quad (4)$$

Point 4 illustrates formula for calculating the fully connect layer. Shown in Point 4, z_j representing output of the number

of j-th neuron, w_{ij} is weight in middle of i-th neuron from previous and then the j-th neuron in fully-connected, x_i is output of i-th value neuron that include from previous layer, and b_j is the value of bias for j-th neuron. This layer helps generate the final prediction, such as classifying whether an image belongs for specific class that based on learned feature throughout the network. The results are then passed to the output layer to display the prediction [37]. CNN layers used in our research illustrated with Figure 2.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 160, 160, 64)	1792
max_pooling2d_4 (MaxPooling2D)	(None, 80, 80, 64)	0
conv2d_5 (Conv2D)	(None, 80, 80, 64)	36928
max_pooling2d_5 (MaxPooling2D)	(None, 40, 40, 64)	0
conv2d_6 (Conv2D)	(None, 40, 40, 96)	55392
max_pooling2d_6 (MaxPooling2D)	(None, 20, 20, 96)	0
conv2d_7 (Conv2D)	(None, 20, 20, 96)	83040
max_pooling2d_7 (MaxPooling2D)	(None, 10, 10, 96)	0
flatten_1 (Flatten)	(None, 9600)	0
dense_3 (Dense)	(None, 512)	4915712
dense_4 (Dense)	(None, 512)	262656
dense_5 (Dense)	(None, 5)	2565

Total params: 5358085 (20.44 MB)
Trainable params: 5358085 (20.44 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 3. CNN Layers

Figure 3 illustrates the CNN layers used in this study. As shown in Figure 3, the model employs combination from Convolutional (Conv2D) and Max Pool so then gradually extract features that in the flower images. The convolution layers help capture important patterns such as the edges and the textures, while the Max Pooling reduce the data dimensions to lower computational complexity while retaining essential information. After feature extraction, the Flatten layer doing job for transforms the data into the vector form to then gave into the Dense layers. The final Dense layer is used for classification, outputting five classes corresponding to the flower categories that need to be identified.

D. Transfer Learning CNN

Transfer learning method is where a CNN model, that already train on large data like ImageNet, using for the new task that using smaller dataset [38]. The pre-trained model can extract features from early to late stages, with the early layers often kept frozen while the later layers are adapted for the new task. This approach speeds up training and improves performance, especially with limited data [39], [40], [41], [42]. GoogleNet leverages Inception modules that combine different kernel sizes in one block to capture features at various scales while reducing the number of parameters through 1x1 convolutions for dimensional mapping, maintaining both efficiency and accuracy [43]. DenseNet connects each layer with all previous layers, facilitating gradient propagation and reducing parameter requirements by reusing features from earlier layers, making it effective at capturing complex features [44]. MobileNet is designed for computational efficiency on devices with the limited environment by using depthwise separable convolution, so the process divide convolution into 2 stages to reducing number from the parameters and the operations, allowing high performance on mobile devices [45]. The layers of the GoogleNet, DenseNet, and MobileNet architectures used detailed in Table 2.

TABLE II.
TRANSFER LEARNING LAYER

Model	Layer
GoogleNet	Model: "sequential_2" inception_v3 (Functional) (None, 3, 3, 2048) 21802784 global_average_pooling2d (GlobalAveragePooling2D) (None, 2048) 0 flatten_2 (Flatten) (None, 2048) 0 dense_6 (Dense) (None, 512) 1049088 dropout (Dropout) (None, 512) 0 dense_7 (Dense) (None, 256) 131328 dropout_1 (Dropout) (None, 256) 0 batch_normalization_94 (BatchNormalization) (None, 256) 1024 dense_8 (Dense) (None, 5) 1285 Total params: 22985509 (87.68 MB) Trainable params: 12297093 (46.91 MB) Non-trainable params: 10688416 (40.77 MB)
DenseNet	Model: "sequential_3" densenet121 (Functional) (None, 7, 7, 1024) 7037504 global_average_pooling2d_1 (GlobalAveragePooling2D) (None, 1024) 0 flatten_3 (Flatten) (None, 1024) 0 dense_9 (Dense) (None, 256) 262400 dropout_2 (Dropout) (None, 256) 0 dense_10 (Dense) (None, 512) 131584 dropout_3 (Dropout) (None, 512) 0 dense_11 (Dense) (None, 5) 2565 Total params: 7434053 (28.36 MB) Trainable params: 564677 (2.15 MB) Non-trainable params: 6869376 (26.20 MB)
MobileNet	Model: "model" input_1 (InputLayer) [(None, 224, 224, 3)] 0 Conv1 (Conv2D) (None, 112, 112, 32) 864 bn_Conv1 (BatchNormalization) (None, 112, 112, 32) 128 Conv1_relu (ReLU) (None, 112, 112, 32) 0 Mobilenet layer global_average_pooling2d GlobalAveragePooling2D) ((None, 1280) 0 flatten (Flatten) (None, 1280) 0 dense (Dense) (None, 512) 655872 dropout (Dropout) (None, 512) 0 dense_1 (Dense) (None, 512) 262656 dense_2 (Dense) (None, 5) 2565 Total params: 3179077 (12.13 MB) Trainable params: 921093 (3.51 MB) Non-trainable params: 2257984 (8.61 MB)

