Flower Species Classification - Final Report (Group P)

Abstract

Accurate identification and classification of flower species is a crucial task for the understanding and conservation of various plant species. However, the lack of available information about the different flower species poses a significant challenge to achieving this goal. This report proposes a Flower Identification System focused on identifying the correct class/specie of a given flower. The project utilizes three different datasets containing a varied number of flower species, that were used to train three different Convolutional Neural Network (CNN) architectures - MobileNetV2, ResNet18, and VGG16. Nine instances were trained from scratch, three via transfer learning, and multiple intsances for hyperparameter tuning. In this report, the outcomes of the scratch and transfer learning training, hyperparameter tuning, Grad-CAM and TSNE visualization are discussed. Analysis determines optimal model-dataset combo for maximum accuracy and hyperparameter tuning. This Study aims to improve flower species identification for conservation and protection.

1. Introduction

In agricultural production, forest management, and other related industries, flower identification is crucial. Because of their enormous presence, complex structure, and unpredictable diversity of classes in nature, automated species identification was initially presented 17 years ago [8]. The problem of accurately classifying flower species remains a challenge in the fields of botany, agriculture, and horticulture. This problem is important because it is necessary to understand and keep track of the diversity of flower species. However, due to the vast amount of variety within a single species, distinguishing between flower species can be difficult. Deep learning has demonstrated its effectiveness in picture classification tasks across several fields and can be leveraged to tackle this problem.

Recent research suggests that deep learning techniques such as convolutional neural networks (CNN) can improve flower classification accuracy [16]. . Other studies have focused on using transfer learning to improve the performance of flower classification models [17]. Despite these advancements, there is still room for improvement in the accuracy of flower species classification. Therefore, addressing this problem is crucial to achieve a better understanding of the diversity of flower species and informing conservation efforts.

Identifying the species of a flower through visual features is a complex problem with numerous associated challenges. Firstly, building a comprehensive and accurate classification model is difficult due to the vast number of flower species. Secondly, there can be significant variation within a single species, which makes it challenging to identify based solely on visual features. Thirdly, inconsistencies in naming conventions can cause confusion when building and evaluating classification models. In addition, the quality of images used for training and testing, overfitting, biased predictions, and lack of annotated data are other significant challenges that can affect the accuracy of flower classification models. Overcoming these challenges requires a combination of domain knowledge and advanced machine learning techniques. [16]. Despite these challenges, accurately classifying flower species has significant applications in various fields, including agriculture and horticulture.

The challenges associated with flower classification have been addressed in various ways in the literature. One of the most common techniques used is traditional machine learning algorithms such as SVM, Random Forest, and k-NN. To address the issue of variability within species, researchers have used feature selection techniques and ensemble methods. Feature selection techniques identify the most informative visual features for classification, such as color, texture, and shape features. SVM and k-NN classifiers have been trained on these features to achieve high accuracy [18]. Ensemble methods have also been used to combine multiple classifiers to improve accuracy and reduce overfitting.

To overcome the challenge of imbalanced data, researchers have employed data augmentation techniques, such as rotation, scaling, and flipping, to increase the size of the dataset and balance the number of samples across different classes. Resampling methods, such as oversampling and undersampling, have also been used to balance the class distribution in the training dataset [19].

However, traditional machine learning-based approaches have some limitations, such as the need for handcrafted features, which can be time-consuming and challenging to design. Additionally, these approaches may not be able to capture complex patterns and relationships in the data that can be learned by deep learning models.

Therefore, deep learning models have been introduced to address these limitations. Convolutional Neural Networks (CNNs) have shown impressive performance in flower classification due to their ability to automatically learn features from the raw image data. Transfer learning, a technique that leverages pre-trained CNN models, has also been used to improve performance on small datasets [17].

Our project aimed to explore the effectiveness of different convolutional neural network (CNN) architectures in identifying flower species, despite challenges such as image variability and data quality issues. In this project, various

combinations of deep learning architectures were trained and evaluated on 3 different datasets after using data preprocessing techniques. Model performances are evaluated with multiple performance metrics. To ensure fair comparison all training metrics were kept constant. All models achieved commendable results, with the models able to accurately classify flower species despite variability within species and labeling ambiguity. Model performance visualization techniques were used to visualize the models' decision-making processes. Overall, the project demonstrated the effectiveness of CNNs in flower species classification and provides insights into potential future improvements, such as using more advanced techniques and larger, more diverse datasets.

