Assignment 4: Flower Image Classification Using CNNs

Student Name: Adir Serruya

ID: 316472455

Submission Date: 2/2/2025

Objective

The goal of this assignment is to classify images of flowers into 102 categories using two pre-trained CNNs: **YOLOv5** and **VGG19**. The models were fine-tuned on the Flowers102 dataset.

Dataset

- **Dataset Name**: Flowers102
- **Total Images**: 8,189
- **Classes**: 102
- **Splits**:
  - Training: 50%
  - Validation: 25%
  - Testing: 25%

Preprocessing

- **Resized images to 224x224.**
- **Normalized using ImageNet's mean and standard deviation.**
  - **Reason**: Ensures compatibility with the pretrained models, as they expect normalized inputs for optimal performance.
- **Label Encoding:**
  - Converted labels to 0-based indexing.
- **Data Loaders:**
  - Batched data efficiently, ensuring shuffling during training and ordered loading during validation/testing.

Model Architecture Details

1. YOLOv5 Architecture Modifications

Original Purpose

- YOLOv5 is originally designed for **object detection** tasks, where it identifies bounding boxes around objects and classifies them. However, for this project, a **preconfigured YOLOv5 classification model** was used, tailored specifically for classification rather than detection.

Key Architectural Features

Reference – Ultralytics documentation
https://docs.ultralytics.com/yolov5/tutorials/architecture_description/

1. Backbone: CSP-Darknet53

What It Is

- The backbone is responsible for extracting features from input images. YOLOv5 uses a **CSP-Darknet53** backbone, which is an improvement over the original Darknet backbone used in earlier YOLO versions.
- CSP stands for **Cross-Stage Partial Networks**, designed to improve learning by splitting the feature map into two parts:
    1. One part is processed through a series of convolutional layers.
    2. The other part bypasses those layers and is combined with the processed output.

2. Neck: SPPF and CSP-PAN

SPPF (Spatial Pyramid Pooling - Fast)

- The **SPP** module used in earlier YOLO versions was replaced with **SPPF**, which is faster and more efficient.
- It processes the feature maps through multiple pooling layers to capture features at different scales.

CSP-PAN (Path Aggregation Network with CSP)

- The Neck is responsible for combining features extracted from different stages of the backbone.
- The **CSP-PAN** fuses multi-scale features, ensuring that both high-level (coarse) and low-level (fine) details are utilized for prediction.
- PANet improves information flow and strengthens the model's ability to detect objects of varying sizes and scales.

Modifications for the Flowers102 Task

1. **Loading a Preconfigured Classification Model**:
    - A YOLOv5 model already adapted for classification was loaded, bypassing the need to modify detection-specific layers manually.
2. **Updating the Final Classification Layer**:
    - The final classification layer was replaced by a linear layer which outputs **102 classes** for the Flowers102 dataset.
3. **Fine-Tuning Strategy**:
    - Only the final linear layer of the YOLOv5 classification model was trained. The pretrained backbone layers were frozen to retain their feature extraction capabilities, while the new classification head was optimized to adapt to the Flowers102 dataset.

2. VGG19 Architecture Modifications

Original Purpose

- VGG19 is a convolutional neural network originally designed for **image classification** tasks, trained on the **ImageNet dataset** for 1,000 classes. Its deep architecture, consisting of 19 layers, emphasizes sequential convolutional layers with small kernel sizes to capture fine-grained spatial features.

Key Architectural Features

1. **Convolutional Layers**:
    - VGG19 uses 3×3 kernels with **ReLU activations** to extract spatial features at multiple levels.
    - These layers are stacked sequentially, allowing the network to build hierarchical feature representations.
2. **Fully Connected Layers**:
    - The convolutional layers are followed by dense, fully connected layers that flatten the extracted features and perform the classification.
    - The final fully connected layer maps the features to the number of classes in the dataset.
3. **Characteristics**:
    - VGG19 has a large number of parameters, making it computationally intensive but effective for feature extraction

Modifications for the Flowers102 Task

1. **Replacing the Fully Connected Layer**:
    - The original fully connected layer, designed for ImageNet's 1,000 classes, was replaced with a new **linear layer** that outputs **102 classes** for the Flowers102 dataset.
2. **Freezing Pretrained Layers**:

    - All convolutional layers were frozen to preserve the feature representations learned during ImageNet training.
3. **Fine-Tuning Strategy**:
    - During the training phase, only the final fully connected layer was trained to adapt to the new task.

