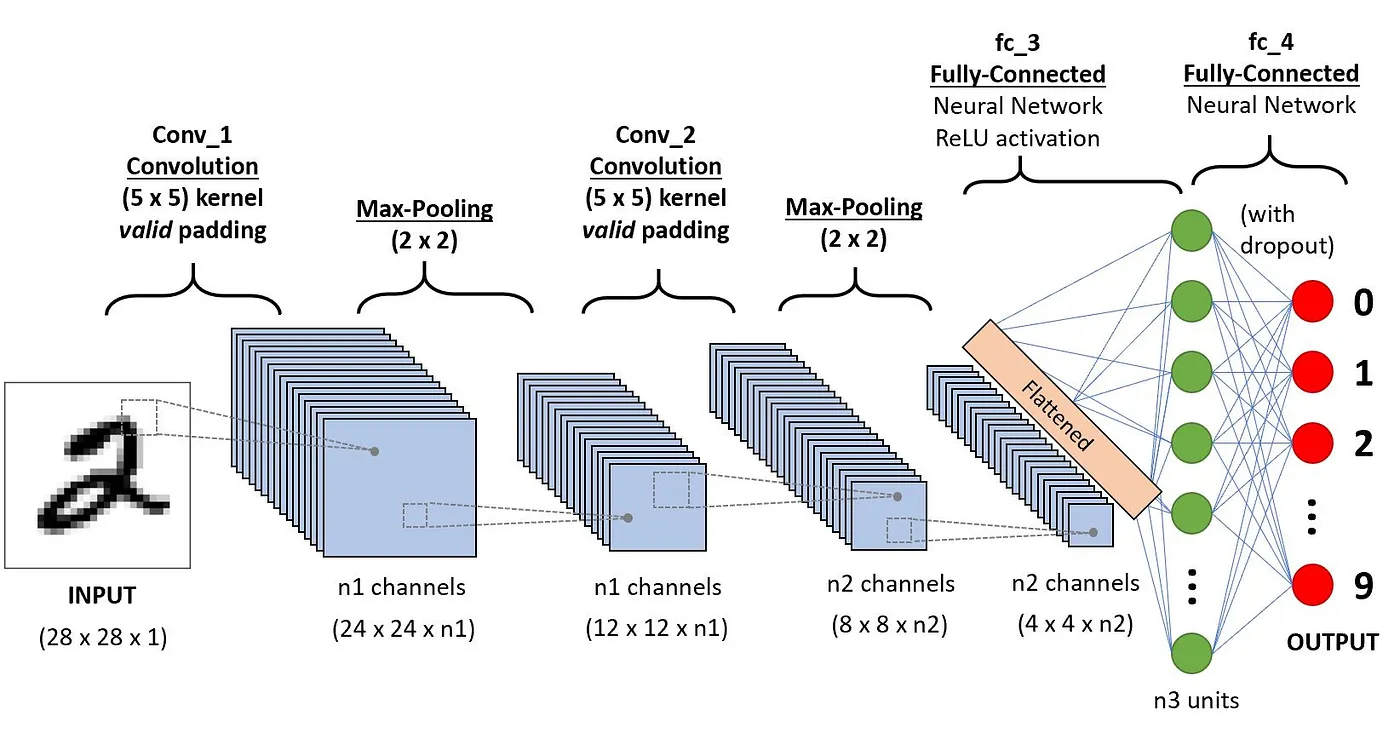
**FLORA GENUS CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK (CNN)**

1. **Introduction**

A Convolutional Neural Network (CNN), also known as ConvNet, is a type of deep learning algorithm mainly designed for tasks related to object recognition (IBM, 2024). Some of the most popular uses of CNN include classifying images into different categories, helping the perception systems in self-driving cars to distinguish surrounding environments, and video analysis (Datagen). CNN is distinguished from classic machine learning models such as SVMs or Decision Trees due to their ability to extract features at a large scale, which increases efficiency by reducing the need for manual feature engineering. A variety of pre-trained CNN architectures include CGG-16, ResNet50, and EfficientNet (Mohanty et al., 2022)