Table 2 shows the specific layers adjusted in this study. Fine-tuning is a method used in transfer learning to tailor a

pre-trained model to a new dataset. This involves unfreezing certain layers so that they can be retrained, allowing the model to better capture unique features relevant to the new task, especially when the new dataset varies significantly from the one used to originally train the model. In this study, fine-tuning was applied differently across models to balance performance and computational efficiency. For the GoogleNet model, layers from 0 to 248 were kept frozen, meaning they retained the original pre-trained weights and were not retrained. Only the layers from layer 249 onward were unfrozen and retrained, enabling these layers to adjust their weights based on the new data. This selective fine-tuning aimed to focus training efforts on higher-level features most relevant to the task. In the DenseNet model, the approach was to freeze all layers except the final 10 layers, which allowed only these last few layers to update their weights during training. This preserved the foundational representations learned by DenseNet while fine-tuning its upper layers for greater adaptability. In the InceptionV3 model used within the GoogleNet framework, the last 50 layers were unfrozen, meaning they were made trainable. This approach allowed the model to retain the benefits of the general features learned during pre-training while adapting the final layers to capture more specific features present in the new dataset. Each of these fine-tuning strategies was chosen to leverage the strengths of pre-trained knowledge while maximizing adaptation to the dataset in this study.

E. Performance Calculation

The confusion matrix is metricx that can used to evaluating performance model with process of compare model's predictions with actual labels, consisting of the value True-Positives, the True-Negatives, the False-Positives, then False-Negatives [46], [47]. Precision measuring the accuracy from the model's positive predictions, indicating how many of the positive predictions are correct, calculation for precision is provided in point 5. Recall is measures model sensitivity or ability to find positive samples, with calculation provided in point 6. F1-Score is harmonic result that is from mean precision value and the recall value, used when a balancing between two is needed, especially in the case of class imbalance, calculation for F1-Score is provided in point 7.

$$\text{Precision} = \frac{True - Postive}{True - Positive + False - Postive} \quad (5)$$

$$\text{Recall} = \frac{True - Postive}{True - Postive + False - Negative} \quad (6)$$

$$\text{F1} = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (7)$$

III. RESULT AND DISCUSSION

This study, the model implementation would using the Python, with Jupyter Notebook as the IDE for writing code. After preprocessing the data, including normalizing the images to a scale of 0 to 255 and reshaping them to a 150 x 150 pixel format, SMOTE will be applied in order to address imbalance data distribution problem in class distribution. Visualization of the data classes before applying SMOTE is provided in Figure 4.

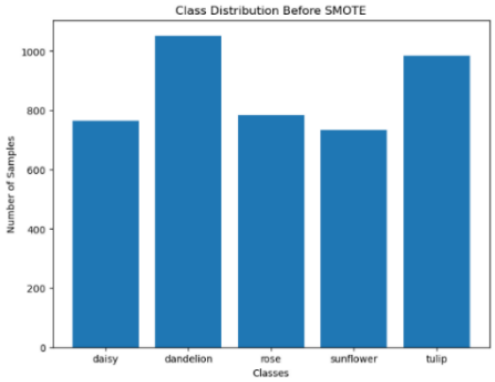

Figure 4. Data Distribution Before SMOTE

Figure 4 shows the visualization of data distribution before applying the SMOTE method. It is clear that there is no balanced in the number of data among the categories, which can lead to bias in the model ability for recognize patterns in the data. The initial data distribution shows 764 images in the daisy category, 1,052 images in the dandelion category, 784 images in the rose category, 733 images in the sunflower category, and 984 images in the tulip category. This imbalance is a factor that needs to be addressed to improve the model's performance in learning each class fairly. Therefore, this study implements SMOTE to balance the classes. The visualization of dataset after doing SMOTE process given in Figure 5.

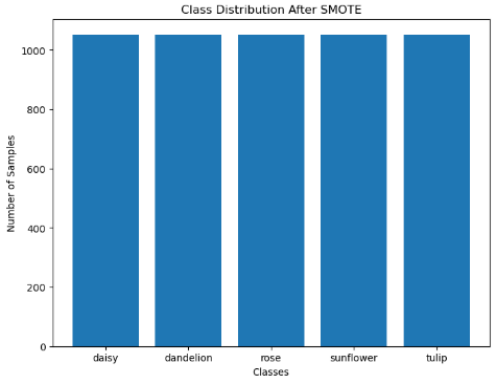

Figure 5. Data Distribution After SMOTE

Figure 5 shown the distribution data right after applying the SMOTE. As seen in Figure 5, the initial data consisted of 764 images in the daisy category, 1,052 images in the dandelion category, 784 images in the rose category, 733 images in the sunflower category, and 984 images in the tulip category. After SMOTE, the data was balanced to 1,052 images in each of the daisy, dandelion, rose, and sunflower categories, and 1,502 images in the tulip category. This process is possible because SMOTE generates synthetic data, which is then used

to augment each class, ensuring a more balanced in data distribution across from each classes. After balancing the class distribution, model development and data augmentation processes were carried out to increase data variability and reduce model bias. Following model development, the model trained using the train and valid data. For comparison, this study also evaluated models using SMOTE and fine-tuning (as indicated in the enhanced column) against models that only used transfer learning with data augmentation (as indicated in the basic column). These all model is trained using 50 epoch, with batch size of 32 so can balancing memory efficiency and computational performance. This batch size allowed the models to update weights after processing 32 samples, optimizing for both stability and convergence speed. In all model also add the learning rate scheduler was applied to dynamically adjust the learning rate during training, using the ReduceLROnPlateau function. This scheduler monitored the validation loss, reducing the learning rate by a factor of 0.2 whenever there was no improvement in validation loss for three consecutive epochs. The minimum learning rate was set to 1e-6 to ensure stability during training, preventing excessively low values that could cause the model to stall. Results from model train, specifically the accuracy values, are given in Table 3.

