International Journal of Sustainable Development and Planning
Vol., No., Month, Year, pp. **-**

Journal homepage: http://iieta.org/journals/ijsdp

Classification of Sunflower Blooming Stages Using CNN and Transfer Learning

Putra bin Sumari*, Chan Khai Shen[2], Hneah Guey Ling[3] Loh Min Yi*, Mohammed Jubarah[2]

School of Computer Sciences, Universiti Sains Malaysia
Penang, Malaysia

Corresponding Author Email: putras@usm.my, chankhaishen@student.usm.my, guey.ling.hneah@student.usm.my, minyiloh@student.usm.my, mdjubarh@student.usm.my

https://doi.org/10.18280/ijsdp.xxxxx
x

Received:
Revised:
Accepted:
Available online:

Keywords:
Deep Learning, CNN, Transfer Learning, AlexNet, VGG16, InceptionV3, pre-trained models

ABSTRACT

The sunflower (Helianthus annuus) is a crop of significant agricultural and economic importance, and monitoring its blooming stages is critical for optimizing pollination, predicting yield, and managing crop health. However, visually distinguishing between its blooming stages—Bud, Partially Bloom, Fully Bloom, and Wilt—can be challenging due to subtle morphological similarities and variations in imaging conditions. Convolutional Neural Networks (CNNs) and transfer learning provide a robust solution for accurate, automated image-based classification. In this study, we developed a system to classify the four key blooming stages of sunflowers using a dataset of 1,076 images, augmented to enhance diversity and increase size. The images were resized to 224×224 pixels for model input. We constructed a custom-built CNN model and adapted three pre-trained models: VGG16, AlexNet, and InceptionV3. Our experiments evaluated the performance of each model, comparing the result between the proposed CNN model and the other three pre-trained models. This study demonstrates the high potential of transfer learning for practical applications in agricultural monitoring and environmental sustainability.

1. INTRODUCTION

The sunflower (Helianthus annuus) is a globally cultivated plant, valued for its oil, seeds, and ornamental appeal. The life cycle of a sunflower includes several distinct blooming stages: Bud, Partially Bloom, Fully Bloom, and Wilt. Accurate identification of these stages is vital for agricultural practices such as determining the optimal time for pollination, estimating potential seed yield, and monitoring plant health for early disease detection.

However, classifying these stages based on visual inspection alone, especially from 2D images, presents a significant challenge. The transition between stages can be subtle, and external factors like lighting, angle, and background clutter often affect visual characteristics. For instance, distinguishing a late bud from an early partial bloom requires attention to fine-grained details of petal unfurling. Relying on images means sensory cues like texture and scent are absent, complicating automated classification.

To address this challenge, this paper proposes using deep learning, specifically Deep Convolutional Neural Networks (DCNN) and transfer learning algorithms, for the automated classification of sunflower blooming stages. Our dataset consists of 1,076 images, which were stratified into training (70%), validation (15%), and testing (15%) subsets to ensure a balanced representation of each class. The images were resized to 224×224 pixels, and augmentation techniques were applied to enhance the dataset's diversity. While machine vision has been applied to many other crops, such as jackfruit, there is a notable gap in research focused specifically on sunflowers. The study presented in this paper aims to construct a classification system for sunflower blooming stages using a custom-built CNN and compare its performance with established transfer learning models, namely VGG16, AlexNet, and InceptionV3. By evaluating these models on a dedicated, self-collected dataset, we aim to identify an effective and reliable method for automated sunflower monitoring, potentially contributing to enhanced agricultural practices and sustainability.

2. RELATED WORK

Recent advancements in machine learning have enabled the classification of flower blooming stages based on RGB images. Traditional machine learning techniques such as Support Vector Classifier (SVC), K-Nearest Neighbour (KNN), and Logistic Regression (LR) have shown reasonable performance. For example, an Android-based application for tomato flower classification and compared several traditional algorithms with Convolutional Neural Networks (CNNs) [2]. Among the traditional methods, SVC achieved the highest accuracy (85%), while CNN reached 70%. The tomato flowers were classified into three categories: ineffective (for pollination), low maturity, and high maturity.

CNNs generally outperform traditional machine learning models due to their ability to capture spatial features through convolutional layers. The performance improves further when transfer learning is applied, as pre-trained models can leverage features learned from large datasets like ImageNet. A five-stage blooming classification model [10] is developed for fresh-cut roses using RGB-D images. They evaluated both traditional and transfer learning-based models, including VGG16, ResNet18, MobileNetV2, and InceptionV3. InceptionV3 achieved the highest accuracy (98%), significantly outperforming traditional classifiers by over 20%. Similarly in [9], the author compared CNN, SVM, and Artificial Neural Network (ANN) for marigold blooming stage classification into bud, partially blooming, and fully blooming. CNN again demonstrated superior accuracy.