Training Process

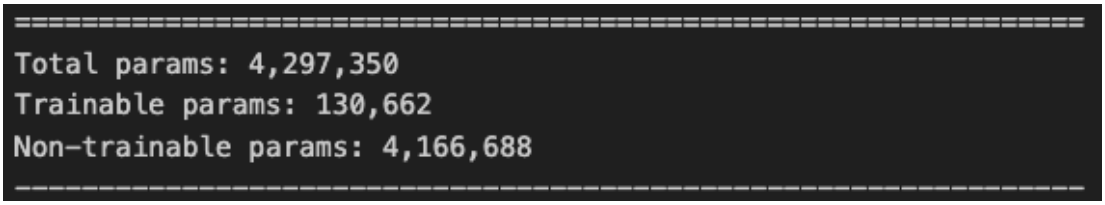
1. **Optimizer**: Adam
2. **Loss Function**: Cross-Entropy Loss
3. **Batch Size**: 128
4. **Early Stopping**:
    - Patience of 3 epochs to prevent overfitting.

Results

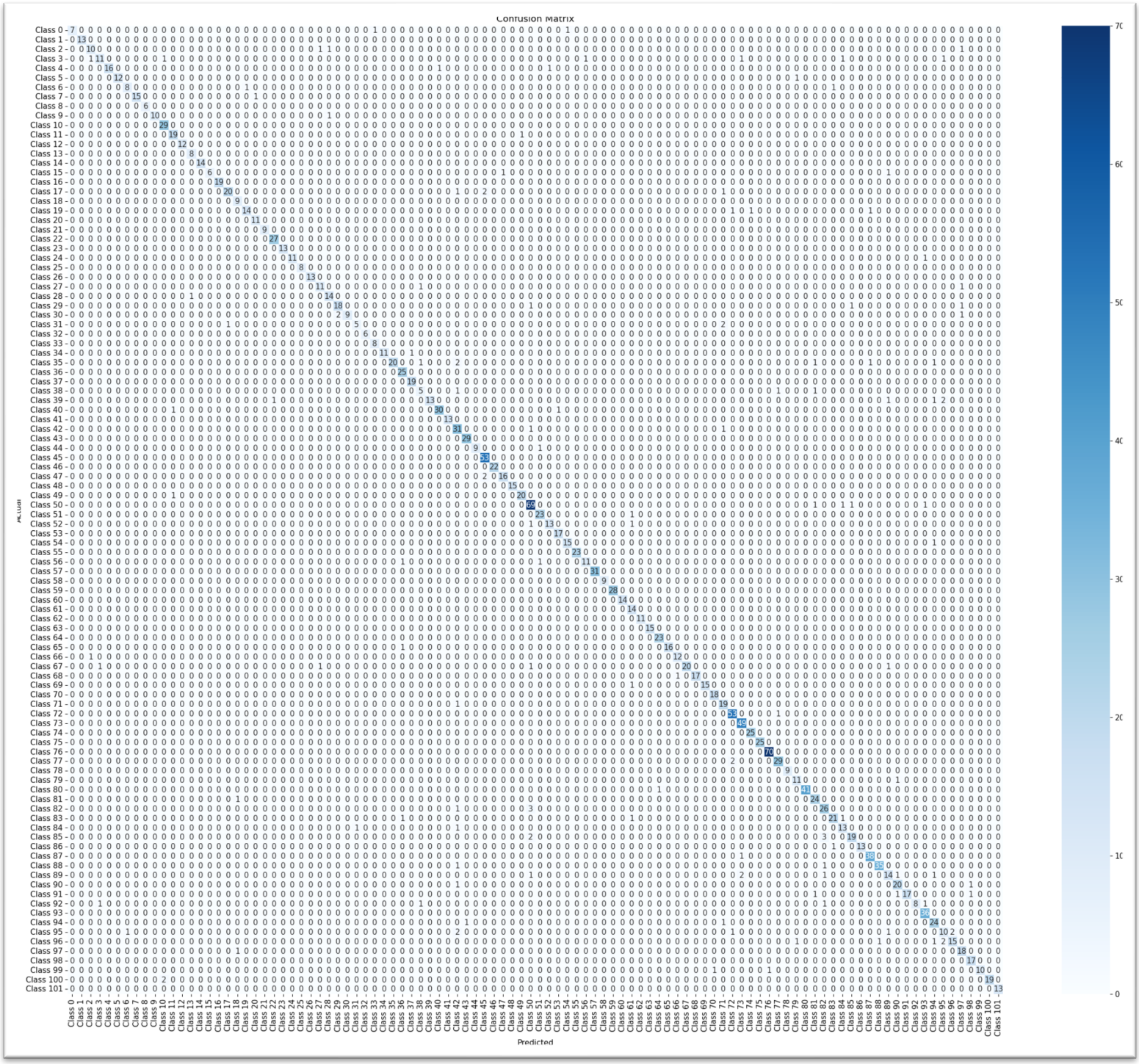
YOLOv5

- **Validation Accuracy**: 92.97%
- **Test Accuracy**: 93.02%
- **Training Time**: ~28 minutes

Trainable Parameters for YoloV5:



Confusion Matrix:



Classification Report:

	Precision	Recall	F1	Support
Macro AVG	0.94	0.92	0.92	2048
Weighted AV	0.93	0.93	0.93	2048

Error Examples:

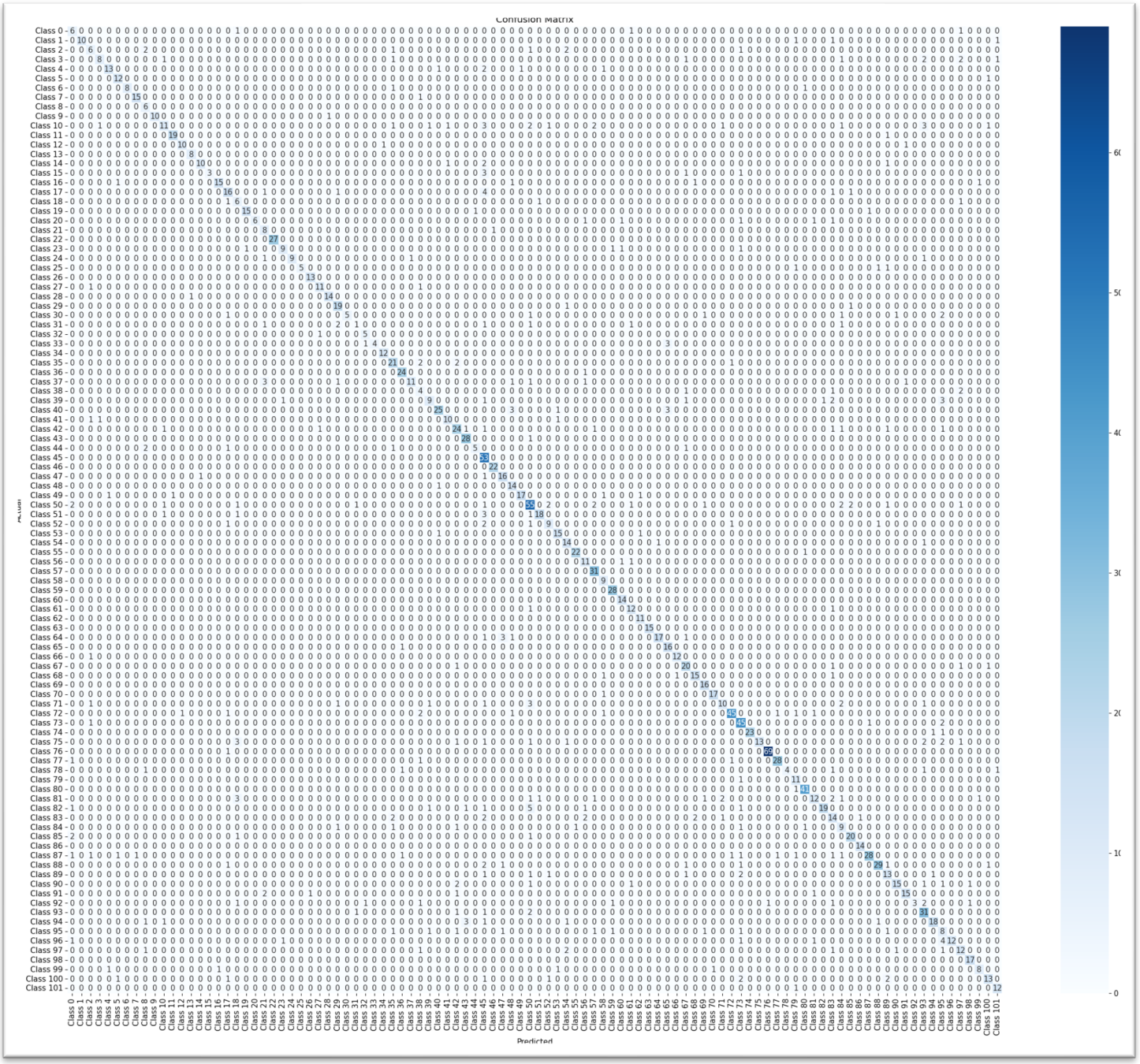
VGG19

- **Validation Accuracy**: 78.90%
- **Test Accuracy**: 79.39%
- **Training Time**: ~37 minutes

Trainable Parameters:

```
=================================================================
Total params: 139,988,134
Trainable params: 417,894
Non-trainable params: 139,570,240
_________________________________________________________________
```

