Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Final Project

Floranet

Authors:
Andrea Moleri - 902011 - a.moleri@campus.unimib.it
Filippo Armani - 865939 - f.armani1@campus.unimib.it

January 23, 2025

Abstract

This report presents Floranet, a deep learning-based approach aimed at developing models capable of correctly identifying flower species from images while minimizing classification errors. The investigation focused on leveraging transfer learning techniques with three base architectures: VGG16, DenseNet121, and InceptionV3. Various methodologies, including full and partial freezing of network layers and the application of data augmentation were tested to evaluate their impact on model performance. The result is a suite of models capable of solving the classification problem with high test accuracy and low error propensity.

1 Introduction

The classification of images has become a cornerstone task in the field of machine learning and deep learning due to its wide range of applications, from medical diagnostics to automated systems in agriculture. This project, titled Floranet, focuses on the classification of flower species using the Oxford Flower Dataset, a dataset curated by Maria-Elena Nilsback and Andrew Zisserman. The dataset comprises 102 flower species, with the number of images per species ranging from 40 to 258. These images present significant challenges due to variations in scale, pose, lighting conditions, and intra-class diversity, as well as inter-class similarity.

The problem addressed by this project is the accurate classification of flower images into their respective species, a task that requires robust models capable of handling the inherent complexity of the dataset. To tackle this problem, transfer learning was used, a technique that leverages pre-trained deep learning models to improve the efficiency and accuracy of new tasks. Specifically, three state-of-the-art architectures were utilized. The aim was to investigate the strengths and limitations of each one, offering insights into the performance of different architectures when applied to complex classification tasks. The subsequent sections will delve into the detailed methodology, experimental results, and an analysis of the findings.

2 Datasets

The dataset used in this project is the Oxford Flower Dataset, introduced in 2008 by Maria-Elena Nilsback and Andrew Zisserman. This dataset is designed for image classification tasks and consists of 102 categories of flowers commonly found in the United Kingdom. The dataset includes three primary components: image files (high-resolution flower images across all 102 categories), segmentation masks (masks that define flower regions in the images), and labels (class labels for each image, provided in MATLAB format).

To prepare the dataset, the required files were downloaded, decompressed, and organized into meaningful directories. A Python script was implemented to automatically handle this process, ensuring that any missing files were downloaded and verified. Class labels were extracted from the MATLAB file (‘imagelabels.mat‘) and adjusted for Python's zero-based indexing. To facilitate analysis, a mapping between class indices and their corresponding flower names was created using the official dataset documentation.

The image files were matched to their labels and organized into a structured Pandas DataFrame, where each row contains the image file name and the associated flower name. While this step is not strictly necessary for training machine learning models, it aids in exploratory data analysis and ensures that the dataset is correctly understood and annotated. The dataset's original benchmark, as described in its publication, utilized hand-crafted features for classification. This project, however, leverages CNN-based approaches, exploring the advantages of transfer learning with pre-trained deep learning architectures.

The dataset poses significant challenges due to its inherent imbalance: the most populated class contains 258 images (Petunia), while the least populated class includes only 40 images (e.g., Pink Primrose). A quantitative analysis [Fig.1] reveals that the average number of images per class is 80.28, with substantial variance indicating the presence of class imbalance. To address this issue, a data augmentation strategy was implemented to ensure a more uniform distribution across classes. The augmentation included transformations such as rotation, width shift, height shift, zoom, and flips.

Qualitative insights into the dataset were gained through visualizations. A bar plot [Fig.2] of class distributions highlights the imbalance, while a

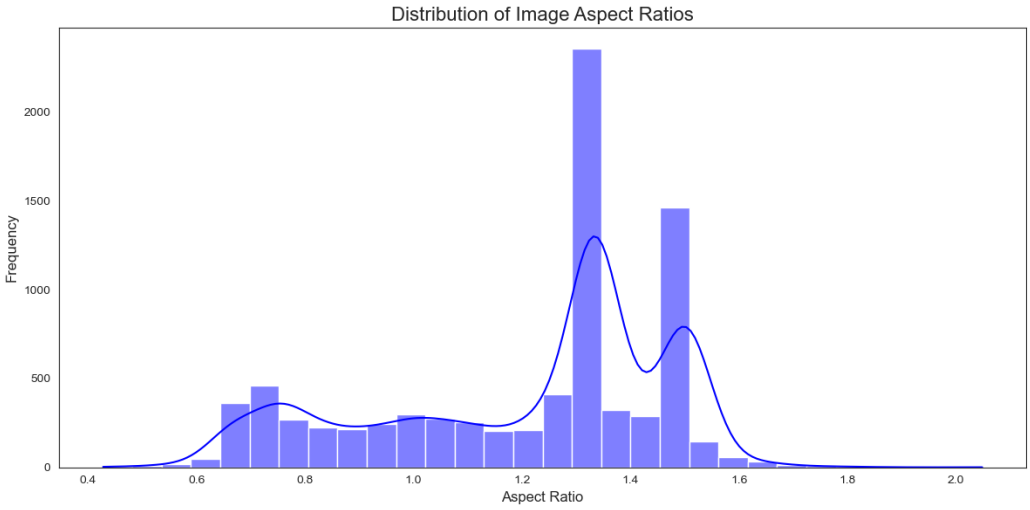

Figure 1: Distribution of Image Aspect Ratios

subset of images from the top five most populated classes [Fig.3] showcases intra-class variability, pose variations, and image quality. For instance, the Petunia class exhibits notable pose diversity, while inter-class similarity is observed between Wallflower and Watercress. Additionally, intra-class variation is evident in the Water Lily class, where samples within the same category differ significantly in appearance. Such variability may increase the complexity of the classification task, necessitating robust feature extraction techniques.

To further understand the dataset's characteristics, the dimensions of all images were analyzed, yielding an average resolution of approximately 630×534 pixels. The aspect ratio distribution centers around 1.2, with minor outliers. This uniformity in image dimensions simplifies preprocessing requirements but necessitates consistent resizing for compatibility with convolutional neural networks. Further analysis can be found on the notebook, for example a comparison is made between flowers extracted from 10 randomly selected classes, and flowers extracted from a specific class. This visualization was made to underscore the diversity within the dataset, and to highlight the consistency of samples for a particular category at the same time.

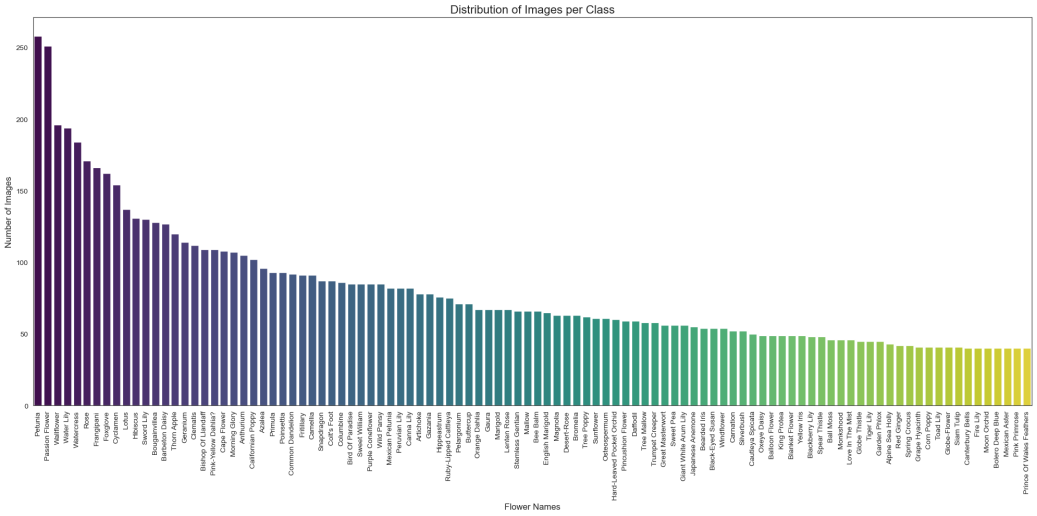

Figure 2: Distribution of Images per Class

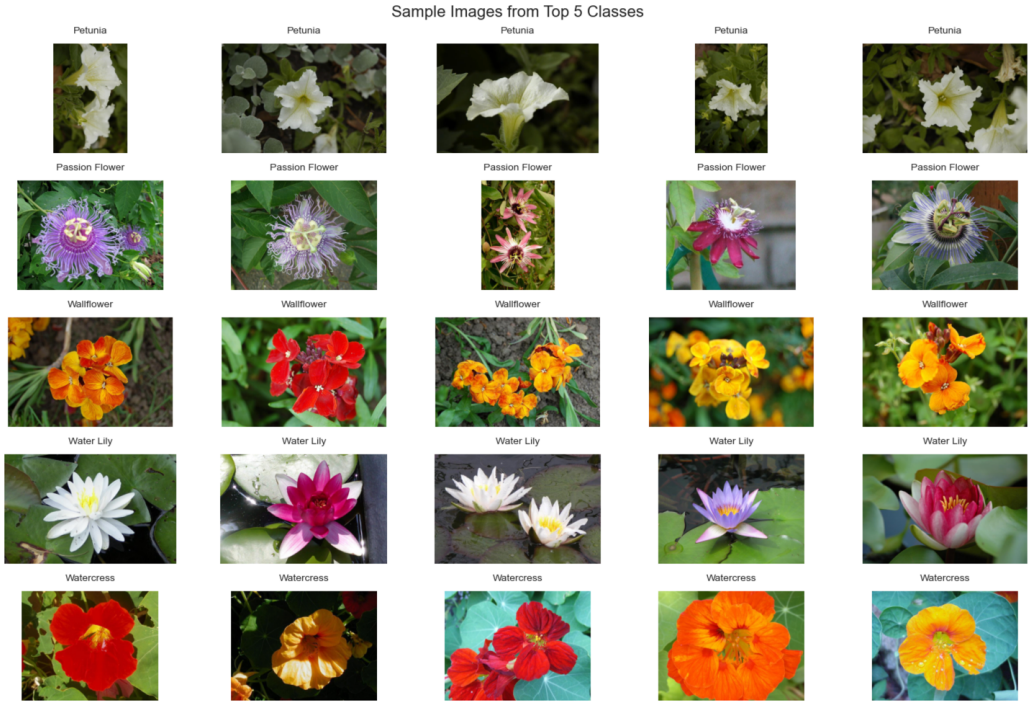

Figure 3: Sample Images from Top 5 Classes

3 The Methodological Approach

3.1 Data Preprocessing

The first step in the methodology involved preparing the dataset. The dataset was prepared by splitting it into training, validation, and test sets (70%, 15%, and 15%, respectively) using a stratified method to maintain class balance. One-Hot Encoding was applied to the target labels to treat each class independently, and the preprocessed data was saved in CSV files for further processing.

3.2 Model Architecture Selection

For the classification task, three advanced convolutional neural network architectures were employed: VGG16, DenseNet121, and InceptionV3. These models were selected due to their proven performance in image classification tasks and their compatibility with transfer learning. Below, we elaborate on the key features and design principles of each architecture, along with the motivations for their use in the context of the Oxford Flower Dataset.

VGG16

VGG16, introduced by Simonyan and Zisserman, is characterized by its simplicity and depth. The architecture [Fig.4] employs a sequence of small 3×3 convolutional filters, stacked in increasing depth, followed by max-pooling layers for spatial dimensionality reduction. This design allows the model to capture complex hierarchical features while maintaining manageable computational complexity. VGG16's consistent layer structure and relatively shallow gradient paths make it a robust choice for transfer learning.

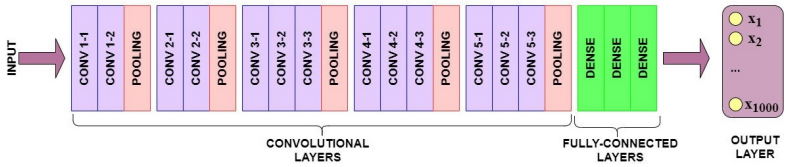

Figure 4: VGG16 Architecture

InceptionV3

InceptionV3, a refined version of the Inception architecture introduced by researchers at Google, presents factorized convolutions, batch normalization, and auxiliary classifiers to improve both training efficiency and accuracy. The architecture [Fig.5] employs inception modules, which capture multiscale spatial features by performing convolutions of varying kernel sizes in parallel. This multi-scale approach is ideal for the Oxford Flower Dataset, as flowers exhibit diverse shapes and sizes.

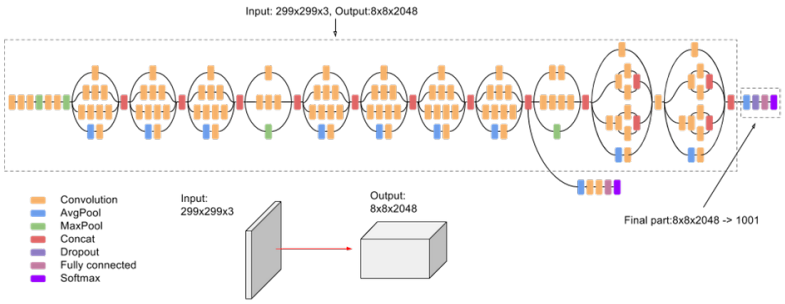

Figure 5: InceptionV3 Architecture

DenseNet121

DenseNet121 leverages dense connectivity patterns where each layer is connected to every other layer within a block. This connectivity encourages feature reuse, significantly reducing the number of parameters compared to traditional architectures while improving gradient flow during training. The compactness and efficiency of the DenseNet121 architecture [Fig.6] make it particularly suitable for the classification task at hand, as the model can effectively learn complex patterns in images, even in limited data scenarios.

Motivation for Model Selection

The decision to use VGG16, DenseNet121, and InceptionV3 was guided by their complementary strengths. VGG16 provides a solid baseline with its straightforward architecture, DenseNet121 offers parameter efficiency and feature reuse, and InceptionV3 excels in capturing multiscale features. Together, these models ensure a comprehensive approach to extracting the rich

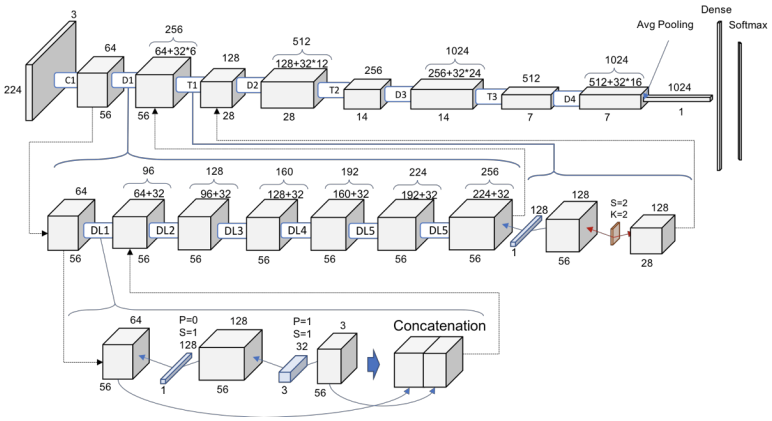

Figure 6: DenseNet121 Architecture

visual information present in the Oxford Flower Dataset, leveraging transfer learning to address the dataset's relatively small size effectively.

3.3 Transfer Learning

Each architecture was initialized with weights pre-trained on ImageNet and adapted for the flower classification task. To better understand the role of different architectures and configurations, several experiments were made for each network:

- *Totally Frozen (100%) Base Model with Standard Training Data*
- *Totally Frozen (100%) Base Model with Augmented Training Data*
- *Partially Frozen (70%) Base Model with Standard Training Data*
- *Partially Frozen (70%) Base Model with Augmented Training Data*

Furthermore, dropout layers were added after the dense layers to further mitigate overfitting and improve model robustness. Each model was compiled using the Adam optimizer, with categorical cross-entropy as the loss function and accuracy as the evaluation metric. The final classification was made using a softmax function.

4 Results and Evaluation

After defining the experimental setups, it is possible to analyze the results using different diagnostic tools. Among these we have two graphs [Fig.7] that show, on the left, the model metrics without validation curves, and on the right, the model metrics with validation curves. We then have a table [Fig.8] that collects the most important values of the experiments conducted, for example their duration, the values of training accuracy, test loss and other useful metrics.

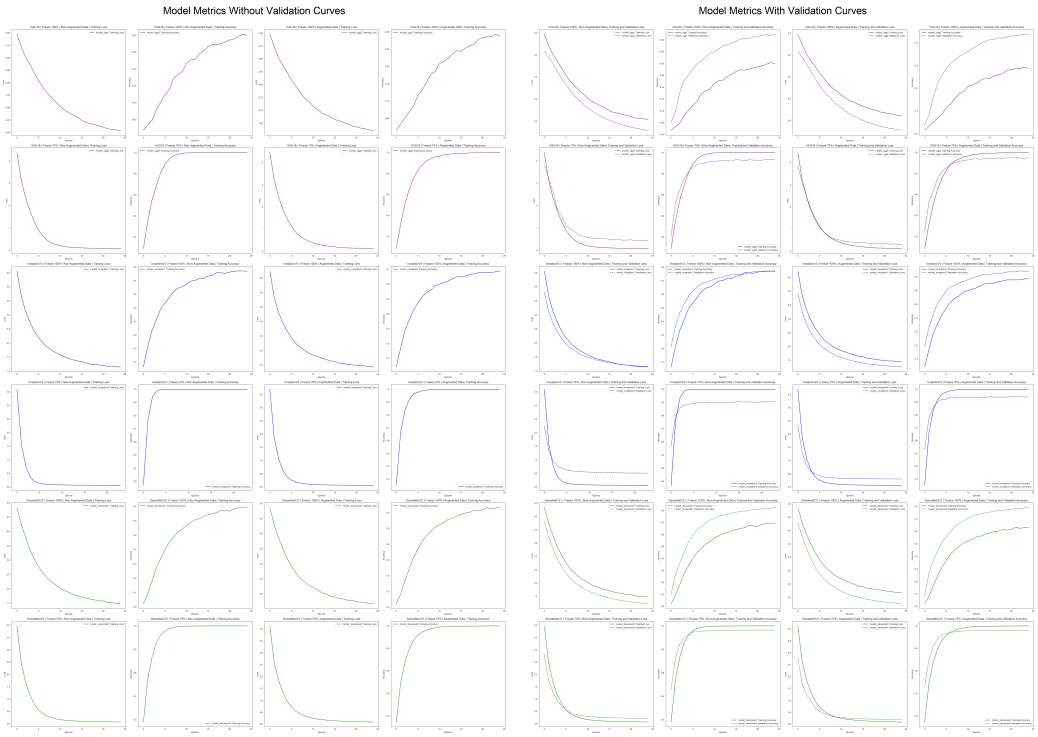

Figure 7: Training Metrics

4.1 Evaluating the Training Metrics

Evaluation of the training metrics revealed a consistent advantage for models with a '70%' freeze over those with a '100%' freeze. These configurations demonstrated superior performance in terms of loss and accuracy, with data

augmentation showing only marginal influence. In the validation phase, models with a ‘100%‘ freeze exhibited pronounced underfitting, whereas ‘70%‘ freeze configurations achieved better alignment between training and validation losses, highlighting their stronger generalization capabilities.

InceptionV3 and VGG16 models followed similar trends, with ‘70%‘ freeze configurations consistently outperforming fully frozen counterparts. Data augmentation slightly improved training metrics but had limited impact overall. Validation results for fully frozen models revealed more severe underfitting, only partially mitigated by data augmentation, while ‘70%‘ freeze models maintained superior validation accuracy. DenseNet121 emerged as the most effective architecture, achieving the highest training accuracy and lowest loss, especially with the ‘70%‘ freeze configuration. This approach significantly improved metrics compared to fully frozen models. The impact of data augmentation remained minor.

4.2 Evaluating the Experiments Report

The report that collects information on experiments shows additional information compared to the graphs. The experiments exhibit a high range of training accuracies, from as low as 45% (model_vgg2) to a perfect 99-100% (model_vgg4, model_densenet3, model_densenet4, and so on). However, the high training accuracies observed must be interpreted with caution, as they may indicate overfitting. In this regard, we note that the models with the higher freeze percentage show relatively lower test accuracies, indicating that these models may not generalize well. Models that instead use a 70% freeze percentage exhibit better generalization with test accuracies nearing or above 90%. In particular, model_densenet3 demonstrate a remarkable test accuracy of 96.98%.

The models with data augmentation tend to show slightly improved test performance compared to their non-augmented counterparts. The augmentation process helps prevent overfitting by introducing variation in the data, which may explain the superior test accuracies. This, however, comes at a cost: experiments without data augmentation take around 15-20 minutes to train (using Google Colab’s T4 GPU Hardware Accelerator), while experiments with data augmentation take around 50 minutes to train. From the analysis of the dataframe, we can also gather further insights. For exam-

ple, loss values also provide insight into model behavior. High training loss values paired with high test loss, such as in model_vgg1, model_vgg2, and model_inception1, suggest that these models struggle to fit the data properly. On the other hand, models like model_densenet3, model_densenet4, model_inception3, and model_inception‘ exhibit lower loss values, which align with their higher generalization abilities.

	Model Name	Duration	Train Accuracy	Train Loss	Val Accuracy	Val Loss	Test Accuracy	Test Loss	Predictions
1	model_vgg1	0:17:01	0.46266573667526245	2.698613166809082	0.4226384460926056	2.7939698696136475	0.44263628125190735	2.7766709327697754	[93 46 55 ... 88 9 93]
2	model_vgg2	0:52:44	0.4508025050163269	2.8014609813690186	0.4364820718765259	2.792041778564453	0.449145644903183	2.7908992767333984	[93 46 55 ... 88 91 5]
3	model_vgg3	0:18:44	0.9909281134605408	0.17514805495738983	0.9332247376441956	0.43470227718353271	0.9308381080627441	0.42559346556663513	[52 46 4 ... 15 37 78]
4	model_vgg4	0:52:25	1.0	0.05750301852822304	0.9511408461196899	0.30543631315231323	0.9471114873886108	0.3168027698993683	[52 46 4 ... 15 37 78]
5	model_densenet1	0:15:55	0.8625261783599854	1.0778467655181885	0.8232899308204651	1.2274885177612305	0.8112286329269409	1.2248973846435547	[50 46 4 ... 47 37 78]
6	model_densenet2	0:50:10	0.8066992163658142	1.263437271118164	0.7899022698402405	1.2657024860382080	0.7811228632926941	1.2548414468765259	[50 46 4 ... 88 76 78]
7	model_densenet3	0:16:11	0.996685266494751	0.08768158406019211	0.965798020362854	0.21927578747272491	0.9698942303657532	0.22156591713428497	[52 46 4 ... 15 37 78]
8	model_densenet4	0:52:16	0.9996510744094849	0.04113602638244629	0.9617263674736023	0.19560351967811584	0.9690805673599243	0.17922396957874298	[52 46 4 ... 15 37 78]
9	model_inception2	0:36:10	0.7918701767921448	1.1527433395385742	0.742671012878418	1.27436327934265140	0.7436940670013428	1.2574583292007446	[52 46 4 ... 88 37 78]
10	model_inception3	0:14:45	0.976622462272644	0.2256433367729187	0.9014657735824585	0.5198075175285339	0.982359664440155	0.5229092240333557	[52 46 4 ... 15 37 78]
11	model_inception4	0:45:28	1.0	0.04970426484942436	0.9429967403411865	0.30157670378685	0.9414157867431641	0.2825128138065338	[52 46 4 ... 15 37 78]
12	model_inception1	0:15:00	0.8860781788825989	0.831899881362915	0.7711726427078247	1.1547964811325073	0.7754271626472473	1.1645433902740479	[52 46 4 ... 50 37 78]

Figure 8: Experiments Report

5 Discussion

The conducted evaluations consistently demonstrate that the ‘70%‘ freeze configuration outperforms other settings across all models. In contrast, the ‘100%‘ freeze configuration leads to underfitting, as the lack of flexibility inhibits the learning of new features, resulting in suboptimal validation accuracy. These findings emphasize the importance of enabling partial adaptability in the model layers for improved performance. Data augmentation provided modest benefits, particularly for InceptionV3, where it reduced overfitting and enhanced generalization. However, its overall impact on performance, notably in DenseNet121, was limited, indicating that robust architectures might derive less benefit from such techniques for the task at hand. Given the increased training time associated with augmentation, its application should be evaluated when the improvements in performance are marginal.

DenseNet121 (with no data augmentation and 70% freezing) emerges as the most effective architecture, achieving the highest validation accuracy and a test accuracy of 96.98%. The additional time needed for including data augmentation in the training process (0:16:11 vs 0:52:16), in DenseNet, did not correspond to substantial accuracy gains, suggesting that augmentation's

utility should be weighed against its computational cost. To further confirm our hypotheses, a deeper analysis of model_densenet3 will be conducted using its classification report and compressed confusion matrix [Fig.9]. The compressed version of the confusion matrix only reports the rows and columns that yielded classification errors, to allow for greater readability. Please note that the uncompressed matrix can be found on the Jupyter Notebook.

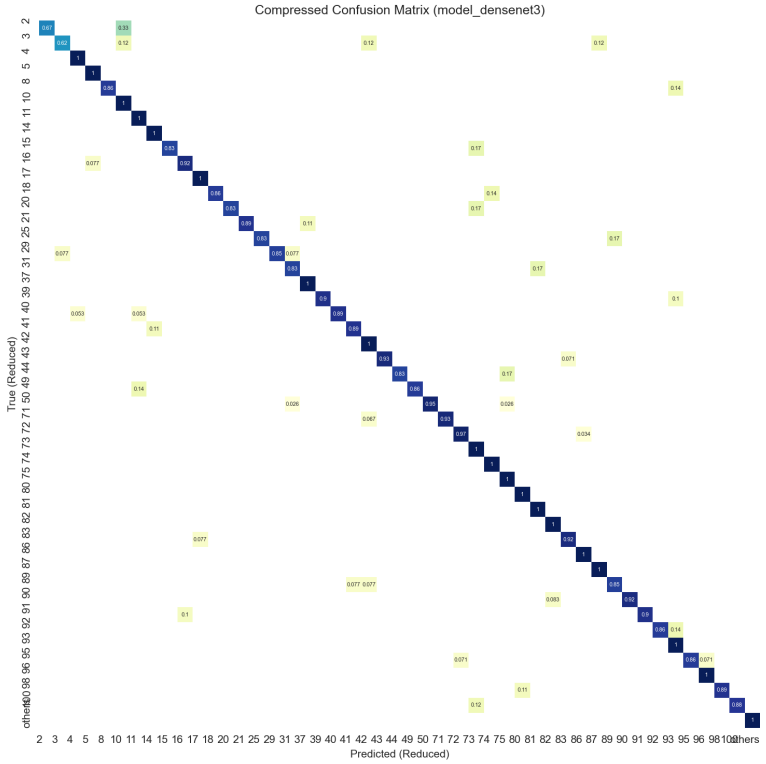

Figure 9: Compressed Confusion Matrix

The classification report for model_densenet3 demonstrates outstanding performance across all categories, with consistently high precision, recall, and F1-scores. Many classes achieve perfect scores (1.00), reflecting the model's capability to accurately identify most samples. Both macro and weighted averages of 0.97 underscore its balanced performance across all classes. The model excels particularly with smaller classes, where precision and recall are nearly flawless. Although minor fluctuations exist, such as class 2 having a

recall of 0.67 but maintaining an F1-score of 0.80, these do not detract from the model's overall robustness and versatility. The compressed confusion matrix further corroborates these findings, highlighting a dominant diagonal line indicative of high accuracy across the majority of classes. Sparse off-diagonal elements and the light intensity of the few misclassifications suggest errors are minimal and isolated rather than systematic.

5.1 Conclusion

This study evaluated the performance of three deep learning architectures across a range of experimental conditions. The primary aim was to investigate the influence of various strategies and architectural choices on model performance, generalization, and computational efficiency. Among the tested configurations, DenseNet121 with 70% of its layers frozen and no data augmentation was identified as the most effective model for the flower classification task. This configuration achieved the highest validation and test accuracies, demonstrating an optimal balance between computational efficiency and model performance. It offered robust results coupled with reduced training time, making it the most efficient and effective solution for the given task, achieving a 96.98% test accuracy.

References

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1097–1105, 2012.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.