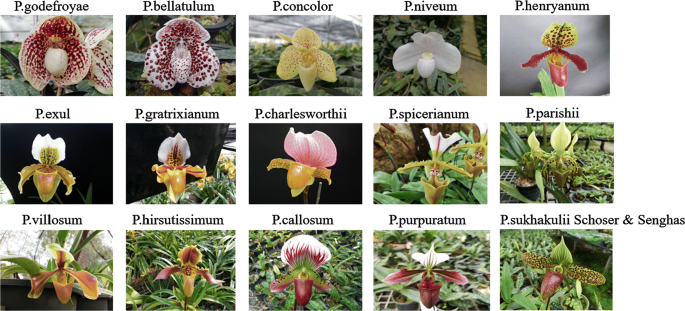
CNN is inspired by the system of the human visual cortex. Like the visual cortex, CNN has a hierarchical structure with simple features extracted in early layers and more complex features built in deep layers. In the context of local connectivity, neurons in the visual cortex connect only to a local region of the input, not the entire visual field (De Cesarei et al., 2021). Similarly, neurons in a CNN layer are exclusively connected to a local region of the input volume through the convolution operation. This local connectivity ensures efficiency (De Cesarei et al., 2021).

Key components of the CNN architecture include the Convolutional layers, Pooling Layers, and Fully connected layers. Images are then flattened after passing a series of convolutional layers and pooling layers to allow for input into the fully connected layers, which perform classification or regression tasks based on the learned features extracted by the preceding layers.



1. **Dataset**

The dataset for this project includes approximately 6700 pages of different genuses of orchids. Orchids are plants that belong to the family Orchidaceae, and are considered to be a resilient type of flower that could grow in any kind of habitat (with the exception of glaciers). Orchids are separated into 5 different subfamilies, Cattleya, Dendrobium, Oncidium, Phalaenopsis, and Vanda (Hsiao et al., 2011). The dataset contains 4000 trained data without background and 800 augmented images per genus. Image augmentation is the process of generating new images from the given dataset to increase diversity. Meanwhile, the test set contains 2500 images (500 images per genus) and 185 images from the internet.



1. **Objective**

Orchids, known for their ornamental appeal, are frequently cultivated, with each genus requiring specific growing methods. Beginner cultivators should understand the characteristics of the orchid genus they intend to grow. However, many novices lack the necessary knowledge and experience, leading to suboptimal orchid growth and flowering. This study created a system to classify orchid genus images, specifically for Cattleya, Dendrobium, Oncidium, Phalaenopsis, and Vanda. The Convolutional Neural Network (CNN) method was used for image classification, processing orchid images based on their genus. The classification involves training and testing phases: the training phase develops a CNN model and adjusts weights, while the testing phase applies the model to new image data. K-Fold Cross Validation was utilized during training, and a Confusion Matrix was used to evaluate the CNN model after testing.

1. **Data Processing and Collection**

Data preprocessing plays a pivotal role in readying the dataset for training a Convolutional Neural Network (CNN) designed to classify five common genera of orchids. The preprocessing process encompasses several key steps. Initially, images are loaded and transformed from BGR to RGB color format, aligning them with the expected input format for CNNs. Standardizing the dimensions of all images to either 224x224 pixels or adjusting them to circumvent memory constraints ensures uniformity across the dataset. Subsequently, shuffling the dataset prevents the model from discerning any underlying order within the data, thereby enhancing its ability to generalize. Further, feature extraction segregates images from their corresponding labels into separate arrays, laying the groundwork for model training. Normalizing image pixel values to a range of 0-1 expedites and stabilizes the training process. Additionally, converting labels from numerical indices to a one-hot encoded format renders them compatible with CNN classification tasks. To optimize memory usage, unused data variables are promptly cleared from memory, a particularly crucial practice in resource-constrained environments like Google Colab. These meticulous preprocessing steps ensure the dataset's proper formatting and optimization, ultimately facilitating superior model performance and efficient learning.