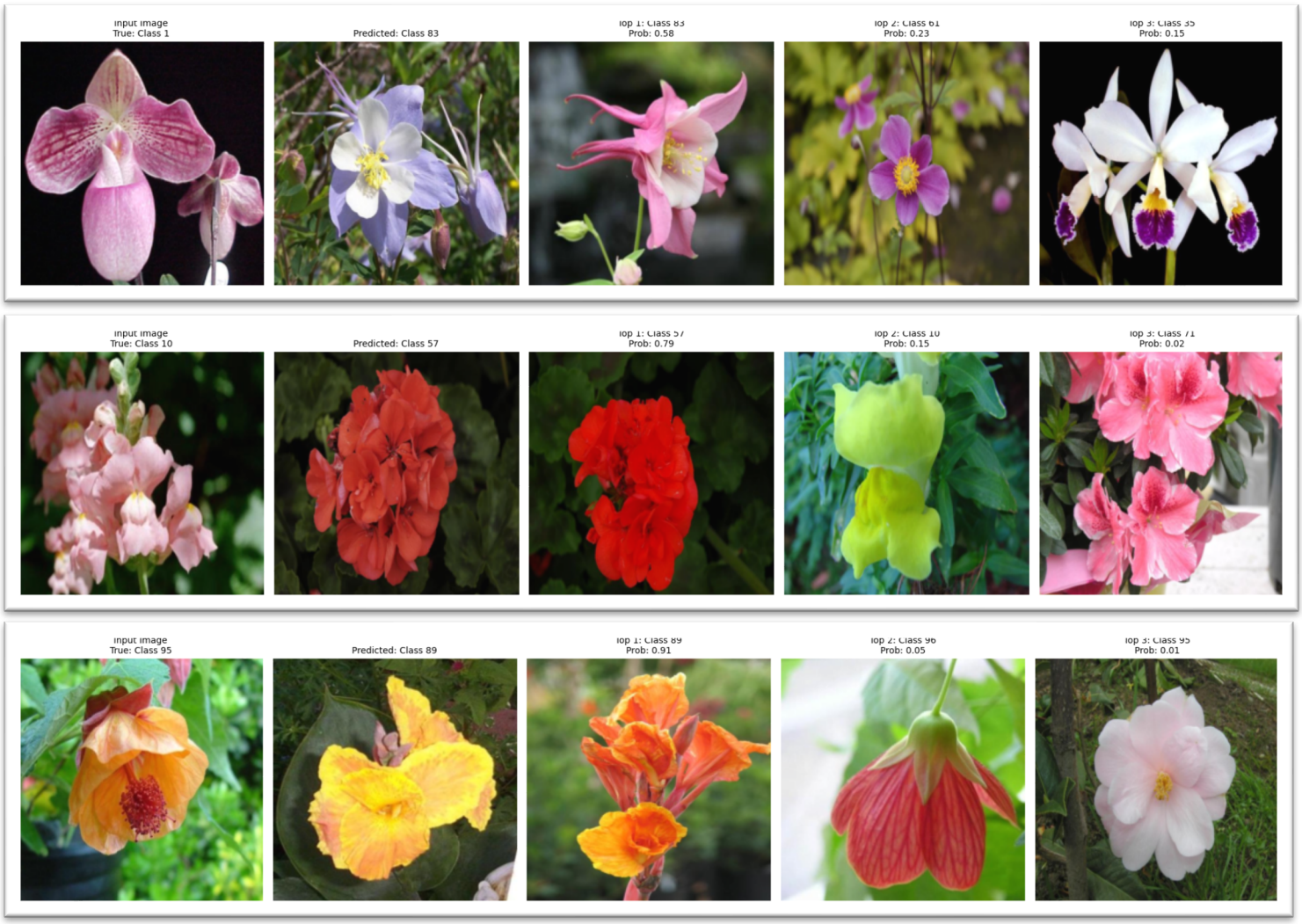
Confusion Matrix:



Classification Report:

	Precision	Recall	F1	Support
Macro AVG	**0.81**	**0.77**	**0.77**	**2048**
Weighted AV	**0.82**	**0.79**	**0.79**	**2048**

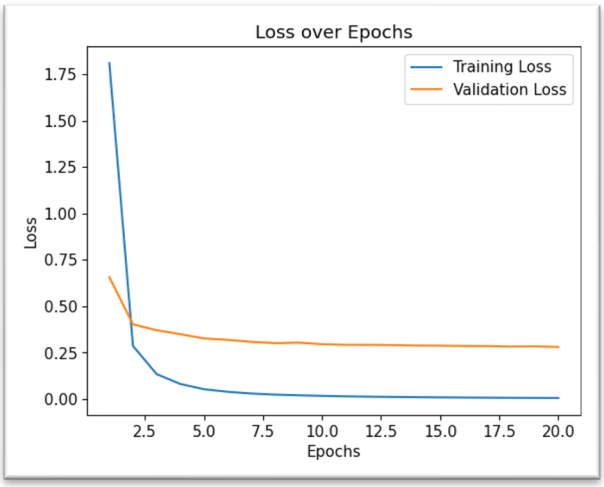
Error Examples:



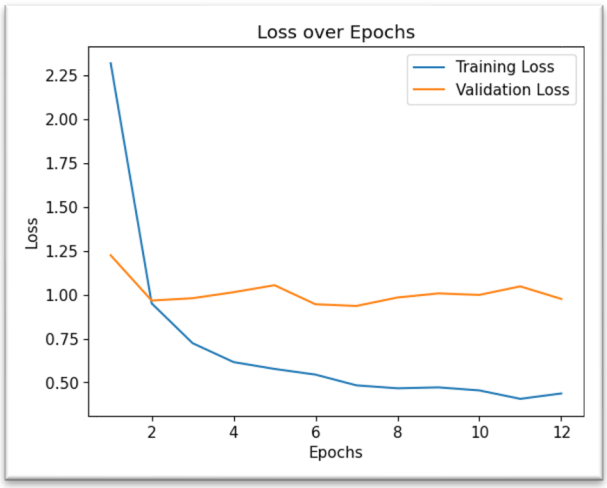
Plots

1. Loss Over Epochs

YOLOv5

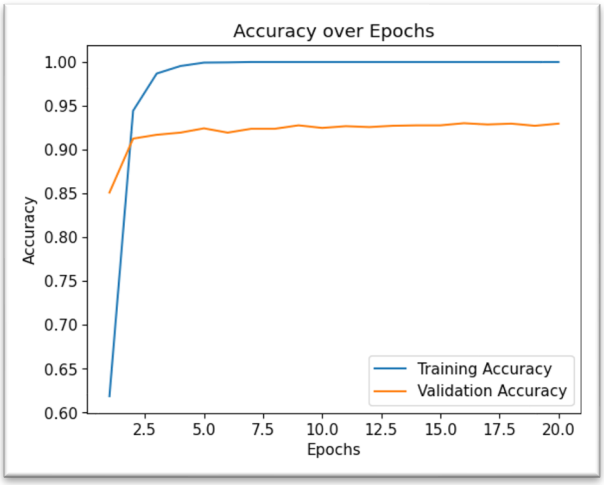


VGG19

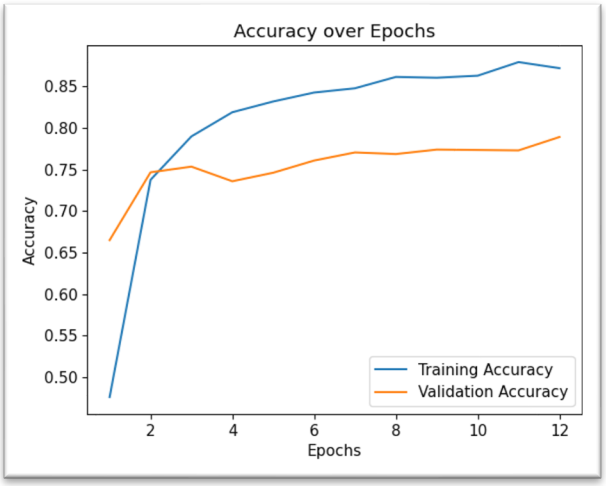


2. Accuracy Over Epochs

YOLOv5



VGG19



Improving VGG19 Performance with Data Augmentation:

The performance of YOLOv5 has been exceptional; however, the fine-tuned VGG19 model has shown significantly poorer results. Upon analyzing the errors and examples, it appears that the model relies heavily on color cues for its predictions. Given that flower classification often depends on shapes and background features, it became evident that the model's reliance on color was a major limitation. To address this, two strategies were proposed:

1. Incorporating data augmentations to reduce the model's dependency on color features, encouraging it to focus on other important characteristics such as shape and texture.
2. Extending the training process with increased patience to allow the model more time to converge.

Data Augmentations Applied:

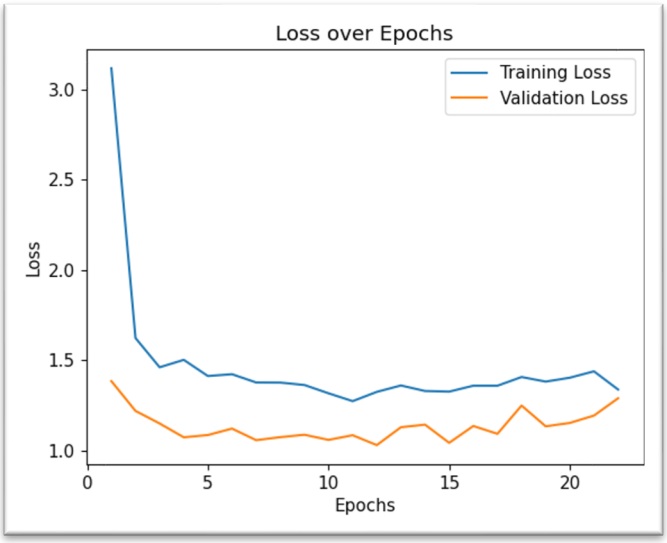
1. **Random Erasing:** Introduced random patches of noise or occlusions to the images, forcing the model to focus on other parts of the image rather than specific regions.
2. **Color Jitter (Hue Adjustment):** Applied subtle random variations in hue to make the model less reliant on specific color patterns.