TABLE III.
MODEL TRAIN AND VALIDATION RESULT

Model	Train Model Accuracy	Validation Model Accuracy
CNN Enhanced	78.26%	72.61%
CNN Basic	80.06%	75.58%
GoogleNet Enhanced	94.11%	90.79%
GoogleNet Basic	89.51%	86.14%
DenseNet Enhanced	93.96%	92.86%
DenseNet Basic	95.58%	91.09%
MobileNet Enhanced	94.11%	87.53%
MobileNet Basic	95.51%	87.46%

Table 3 presents a comparison of training and validation accuracy between models optimized with SMOTE, fine-tuning, and data augmentation (enhanced models) and models using only data augmentation (basic models). For the CNN model, the basic CNN achieved a higher training accuracy of 80.06% compared to the enhanced CNN, which reached only 78.26%. Similarly, in validation accuracy, the basic CNN outperformed with 75.58% versus 72.61% for the enhanced CNN. This suggests that adding SMOTE and fine-tuning to the CNN model did not effectively improve its performance and instead reduced its ability for learn patterns in the train data and generalize to validation data. For the GoogleNet model, the results show a significant performance improvement with the enhanced model. The training accuracy of the enhanced GoogleNet reached 94.11%, higher than the basic GoogleNet's 89.51%. Validation accuracy also saw a substantial increase from 86.14% in the basic model to 90.79% in the enhanced model. This improvement indicates that SMOTE and fine-tuning techniques effectively enhanced the generalization capability of the GoogleNet model, allow for learn complex and accurate patterns from unseen data. The results occurred because the CNN model begins by extract simple features from data, such as the edges feature or textures, using pooling to reduce spatial dimensions while keeping the feature maps straightforward. As more convolutional layers are added, the model becomes capable of recognizing more complex patterns. However, due to the smaller number of parameters compared to GoogleNet, CNN struggles to generalize well with more complex datasets. Basic CNN performs better without SMOTE, as it avoids the risk of overfitting. With the addition of SMOTE and fine-tuning, CNN tends to overfit on a balanced dataset, making it difficult to generalize to validation data. In contrast, GoogleNet uses global average pooling to reduce parameters without losing spatial information, which helps prevent overfitting. The dense and dropout layers allow the model to learn more complex representations while further preventing overfitting. With SMOTE and fine-tuning, GoogleNet shows significant improvement, effectively leveraging the better class balance to learn representative patterns without overfitting.

The DenseNet model exhibited a wider range of results, with some key differences between the base and enhanced versions. The base DenseNet achieved a higher training accuracy of 95.58%, whereas the enhanced version reached 93.96%. Despite this, the enhanced DenseNet outperformed slightly in validation accuracy, scoring 92.86% compared to the base model's 91.09%. This suggests that while the base DenseNet learned the training data more effectively, the enhanced version handled overfitting better, leading to improved performance on unseen data. The MobileNet model displayed less variation. The base MobileNet attained a training accuracy of 95.51%, slightly higher than the enhanced version's 94.11%. However, the validation accuracy remained almost identical between the two, with the enhanced MobileNet scoring 87.53% and the base version at 87.46%. This minimal improvement suggests that while optimization techniques were applied to MobileNet, their impact on validation performance was marginal. The reason behind DenseNet's better generalization lies in its architecture, where dense connections allow each layer to receive inputs from all previous layers. This design promotes feature reuse and mitigates the vanishing gradient issue, enabling the model to capture complex features and improve accuracy. DenseNet consists of layers such as Dense, Dropout, and Global Average Pooling, which are key for classification tasks. The Dropout layer, set to 50%, prevents overfitting by deactivating random neurons during training, and pooling layers help reduce dimensionality. Although the base DenseNet showed high training accuracy, its slightly lower validation accuracy indicates some overfitting. On the other hand, the enhanced DenseNet, while slightly sacrificing training accuracy, demonstrated better validation accuracy,

likely due to techniques like data augmentation, Dropout, and SMOTE, which introduce more diversity in the data and improve generalization. MobileNet, known for its efficiency, utilizes depthwise separable convolutions to minimize computational costs and reduce the number of parameters. This architecture makes it ideal for lightweight models but limits its ability to capture as rich a feature set as DenseNet. The base MobileNet, despite its high training accuracy, showed a clear sign of overfitting with its lower validation accuracy. Optimization techniques in the enhanced MobileNet provided a slight boost in validation performance, but the improvements were not significant. Due to its lighter design, MobileNet naturally tends to avoid severe overfitting, but its reduced parameter count makes it less effective in capturing detailed feature representations.

Overall, the results demonstrate that the impact of SMOTE, fine-tuning, and data augmentation varies depending on the model architecture. For some models like GoogleNet, optimization techniques provided significant performance improvements, while for other models like CNN and MobileNet, the improvements were not always consistent or significant. After this analysis, it can be seen that the models are not experiencing overfitting, as there no significant differencing with train and valid accuracy. With the performance analysis complete, next step is for test the model for ensuring robustness and optimal accuracy on new data. The results for model test accuracy are given with Table 4.

