Assignment No. 4 - CNN for Flower Classification

Shaked Tayouri – 323866749

Noa Magrisso – 206934978

1. Introduction

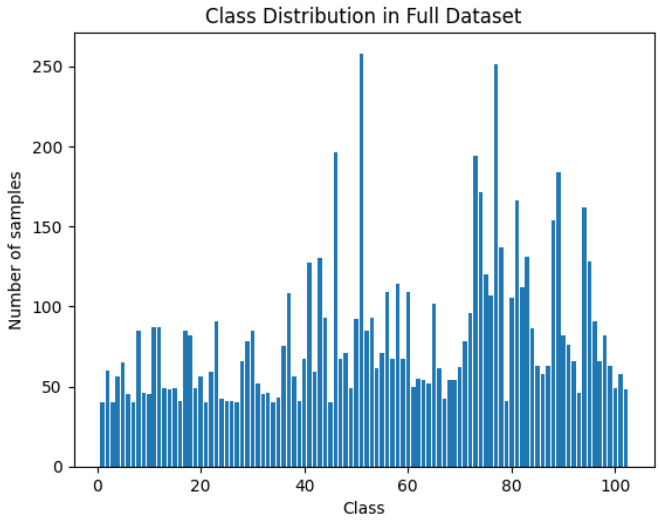
This report presents the implementation and results of using Convolutional Neural Networks (CNNs) for flower classification using transfer learning.

Three approaches were implemented and compared:

1. VGG19 (Classification)
2. YOLOv5 (Classification)
3. YOLOv5 (Object Detection)

2. Dataset

The primary dataset used in this study is the Oxford 102 Flower Dataset (Nilsback and Zisserman, 2008, https://www.robots.ox.ac.uk/~vgg/data/flowers/102/), which consists of 102 flower categories commonly found in the United Kingdom. The dataset contains 8,189 images in total, with each class having between 40 to 258 images.



The class imbalance in the dataset presents challenges such as model bias towards majority classes, underrepresentation of minority classes leading to poor predictions, and evaluation difficulties.

To mitigate the impact of this class imbalance, we employ **stratified sampling** during the train-test split. This approach ensures that each class is proportionally represented in the training, validating and testing sets, preserving the class distribution across all splits.

3. Data Preprocessing

Image Resizing:

- For VGG19, the images were resized to a fixed dimension of **224×224 pixels**.
- For YOLOv5, the images were resized to **320×320 pixels**.

In terms of *image augmentation*, for VGG19, several techniques were applied including random horizontal and vertical flip, random rotations, and adjustments to brightness and contrast. YOLOv5 utilizes its built-in augmentation pipeline, which automatically applies various transformations to the images.

For *normalization*, all images were scaled to a pixel range of [0, 1]. Additionally, the images underwent mean subtraction and standard deviation division using ImageNet statistics to standardize the data.

The dataset was then split into three sets: 50% for training, 25% for validation, and 25% for testing.

4. Transfer Learning Approach

- **VGG19** - The base VGG19 model was adapted for flower classification by replacing the fully connected layers with a custom classifier. The origin architecture consists of 16 convolutional layers and 5 MaxPooling2D layers (down-sampling operation):

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

Total params: 20,024,384 (76.39 MB)
Trainable params: 20,024,384 (76.39 MB)
Non-trainable params: 0 (0.00 B)

- :

  We freeze these layers and add fully connected layer, and a softmax output layer for 102 classes:

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg19 (Functional)	(None, 7, 7, 512)	20,024,384
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 4096)	102,764,544
dense_1 (Dense)	(None, 102)	417,894

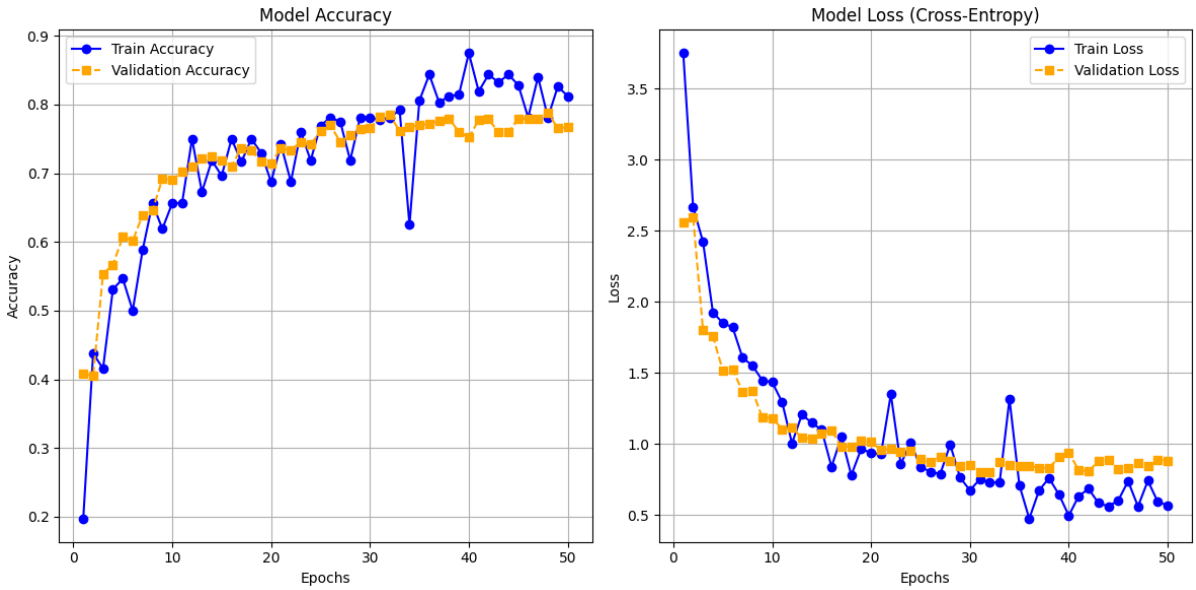
Total params: 123,206,822 (470.00 MB)
Trainable params: 103,182,438 (393.61 MB)
Non-trainable params: 20,024,384 (76.39 MB)

- **YOLOv5 (Classification)** - We adapted YOLOv5's architecture for classification by replacing the detection head with a classification head. The model uses YOLO's efficient feature extraction with a global average pooling layer and a dense output layer for 102 classes. Train on 20 epochs.
- **YOLOv5 (Object Detection)** - We also explored YOLOv5 for object detection, where the model predicts bounding boxes and class labels for flowers. This setup uses the YOLOv5 object detection pipeline, where images and labels are prepared for training and evaluation. Train on 10 epochs.

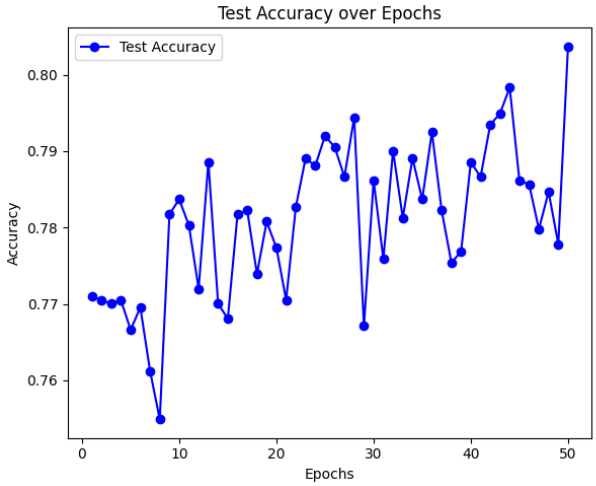
5. Results

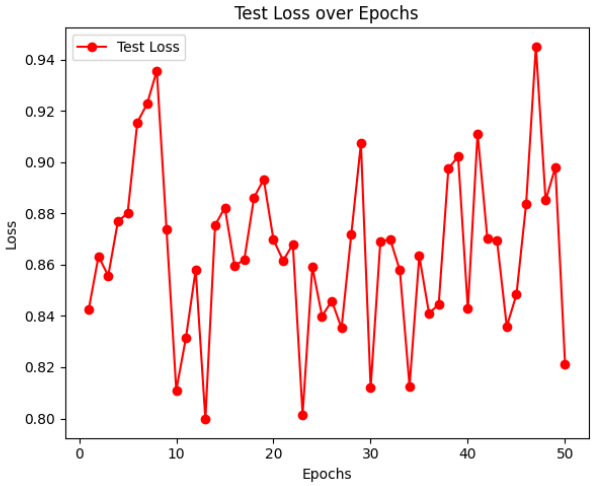
VGG19

The model learns the task effectively and stabilizes within approximately 30 epochs.



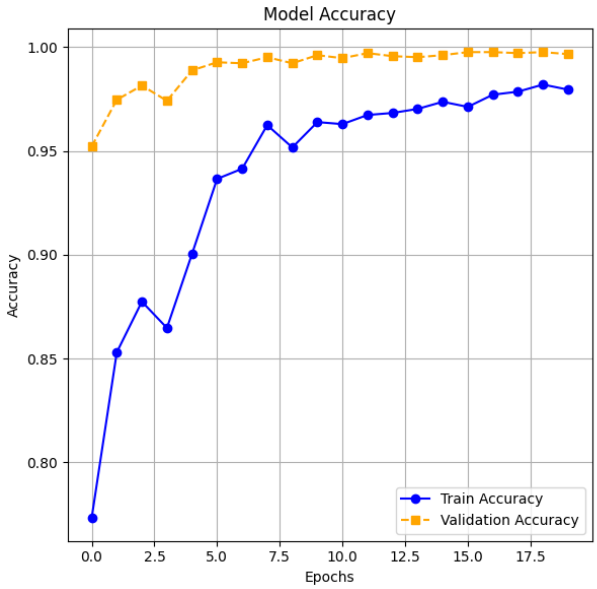
Test Accuracy: 76.7578%, Test Loss: 0.8355

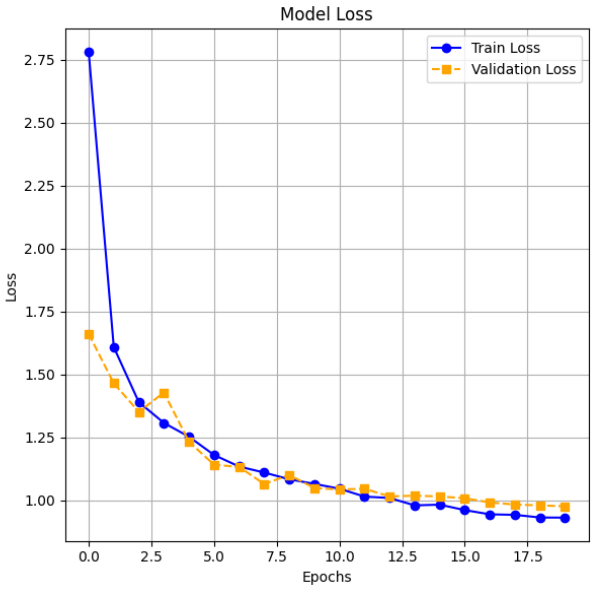




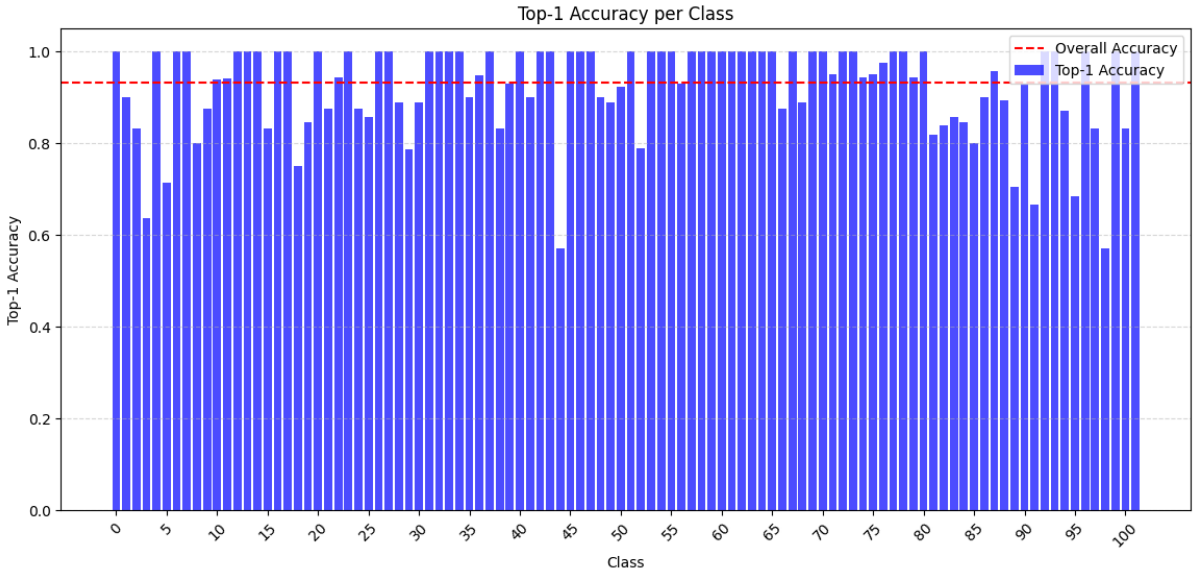
YOLOv5 (Classification)

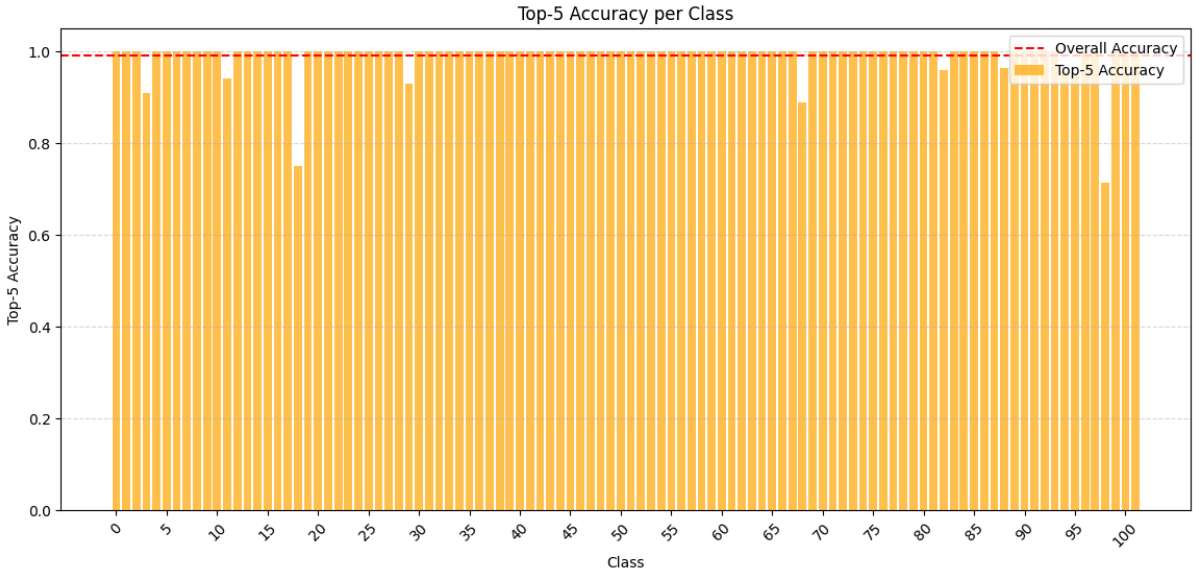
Yolo perform better than VGG19 with higher accuracy.





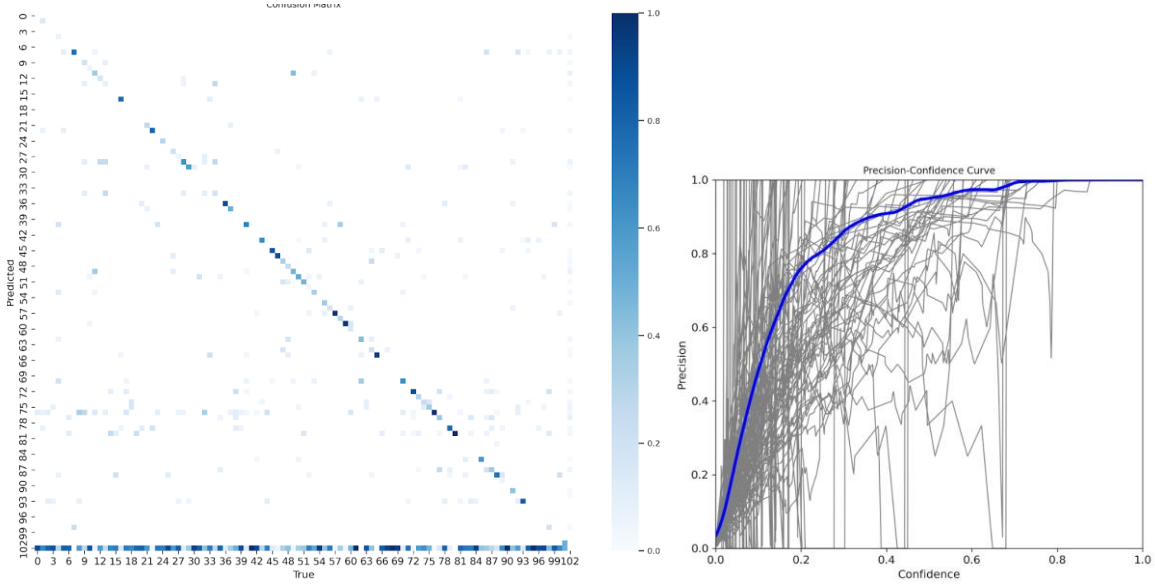
Since YOLO tests do not run on epochs, the following graph shows the accuracy performance for each class.