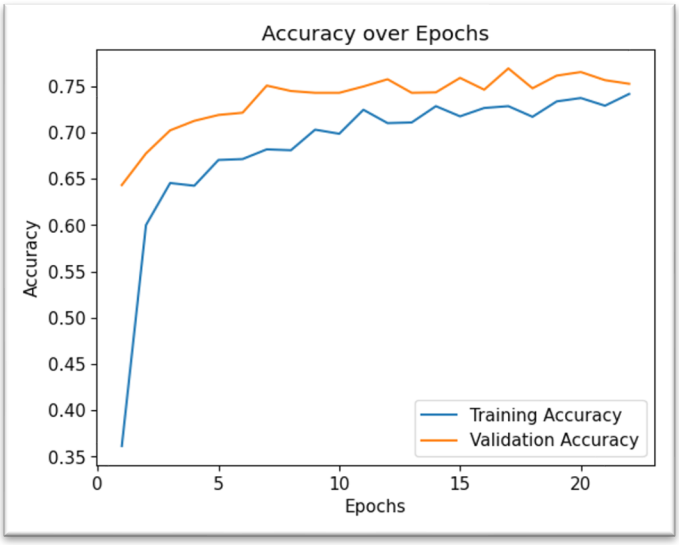
These augmentations aim to diversify the training data, helping the model generalize better to unseen data.

Example :

Results of VGG19 Performance Improvement Trials

1. **First Trial: Data Augmentation**
    - **Objective**: Reduce the model's over-reliance on color by applying data augmentations like Random Erasing and Color Jitter, and extend training with increased patience.
    - **Outcome**: While the augmentations successfully reduced overfitting (training and validation losses were closer), the overall performance remained unchanged. The test accuracy and validation accuracy did not improve, suggesting that the model's fundamental limitations were not addressed by these strategies.