In recent years, object detection algorithms such as YOLO (You Only Look Once), Faster R-CNN, Mask R-CNN, and Single Shot Detector (SSD) have been widely adopted for flower detection tasks. These models integrate CNN backbones and utilize transfer learning for improved accuracy. A YOLOv5l-based model [4] was developed that was capable of detecting ten kiwifruit flower classes, including seven blooming stages, occluded flowers, and two pollination stages, achieving a mean average precision of 93%. In [11], the author demonstrated that YOLOv8 outperformed YOLOv4 in classifying cantaloupe buds and flowers in greenhouse environments, based on datasets from different orchards. Patel [7] applied Faster R-CNN with a ResNet50 V1 backbone for field-based marigold flower detection and found it superior to SSD with MobileNet V1, achieving a mean average precision of 89%.

Further research has investigated enhancements to object detection performance using CNN-based YOLO. An enhanced YOLOv5 [5] for tea flower detection by incorporating a squeeze-and-excitation module and achieved a mean average precision of 87%. The model successfully identified bud, blooming, and withered flowers across various tea accessions. YOLOv4-Tiny was improved [7] using circular bounding boxes to classify chrysanthemums into budding, early flowering, and full bloom, enhancing both accuracy and computational efficiency. In [12], the author compared four attention mechanisms in YOLOv5s for cucumber flower classification and found that the squeeze-and-excitation network yielded the best results. Their model classified cucumber flowers into bud, bloom, and faded stages.

Several studies have specifically focused on apple flower blooming stage detection. In [13], the author compared YOLOv4 and YOLOv7 for recognizing tight cluster, pink, bloom, and petal fall stages under different annotation quality levels, concluding that YOLOv7 offered better accuracy. In [6], the author used Mask R-CNN with a ResNet101 backbone to detect apple buds, occluded flowers, and exposed flowers accurately. In [3], the author tested four YOLOv5 variants for early apple flower detection and centroid localization and their results indicated that YOLOv5s provided the best overall performance.

In conclusion, while traditional machine learning methods have shown effectiveness in classifying flower blooming stages, CNN-based models—especially those using transfer learning—consistently outperform them. Furthermore, the majority of research has focused on crops like tomato, rose, tea, apple, and marigold. Notably, there is limited literature on sunflower blooming stage classification, indicating a clear research gap. This motivates our current study, which aims to address this gap by using CNN and transfer learning for sunflower blooming stage classification.

3. PROPOSED WORK

3.1 Data Preprocessing

The dataset used in this study consists of images categorized into four distinct classes, each representing a specific visual category relevant to the research objective. These images were originally stored in a single directory, with subdirectories corresponding to each class label. In order to enable effective supervised learning and model validation, the dataset was partitioned into three subsets: training, validation, and testing.

The splitting was conducted in a stratified manner, where 70% of the data from each class was allocated to the training set, 15% to the validation set, and the remaining 15% to the test set. This approach ensures that all subsets reflect the original distribution of classes, thereby minimizing the risk of class imbalance, which can severely degrade model performance. The splitting procedure was implemented using a random shuffling mechanism with a fixed random seed to ensure reproducibility. This guarantees that future iterations of the experiment would yield consistent results.

Once the split was completed, dedicated directories were created for each subset and for each class within those subsets. Images were then copied into their corresponding directories. This structure is essential for compatibility with the ImageDataGenerator utility in TensorFlow, which expects data to be organized in a hierarchical folder format for efficient loading and label inference.

Before feeding images into the neural network, a preprocessing pipeline was applied to standardize their format and scale. All images were resized to 224×224 pixels, the default input dimension expected by the VGG16 architecture. Resizing not only standardizes input dimensions across the dataset but also reduces computational cost by constraining the spatial size of the input tensors.

Furthermore, all pixel values were normalized by rescaling them from the 0–255 range to a 0–1 range. This normalization step is crucial for stabilizing the training process, especially when using pre-trained models, which are typically optimized on normalized data. Without normalization, the large range of pixel intensities could lead to exploding gradients or unstable convergence during training.

The images were loaded using TensorFlow's ImageDataGenerator, which was configured to perform real-time data loading and preprocessing. For the training and validation sets, the generator was set to shuffle data at the start of each epoch, ensuring the model is exposed to data in a different order each time, thus improving generalization. In contrast, the test set was processed without shuffling to preserve the label order, which is necessary for accurate post-hoc analysis and metrics computation.

Although this study did not apply advanced data augmentation techniques such as rotation, zooming, or flipping, such techniques are often beneficial in expanding the diversity of the training set. Future iterations of this work may explore the effect of data augmentation on model robustness.

3.2 Proposed Model

3.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) is a type of deep neural network that is mainly used in building image classification tasks. In this study, we proposed a custom CNN to classify sunflower blooming stages.

The proposed CNN architecture comprises five convolutional blocks, each followed by Batch Normalization, a ReLU activation, MaxPooling, and a Dropout layer for regularization. This progressive design, increasing filter counts from 32 to 512, allows the model to capture increasingly complex features from the input images. A Flatten layer connects these convolutional features to two subsequent fully connected (Dense) layers: a hidden layer with 256 units and a final output layer with 4 units (corresponding to the number of classes) using a softmax activation for classification probabilities. Data augmentation (horizontal flips, rotations, and zooms) was incorporated into the training pipeline to enhance the model's generalization capabilities and mitigate overfitting. Figure 1 shows the overall architecture of the proposed CNN.

