CUSTOM CNN-BASED FLOWER CLASSIFICATION AND DEEPLAB V3+ FLOWER SEGMENTATION IN MATLAB

Aditi Nair

School of Computer Science, University of Nottingham, UK

ABSTRACT

This paper addresses the successful application of convolutional neural networks of flower classification and segmentation. For classification, a custom CNN is built and trained from scratch to recognise the 17 different flower categories. For segmentation, a pretrained DeepLab v3+ model with a ResNet-18 backbone is used for distinguishing flowers from background. The classification achieved a high accuracy with validation performance improved though data augmentation, batch normalisation and dropout. The segmentation model is assessed using Mean Intersection Over-Union and Mean Boundary F1 score both of which showed strong performance. These results demonstrate that a custom CNN can also perform well with proper regularisation and transfer learning which is an effective strategy for achieving accurate pixel-wise predictions with limited labelled data

***Index Terms*—** CNN, Image Classification, Semantic Segmentation, Oxford Flower Dataset, DeepLab v3+

1. INTRODUCTION

Convolutional Neural Networks (CNNs) are widely used in computer vision to solve tasks such as image classification and semantic segmentation. CNNs are particularly effective because they can learn hierarchical patterns in images, making them ideal for a wide range of visual recognition problems.[1] In this project, CNNs were used in two applications: image classification using the Oxford 17 Flower Dataset, and pixel-level segmentation of daffodils using a DaffodilSeg dataset. The classification model is built and trained from scratch, which posed challenges due to the small dataset size and subtle visual differences between flower categories. Data augmentation methods such as image rotation and flipping were applied to improve model generalisation. For the segmentation task, a pretrained DeepLab v3+ model with a ResNet-18 backbone was adapted to identify flower regions within images. Given the small number of labelled segmentation images available, using a pretrained network improved accuracy while requiring less training data.

This report outlines how both models were implemented, trained, and tested in MATLAB. It details the dataset preparation, network design, training techniques, and evaluation procedures. The evaluation includes quantitative results such as classification accuracy and segmentation IoU, alongside qualitative visual outputs. Finally, it addresses the challenges of working with limited data and demonstrates that with appropriate techniques, CNNs can still perform effectively in such conditions.

2. METHODS

2.1 IMAGE CLASSIFICATION

The section outlines the methodology used for image classification with the target to assign each input image to one of pre-defined classes. A CNN is created from scratch where it involves dataset preparation, designing a CNN architecture, configuring training parameters and assessing model performance through accuracy metrics and class-wise analysis.

The Oxford 17 flowers dataset was initially a set of 1,360 labelled images [2] titled 'image_001.jpg' to 'image_1360.jpg' in one folder. The dataset was reorganised into 17 flower class subfolders to enable automatic label extraction using '*imageDatastore*'. The label for each image was obtained by taking the index number of the image and diving it by 80 as each class contains 80 images. A MATLAB script was written to separate images into the respective folders called *newsorted17flowers.* All images were resized to 256 x 256 x 3 pixels to provide standardised input size which promotes efficiency training with reduced computational requirements. An 80/20 split was done across the classes to achieve training, and validation sets for getting a balanced representation.

To improve the model's ability to generalise, data augmentation is performed on the training data. The augmentation techniques included random rotation (up to ± 10 degrees), horizontal translation (± 10 pixels), vertical translation (±15 pixels), scaling (between 90% and 105%), and horizontal flipping.

These processes added variability to the training data making the model more robust to real-world scenarios.[3] The augmented dataset was constructed using MATLAB's *imageDataAugmenter* and introduced into the training network using *augmentedImageDatastore*.

The CNN network comprised five convolutional blocks with growing depth (32 to 384) filters. Each block included convolutional layers, batch normalisation layer, ReLU layer and maxPooling2dLayer to reduce spatial dimensions while preserving salient features. This hierarchical design allows the network to extract low-level to high-level features effectively. Instead of using a traditional flattening layer, a global average pooling layer is used to reduce the number of trainable parameters while enhancing the generalisation by averaging spatial features. The head of the classification consisted of two fully connected layers (with 512 and 256 units) each followed by batch normalisation, ReLU layer and a dropout later (rate of 0.2) to reduce overfitting. The final layer is a softmax layer for 17-class output. The architecture aimed to strike a balance between representational power and regularisation to avoid overfitting.

The network is trained with 'adam' optimiser which is selected due to its efficiency and adaptive learning rates compared to traditional methods like SGDM, making it well suited for complex image data. An initial learning rate of 0.0003 was applied, with a piecewise schedule halving the rate every five epochs to promote stable convergence. To reduce overfitting, L2regularisation is also used to further penalise model complexity. Then the training is performed for 60 epochs with a mini batch size of 32 and validation loss is monitored throughout. Early stopping is also implemented to stop the training if the validation loss shows no improvement after five consecutive evaluations through *ValidationPatience* parameter

Upon completion, the trained model is saved as '*classnet.mat'* using MATLAB's *'save'* function. The model performance is assessed on the overall accuracy on the validation set which is supplemented by per-class accuracy analysis and a confusion matrix. This gives a detailed view of class-wise performance and helps to identify classes with higher misclassification rates.

2.2 IMAGE SEGMENTATION

The section outlines the methodology for training and evaluating a semantic network capable of identifying a daffodil flower region in images. The process involved preparing the dataset, converting label maps for binary segmentation, configuring a DeepLab v3+ network architecture, training the model using optimised parameters and assessing its performance through pixel-level metrics and qualitative visual outputs. The segmentation task utilises the DaffodilSeg dataset that consists of 71 RGB images per pixel distinguishing flower, background, sky and leaves. The images are in the *daffodilSeg/imagesRsz256* folder, while the labels are stored in the *daffodilSeg/LabelsRsz256*. Since only flower segmentation is needed, a custom label conversion function (*convertClassLabel*) is used which converts all pixels with label 1 (flower) to class 1 and everything else to class 0 (background). The image dataset is loaded using MATLAB's *imageDatastore* while the labels are accessed using *pixelLabelDatastore.* Both label masks and input images are resized to 256 x 256 x 3 pixels to have consistent input sizes to reduce computational complexity during training. The dataset is then randomly divided into an 80% training set and 20% validation set ratio using *randperm,* which provides a balanced distribution of the data across the sets. The training and validation data are stored in separate i*mageDatastore* and *pixelLabelDatastore* which are then combined using 'combine' function to form the training and validation datasets.

The segmentation model is based on the DeepLab v3+ network with a ResNet- 18 pertained backbone which is selected since its lightweight, computationally efficient design, offering faster convergence with acceptable accuracy on small datasets, in contrast to deeper architectures like ResNet-50.[4] This choice of architecture has been proven effective in other segmentation tasks such as post-disaster imagery segmentation where DeepLab v3+ was used for aerial segmentation of flooded regions [5]. The size of the input layer is modified to the size of the resized images (256 x 256 x 3) and the model is designed to classify each of the pixels into one of the two classes: 'background' and 'flower'. The architecture is constructed using the *deeplabv3plusLayers* function which specifies the image size and the number of classes.

The model is trained using 'adam' optimiser with an initial learning rate of 1e-4, for 50 epochs, and a mini-batch size of 4. These parameters are used since they offer training stability as well as allow the model to converge properly. Training is also configured to shuffle data at every epoch with progress being checked through training plots. The validation data is also provided during training to evaluate the model performance after each epoch. The training set is combined using the *'combine'* function to link the image and pixel label datastores likewise for the validation set. A local function, *convertClassLabel* function is defined to process the images. This function converts the pixel values in the label images into a binary mask where pixels corresponding to the flower class are set to 1, and all other pixels are set to 0. This conversion ensures that the labels are in the correct format for the binary classification task. After training, the final network is saved as *'segmentnet.mat'* using MATLAB's *'save'* function. The network is evaluated on the validation set using MATLAB's *evaluateSemanticSegmentation* function. The model's performance is assessed based on the Mean Intersection over Union (MIoU), Global accuracy, mean per-class accuracy and Per-class IoU. These measures provide a comprehensive assessment of the model's ability to distinguish flower regions pixel-wise. To visually inspect the model's performance, the predicted labels for six validation images are overlayed onto the original images. This is done using the *labeloverlay* function which combines the predicted segmentation mask with the original image.

3. EVALUATION AND DISCUSSION

3.1 IMAGE CLASSIFICATION

The performance of the classification network was evaluated on a validation set made up of 20% of the images from each flower class which were not used during the training. This helps measure how well the model performs on the unseen data from all categories. The overall validation accuracy achieved was 74.63% as reported by MATLAB's training (see Fig 1). This is reasonably a good performance given the 17-flower classification problem and the relatively small size of the dataset.

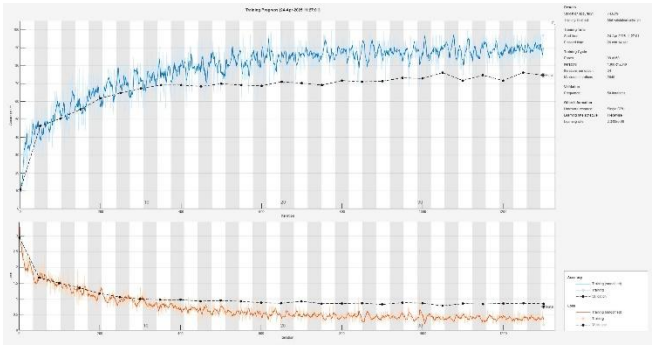

Figure 1. Training and validation accuracy and loss curves illustrating converging behaviour over training epochs

The training can be observed (see Fig 1) through the accuracy and loss curves. The training showed a stable convergence trend over 60 epochs. The validation accuracy increased steadily and reached its best performance around 40 epochs after which training stopped early. Since the training and the validation loss both reduced together and their results stayed close, it shows the model learned general patterns well.

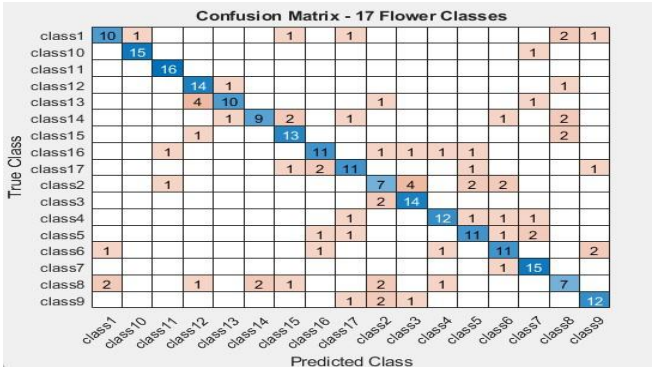

Figure 2. Confusion Matrix displaying results for 17 flower classes on the validation set

The confusion matrix (Fig. 2) provides class-specific performance insights by visualising the distribution of predicted versus true labels. The matrix indicates most of the classes have high diagonal values which represent the correct predictions. However, there is some confusion between visually similar classes like class 5 (Crocus) and class 6 (Iris), between classes class 1 (Daffodil) and class 13 (Dandelion). These misclassifications suggest that certain flower categories share overlapping visual features such as colour, and shape which makes it difficult to distinguish them, particularly under the influence of data augmentation. (see Fig 3) To accurately quantify this, the per-class accuracy (see Figure 4) is computed from the confusion matrix. There are certain classes like class 2, class 3 and class 15 have high classification accuracy (>= 95%) likely due to distinctive features to distinguish them whereas classes like class 6, class 10, and class 16 performed worse saying that those classes may have fewer discriminative features or are more vulnerable to augmentation noise. The lowest per-class accuracy is around 45% which indicates room for improvement.

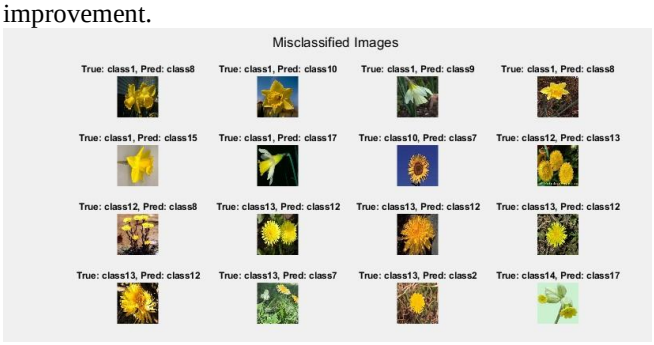

Figure 3. Misclassified images

This was likely due to the use of data augmentation and dropout which reduced overfitting. Overall, the classification model performed well in most classes and showcased good behaviour. Enhancing the dataset through class rebalancing or targeted augmentation for low-performing classes can help reduce the observed performance disparities. Additionally, adopting advanced data augmentation and class balancing techniques can significantly improve classification accuracy.

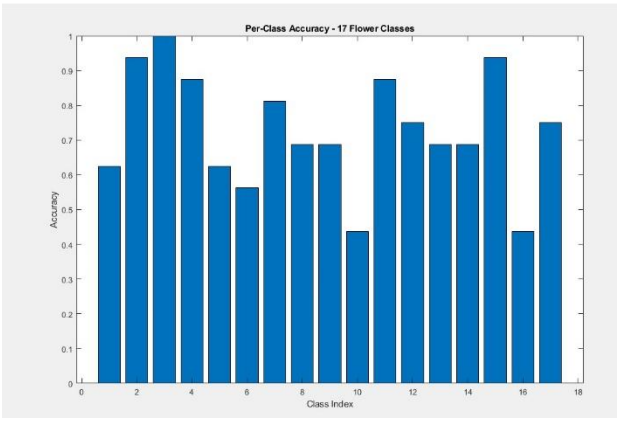

Figure 4. Per-class classification accuracy for the 17 flower categories highlighting variation in model performance across classes.

3.2 IMAGE SEGMENTATION

The performance of the segmentation model is also compared against a validation set consisting of 20% of the daffodil

images. Despite having a limited set of data, the DeepLab v3+ network with a ResNet-18 demonstrated excellent performance across all evaluation metrics. The Mean Intersection over Union (IoU) achieved a rate of 0.94593 demonstrating the mode's ability to distinguish between flower and background regions with high precision. The Global accuracy reached 97.82% and the Mean Accuracy was 97.09% (see Fig 5) which indicates the model is capable of consistently predicting the pixels correctly across both the classes.

GlobalAccuracy	MeanAccuracy	MeanIoU	WeightedIoU	MeanBFScore
0.9782	0.97089	0.94593	0.95753	0.86964

Mean IoU: 0.94593
Per-class IoU:

	Accuracy	IoU	MeanBFScore
background	0.98654	0.97076	0.92084
flower	0.95524	0.9211	0.81844

Figure 5. overall segmentation performance metrics including Global accuracy, Mean Accuracy, Mean IoU and Mean Boundary F-Score.

The per-class indicators also show a good segmentation quality for both classes. (see Fig 5) The background class achieved an IoU of 0.97076 and an accuracy of 98.65%, while the flower class reached an IoU of 0.9211 and an accuracy of 95.52%. The high IoU and Boundary score suggest that the network is highly confident and precise in distinguishing non-flower regions even near object boundaries. The Mean Boundary F1 score (MeanBFScore) further highlights the segmentation quality scoring a 0.86964 overall. While the background has a higher boundary score (0.92084), the flower class saw a lower value (0.81844), this indicates the difficulty in capturing petal boundaries with complete precision. This is typical for thin, irregular shapes like flower edges, where pixel-wise accuracy is more challenging.

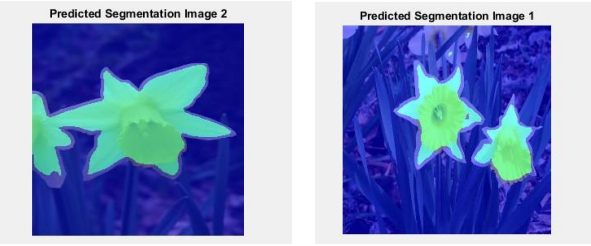

Figure 6. Example Segmentation result showing predicted flower regions (in green) overlaid on a validation image.

Visually, the segmentation produced is nicely preserved along the boundaries as seen from the (see Fig 6). The predicted flower mask significantly overlays the actual flower area, even in images with occluded or cluttered backgrounds or partial occlusion. This indicates that the DeepLab v3+ model with ResNet-18 has captured the features that are typical of the daffodil class.

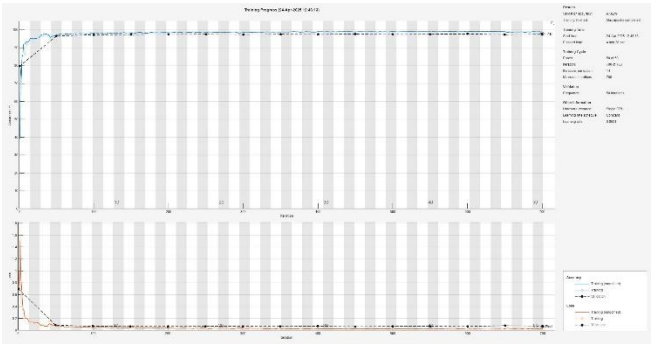

Figure 7. Training and Validation accuracy and loss curve for the segmentation model, showing consistent convergence.

The training curves as shown in (see Fig 7) indicate that the model learned quickly and performed consistently. Both the training and the validation accuracy increased sharply at the start and stayed high showing the model understood the data from the beginning. The validation loss also dropped quickly and stayed low which means that the model did not overfit to the training data. In conclusion, the segmentation model performs quantitively and qualitatively well at the task of segmenting daffodils. It generalises well despite having limited training data. The model can be further improved by incorporating more sophisticated model enhancements like edge-aware loss functions like dice loss or boundary loss or training with higher resolution masks which can lead to further improvements in accuracy and robustness.

5. CONCLUSION

This project developed and evaluated convolutional neural networks for flower classification and semantic segmentation using the Oxford Flower and DaffodilSeg datasets. The custom classification model achieved a 74.63% validation accuracy, effectively generalising despite limited data and visually similar classes. Data augmentation, batch normalisation, and dropout were key in improving robustness. The DeepLab v3+ segmentation model with a ResNet-18 backbone achieved strong performance, with a Mean IoU of 0.95 and global accuracy of 97.82%. While overall segmentation quality was high, capturing fine petal boundaries remained challenging due to class imbalance and label noise. To further improve performance, the use of transfer learning through the fine-tuning of an already pre-trained model like ResNet-50 which can offer more robust feature extraction through fine visual variation. Additionally, incorporating advanced augmentation techniques like Mixup or CutMix can be employed to introduce intra-class variation and help the model learn more discriminative features, particularly for classes frequently misclassified like daffodil vs dandelion. For segmentation, improvements can include edge-aware loss functions, higher-resolution masks, and larger or synthetic datasets to improve generalisation and boundary accuracy. These steps are expected to offer more consistent accuracy over all classes and make the model stronger under different visual conditions.

6. REFERENCES

[1] Gurnani, A. and Mavani, V. (2017) 'Flower Categorization using Deep Convolutional Neural Networks,' arXiv (Cornell University) [Preprint]. https://doi.org/10.48550/arxiv.1708.03763.

[2] Nilsback, M.E. and Zisserman, A., 2008, December. Automated flower classification over a large number of classes. In 2008 Sixth Indian conference on computer vision, graphics & image processing (pp. 722-729). IEEE.

[3] Wang, J. and Perez, L., 2017. The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit, 11(2017), pp.1-8.

[4] Gupta, A. et al. (2025) 'Comparative analysis of RESNET-18 and RESNET-50 architectures for pneumonia detection in medical imaging,' in Lecture notes in networks and systems, pp. 355– 365. https://doi.org/10.1007/978-981-97-7178-3_31

[5] Sundaresan, A.A. and Solomon, A.A. (2024) 'post-disaster flooded region segmentation using DeepLabV3+ and unmanned aerial system imagery,' Natural Hazards Research [Preprint]. https://doi.org/10.1016/j.nhres.2024.12.003.