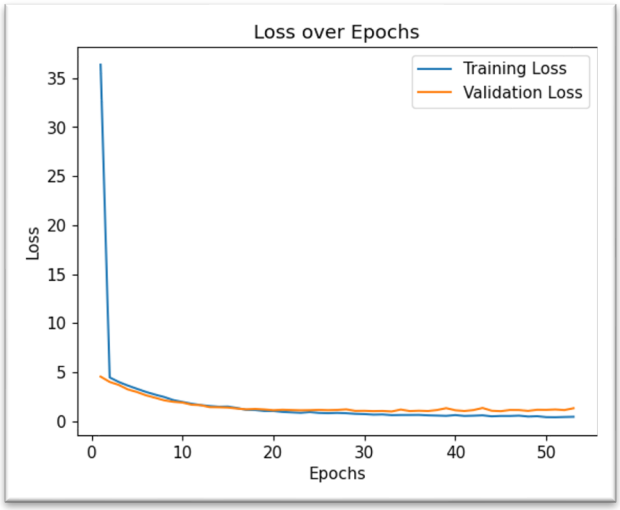


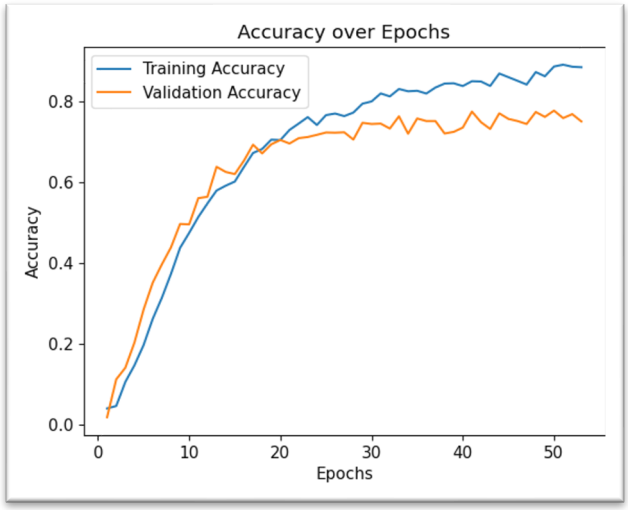


2. **Second Trial: Unfreezing Layers**
    - **Objective**: Unfreeze some of the deeper layers in the VGG19 architecture to allow the model to adapt its feature extraction process to the dataset, focusing on fine-grained details specific to flower classification.
    - **Outcome**: Similar to the first trial, unfreezing layers did not yield any significant improvement in test accuracy or overall performance. The results were comparable to the initial fine-tuned model, indicating that this approach was also insufficient.

In this trial we added trainable parameters due to the unfreezing of some deeper layers causing our trainable parameters count to increase to :

```
=================================================================
Total params: 139,988,134
Trainable params: 5,137,510