1.1. Related work

In recent years, machine learning techniques, specifically convolutional neural networks (CNNs), have been used to classify different plant species based on their images. Krizhevsky et al. utilized a deep CNN with 11 layers to classify 102 different species of flowers with an accuracy of 82.7% [13]. The authors preprocessed the images by resizing and augmenting the data and trained their model on a GPU. Similarly, Nilsback and Zisserman used a CNN model to classify flower species in the wild, achieving an accuracy of 72.8% on a dataset of 1 million images covering over 10,000 different species [14]. They also preprocessed the images and trained their model on a GPU.

Another study that used CNN models for plant identification, including flowers, is 'Deep-Plant: Plant Identification with Convolutional Neural Networks' [15]. The authors trained their model on a dataset of over 10,000 plant species and achieved an accuracy of 92.5%. The authors preprocessed the images by resizing and augmenting the data and trained their model on a GPU. In a comparative study of different deep learning methods for flower classification, including CNNs, deep belief networks (DBNs), and convolutional deep belief networks (CDBNs), Thakur et al. achieved the best performance with CNNs on a dataset of 17 flower categories [4].

In summary, CNNs have been shown to be an effective technique for flower classification using image data. The studies discussed above demonstrate the success of CNNs in achieving high accuracy for flower recognition tasks. These models rely on preprocessing techniques such as data augmentation and resizing to improve their performance.

2. Methodology

2.1. Datasets

The 3 datasets that were picked are highly rated flower datasets from Kaggle.

Dataset-1 has 5 evenly balanced classes with 1600 images per class (8000 total images). Dataset-2 has 10 evenly distributed classes with 1500 images per class (15000 total images). Dataset-3 has 16 classes unevenly distributed with a total of 15,740 images. There are 980 images on average per class in Dataset-3 where the number of images per class fell in the range of 737-1054.

Dataset Type	No. of Images	Image Dimension	No. of Classes	Formats available	Avg File Size	Avg Standard Deviation	Avg Channels
Dataset 1	8k	450x380	5	.jpg: 7463 .jpeg: 234 .png: 303	~ 63 kb	67.4	3
Dataset 2	15k	240x215	10	.jpeg: 14996 .png: 4	~ 10 kb	64.2	3
Dataset 3	15.7k	290x280	16	.jpg: 15740	~ 15 kb	64.14	3

Figure 1. Dataset Statistics

Samples from each datasets is shown in Figure[2].

Figure 2. Sample Images from Datasets

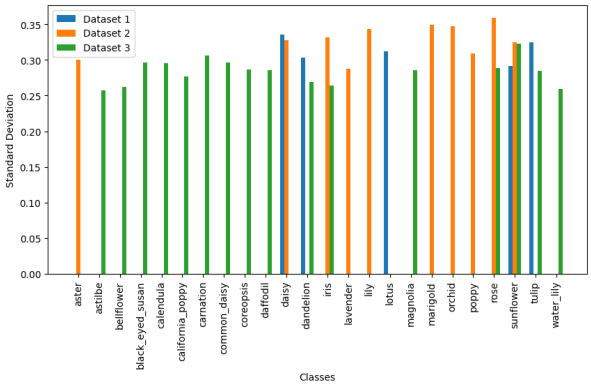


Figure 3. Standard Deviations

Figure[3] suggests that there is a considerable amount of variability within each class in all three datasets. The similarity of the average standard deviations across all three datasets suggests that there is no significant difference in the degree of variability among the classes in the three datasets.

For Data preprocessing, the following steps were performed to prepare our image datasets for training our models. A set of data augmentation techniques were applied to

increase the diversity of our training data and prevent overfitting. These techniques included random resizing, cropping, rotation, colour jitter, horizontal and vertical flipping, and perspective and affine transformations. Furthermore, training data was normalized using mean and standard deviation values.

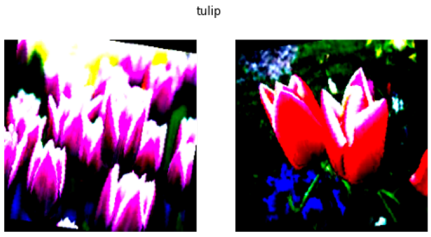


Figure 4. Sample Image after Data Preprocessing

2.2. CNN Models

The project was implemented with three popular convolutional neural network (CNN) architectures: ResNet18, VGG16, and MobileNetV2, to train and evaluate 12 model instances. 9 instances were trained from scratch on three different datasets for each CNN architecture. Additionally, three transfer learning models were trained on Dataset 3. So, there are a total of 12 model instances whose results are discussed below. The reason for selecting these three models is based on an analysis of popular CNN architectures trained for object identification/classification, taking into account their performance and the time required for training. [11]

ResNet18 has a simple architecture with 18 deep layers, consisting of convolutional layers, batch normalization, and ReLU activation functions. It also includes skip connections that allow the model to bypass layers and preserve the gradient signal, which helps prevent the problem of vanishing gradients. MobileNetV2 has a more complex architecture with 53 deep layers, consisting of depth-wise separable convolutions, batch normalization, and ReLU activation functions. It also includes a bottleneck design that uses 1x1 convolutions to reduce the number of parameters and make the model more efficient. VGG16 has a very deep architecture with 16 layers, consisting of multiple convolutional layers with small filters, max-pooling layers, and fully connected layers. It has a very high number of parameters due to its deep architecture and uses a large number of filters to extract features at different scales.

The fact that ResNet18 executes faster than other ResNet iterations and contributes to maintaining low error rates much deeper in the network with fewer flops are additional reasons to choose it. The reason behind choosing MobilenetV2 is because of channel reduction implementation which means channels number in the 1x1 convolutions are divided by 6 and only the depth wise convolution uses this high number of channels. Compared to other models, VGG16 uses 3x3 convolutional filters throughout the network which makes it a key feature of the architecture.This helps in reducing the number of parameters and makes the model computationally efficient.

While the estimated number of FLOPs for a standard ResNet18 model with a 224x224 input size is around 1.8 billion, that figure is only about 0.3 billion for a standard MobileNetV2 model. For VGG16 architecture, the estimated number of FLOPs is approximately 15 billion.

In terms of wall clock time for one epoch, the computational complexity of all three training and validation models is as follows:

Models	Dataset1	Dataset2	Dataset3
Resnet18	96	354	94.9
MobileNetV2	103	134.19	94.9
VGG16	229	315.83	216.99
Resnet18 TransferLearning			95
MobileNetV2 TransferLearning			105
VGG16 TransferLearning			289

Figure 5. Wall clock time for instances per epoch

2.3. Optimization Algorithm

The available data was divided into two categories for each epoch: Validation (10% of the data) and Test (20% of the data), from which all evaluation metrics were calculated and presented in this report.

For this project, Adam [12] optimization algorithm is selected and considered as the preferred method for optimizing all twelve dataset-model instances. Adam is a computationally efficient algorithm that is well-suited for large-scale problems with a significant amount of data and/or parameters. Additionally, it rectifies the issue of vanishing learning rates, which results in faster training times.

The Scikit-learn library is utilized to evaluate the performance of the models, generating a comprehensive classification report that includes accuracy, precision, recall, F1 score, and ROC-AUC score for each class of the dataset.

To ensure consistent performance across all instances, hyperparameters were kept constant, including the Adam optimizer, cross-entropy loss, learning rate of 0.0001, and a batch size of 128. The choice of these parameters was based on both initial experimentation and prior knowledge, as well as attempts to minimize computational costs and avoid out-of-memory errors during training.

The Early stopping technique was adopted to help prevent this overfitting and improve the generalization performance of the model [10]. The early stopping technique used in this project was based on monitoring the validation loss during training. If the validation loss does not improve for a certain number of epochs (7 epochs in this case), the training is stopped early to prevent overfitting and save computation time.

3. Results

3.1. Experiment Setup

Training large neural networks necessitates substantial computing resources. Initially, models were trained on local machines such as Apple silicon chip's "mps" and NVIDIA GTX 1060 GPU. However, due to GPU availability limitations and time constraints, cloud service providers such as Kaggle and Colab were utilized. Based on experimentation and analysis, it was determined that Kaggle was better suited for long-term requirements due to its provision of faster GPUs (NVIDIA Tesla T4 and NVIDIA Tesla P100) and the ability to run uninterrupted overnight training sessions.

1. **Data Setup:** During the course of this project, challenges related to data quality [Suppl. 21] were encountered, specifically regarding a subset of images within Dataset-2. Some images were found to contain text, which could potentially hinder the model's classification accuracy of the depicted flower. Additionally, certain images were composed in a manner that did not prominently feature the flower or make its species easily recognizable, thereby impeding accurate image classification by the model. Due to relatively higher standard deviations compared to Dataset 1 and 3, an attempt was made to prune the problematic images. At this stage, data augmentation and normalization techniques were applied to the datasets.

2. **Optimisation Setup:** The effects of different hyperparameters, such as learning rate, and batch size were evaluated. (Refer to 3.3). The datasets were split into 70/10/20 splits, with 20% of the original images acting as the test (hold-out) data which was used to calculate various evaluation metrics. A batch size of 128 and learning rate of 0.0001 was adopted along with Adam optimizer and cross-entropy loss function. The models were then trained with early stopping criteria to determine epoch count.

3. **Evaluation:** To evaluate the performance of the model metrics such as precision, recall, accuracy and F1 score were measured. Besides these, confusion matrices, ROC curves and classification reports were also generated to provide a more granular understanding of the model performances.

3.2. Main Results

Figure[6] demonstrates the performance metrics namely F1 score, precision, recall and accuracy for all the nine models trained from scratch.

It is observed that all three models have performed similarly with Dataset 1. In Dataset 1, the models struggled to differentiate between "Daisy" and "Tulip" in comparison to

Model	Dataset	F1 score	Recall	Precision	Accuracy
Resnet18	1	83.91	83.81	84.01	83.81
VGG16	1	83.91	83.77	84.05	83.75
MobileNetV2	1	82.97	82.89	83.05	82.88
Resnet18	2	77.18	77.15	77.21	77.11
VGG16	2	72.49	73.26	71.74	72.7
MobileNetV2	2	68.61	69.22	68.02	68.2
Resnet18	3	89.16	88.94	89.37	88.98
VGG16	3	69.11	68.46	69.77	69
MobileNetV2	3	77.13	76.83	77.44	77.54

Figure 6. Metrics

the other flower classes. And in Dataset 2, below-par evaluation metrics were obtained while training from scratch due to poor image quality among all classes across the Dataset. The models performed particularly well in the "Sunflower" class, with high F1 scores. However, the model struggles with differentiating [7] between the "Orchid" and "Lily" classes which raises the question about the quality of images in a few species. The datasets contain noisy or erro-

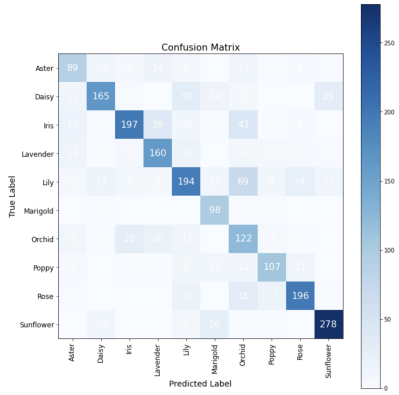


Figure 7. Confusion Matrix - Dataset 2

neous data whose underlying patterns the models may not be able to accurately learn, thereby affecting their performance. Similarly, some flower species require more complex feature representations which the model may struggle to capture. These factors explain why Dataset 1 and Dataset 2 are performing poorly across all three models shown in Fig[8],Fig[9] despite having only five and ten classes respectively.

Figure 8. Loss and Accuracy on Dataset 1

The Figure[6] shows that Resnet18 performs best on Dataset3 attaining an accuracy of 88.98%, F1 score of 89.16, Precision of 89.37 and a Recall of 88.94. Furthermore, this instance is the overall best-performing train-

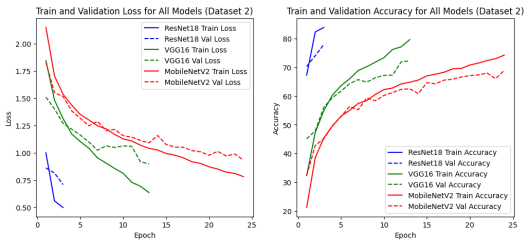


Figure 9. Loss and Accuracy on Dataset 2

from-scratch model of the nine instances.

The outcomes of our study revealed that the ResNet18 architecture consistently outperformed the other architectures on all three datasets. It exhibited the highest accuracy of 88.98% on Dataset 3, 83.81% on Dataset 1, and 77.11% on Dataset 2. This can be attributed to its deeper architecture, which allows it to capture more intricate features and patterns in the data.

Figure[10] shows the performance of Resnet18 with each Dataset.

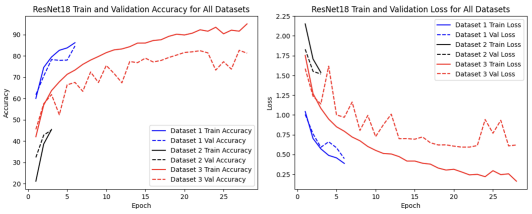


Figure 10. Resnet18 Accuracy and Loss on all Datasets

The t-SNE analysis for the instances shown in Fig[11] demonstrates that for almost all models which have been trained from scratch, clusters are easily visible which indicates that the model has learned to represent different flowers classes in a way that separates them into distinct groups based on their features. This suggests that the models perform well and have learned to distinguish between different flower classes.

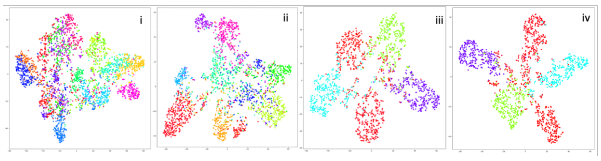


Figure 11. t-(SNE) i)VGG16-D3 ii)Resnet18-D2 iii)MobileNetV2-D1 iv)VGG16-D1

In the model instance of VGG16 architecture trained on Dataset 2, the GradCAM visual[12] shows that the model has learned to recognize a flower based on the shape and colour of its petals which is apparent by the highlight of the petals of the input image.

Finally, the supplementary[A] has attached the accuracy

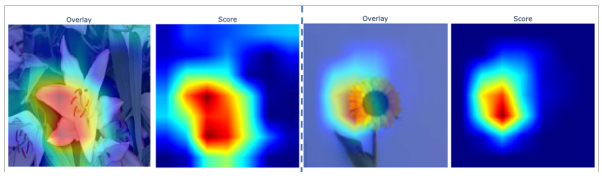


Figure 12. Overlay vs Score - Grad-CAM (VGG-16 on Dataset 2)

and loss plots of all models across Dataset-3 and different datasets across the VGG16 model and MobileNetV2 model.

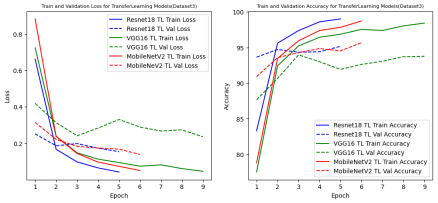


Figure 13. TransferLearning Models performance on Dataset3

On Dataset-3, three instances of Transfer learning models(Resnet18, VGG16, MobileNetV2) were trained, maintaining the same hyperparameters which produced the following results.

Model	F1 score	Recall	Precision	Accuracy
Resnet18	89.16	88.94	89.37	88.98
Resnet18-TransferLearning	94.68	94.62	94.73	94.73
VGG16	69.11	68.46	69.77	69
VGG16-TransferLearning	93.61	93.56	93.65	93.61
MobileNetV2	77.13	76.83	77.44	77.54
MobileNetV2-TransferLearning	95.93	95.89	95.96	96.06

Figure 14. Metrics Comparison Transfer Learning vs Train Scratch Models on Dataset3

Figure[14] draws comparisons between the performances of the models trained with and without pre-trained weights being used.

For the VGG16 model, the improved accuracy was around 93.61 percent from 69 percent (train from scratch model instance), demonstrating the real significance of transfer learning. This finding suggests that training CNNs from learned weights (transfer learning) can help achieve better results more quickly than training from scratch.

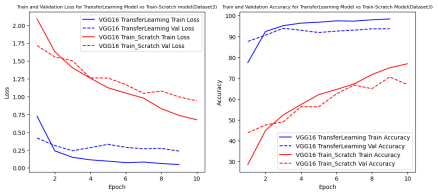


Figure 15. TransferLearning vs Train-Scratch VGG16 on Dataset3

Runtime for ResNet18 was generally comparable to Mobilenetv2, but VGG16 took almost thrice as long to train a

single epoch[5]. Results across various evaluation metrics prove that MobileNetV2 is better suited for flower classification on Dataset-3.

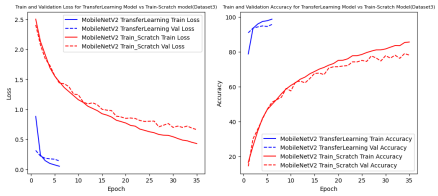


Figure 16. TransferLearning vs Train-Scratch MobileNetV2 on Dataset3

3.3. Ablative Study

In an attempt to better understand the effects of different hyperparameters, an experiment was conducted where the hyperparameters were varied within the scope of one of the models trained from scratch. For this experiment, the Resnet18 model trained on Dataset-3 was picked.

Bayesian optimization technique was implemented to find the best hyperparameter to tune. The algorithm starts with an initial set of hyperparameters, evaluates the objective function (in this case, validation loss would be the objective value), and updates the distribution based on the results. The next set of hyperparameters to try is chosen by sampling from this distribution. This process is repeated until a stopping criterion is met. The experiment calculates the relative importance of each hyperparameter based on the results of the trials in the search. Specifically, it uses a statistical technique called Analysis of Variance (ANOVA) to decompose the variation in the objective function across different hyperparameter values into contributions from each individual hyperparameter.

Batch Size and Learning rates were the hyperparameters that were picked for this Bayesian experiment and it was revealed that the tweaking of learning rate has a higher impact on model success as compared to Batch Size shown in Figure[17].

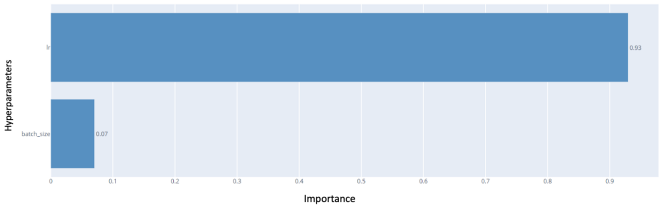


Figure 17. Hyperparameter Importance

In attempts to further investigate this across all models on dataset 3, another hyperparameter tuning experiment was set up. In this experiment, the batch size was fixed as 128 and the models have trained from scratch again with learning rates from the set of [0.01, 0.001, 0.0001, 0.00005].

For each of the three architectures (Resnet18, MobileNetV2 and VGG16), a model was trained on Dataset-3 with batch size as 128, number of epochs as 5 and a learning rate picked from the set of rates to be experimented with. This totalled to training an additional 12 models (each run for 5 epochs) which helped us determine the best learning rate.

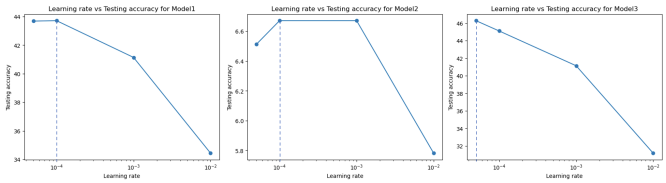


Figure 18. Learning Rate Comparison

Based on the results of the experiment, the following interpretations were concluded The learning rate of 0.0001 was consistently a good choice for all three models across dataset 3 indicating that a learning rate larger than this might have been too big a step size which could have led to a missing convergence on the minima.

A higher learning rate of 0.01 performed poorly across all models which further reinforced our inference about the gradients leaping over the local minima.

For only five epochs, the VGG16 model did not train as well compared to the other two models. Because of this reason, the VGG16 model was trained for additional 5 epochs(totalling 10 epochs) in the end gave the same result(0.0001 is best).

For MobileNetV2, the learning rate of 0.00005 provided the most promising results (slightly better performance than 0.0001). Therefore a new MobileNetV2 model was trained with a learning rate of 0.00005 and all other hyperparameters were kept constant. This new model was able to achieve slightly higher accuracy and F1 score.

Model	Learning rate	F1 score	Recall	Precision	Accuracy
MobileNetV2	0.0001	77.13	76.83	77.44	77.54
MobileNetV2	0.00005	78.78	78.77	78.79	79

Figure 19. MobileNetV2 optimisation

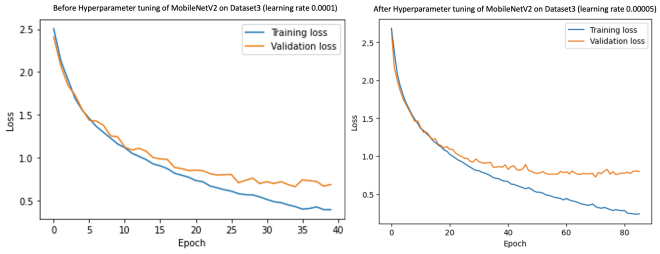


Figure 20. MobileNetV2 Loss before and after optimisation

Thus for the models of Resnet18 and VGG16, the initial learning rate of 0.0001 was the best option. This concludes that peak learning rate tuning has already been achieved.

References

[1] Flowers Dataset 1, URL: https://www.kaggle.com/datasets/nadyana/flowers.

[2] Flowers Dataset 2, URL: https://www.kaggle.com/datasets/utkarshsaxenadn/flower-classification-5-classes-roselilyetc

[3] Flowers Dataset 3, URL: https://www.kaggle.com/datasets/l3llff/flowers.

[4] Z. Ardalan and V. Subbian, "Transfer Learning Approaches for Neuroimaging Analysis: A Scoping Review," Frontiers in Artificial Intelligence, vol. 5, Feb. 2022, doi: https://doi.org/10.3389/frai.2022.780405 (Accessed: March 05, 2023).

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, Jun. 2016, doi: https://doi.org/10.1109/cvpr.2016.90 (Accessed: March 09, 2023).

[6] K. Dong, C. Zhou, Y. Ruan, and Y. Li, "MobileNetV2 Model for Image Classification," 2020 2nd International Conference on Information Technology and Computer Application (ITCA), Dec. 2020, doi: https://doi.org/10.1109/itca52113.2020.00106 (Accessed: March 12, 2023).

[7] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," IEEE Xplore, Nov. 01, 2015. https://ieeexplore.ieee.org/document/7486599/ (Accessed: March 12, 2023).

[8] T. Abbas et al., "Deep Neural Networks for Automatic Flower Species Localization and Recognition," Computational Intelligence and Neuroscience, vol. 2022, p. e9359353, Apr. 2022, doi: https://doi.org/10.1155/2022/9359353 (Accessed: March 12, 2023). 1

[9] E. M. Dogo, O. J. Afolabi, and B. Twala, "On the Relative Impact of Optimizers on Convolutional Neural Networks with Varying Depth and Width for Image Classification," Applied Sciences, vol. 12, no. 23, p. 11976, Nov. 2022, doi: https://doi.org/10.3390/app122311976 (Accessed: March 12, 2023).

[10] Y. Bai et al., "Understanding and Improving Early Stopping for Learning with Noisy Labels," arXiv:2106.15853 [cs], Dec. 2021, Accessed: Mar. 19, 2023. [Online]. Available: https://arxiv.org/abs/2106.15853 (Accessed: March 12, 2023). 3

[11] P. Zhao et al., "A comparative study of deep learning classification methods on a small environmental microorganism image dataset (EMDS-6): From Convolutional Neural Networks to visual transformers," arXiv:2107.07699 [cs, q-bio], Jul. 2022, Accessed: Apr. 12, 2023. [Online]. Available: https://arxiv.org/abs/2107.07699 (Accessed: March 12, 2023). 3

[12] Kingma, D.P. and Ba, J. (2017) Adam: A method for stochastic optimization, arXiv.org. Available at: https://arxiv.org/abs/1412.6980 (Accessed: March 12, 2023). 3

[13] Inc, A.K.G. et al. (2017) ImageNet classification with deep convolutional Neural Networks, Communications of the ACM. Available at: https://dl.acm.org/doi/10.1145/3065386 (Accessed: March 16, 2023). 2

[14] M. -E. Nilsback and A. Zisserman, "Automated Flower Classification over a Large Number of Classes," 2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing, Bhubaneswar, India, 2008, pp. 722-729, doi: 10.1109/ICVGIP.2008.47 (Accessed: April 05, 2023). 2

[15] S. H. Lee, C. S. Chan, P. Wilkin and P. Remagnino, "Deep-plant: Plant identification with convolutional neural networks," 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, 2015, pp. 452-456, doi: 10.1109/ICIP.2015.7350839 (Accessed: April 07, 2023). 2

[16] Zawbaa, Hossam Abbass, Mona Basha, Sameh Hazman, Maryam Hassanien, Aboul Ella. (2014). An Automatic Flower Classification Approach Using Machine Learning Algorithms. Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014. 10.1109/ICACCI.2014.6968612 (Accessed: April 10, 2023). 1

[17] Khuwaja, Parus Khowaja, Sunder Memon, Bisharat Rasool Memon, M Laghari, G Dahri, Kamran. (2020). Automated Flower Classification using Transfer Learning and Meta-Learners in Deep Learning Framework. 52. 93-102. 10.26692/sujo/2020.03.14 (Accessed: April 10, 2023). 1

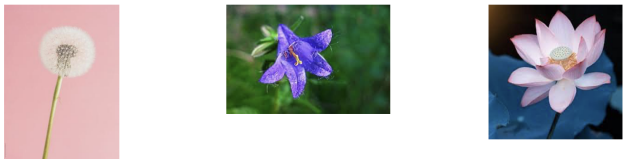
[18] Musa Cıbuk, Umit Budak, Yanhui Guo, M. Cevdet Ince, Abdulkadir Sengur, Efficient

deep features selections and classification for flower species recognition,Measurement, Volume 137, 2019, Pages 7-13, ISSN 0263-2241, https://doi.org/10.1016/j.measurement.2019.01.041 (Accessed: April 10, 2023). 1

[19] Stefanowski, Jerzy. (2016). Dealing with Data Difficulty Factors While Learning from Imbalanced Data. Studies in Computational Intelligence. 605. 333-363. 10.1007/978-3-319-18781-5_17 (Accessed: April 12, 2023). 1

A. Supplementary

Samples Dataset-1



Samples Dataset-2

Samples Dataset-3

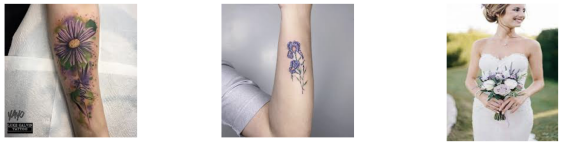


Figure 21. Problematic Images in Dataset 2

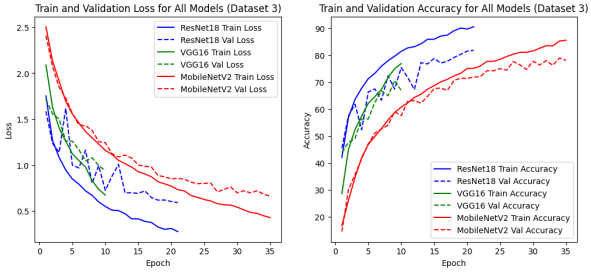


Figure 22. Loss and Accuracy on Dataset 3

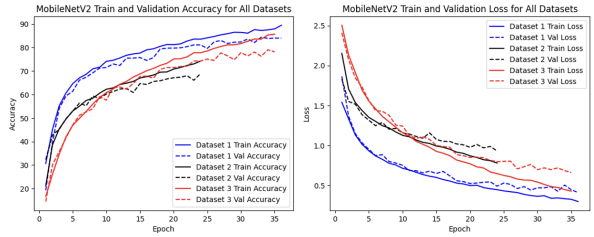


Figure 23. VGG16 Loss on all Datasets

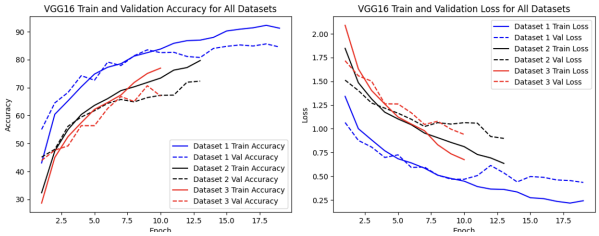


Figure 24. VGG16 Loss on all Datasets

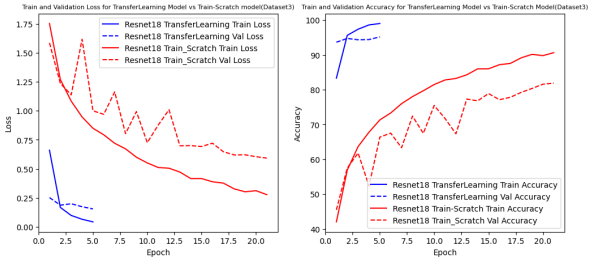


Figure 25. TransferLearning vs Train-Scratch Resnet18 on Dataset3