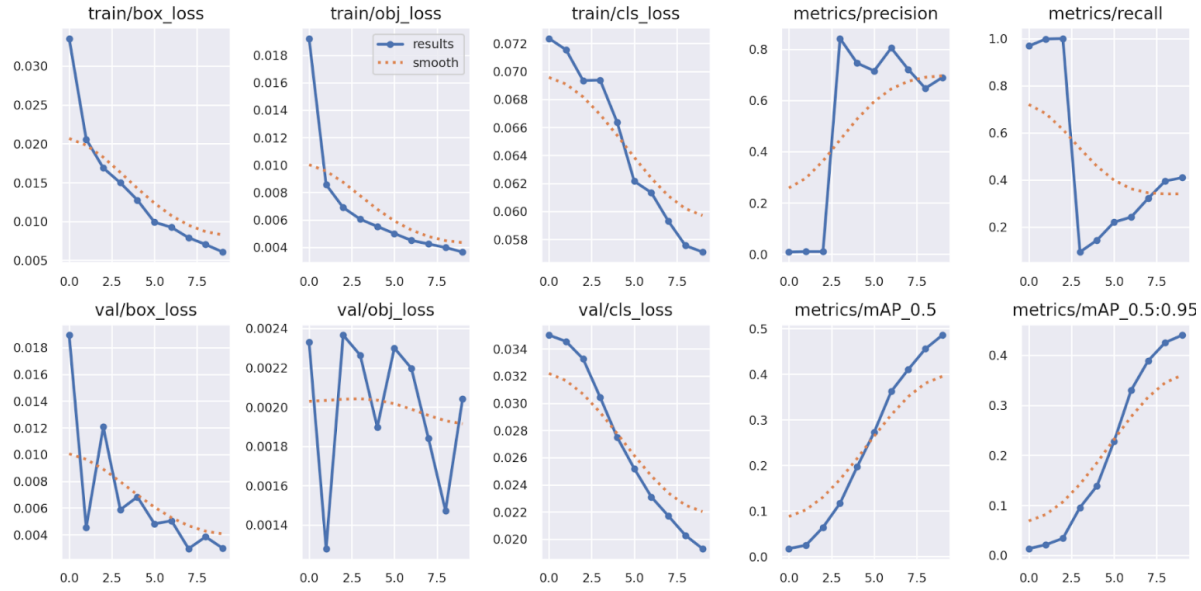

Top-1 Accuracy per Class
Overall Accuracy
Top-1 Accuracy
Top-1 Accuracy
Class
0.0
0.2
0.4
0.6
0.8
1.0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100


Top-5 Accuracy per Class
Overall Accuracy
Top-5 Accuracy
Top-5 Accuracy
Class
0.0
0.2
0.4
0.6
0.8
1.0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

YOLOv5 (Object Detection)

The metrics in the Object Detection is different than the classification. We show here the confusion matrix that shows that the model predict the label in high manner.





6. Conclusion

All transformer models we implemented exceeded 75% accuracy on the test set, showcasing impressive performance. We successfully classified 102 classes, significantly outperforming random predictions. The model also showed exceptional ability in accurately identifying flowers, demonstrating strong predictive intuition.