Non-trainable params: 134,850,624
_________________________________________________________________
```



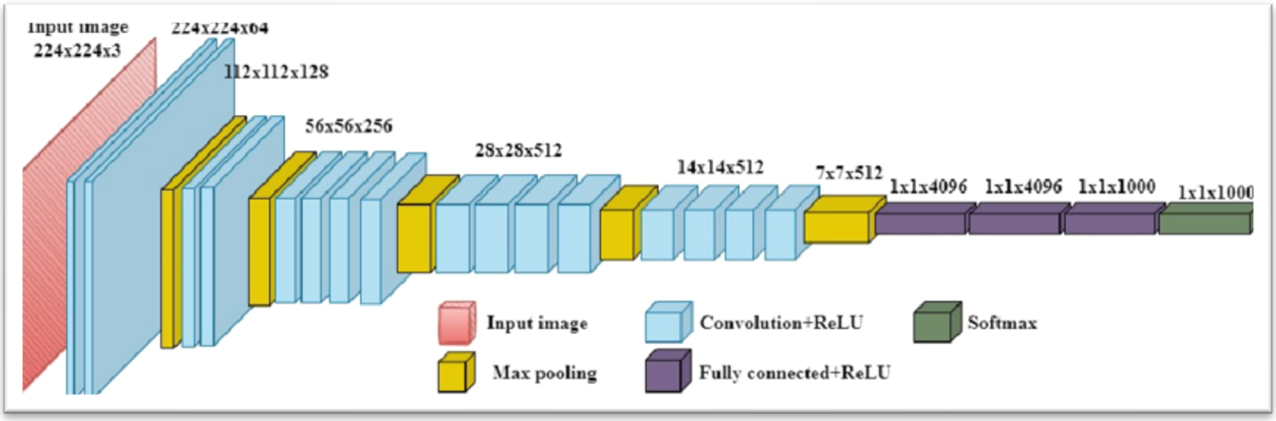


3. **Conclusion**:
    - These experiments suggest that augmentations and layer unfreezing alone may not be enough to address the model's limitations.
    - Future directions could involve Hyperparameter Optimization (HPO), exploring alternate architectures, or ensemble methods to achieve meaningful performance gains.

Final Conclusion:

- **YOLOv5** outperformed **VGG19**, achieving a test accuracy of **93.02%**.
- The results demonstrate the effectiveness of transfer learning for flower classification.
- Further improvements could be made by exploring additional pre-trained models or using ensemble methods, doing HPO, using data augmentation techniques and trying different architectures.

VGG 19 Model Architecture



Yolo V5 Model Architecture: