**Project: Flower Classifier**

## Table of Contents

1. Introduction
2. Project Overview
3. Dataset
4. Neural Network Architecture
5. Training Process
6. Graphical User Interface (GUI)
7. Results and Conclusion

**1. Introduction**  
Image classification is a fundamental task in computer vision, with applications ranging from medical diagnosis to autonomous vehicles. In this project, we aim to develop an image classification system capable of identifying different species of flowers. Leveraging deep learning techniques, we create a model trained on a dataset consisting of images from Kaggle. The project is based on the notebook available on Kaggle: [Flower Recognition CNN Keras](<https://www.kaggle.com/code/rajmehra03/flower-recognition-cnn-keras/notebook>).

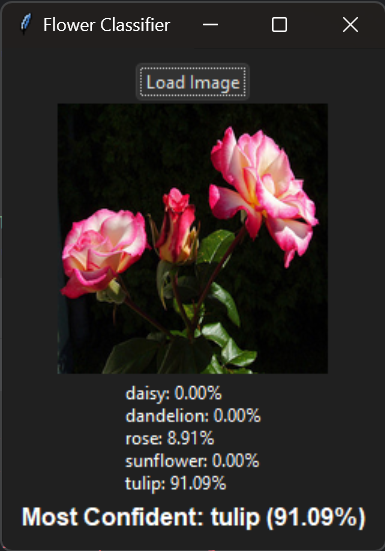
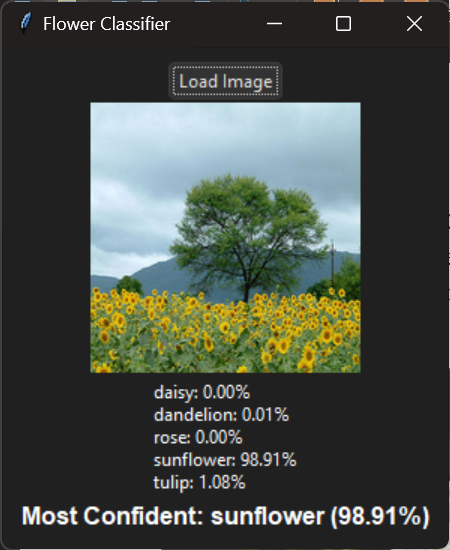
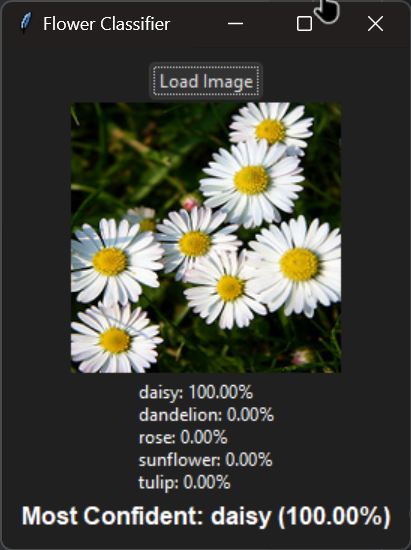
**2. Project Overview**  
The dataset comprises images of five different types of flowers: roses, daisies, sunflowers, tulips, and dandelions. These images are split into 80% training and 20% validation subsets using TensorFlow's built-in functionality. Our goal is to train a neural network to classify these images into their respective classes.

**3. Dataset**  
The dataset is sourced from Kaggle and contains images of roses, daisies, sunflowers, tulips, and dandelions. Each class represents a different flower type, with images exhibiting variations in size, color, and background. The dataset is divided into training and validation subsets to facilitate model training and evaluation

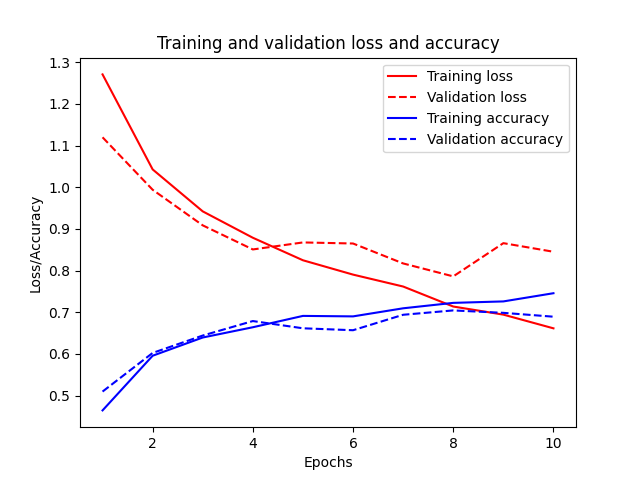
**4. Neural Network Architecture**  
The neural network architecture consists of multiple convolutional layers designed to extract features from input images. Specifically, our architecture includes three convolutional layers, each followed by max-pooling layers to downsample the feature maps. The convolutional layers employ filters to detect patterns and features within the images, with increasing complexity and abstraction in deeper layers. Fully connected layers at the end of the network enable classification based on the learned features. The number of neurons in the fully connected layers corresponds to the number of flower classes (five in our case).

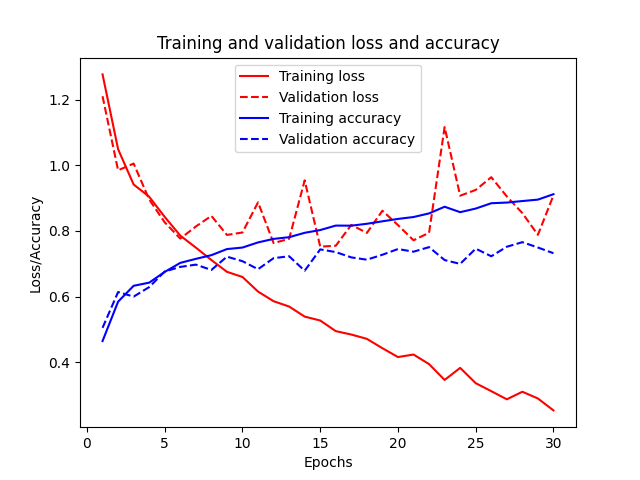
**5. Training Process**  
We conduct three training sessions with different numbers of epochs: 10 epochs, 30 epochs, and 50 epochs. During training, the model learns to classify flower images by minimizing a predefined loss function using stochastic gradient descent with the Adam optimizer. The training process involves feeding batches of images through the network, computing the loss, and updating the model parameters based on backpropagation. We monitor training progress using metrics such as accuracy and loss to assess model performance and convergence.

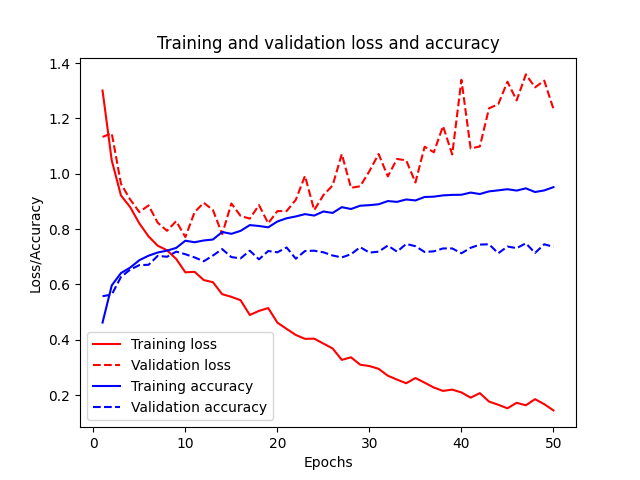
**6. Graphical User Interface (GUI)**  
After training the model, we implement a graphical user interface (GUI) to enable users to interact with the trained model. The GUI allows users to load images of flowers and obtain classification results along with confidence scores for each class. Users can visualize the predicted class probabilities and make informed decisions based on the model's outputs.



**7. Results and Conclusion**  
Following training and GUI implementation, we evaluate the model's performance on the validation dataset. We assess classification accuracy and examine the model's ability to generalize to unseen data. The GUI provides a user-friendly interface for visualizing classification results, enhancing the usability and accessibility of the model. In conclusion, our flower image classification system demonstrates the effectiveness of deep learning in accurately identifying different types of flowers.







**Graph Analysis**  
Graph 1:  
- Training Loss (Red Solid Line): The training loss decreases steadily, indicating that the model is learning the training data well.  
- Validation Loss (Red Dashed Line): The validation loss decreases initially but then starts to increase, showing signs of overfitting as the model becomes too specialized in the training data.  
- Training Accuracy (Blue Solid Line): The training accuracy increases steadily, confirming that the model is learning the training data.  
- Validation Accuracy (Blue Dashed Line): The validation accuracy initially increases but then starts to fluctuate and slightly decrease, further indicating overfitting.  
  
Graph 2:  
- Training Loss: The training loss decreases, which is expected as the model learns the training data.  
- Validation Loss: The validation loss decreases initially but then flattens or increases, showing early signs of overfitting.  
- Training Accuracy: The training accuracy increases.  
- Validation Accuracy: The validation accuracy increases initially but then starts to decrease or fluctuate, indicating overfitting.  
  
Graph 3:  
- Training Loss: The training loss decreases continuously.  
- Validation Loss: The validation loss fluctuates and does not decrease as much as the training loss, indicating overfitting.  
- Training Accuracy: The training accuracy increases.  
- Validation Accuracy: The validation accuracy shows fluctuations and does not improve significantly, indicating overfitting.

In summary, all three graphs show signs of overfitting, where the model performs well on training data but struggles to generalize to validation data. This suggests a need for further techniques to prevent overfitting, such as early stopping, data augmentation, or model regularization.