TABLE IV.
MODEL TESTING RESULT COMPARISON

Model	Test Accuracy
CNN Enhanced	74.61%
CNN Basic	75.07%
GoogleNet Enhanced	87.45%
GoogleNet Basic	84.49%
DenseNet Enhanced	93.92%
DenseNet Basic	88.97%
MobileNet Enhanced	88.34%
MobileNet Basic	86.34%

Table 4 displays the test accuracy results for both enhanced and basic models. As shown in Table 4, the test accuracy results for enhanced and basic models vary depending on the architecture used. For the CNN model, the basic model demonstrates higher test accuracy of 75.07% compared to enhanced CNN model, which reaches 74.61%. This suggests that the optimizations applied to the enhanced model, did not significantly improve performance and even slightly reduced accuracy. For the GoogleNet model, the improvement in test accuracy is more pronounced, with the enhanced model achieving an accuracy of 87.45%, compared to 84.49% for the basic GoogleNet model. This shows that the optimization techniques applied to GoogleNet successfully enhanced the ability to classify new data more effectively. The enhanced GoogleNet was able to improve model generalization, resulting in a significant performance boost. For the DenseNet model, the enhanced version also shows a substantial improvement, with a test accuracy of 93.92% compared to 88.97% for the basic DenseNet model. This significant increase indicates that DenseNet responds well to the applied optimization techniques. In the MobileNet model, the enhanced version also shows improvement, with a test accuracy of 88.34% compared to 86.34% for the basic model.

The results can be explained by the relatively shallow convolutional layers of CNN. With fewer layers and parameters, CNN model struggles to capturing the the pattern that complex from data, relying more on simpler features like edges and textures. As a result, adding synthetic data through techniques like SMOTE or applying extensive fine-tuning may not offer much benefit. In fact, it could introduce noise or unnecessary complexity, increasing the risk from overfitting and reducing ability to generalize new or unseen data. The enhanced version of CNN likely experienced overfitting due to aggressive fine-tuning and the introduction of synthetic data. Given the shallow nature of the model, CNN might have over-learned from the balanced data, making it less capable of adapting to new patterns. In contrast, GoogleNet, with its much deeper architecture and multiple Inception modules, is better equipped to capture a large amount range of patterns from the input data. GoogleNet had ability to perform convolution at different scales—using parallel filters of varying sizes—enables it to detect complex features that a simpler CNN might miss. This depth and complexity also make GoogleNet more adaptable to data augmentation and fine-tuning. The balanced data provided by SMOTE is utilized more effectively by GoogleNet, allowing it to explore a wider variety of input-output relationships and improve its generalization to new data. In simpler architectures like CNN, the addition of synthetic data may only contribute to noise, but GoogleNet's larger number of parameters allows fine-tuning to be more effective. Unlike CNN, GoogleNet can refine a broader set of features, focusing on the most relevant patterns while discarding irrelevant ones, leading to improved performance.

The DenseNet model achieves superior performance due to its unique structure, which connecting one layer to the other layer through dense blocks. This architecture encourages feature reuse across the network, enabling DenseNet to capture more detailed and hierarchical features while mitigating the risk of vanishing gradients. As a result, when optimization techniques are applied, DenseNet shows a significant performance improvement. This structure allows DenseNet to learn more complex patterns, contributing to its high training accuracy. Additionally, the use of SMOTE helps balance the dataset, which is essential for DenseNet, enhances the model ability for generalize by reducing class imbalance. This leads to an improvement in test accuracy, reaching 93.92% compared to the baseline accuracy of 88.97%. Fine-tuning pre-trained layers further refines the model's generalization, particularly in DenseNet’s deep architecture, reducing the gap between training and test performance and avoiding overfitting. Then, the MobileNet model used depthwise separable convolutions, that which can occur reducing of both the number parameters and computational

costs, so then making it suitable for real-time app or in mobile devices. However, this efficiency limits its capacity to learn complex patterns when compared to DenseNet. While techniques like SMOTE and optimization reduce overfitting and improve accuracy, their impact on MobileNet is less significant. MobileNet's simplified architecture restricts the ability for take complex from relationships in data, resulting in only a slight improvement in test accuracy, from 86.34% to 88.34%. Although MobileNet excels in resource-constrained environments, it is less effective than DenseNet for more complex tasks such as image classification with imbalanced datasets. Overall, the results in Table 4 show that the impact of optimization depends on model. For simpler models like CNN, optimization does not make a significant difference, while for more complex models like GoogleNet and DenseNet, techniques like SMOTE, fine-tuning proces, then the augmentation significantly improve test accuracy and ability to generalize new data. Following the accuracy analysis, further examination of value from precision, the recall, and then F1-score will be conducted. The value obtained given with Table 5.

TABLE V.
DETAIL MODEL PERFORMANCE RESULT

Model	Precision	Recall	F1-Score
CNN Enhanced	74%	75%	74%
CNN Basic	75%	75%	75%
GoogleNet Enhanced	87%	88%	87%
GoogleNet Basic	86%	84%	84%
DenseNet Enhanced	94%	94%	94%
DenseNet Basic	89%	89%	89%
MobileNet Enhanced	88%	88%	88%
MobileNet Basic	86%	86%	86%

Table 5 highlights the differences in precision, recall, and F1-score between enhanced (with SMOTE and fine-tuning) and basic versions of each model, revealing how these optimization techniques affect performance. For CNN, minimal gains are seen with enhancements, as the basic model scores 75% across all metrics, while the enhanced version shows a minor drop to 74% in precision and F1-score, maintaining 75% recall. This suggests CNN's limited complexity restricts it from fully benefiting from SMOTE and fine-tuning. GoogleNet, however, shows notable improvement, with the enhanced model achieving 87% precision, 88% recall, and 87% F1-score, compared to the basic model's 86%, 84%, and 84%. This indicates GoogleNet's capacity to better leverage the augmented data and fine-tuning adjustments in its final layers. DenseNet achieves the highest gains, with the enhanced version scoring consistently at 94% across metrics, compared to 89% in the basic model. DenseNet's architecture, with densely connected layers, makes it particularly effective in capturing complex data features through SMOTE and fine-tuning, resulting in optimal performance. For MobileNet, enhancements lead to moderate gains, with scores improving from 86% in the basic model to 88% in the enhanced. MobileNet's simpler architecture benefits from SMOTE and fine-tuning, though to a lesser extent than more complex models, indicating that while it achieves better balance, it has a limited capacity for deeper feature adjustments. For detailed value from precision, recall and F1-score in each class given in table 6.

TABLE VI.
DETAILED PERFORMANCE EACH CLASS

Model	Class	Precision	Recall	F1-Score
CNN Enhanced	Daisy	76%	81%	78%
	Dandelion	85%	75%	80%
	Rose	63%	63%	63%
	Sunflower	78%	82%	80%
	Tulip	70%	73%	72%
CNN Basic	Daisy	75%	82%	78%
	Dandelion	81%	77%	79%
	Rose	63%	74%	68%
	Sunflower	80%	80%	80%
	Tulip	76%	64%	70%
GoogleNet Enhanced	Daisy	94%	85%	89%
	Dandelion	92%	91%	91%
	Rose	81%	87%	84%
	Sunflower	87%	88%	87%
	Tulip	84%	87%	85%
GoogleNet Basic	Daisy	94%	83%	88%
	Dandelion	93%	88%	90%
	Rose	86%	74%	80%
	Sunflower	84%	82%	83%
	Tulip	71%	92%	80%
DenseNet Enhanced	Daisy	97%	93%	95%
	Dandelion	92%	97%	94%
	Rose	90%	94%	92%
	Sunflower	97%	96%	97%
	Tulip	93%	88%	90%
DenseNet Basic	Daisy	90%	89%	90%
	Dandelion	94%	93%	93%
	Rose	86%	85%	85%
	Sunflower	86%	90%	88%
	Tulip	87%	87%	87%
MobileNet Enhanced	Daisy	90%	85%	87%
	Dandelion	93%	92%	92%
	Rose	83%	91%	87%
	Sunflower	86%	94%	90%
	Tulip	91%	80%	85%
MobileNet Basic	Daisy	78%	90%	83%
	Dandelion	94%	89%	92%
	Rose	86%	84%	85%
	Sunflower	89%	81%	85%
	Tulip	85%	85%	85%

Table 6 show detailed precision, recall and f1-score from the model. Had been seen in table 6, The DenseNet model demonstrates significant improvement in classification performance with the enhanced configuration, achieving a

consistent 94% in precision, recall, and F1-score higher than the basic model's 89% across these metrics. The DenseNet architecture, with its densely connected layers, enables robust feature propagation and better capture of subtle data patterns across various classes. The use of SMOTE to balance classes benefits the model by improving both precision and recall, while fine-tuning further enhances DenseNet's ability to adapt to specific dataset characteristics. For instance, in DenseNet Enhanced, class-specific performance metrics are particularly high, with classes like Daisy reaching 95% F1-score and Sunflower achieving 97% precision and recall, indicating balanced sensitivity and specificity across diverse classes. This strong result suggests DenseNet's capacity for accurate, well-rounded classification due to its depth and architectural complexity. In contrast, the MobileNet model designed for efficiency achieves moderate gains from enhancements, with precision, recall, and F1-score all reaching 88%, compared to 86% in the basic version. MobileNet's architecture employs depthwise separable convolutions, which reduce computational demands but limit its ability to capture highly detailed features. Nevertheless, SMOTE and fine-tuning help MobileNet to address class imbalance, as seen in its improved F1-scores for classes like Rose 87% and Sunflower 90% in the enhanced configuration. However, the improvement remains limited compared to DenseNet due to MobileNet's streamlined design, which restricts its capacity for capturing complex feature patterns introduced by the optimizations. In the case of GoogleNet, the enhanced model shows significant gains, with 87% precision, 88% recall, and 87% F1-score, compared to the basic version's lower recall and F1-scores. GoogleNet's responsiveness to SMOTE and fine-tuning is apparent, particularly in class-specific performance: the Daisy class reaches 89% F1-score and Dandelion achieves 91% across all metrics, showing that GoogleNet effectively learns from balanced and augmented data, especially in the final layers where fine-tuning is applied. CNN, on the other hand, shows minimal impact from SMOTE and fine-tuning, with the enhanced model scoring close to or slightly below the basic version. For example, the enhanced CNN model sees a minor dip in precision for Tulip from 76% to 70% and Rose (from 63% to 63%), likely due to CNN's simpler architecture, which lacks the depth to capture nuanced patterns effectively. Overall, Table 6 illustrates that SMOTE and fine-tuning optimizations consistently boost performance across more complex architectures like DenseNet and GoogleNet, with DenseNet achieving the highest improvements due to its intricate layer connections. Meanwhile, MobileNet and CNN exhibit smaller enhancements, reflecting their limited capacity for intricate pattern recognition. These results emphasize the importance of model architecture complexity in leveraging data augmentation and fine-tuning techniques for balanced, accurate classification across classes. Following further analysis, the next step will be comparing this research with previous studies, as presented in Table 7.

TABLE VII.
COMPARISON WITH PREVIOUS STUDY

Journal	Method	Accuracy
[29]	InceptionResNetV2	92.25%
[48]	CNN with Classic Training, Boostrap Agregation, with Random splits, Channel I, Channel II, Channel III and Keep N Max	93.12%, 50%, 64.50%, 89.87%, 88.75%, 90%, 90.37%
[49]	CNN with VGG16 architecture	80%
[50]	Neural Network and Logistic Regression	90.1% and 84.2%
[51]	Improved CNN	91%
Our	Proposed Method (DenseNet Enhanced)	93.91%

Table 7 showing comparison of accuracy results between the proposed method in this study, DenseNet Enhanced, and methods used in previous research. As illustrated in Table 4, the proposed DenseNet Enhanced method achieves an accuracy of 93.91%, which is highly competitive and even surpasses several existing methods in the literature. Research in [48] used a CNN approach with techniques such as Training with Classic method, the Bootstrap Aggregation, then the Splitting Random, considering multiple of the cannel of data (Channel I, Channel II, Channel III) and methods like Keep N Max, yielding accuracy results ranging from 50% to 93.12%. However, the best method from that study is still slightly lower than the 93.91% accuracy achieved by the DenseNet Enhanced method. This indicates that the optimization approach applied in this study, including SMOTE, fine-tuning, and data augmentation, results in better performance compared to conventional methods used in [48].

Meanwhile, the study in [49], which employed the VGG16 architecture, achieved only 80% accuracy, significantly lower than the results obtained with the proposed method in this study. This suggests that using more modern and complex architectures like DenseNet, combined with appropriate optimization techniques, can lead to substantial improvements in classification accuracy, especially with the data used. Overall, the results from Table 4 demonstrate proposed method in this study is competitive and even surpasses existing methods in the literature. The application of optimization techniques such as SMOTE and fine-tuning plays a crucial role in achieving better results, making DenseNet Enhanced a more effective solution compared to previous methods.

IV. CONCLUSION

This study, flower classification model develop using deep learning techniques, specifically evaluating the performance of CNN architectures on real-world data. So data not only images of flowers but also various background elements, making the task more challenging since the models had to focus on identified flowers with wide range of background distractions. Despite this, the CNN model achieved an accuracy of 74.61%, GoogleNet reached 87.45%, MobileNet

obtained 88.34%, and DenseNet, optimized with SMOTE and Fine-Tuning, achieved the highest accuracy at 93.92%. Among these, DenseNet, enhanced with SMOTE and Fine-Tuning, provided the highest accuracy of 93.92%, surpassing previous studies. The results occurred because the CNN model begins by extract simple features from data, such as the edges feature or textures, using pooling to reduce spatial dimensions while keeping the feature maps straightforward. As more convolutional layers are added, the model becomes capable of recognizing more complex patterns. GoogleNet uses global average pooling to reduce parameters without losing spatial information, which helps prevent overfitting. The dense and dropout layers allow the model to learn more complex representations while further preventing overfitting. The reason behind DenseNet's better generalization is in its architecture, where dense connections allow each layer to receive inputs from all previous layers. This design promotes feature reuse and mitigates the vanishing gradient issue, enabling the model to capture complex features and improve accuracy. MobileNet, despite its high training accuracy, showed a clear sign of overfitting with its lower validation accuracy. Due to its lighter design, MobileNet naturally tends to avoid severe overfitting, but its reduced parameter count makes less effective in capture detail feature representations. This study highlights the importance of addressing real-world data challenges. Dataset used included not only the flowers but also complex backgrounds, simulating real-life conditions where the flower may not always be the central object in the image. Despite these complexities, the DenseNet model, with its enhanced architecture and optimization techniques, performed exceptionally well in distinguishing flower species amidst noisy and varied backgrounds. These results demonstrate that DenseNet, enhanced with SMOTE and Fine-Tuning techniques, outperformed the other models, in handle complexities of real-world data where the focus isn't solely on the flower itself. The performance boost provided by SMOTE and Fine-Tuning was instrumental in improving classification accuracy, especially in addressing the class imbalance and capturing the finer details needed to differentiate flower species amidst background noise. The success of DenseNet demonstrates that applying SMOTE and Fine-Tuning plays the crucial role with improving ability from model to generalizing and handling the inconsistencies found in real data. Future research should focus on further optimizing this process, potentially by incorporating hyperparameter tuning or ensemble methods so hopefully can get better performance, so can achieve even higher levels of accuracy in challenging real-life classification tasks.

REFERENCES

[1] S. Islam, Md. F. A. Foysal, and N. Jahan, A Computer Vision Approach to Classify Local Flower using Convolutional Neural Network, *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Jun. 2020, doi: 10.1109/ICICCS48265.2020.9121143.

[2] L. Jia, H. Zhai, X. Yuan, Y. Jiang, and J. Ding, A Parallel Convolution and Decision Fusion-Based Flower Classification Method, *Mathematics*, vol. 10, no. 15, Aug. 2022, doi: 10.3390/math10152767.

[3] I. Patel and S. Patel, Flower identification and classification using computer vision and machine learning techniques, *Int J Eng Adv Technol*, vol. 8, no. 6, pp. 277–285, Aug. 2019, doi: 10.35940/ijeat.E7555.088619.

[4] F. Khalid, A. H. Abdullah, and L. N. Abdullah, Smartflora Mobile Flower Recognition Application Using Machine Learning Tools, in *2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA)*, IEEE, 2022, pp. 204–209.

[5] M. Toğaçar, B. Ergen, and Z. Cömert, Classification of flower species by using features extracted from the intersection of feature selection methods in convolutional neural network models, *Measurement (Lond)*, vol. 158, Jul. 2020, doi: 10.1016/j.measurement.2020.107703.

[6] A. Yehya Hussien, Flower Species Recognition Using Machine Learning Classifiers, *Academic Journal of Nawroz University*, vol. 11, no. 4, pp. 469–475, Dec. 2022, doi: 10.25007/ajnu.v11n4a1636.

[7] M. M. Chandra and Yoannita, Klasifikasi Jenis Bunga Menggunakan Metode Svm Berdasarkan Citra Dengan Fitur Hsv, *Jurnal Indonesia Sosial Teknologi*, vol. 4, no. 2, pp. 255–264, Feb. 2023, doi: https://doi.org/10.59141/jist.v4i02.585.

[8] L. Qiu, M. Zhang, B. Bhandari, and B. Wang, Effects of infrared freeze drying on volatile profile, FTIR molecular structure profile and nutritional properties of edible rose flower (Rosa rugosa flower), *J Sci Food Agric*, vol. 100, no. 13, pp. 4791–4800, Oct. 2020, doi: 10.1002/jsfa.10538.

[9] M. Mileva *et al.*, Rose flowers—a delicate perfume or a natural healer?, Jan. 01, 2021, *MDPI AG*. doi: 10.3390/biom11010127.

[10] A. Khan, R. Arumugam Senthil, J. Pan, Y. Sun, and X. Liu, Hierarchically Porous Biomass Carbon Derived from Natural Withered Rose Flowers as High-Performance Material for Advanced Supercapacitors, *Batter Supercaps*, vol. 3, no. 8, pp. 731–737, Aug. 2020, doi: 10.1002/batt.202000046.

[11] O. H. Kwon, H. G. Choi, S. J. Kim, Y. R. Lee, H. H. Jung, and K. Y. Park, Changes in Yield, Quality, and Morphology of Three Grafted Cut Roses Grown in a Greenhouse Year-Round, *Horticulturae*, vol. 8, no. 7, Jul. 2022, doi: 10.3390/horticulturae8070655.

[12] R. Kaur, A. Jain, and S. Kumar, Optimization classification of sunflower recognition through machine learning, in *Materials Today: Proceedings*, Elsevier Ltd, 2021, pp. 207–211. doi: 10.1016/j.matpr.2021.05.182.

[13] E. Mladenovic *et al.*, Effect of plant density on stem and flower quality of single-stem ornamental sunflower genotypes, *Horticultural Science*, vol. 47, no. 1, pp. 45–52, 2020, doi: 10.17221/10/2019-HORTSCI.

[14] D. T. C. Nguyen *et al.*, The sunflower plant family for bioenergy, environmental remediation, nanotechnology, medicine, food and agriculture: a review, Oct. 01, 2021, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s10311-021-01266-z.

[15] W. Handayati and D. Sihombing, Study of NPK fertilizer effect on sunflower growth and yield, in *AIP Conference Proceedings*, American Institute of Physics Inc., Jul. 2019. doi: 10.1063/1.5115635.

[16] S. Dong, P. Wang, and K. Abbas, A survey on deep learning and its applications, May 01, 2021, *Elsevier Ireland Ltd*. doi: 10.1016/j.cosrev.2021.100379.

[17] C. Janiesch, P. Zschech, and K. Heinrich, Machine learning and deep learning, *Electronic Markets*, vol. 31, pp. 687–694, Apr. 2021, doi: https://doi.org/10.1007/s12525-021-00475-2.

[18] L. von Chamier *et al.*, Democratising deep learning for microscopy with ZeroCostDL4Mic, *Nat Commun*, vol. 12, no. 1, Dec. 2021, doi: 10.1038/s41467-021-22518-0.

[19] N. O'Mahony *et al.*, Deep Learning vs. Traditional Computer Vision, in *Advances in Intelligent Systems and Computing*, Springer Verlag, 2020, pp. 128–144. doi: 10.1007/978-3-030-17795-9_10.

[20] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, A Survey of Deep Learning and Its Applications: A New Paradigm to

Machine Learning, *Archives of Computational Methods in Engineering*, vol. 27, no. 4, pp. 1071–1092, Sep. 2020, doi: 10.1007/s11831-019-09344-w.

[21] A. Esteva *et al.*, Deep learning-enabled medical computer vision, Dec. 01, 2021, *Nature Research*. doi: 10.1038/s41746-020-00376-2.

[22] P. Wang, E. Fan, and P. Wang, Comparative analysis of image classification algorithms based on traditional machine learning and deep learning, *Pattern Recognit Lett*, vol. 141, pp. 61–67, Jan. 2021, doi: 10.1016/j.patrec.2020.07.042.

[23] G. Wang, J. C. Ye, and B. De Man, Deep learning for tomographic image reconstruction, Dec. 01, 2020, *Nature Research*. doi: 10.1038/s42256-020-00273-z.

[24] H. Liu and B. Lang, Machine learning and deep learning methods for intrusion detection systems: A survey, Oct. 01, 2019, *MDPI AG*. doi: 10.3390/app9204396.

[25] D. R. Sarvamangala and R. V. Kulkarni, Convolutional neural networks in medical image understanding: a survey, Mar. 01, 2022, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s12065-020-00540-3.

[26] Y. Li, J. Nie, and X. Chao, Do we really need deep CNN for plant diseases identification?, *Comput Electron Agric*, vol. 178, Nov. 2020, doi: 10.1016/j.compag.2020.105803.

[27] M. B. Bora, D. Daimary, K. Amitab, and D. Kandar, Handwritten Character Recognition from Images using CNN-ECOC, in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 2403–2409. doi: 10.1016/j.procs.2020.03.293.

[28] C. Narvekar and M. Rao, Flower classification using CNN and transfer learning in CNN-Agriculture Perspective, in *Proceedings of the 3rd International Conference on Intelligent Sustainable Systems, ICISS 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 660–664. doi: 10.1109/ICISS49785.2020.9316030.

[29] F. BOZKURT, A Study on CNN Based Transfer Learning for Recognition of Flower Species, *European Journal of Science and Technology*, Jan. 2022, doi: 10.31590/ejosat.1039632.

[30] S. Wang, Y. Dai, J. Shen, and J. Xuan, Research on expansion and classification of imbalanced data based on SMOTE algorithm, *Sci Rep*, vol. 11, no. 1, Dec. 2021, doi: 10.1038/s41598-021-03430-5.

[31] A. Ishaq *et al.*, Improving the Prediction of Heart Failure Patients' Survival Using SMOTE and Effective Data Mining Techniques, *IEEE Access*, vol. 9, pp. 39707–39716, 2021, doi: 10.1109/ACCESS.2021.3064084.

[32] A. Çinar and M. Yildirim, Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture, *Med Hypotheses*, vol. 139, Jun. 2020, doi: 10.1016/j.mehy.2020.109684.

[33] G. Shrestha, Deepsikha, M. Das, and N. Dey, Plant Disease Detection Using CNN, in *Proceedings of 2020 IEEE Applied Signal Processing Conference, ASPCON 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 109–113. doi: 10.1109/ASPCON49795.2020.9276722.

[34] M. A. Hossain and Md. S. A. Sajib, Classification of Image using Convolutional Neural Network (CNN), *Global Journal of Computer Science and Technology: D Neural & Artificial Intelligence*, vol. 19, no. 2, 2019, Accessed: Oct. 22, 2024. [Online]. Available: https://globaljournals.org/GJCST_Volume19/2-Classification-of-Image-using-Convolutional.pdf

[35] H. Gholamalinezhad and H. Khosravi, Pooling Methods in Deep Neural Networks, a Review, *arXiv preprint*, 2020, doi: https://doi.org/10.48550/arXiv.2009.07485.

[36] Z. A. Sejuti and M. S. Islam, An Efficient Method to Classify Brain Tumor using CNN and SVM, in *International Conference on Robotics, Electrical and Signal Processing Techniques*, 2021, pp. 644–648. doi: 10.1109/ICREST51555.2021.9331060.

[37] L. Chen, X. Kuang, A. Xu, S. Suo, and Y. Yang, A Novel Network Intrusion Detection System Based on CNN, in *Proceedings - 2020 8th International Conference on Advanced Cloud and Big Data, CBD 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 243–247. doi: 10.1109/CBD51900.2020.00051.

[38] S. M. Hassan, A. K. Maji, M. Jasiński, Z. Leonowicz, and E. Jasińska, Identification of plant-leaf diseases using cnn and transfer-learning approach, *Electronics (Switzerland)*, vol. 10, no. 12, Jun. 2021, doi: 10.3390/electronics10121388.

[39] K. Thenmozhi and U. Srinivasulu Reddy, Crop pest classification based on deep convolutional neural network and transfer learning, *Comput Electron Agric*, vol. 164, Sep. 2019, doi: 10.1016/j.compag.2019.104906.

[40] T. Rahman *et al.*, Transfer learning with deep Convolutional Neural Network (CNN) for pneumonia detection using chest X-ray, *Applied Sciences (Switzerland)*, vol. 10, no. 9, May 2020, doi: 10.3390/app10093233.

[41] N. Aneja and S. Aneja, Transfer Learning using CNN for Handwritten Devanagari Character Recognition, *2019 1st International Conference on Advances in Information Technology (ICAIT)*, pp. 293–296, Feb. 2019, doi: 10.1109/ICAIT47043.2019.8987286.

[42] K. M. Hosny, M. A. Kassem, and M. M. Foaud, Classification of skin lesions using transfer learning and augmentation with Alex-net, *PLoS One*, vol. 14, no. 5, May 2019, doi: 10.1371/journal.pone.0217293.

[43] L. Yang *et al.*, GoogLeNet based on residual network and attention mechanism identification of rice leaf diseases, *Comput Electron Agric*, vol. 204, p. 107543, 2023.

[44] N. Hasan, Y. Bao, A. Shawon, and Y. Huang, DenseNet Convolutional Neural Networks Application for Predicting COVID-19 Using CT Image, *SN Comput Sci*, vol. 2, no. 5, Sep. 2021, doi: 10.1007/s42979-021-00782-7.

[45] C. Bi, J. Wang, Y. Duan, B. Fu, J.-R. Kang, and Y. Shi, MobileNet based apple leaf diseases identification, *Mobile Networks and Applications*, pp. 1–9, 2022.

[46] Y. Li, J. Nie, and X. Chao, Do we really need deep CNN for plant diseases identification?, *Comput Electron Agric*, vol. 178, Nov. 2020, doi: 10.1016/j.compag.2020.105803.

[47] M. Te Wu, Confusion matrix and minimum cross-entropy metrics based motion recognition system in the classroom, *Sci Rep*, vol. 12, no. 1, Dec. 2022, doi: 10.1038/s41598-022-07137-z.

[48] J. Musaev, A. Anorboev, N. T. Nguyen, and D. Hwang, KeepNMax: Keep N Maximum of Epoch-Channel Ensemble Method for Deep Learning Models, *IEEE Access*, vol. 11, pp. 9339–9350, 2023, doi: 10.1109/ACCESS.2023.3239658.

[49] G. Wega Intyanto, Klasifikasi Citra Bunga dengan Menggunakan Deep Learning: CNN (Convolution Neural Network), *Jurnal Arus Elektro Indonesia*, vol. 7, no. 3, pp. 80–83, 2021, doi: https://doi.org/10.19184/jaei.v7i3.28141.

[50] R. Kursun, I. Cinar, Y. S. Taspinar, and M. Koklu, Flower recognition system with optimized features for deep features, in *2022 11th Mediterranean conference on embedded computing (MECO)*, IEEE, 2022, pp. 1–4.

[51] G. Yifei, Q. Chuxian, X. Jiexiang, M. Yixuan, and T. T. Toe, Flower image classification based on improved convolutional neural network, in *2022 12th International Conference on Information Technology in Medicine and Education (ITME)*, IEEE, 2022, pp. 81–87.