1. **Building the CNN Architecture**

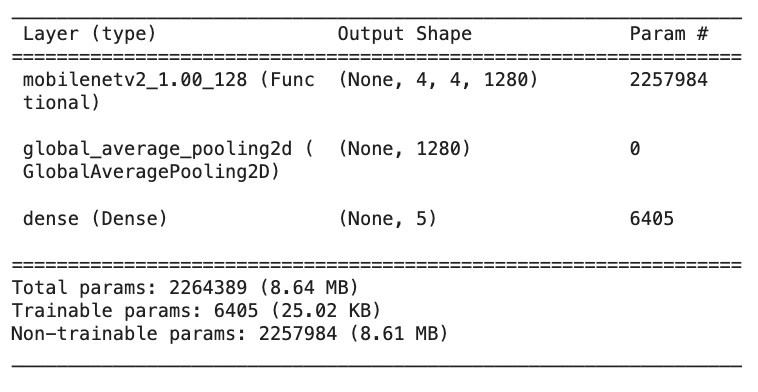
To initialize and configure a pre-trained MobileNetV2 model for transfer learning tasks, we employed the Keras library, leveraging components such as MobileNetV2 and plot\_model. Key parameters for configuring the model were meticulously explained in accompanying notes within the code. This setup involved loading pre-trained weights from ImageNet, excluding fully connected layers to enable custom layers (include\_top=False), and ensuring the convolutional base remains untrainable by setting conv\_base.trainable=False. Subsequently, we instantiated the MobileNetV2 model (conv\_base) with these settings, specifying an input size of 224x224 pixels and 3 channels for RGB images. To prevent weight retention, the convolutional base was frozen, and a summary of the model architecture was printed for comprehensive review. Visualizing the architecture was facilitated by the plot\_model function, generating a graphical representation saved as an image file for further scrutiny. This configuration facilitates efficient adaptation of the model to new tasks through transfer learning, with a focus on fine-tuning the final layers.

1. **Modifying the MobileNetV2 Architecture**

The architecture of our model is based on MobileNetV2, a pre-trained convolutional neural network utilized as a feature extractor within a Sequential model in Keras. Several modifications were applied to this model architecture:

1. Feature Extractor: MobileNetV2 serves as a potent feature extractor, capturing hierarchical features from input images to facilitate subsequent classification tasks.
2. Global Average Pooling 2D: Following feature extraction, global average pooling is applied to condense spatial dimensions while preserving channel-wise information. This operation reduces computational complexity and enhances feature robustness.
3. Dense Layer: A dense layer with 5 neurons and softmax activation was appended for multi-class classification. Softmax activation generates class probabilities, making it ideal for multi-class classification tasks.

The model was compiled using the Adam optimizer with a learning rate of 0.001. Adam optimizer's adaptive learning rate mechanism enhances training efficacy. Categorical cross-entropy loss function was chosen, well-suited for multi-class classification problems, facilitating comparison of model output probabilities with true class labels during training.



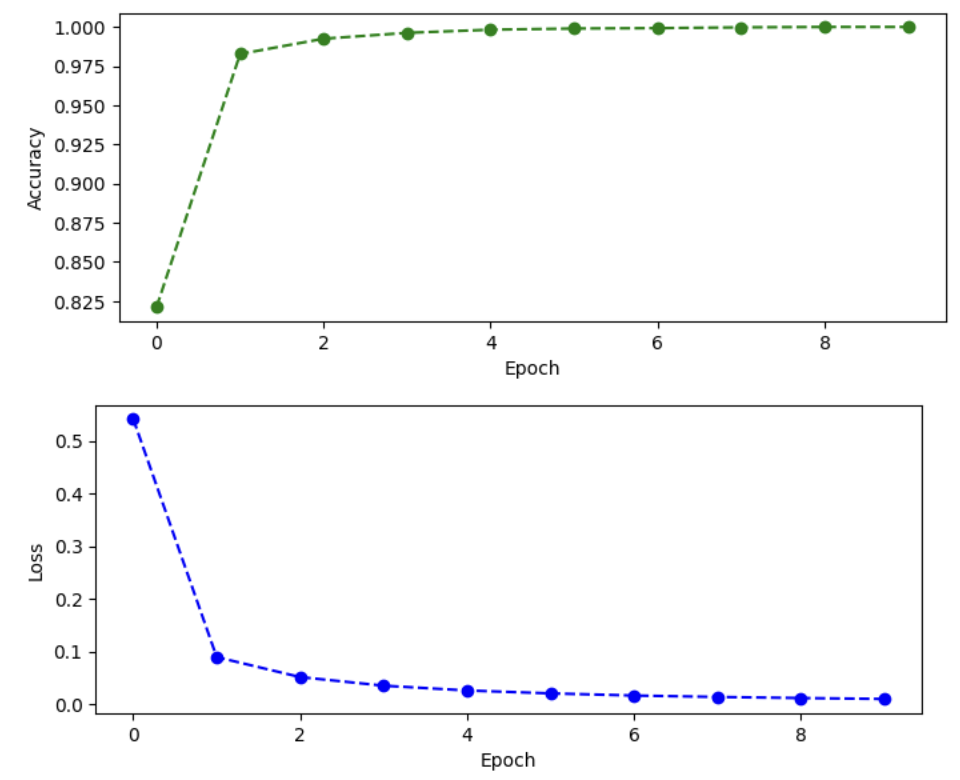
1. **Training the Data**

We trained the data with k-fold cross validation to assess its performance across multiple folds of the dataset. First, we split the dataset into k folds, ensuring shuffling and a consistent random state (seed) for reproducibility. For each fold, a new instance of the MobileNetV2-based model is defined. With regards to model training and evaluation, the data is divided into training and validation sets based on the current fold. The model is trained on the training set for the specified number of epochs and batch size, with validation performed using the validation set. Evaluation metrics, including accuracy and loss, are calculated for each fold.

For each fold, the accuracy and loss metrics are printed. The collected scores and training histories are returned for further analysis. The number of key parameters we included are 5 folds, 10 epochs, and a batch size of 64.

With these hyperparameters, we got a training accuracy of 99.625% and training loss of 0.018.

The k-fold cross-validation process allows for robust evaluation of the model's performance by training and testing on different subsets of the dataset. This approach helps in assessing the model's generalization capabilities and identifying potential overfitting or underfitting issues.

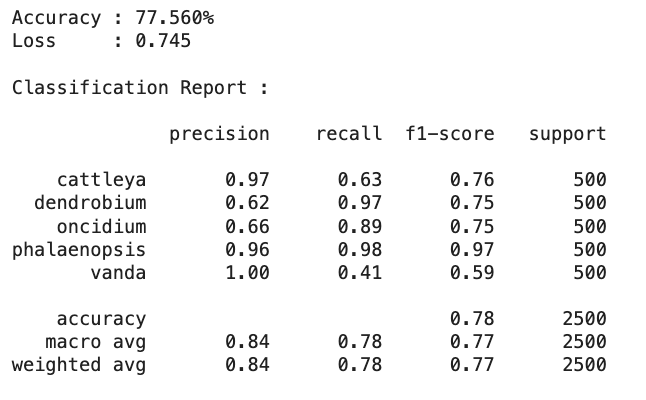


*Without k-fold Cross Validation*

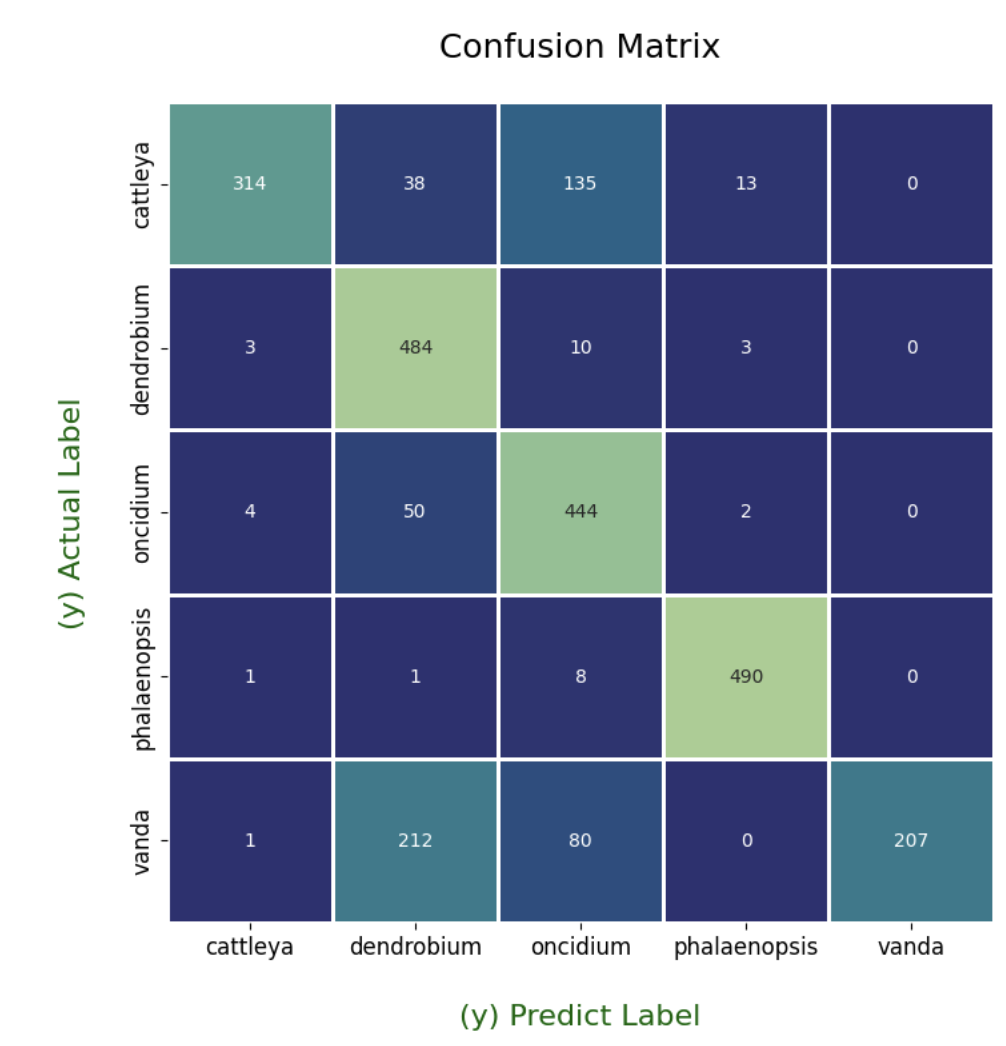
We also tested our data without k-fold cross validation to provide us with a quick insight into the model's performance on the training data. With 10 epochs and batch size of 64, we obtained a training accuracy of 100% and loss of 0.0097.