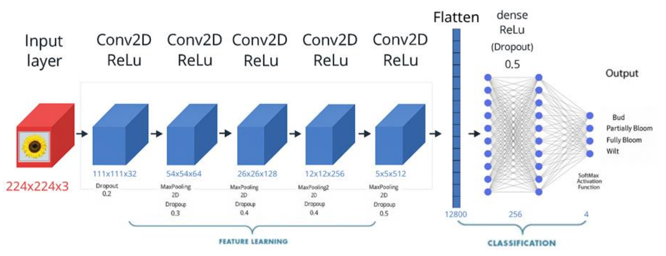

Figure 1. The architecture of proposed CNN

3.2.2 VGG16

VGG16 is a deep convolutional neural network developed by the Visual Geometry Group at the University of Oxford as shown in the figure above. It consists of 13 convolutional layers and 3 fully connected layers, and was originally trained on the ImageNet dataset, which contains over one million images across 1,000 object categories. The architecture is known for its uniform design using small 3×3 convolution filters, which enable the network to capture spatial hierarchies in the image data while maintaining manageable computational complexity. The architecture of VGG16 is shown in Figure 2.

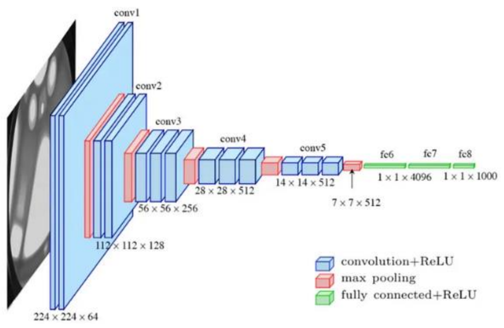

Figure 2. The architecture of VGG16

In this study, VGG16 was employed as the backbone of a transfer learning strategy for image classification. Transfer learning allows for the reuse of knowledge from large-scale datasets like ImageNet in domain-specific tasks with limited training data. By retaining the convolutional base of VGG16 and excluding its top classification layers, the model acts as a fixed feature extractor that provides rich, high-level feature representations of input images. All layers in the base model were frozen to preserve the pretrained weights during training.

To tailor the architecture to the specific classification task, a custom classification head was appended to the base model. This head includes a flattening layer to convert 2D feature maps into a 1D feature vector, a dense layer with 256 neurons and ReLU activation, and a dropout layer with a rate of 0.5 to reduce overfitting. The output layer consists of four neurons with softmax activation, corresponding to the four target classes. The model outputs the class probabilities for each input image.

The Adam optimizer was used for training, due to its efficiency and ability to converge quickly in practice. The default learning rate of 0.001 was selected to maintain a balance between convergence speed and training stability. The categorical cross-entropy loss function was used to optimize multi-class classification with one-hot encoded labels, and accuracy was chosen as the evaluation metric to reflect the model’s classification performance.

The model was implemented using TensorFlow and Keras libraries. Input images were resized to 224×224 pixels with three color channels (RGB) to match the input format required by VGG16. The pretrained VGG16 model was loaded with include_top=False, which removed the default fully connected layers and retained only the convolutional base for feature extraction.

The convolutional layers were frozen throughout the training process to preserve the knowledge acquired from the ImageNet dataset. A custom classification head was constructed and appended to the base model. This included a Flatten layer, a dense layer with 256 ReLU-activated units, a Dropout layer with a rate of 0.5 to reduce overfitting, and a final dense layer with 4 softmax-activated units for multi-class classification.

The model was compiled using the Adam optimizer and categorical cross-entropy loss function. Training was conducted for 10 epochs with a batch size of 32. During each epoch, the model was trained on the full training dataset and validated against a separate validation set. Validation accuracy and loss were monitored after each epoch to assess generalization performance and to detect early signs of overfitting or underfitting.

3.2.2 AlexNet

AlexNet is a type of deep convolutional neural network architecture which was created for ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. This architecture had won the challenge as it showed that image recognition can be performed using deep learning. In this architecture, several new key features had been introduced in 2012 causing AlexNet to become a powerful image recognition architecture at that time.

There are a total of 8 layers, including 5 convolution layers and 3 fully connected layers in AlexNet architecture. The convolution layers contain a convolution filter and a ReLU activation function. There are a total of 3 pooling layers which apply Max Pooling technique in the first three convolution layers. The input size of the model is 224 x 224 x 3. Padding technique is also applied in each convolution layer using 2x2 padding in the first two layers and 1x1 padding in the rest of the layers. This is to ensure that the size of the final output can remain the same. The activation of the final output layer will be using Softmax Function. There will be a total of 60 million parameters and 650,000 neurons in Alexnet. Cross entropy will be used as a loss function and accuracy will be the main matric to evaluate the model performance. The architecture of AlexNet is shown in Figure 3 below.

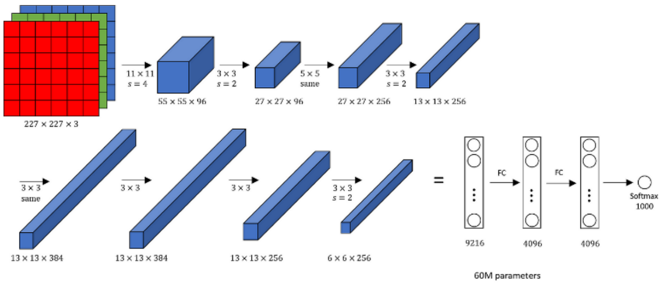

Figure 3. The architecture of AlexNet

In this study, AlexNet was chosen to be one of the transfer learning to perform classification. We customized AlexNet by changing the final layer of AlexNet to replace the original label, which consists of 1000 labels with a new layer to match the number of labels, which is 4 labels in our classification task.

The original optimizer in AlexNet is Adam optimizer. In our implementation, several different optimizers such as Adam, Adagrad, SGD and RMS Prop will be used in the experiment to get the best optimizer. Besides, different unfrozen layers will be used in experiment 2 to get the best result with the best optimizer. A total of 10 epochs will be used in the training process with a learning rate 0.001 across the experiment.

3.2.2 Inception V3

Inception V3 is a convolutional neural network architecture developed by Google in 2014, designed to improve computational efficiency and classification accuracy [1]. It is pre-trained on the ImageNet dataset, which contains over 1,000 object classes. The architecture is composed of three main components: the stem, the inception modules, and the output classifier [1].

The stem contains standard convolutional and max-pooling layers, while the output classifier is a fully connected neural network [1]. The distinctive innovation lies in the inception module, which performs convolution operations using multiple filter sizes at the same level—typically 1×1, 3×3, and 5×5 filters, along with 3×3 max-pooling [1]. This design enables the model to capture both local and global features simultaneously. Outputs from these operations are concatenated along the depth axis, and padding ensures uniform feature map size [1].

To reduce computational cost, 1×1 convolutions are applied before 3×3 and 5×5 filters, and after pooling layers, to decrease the number of input channels [1]. From Inception V2 onwards, a single 5×5 filter is replaced with two consecutive 3×3 filters, reducing multiplications from 25 to 18 [1]. Further optimization is done by factorising 3×3 convolutions into asymmetric 1×3 and 3×1 convolutions, which reduces computation by roughly one-third while preserving spatial expressiveness [1]. The key structure of Inception V3 is depicted in Figure 4.

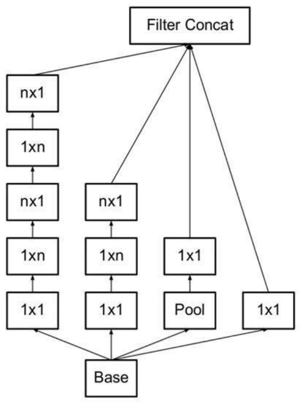

Figure 4. Key structure of Inception V3 [8]

Another key feature is the use of auxiliary classifiers—small classifiers inserted in intermediate layers to mitigate the vanishing gradient problem [1]. These auxiliary networks contribute to the total loss function and improve convergence. In Inception V3, batch normalization is applied to the auxiliary classifiers to further enhance stability and performance [8]. The model also employs the RMSProp optimizer and a regularization technique called label smoothing to prevent overfitting [1].

In this study, the pre-trained Inception V3 model from TensorFlow (trained on ImageNet) was used. The input image size was resized to 224×224×3, representing RGB colour channels. The original fully connected classifier for 1,000 classes was replaced with a custom classifier tailored for four classes, corresponding to the four blooming stages of sunflower.

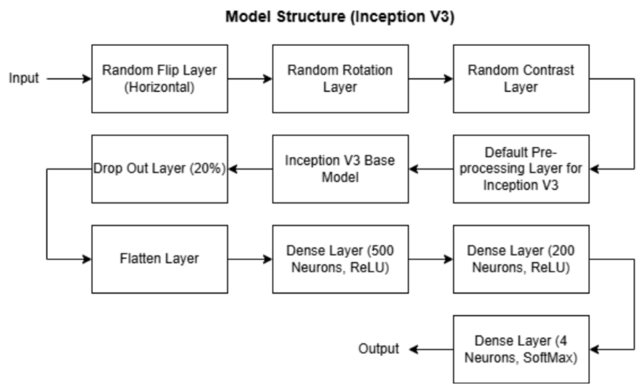
Model Structure (Inception V3)
Input
Random Flip Layer (Horizontal)
Random Rotation Layer
Random Contrast Layer
Default Pre-processing Layer for Inception V3
Inception V3 Base Model
Drop Out Layer (20%)
Flatten Layer
Dense Layer (500 Neurons, ReLU)
Dense Layer (200 Neurons, ReLU)
Dense Layer (4 Neurons, SoftMax)
Output

Figure 5. Model structure of Inception V3

As shown in Figure 5, the input image first passes through random data augmentation layers, which include horizontal flipping, rotation, and contrast adjustment (applied only during training to enhance generalization and reduce overfitting). Next, the image is fed into Inception V3's pre-process layers and feature extraction layers, followed by a dropout layer with a dropout rate of 0.2 to reduce overfitting. A flatten layer then converts the feature maps into a 1D vector for final classification.

The final classification is done by a custom fully connected neural network with three dense layers. The first dense layer has 500 neurons with rectified linear unit (ReLU) activation, the second dense layer has 200 neurons with ReLU activation and the last layer has 4 neurons with SoftMax activation. The last layer of the classifier outputs the probability that the image belongs to each class.

4. RESULTS AND DISCUSSIONS

4.1 Proposed CNN model

This section presents the experiment result and the result analysis of the proposed CNN model.

The experimental methodology for the proposed CNN involved one primary phase:

(1) Experiment 1: Optimizer and Learning Rate Tuning This phase focused on identifying the optimal combination of optimizer and learning rate that yielded the best performance for the proposed CNN architecture.

For consistency with other experimental sections (e.g., VGG16, AlexNet), each model configuration was trained for a fixed 10 epochs. Performance was evaluated using training accuracy, validation accuracy, and test accuracy, alongside loss, precision, recall, and F1-score.

4.1.1 Experiment 1: Optimizer and Learning Rate Tuning

In this experiment, the proposed CNN architecture, including the integrated data augmentation and Batch Normalization layers, was trained using various optimizer and learning rate combinations to determine the most effective setup. All models were trained for a consistent 10 epochs to ensure direct comparability. The learning rate was set to 0.001 for all optimizers unless otherwise specified.

The following table summarizes the performance of the proposed CNN across different optimizer and learning rate configurations. The results reflect the metrics achieved at the end of the 10th epoch.

Table 1. proposed CNN Model Performance with Different Optimizers and Learning Rates

Optimizer	Learning Rate	Train Accuracy	Validation Accuracy	Test Accuracy
Adam	0.001	66.88%	44.65%	44.65%
Adam	0.0005	65.75%	35.35%	12.50%
Adam	0.0001	65.63%	35.35%	12.50%
SGD	0.01	37.12%	66.05%	67.20%
SGD	0.001	66.84%	36.28%	30.75%
RMSprop	0.001	70.77%	67.44%	71.19%
RMSprop	0.0005	71.24%	53.95%	68.73%
Adagrad	0.001	61.62%	35.35%	12.50%

The results from Table 1 provide comprehensive insights into the performance of the proposed CNN when trained with various optimizers for a fixed duration of 10 epochs.

The RMSprop optimizer with a learning rate of 0.001 stands out as the best performer among all tested configurations for the proposed CNN. It achieved the highest validation accuracy of 67.44% and an F1-score of 64.33% at the end of 10 epochs. This suggests that RMSprop, with its adaptive learning rate for each parameter, effectively navigated the loss landscape and converged to a robust solution that generalized well to unseen data.

The SGD optimizer with a learning rate of 0.01 and momentum of 0.9 also showed strong generalization, reaching a validation accuracy of 66.05% and an F1-score of 55.97%. Its performance was comparable to RMSprop 0.001, indicating that with appropriate momentum and a well-tuned learning rate, SGD can be very effective, even though its training accuracy at 10 epochs was lower, suggesting a more conservative but stable learning path.

Conversely, the Adam optimizer, while starting strong with a 0.001 learning rate (44.65% validation accuracy), showed diminished performance when the learning rate was decreased (0.0005 and 0.0001 both yielded 35.35% validation accuracy). This indicates that Adam might have converged to less optimal local minima or became too conservative at lower learning rates within the limited 10 epochs. Similarly, reducing the RMSprop learning rate to 0.0005 also led to a drop in validation accuracy to 53.95%, reinforcing the sensitivity of optimizers to learning rate tuning for optimal generalization.

The Adagrad optimizer with a learning rate of 0.001 performed the worst among all tested optimizers, achieving a validation accuracy of only 35.35% and an F1-score of 18.46%. Adagrad's aggressive decrease in learning rates for frequently updated parameters likely caused it to stall prematurely or find a very poor local minimum, leading to a lack of significant learning and generalization within the 10-epoch time frame.

Overall, the consistent application of data augmentation (horizontal flips, rotations, and zooms) across all proposed CNN experiments was a crucial factor. By artificially expanding the training dataset, it helped mitigate overfitting and improved the model's ability to generalize to unseen sunflower images. The inclusion of Batch Normalization layers after each convolutional block also contributed significantly to training stability and efficiency, allowing for smoother optimization and potentially enabling the model to learn more complex features effectively within the limited epoch count.

The established performance of this proposed CNN, particularly with the RMSprop 0.001 and SGD 0.01 (momentum) configurations, serves as a strong point of comparison for the subsequent analysis of transfer learning models.

4.2 Transfer Learning

This section presents the experimental results and comparative analysis of transfer learning using the VGG16, AlexNet and Inception V3 for sunflower blooming stage classification.

Two experimental phases were conducted across the three pre-trained model:

(1) Experiment 1: Evaluating the performance of different optimizers while keeping all convolutional layers of the pre-trained model base frozen.

(2) Experiment 2: Fine-tuning the pre-trained model model by unfreezing different percentages of convolutional layers using the best-performing optimizer identified in Experiment 1.

Each model was trained for 10 epochs and 0.001 learning rate, and evaluated using training accuracy, validation accuracy, and test accuracy as performance metrics.

4.2.1 VGG16

In the first experiment, all 13 convolutional layers of the VGG16 base model were set as non-trainable, effectively transforming the base model into a fixed feature extractor. Only the classification head was updated during training. Four optimizers were compared: SGD, Adam, RMSprop, and Adagrad. 10 epochs are used for this experiment.

Table 2. Model Performance with VGG16 Frozen Convolutional Layers

Optimizer	Train Accuracy	Validation Accuracy	Test Accuracy
SGD	83.68%	87.74%	87.82%
Adam	98.43%	91.61%	92.95%
RMSprop	98.04%	88.38%	92.30%
Adagrad	99.73%	92.90%	94.23%

The results clearly demonstrate that the choice of optimizer significantly influences model performance. Adagrad achieved the highest training (99.73%), validation (92.90%), and test (94.23%) accuracy among all optimizers. As an adaptive optimizer, Adagrad adjusts the learning rate individually for each parameter, which is beneficial when some parameters require larger updates than others. This fine-grained control appears to be particularly advantageous in our sunflower classification task, where subtle differences in petal openness, structure, and withering stages must be learned.

SGD, a basic gradient descent method with a constant learning rate, exhibited the weakest performance, which is 87.82% test accuracy. This reflects its limitations in convergence speed and stability. Adam and RMSprop, which incorporate adaptive learning rate strategies and momentum, showed stronger generalization, but still slightly short of Adagrad's consistency and precision.

Overall, freezing all convolutional layers yielded good performance while significantly reducing computational complexity and risk of overfitting. This supports the idea that pre-trained CNNs already encode useful visual features that can be effectively reused in similar visual domains, such as plant image classification.

In the second phase, the experiment aimed to assess whether partial fine-tuning of the VGG16 model could further enhance classification performance. Based on the results from Experiment 1, the Adagrad optimizer was used in all configurations in this phase. Furthermore, each model was trained for a consistent 10 epochs to ensure comparability.

The experiment involved unfreezing the top 2, 4, and 8 convolutional layers, corresponding to approximately 15%, 31%, and 62% of the VGG16 base. The remaining layers remained frozen to preserve low-level generic features.

Table 3. Performance with Selective Fine-Tuning for VGG16

% of Unfrozen Layer	Train Accuracy	Validation Accuracy	Test Accuracy
15	92.42%	89.68%	91.03%
31	91.64%	92.26%	92.30%
62	92.56%	89.03%	92.31%

The result shows that unfreezing additional layers did not surpass the performance achieved by the frozen model in Experiment 1. The highest test accuracy among the fine-tuned models (92.31%) was observed when unfreezing 62% of the convolutional layers, yet this remained inferior to the 94.23% achieved with the fully frozen configuration

Given that the dataset used in this study is relatively small containing fewer than 1,000 images and the classification task shares visual similarities with the pre-training domain (ImageNet), the configuration that kept all convolutional layers frozen and employed the Adagrad optimizer proved to be the most robust and generalizable.

4.2.2 AlexNet

In the first experiment, all 8 convolutional layers of the AlexNet base model were set as non-trainable, effectively transforming the base model into a fixed feature extractor. Only the classification head was updated during training. Total of 4 optimizers, which are Adam, SGD, Adagrad and RMS Prop are being chosen to conduct the experiment. The result for each optimizer is shown in Table 4.

Table 4. Model Performance with AlexNet Frozen Convolutional Layers

Optimizer	Train Accuracy	Validation Accuracy	Test Accuracy
Adam	99.23%	94.19%	95.51%
Adagrad	94.91%	94.83%	96.79%
SGD	94.00%	93.54%	94.87%
RMSprop	96.56%	96.12%	96.79%

From the tables above, we can observe that the overall result for each optimizer gave similar performance. In terms of validation accuracy result, RMSprop optimizer a gave the best result which is 96.56%, which just exceeded 2% of accuracy compared to the other optimizer. While in terms of test accuracy, RMS Prop and Adagrad optimizer gave the best result which is 96.79%. Therefore, we can conclude that the RMSProp optimizer gave the best result as the validation accuracy and the test accuracy achieved the highest value of accuracy among the optimizer followed by the Adagrad optimizer which achieved 94% of validation accuracy and 96.79% of test accuracy. Therefore, RMSprop optimizer will be selected to proceed to experiment 2.

There are only a total of 5 convolutional layers in the AlexNet based model. Therefore, in experiments 2, the first two layers of AlexNet base model will be frozen by freezing 20% and 40% of the base model of the convolutional layer. Table 5 shows the result of experiment 2.

Table 5. Performance with Selective Fine-Tuning for AlexNet

% of Unfrozen Layer	Train Accuracy	Validation Accuracy	Test Accuracy
20	88.04%	87.74%	89.00%
40	96.81%	93.54%	93.00%

The test accuracy for 20% and 40% unfrozen layers are 89% and 93% respectively. We can observe that by using 40% unfrozen layers, the train, validation and test accuracy outperform than 20% unfrozen layers. However, we can notice that the training and validation result for 40% unfrozen did not have much difference compared to the result from experiment 1. From the result, it was expected that the performance of 40% unfrozen layers will outperform the 20% unfrozen layers. This is because in 20% unfrozen layers, the model only learns the high-level representation of our image datasets while in 40% unfrozen layers, the model has a chance to learn more pattern or information from our datasets.

In terms of training results, 100% frozen layers achieved the best result with 96% accuracy compared to 40% unfrozen layers which only achieved 93% accuracy. This might be due to the size of our dataset being small and the original ImageNet datasets that were trained on AlexNet have a high similarity with our datasets. Therefore, the 100% frozen layers approach might be suitable for our case.

4.2.3 Inception V3

In the first experiment, all layers in the pre-trained base model were kept frozen, aimed at identifying the optimal optimizer for training the Inception V3 model with all pre-trained layers frozen. The training accuracy, validation accuracy, and test accuracy at the 10th epoch are presented in Table 6.

Table 6. Model Performance with Inception V3 Frozen Convolutional Layers

Optimizer	Train Accuracy	Validation Accuracy	Test Accuracy
Adam	96.21%	91.61%	91.67%
RMSprop	95.69%	87.74%	86.54%
SGD	79.11%	85.16%	85.26%
Adagrad	88.38%	88.39%	90.38%

Among all optimizers, Adam achieved the best performance, with a validation accuracy of 91.61% and a test accuracy of 91.67%, followed by Adagrad, then RMS Prop, and finally SGD, which had the lowest performance. Therefore, Adam was selected as the best optimizer and used in Experiment 2.

In Experiment 2, models were trained using varying percentages of unfrozen layers in the Inception V3 base model to determine the best configuration. All models were trained with the Adam optimizer and a learning rate of 0.0001. The configurations that are tested are 0% unfrozen (all layers frozen), 10% unfrozen (31 layers unfrozen), 15% unfrozen (47 layers unfrozen), 20% unfrozen (62 layers unfrozen), and 25% unfrozen (78 layers unfrozen).

The training accuracy, validation accuracy, and test accuracy at the 10th epoch are presented in Table 7.

Table 7. Model Performance with Inception V3 with Different Percentage of Unfrozen Convolutional Layers

% of Unfrozen Layer	Train Accuracy	Validation Accuracy	Test Accuracy
0	96.21%	91.61%	91.67%
10	98.69%	94.19%	96.79%
15	98.69%	93.55%	94.23%
20	98.96%	94.84%	95.51%
25	99.22%	96.77%	92.95%

The configuration with 0% unfrozen layers (i.e., fully frozen pre-trained model) yielded the lowest performance, though still high. This suggests that Inception V3, pre-trained on ImageNet, generalizes reasonably well to the sunflower blooming stage dataset.

The best test accuracy was achieved at 10% unfrozen layers (96.79%), while the highest validation accuracy was observed at 25% unfrozen layers (96.77%). However, the drop in test accuracy to 92.95% at 25% indicates potential overfitting. Configurations with 15% and 20% unfrozen layers achieved moderate validation and test accuracies, indicating balanced but not optimal performance. The model with 10% unfrozen layers strikes the best balance between generalization and performance, with high validation accuracy (94.19%) and the highest test accuracy (96.79%), suggesting minimal overfitting.

So, the best overall performance achieved using the Inception V3 model was when 10% of layers are unfrozen, and trained with the Adam optimizer and a learning rate of 0.0001. This configuration provides the best generalization and accuracy on unseen data.

4.2.4 Comparison of classification accuracies of the proposed and transfer learning models

The following results comparison will be based on the best result for each model, and test accuracy will be used as the main indicator to compare. Figure 6 shows the test accuracy obtained from the proposed model and transfer learning model.

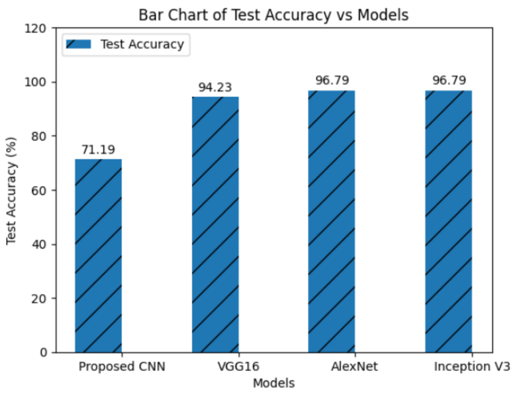

Figure 6. The result comparison between all models

By comparing the result between the proposed CNN model and transfer learning model, we can conclude that all the transfer learning models produce higher accuracy (above 90%) than our proposed CNN model (71.19%). While among the transfer learning models which are VGG16, AlexNex and Inception V3 models, both AlexNet and Inception V3 gave the best accuracy results (96.79%).

Table 8 shows the best optimizer and the % of unfrozen layers for each model.

Table 8. Best Optimizer and % of unfrozen layers

Model	Optimizer	% of unfrozen layers
Proposed CNN	RMSprop	-
VGG16	Adagrad	0
AlexNet	RMSprop	0
Inception V3	Adam	10

From Table 8, we can observe that the RMSprop optimizer contributed to the both proposed CNN and AlexNet model to achieve their best result. While in terms of the % of unfrozen layers, 0% of unfrozen layers gave the best test accuracy for both VGG16 and AlexNet models. This might be due to our data size being small and the original ImageNet datasets that were trained on AlexNet and VGG16 have a high similarity. While Inception V3 is able to achieve the best performance when unfreezing 10% of the layers. We suspect that the architecture of Inception V3 is able to capture more important features and as a result yield more improvement when fine tuning the unfrozen layers.

5. CONCLUSION

The classification of sunflower blooming stage is important for agriculture fields. In this paper, we proposed a CNN model that is able to perform an image classification on the sunflower blooming stage. Moreover, we also fine tuned on three types of transfer learning models which are VGG 16, AlexNet and Inception V3 and the result of the transfer learning model is compared to our proposed CNN model. We can observe that our fine-tuned transfer learning model, the AlexNet and Inception V3 model, is able to outperform our proposed CNN model. The difference of hyperparameters such as different types of optimizer and % of unfrozen layers is able to give a significant effect for the final performance of the models. Thus, our fine tune transfer learning models, which are AlexNet and Inception V3, are able to perform well in this classification problem which have a 96.79% accuracy.

REFERENCES

[1] Bansal, L. (2021, September 18). Inception and Versions of Inception Network. Medium. Retrieved May 31, 2025, from https://luv-bansal.medium.com/inception-and-versions-of-inception-network-c350758ba3e6

[2] Bataduwaarachchi, S. D., Sattarzadeh, A. R., Stewart, M., Ashcroft, B., Morrison, A., & Huynh, V. T. (2023). Towards autonomous cross-pollination: Portable multi-classification system for in situ growth monitoring of tomato flowers. Smart Agricultural Technology, 4, 100205-100212.

[3] Khanal, S. R., Sapkota, R., Ahmed, D., Bhattarai, U., & Karkee, M. (2023). Machine Vision System for Early-stage Apple Flowers and Flower Clusters Detection for Precision Thinning and Pollination. International Federation of Automatic Control Papers On Line, 56(2), 8914-8919.

[4] Li, G., Fu, L., Gao, C., Fang, W., Zhao, G., Shi, F., Dhupia, J., Zhao, K., Li, R., & Cui, Y. (2022). Multi-class detection of kiwifruit flower and its distribution identification in orchard based on YOLOv5l and Euclidean distance. Computers and Electronics in Agriculture, 202, 107342-107352.

[5] Mi, Q., Yuan, P., Ma, C., Chen, J., & Yao, M. (2025). TflosYOLO+TFSC: An Accurate and Robust Model for Estimating Flower Count and Flowering Period. arXiv.

[6] Mu, X., He, L., Heinemann, P., Schupp, J., & Karkee, M. (2023). Mask R-CNN based apple flower detection and king flower identification for precision pollination. Smart Agricultural Technology, 4, 100151-100160.

[7] Park, H.-M., & Park, J.-H. (2023). YOLO Network with a Circular Bounding Box to Classify the Flowering Degree of Chrysanthemum. Agri Engineering, 5, 1530-1543.

[7] Patel, S. (2023). Marigold Flower Blooming Stage Detection in Complex Scene Environment using Faster RCNN with Data Augmentation. International Journal of Advanced Computer Science and Applications, 14(3), 676-684.

[8] Shah, A. (2022, November 25). Understanding Inception-V3. Medium. Retrieved May 31, 2025, from https://medium.com/@akshayhitendrashah/understanding-inception-v3-d43fe7ca66a9

[9] Shuaeba, S. M. A. A., Alamb, S., Alic, M. H., & Kamruzzaman, M. (2021). Marigold Blooming Maturity Levels Classification Using Machine Learning Algorithms. International Journal of Computer (IJC), 40(1), 50-65.

[10] Sun, X., Li, Z., & Ni, C. (2021). Four-Dimension Deep Learning Method for Flower Quality Grading with Depth Information. Electronics, 10, 1353-1365.

[11] Tai, N. D., Lin, W. C., & Thinh, N. T. (2023, December 22). Real-time Feature Recognition of Cantaloupe Flowers and Buds in Greenhouse based on Deep Learning for Autonomous Pollination. SSRN. https://ssrn.com/abstract=4673081

[12] Xu, X., Wang, H., Miao, M., Zhang, W., Zhang, Y., Dai, H., Zheng, Z., & Zhang, X. (2023). Cucumber Flower Detection Based on YOLOv5s-SE7 Within Greenhouse Environments. IEEE Access, 11, 64358-64369.

[13] Yuan, W. (2023). Accuracy Comparison of YOLOv7 and YOLOv4 Regarding Image Annotation Quality for Apple Flower Bud Classification. Agri Engineering, 5, 413-424.