Results (with 5-fold CV)

We gathered a few insights from the classification report below. 97% of the instances predicted as Cattleya were truly Cattleya. For Phalaenopsis, the recall is 98%, meaning that the model incorrectly identified 98% of all actual Phalaenopsis instances. Overall, metrics such as accuracy indicates that 78% of all predictions made by the model were correct.



Based on the confusion matrix below, the model correctly predicted 324 instances of Cattleya, 484 instances of Dendrobium, 444 instances of Onicidium and 207 instances of Vanda. We obtained an accuracy of 82%.



1. **Conclusion**

Our Convolutional Neural Network (CNN) utilizing the MobileNetV2 architecture for classifying orchid genera has shown considerable promise in aiding both beginner and experienced orchid cultivators. It achieves a testing accuracy of 77.56%.

However, the model does exhibit signs of overfitting, as evidenced by the perfect training accuracy (without k-fold cross-validation) and significantly higher accuracy during cross-validation. This suggests that while the model performs exceptionally under controlled testing conditions, its ability to generalize on truly unseen data could be improved.

To address these challenges, future work could include implementing regularization techniques to combat overfitting, enhancing the diversity of the training dataset to better represent real-world variability, and potentially adjusting class distributions to manage class-specific difficulties. Exploring adjustments to the model's architecture and training process will be crucial in maximizing its effectiveness and efficiency.

Additionally, deploying this model on mobile platforms could greatly increase its accessibility, making it a practical tool for orchid enthusiasts to use directly in their cultivation activities. This move would mark a significant step toward integrating advanced machine learning tools in everyday botanical applications, thereby enriching the resources available for precise plant care and cultivation.

**Bibliography**

# Convolutional Neural Network: Benefits, Types, and Application. Datagen. <https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/>

# De Cesarei, A., Cavicchi, S., Cristadoro, G., & Lippi, M. (2021). Do Humans and Deep Convolutional Neural Networks Use Visual Information Similarly for the Categorization of Natural Scenes?. Cognitive science, 45(6), e13009. <https://doi.org/10.1111/cogs.13009>

Mohanty, Niharika & Pradhan, Manaswini & V N Reddy, Annapareddy & Kumar, Sachin & Alkhayyat, Ahmed. (2022). Integrated Design of Optimized Weighted Deep Feature Fusion Strategies for Skin Lesion Image Classification. Cancers. 14. 5716. 10.3390/cancers14225716.

# What are convolutional neural networks? (2024) IBM. <https://www.ibm.com/topics/convolutional-neural-networks>

# Hsiao Y., Zhao-Jun Pan, Chia-Chi Hsu, Ya-Ping Yang, Yi-Chin Hsu, Yu-Chen Chuang, Hsing-Hui Shih, Wen-Huei Chen, Wen-Chieh Tsai, Hong-Hwa Chen, Research on Orchid Biology and Biotechnology, Plant and Cell Physiology, Volume 52, Issue 9, September 2011, Pages 1467–1486, <https://doi.org/10.1093/pcp/pcr100>

# 