Flower Classification with Convolutional Neural Network

Achini Herath, **Nourin Ahmed**
(heratha, ahamed7k)@uwindsor.ca
Department of Computer Science, University of Windsor

Abstract

In this paper, we address the problem of natural flower classification. It is a challenging task due to the non-rigid deformation, illumination changes, and inter-class similarity between images of different types of flower. This paper aims to develop an effective flower classification model using convolution neural networks. To improve the accuracy of the model, we present several approaches, namely, applying deep learning optimization techniques (regularization, optimizers, activation functions etc.), transfer learning and data augmentation. For the experiments we are using a kaggle dataset containing over 100 types of flowers images. The results demonstrated that the applied combination of techniques significantly improved the accuracy of the model, and also improved its robustness and generalization ability.

Keywords: *Deep Convolutional Neural Networks; Image Classification; CNN Optimization Techniques; Transfer Learning; Data Augmentation*

1. Introduction

Flowers are an essential part of the life cycle of many species, and they are also the food of all insects, birds, animals, and even humans. Additionally, they have medicinal value because of their potential to cure many diseases. Thus proper flower classification is useful for doctors, botanists, ecological researchers, and plant conservation specialists [1]. Flower classification is a fundamental task in botany, and the application expands across agriculture, botanical research, medicine development, biodiversity conservation, and floriculture, to name a few.

The flower classification task is significantly more complex than simple object classification because flowers are often similar in color, shape, texture, and appearance. More than 250,000 known species of flowering plants are classified into 350 families [2]. Furthermore, the task is complex because within the same class there are several different types of flowers due to subtle change in color or other variation. Also, different classes of flowers can have similarities between them. On top of that there are over 400,000 different types of flowers known to us. For this project we are focusing on the publicly available Kaggle dataset for the Petals to the Metal - Flower Classification competition [3] (Fig -1). Proper flower classification can be achieved successfully under the guidance of scientific research personnel who have rich professional knowledge and experience. There are limited personals with such expertise and domain-specific knowledge. Therefore, developing automatic flower or plant

classification systems have become valuable assignments that have been widely explored in Artificial Intelligence research.

Conventional flower classification techniques combines texture base feature extraction methods such as HOG, SIFT, speeded up robust features with machine learning classifiers such as SVMs [4]. But these hand-crafted solutions start to fall apart when class imbalance is present in the dataset, and for numerous flower classes. Because of the dependency on the hand-made features, it becomes hard to distinguish between flower classes which have similar features. Thus we need a robust flower classification system which is independent of custom feature design and selection. We need a system that has the generalization capability to further beyond any hand-crafted method can reach.

Figure 1. Sample of the dataset

Thus, researches have turned towards deep learning techniques and specially convolution neural networks (CNN). Recent research has demonstrated that the convolution neural network has great advantages in feature extraction and has certain degree of invariance to the operation. CNN are increasing used for image classification due to their ability to automatically learn features from the raw pixel values of the input images. CNN can detect and learn hierarchical representations of patterns and structures in the images, making them suitable for handling complex and varied image data such as flowers with their diverse shapes, colors, and textures. However, CNN models are affected by issues such as data sparsity and are prone to over-fitting or fall into local optimization problems.

The focus of this research project is to develop a CNN model for classifying flowers and enhance its performance and address its limitations by utilizing several approaches, namely - deep learning optimization techniques, transfer learning techniques and data augmentation techniques. Our baseline CNN model demonstrated a poor accuracy of 38.9% but the final model with the incremental combination of above techniques exhibited state-of-performance of 94.7% accuracy. The paper will present a comprehensive discussion on how we arrived at that result.

2. Problem Statement

The goal of this research project is to experiment with developing CNN models combined with different optimizing approaches and techniques to classify flower species based on their images. The target of the models is to be able to generalize well to unseen images of flowers and achieve high accuracy in classifying them into their respective categories. Additionally, our goal is to apply techniques will address three main challenges of CNN based classification models. Firstly, the data sparsity arising due to the small size of the dataset. Secondly,

the tendency of the model to over fit on the training dataset. Thirdly, to handle the class imbalance of dataset that will cause the model to be biased towards the more frequent classes and performing poorly on the less frequent ones. Finally, we aim to present a comparative analysis on the findings from the research.

3. Literature Review

Development of the recognition systems for plants and flower species will be beneficial to fields such as the pharmaceutical, industry, botany, agricultural, and trade activities. It has been reported that there are over hundreds of thousands of species of flowers, and due to the diversity and the similarities between species it is a challenging task to accurately classify them [5]. There has been a significant amount of research on flower classification using various machine learning techniques. Mete et al. [6] has presented an experimental analysis on using support vector machines (SVM), random forest (RF), K nearest neighbor (KNN) and deep CNN on the Oxford-17 Flowers Dataset. In the study they obtained accuracies of 99.60%, 95.40%, 99.10% and 99.80% respectively for above four techniques, thus concluding that CNN is a better classifier surpassing other techniques. CNN is becoming one of the most commonly used models in deep learning and it has become a research hotspot in the field of computer vision [7]. Unlike traditional neural networks, CNNs take into account the spatial relationships between pixels in an image, allowing them to learn features such as edges, textures, and shapes from raw pixel data. A CNN consists of several layers including convolutional layers, pooling layers, and fully connected layers. In the convolutional layer, a set of filters convolve over the input image to produce a set of feature maps. The pooling layer reduces the size of the feature maps, which helps to reduce the computational complexity of the network. The fully connected layers use the extracted features to make the final classification decision [7] (Fig - 2).

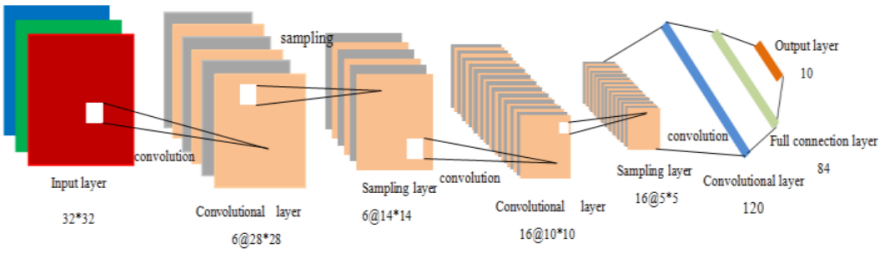

Figure 2. Traditional convolution neural network structure (Source:[7])

Research in CNN demonstrates that there are number of optimization techniques that can be incorporated to improve its performance. Overfitting in the fully connected layers is a potential issue for most CNN models. Krizhevsky et al. [8] proposed that employing the dropout regulation technique is an effective method of addressing the problem of overfitting. His proposed network achieved top-1 and top-5 error rates of 37.5% and 17.0% on the ILSVRC2012 dataset, which is considerably better than the previous state-of-the-art. Dahl et al. [9] proposed using the alternative of Sigmoid function, the Rectified Linear Unit (ReLU) activation, can significantly avoid gradient dispersion problems and speed up training of a CNN network. Giusti et al. [10] discussed on the use of max pooling layers to down sample the feature maps produced by the convolutional layers. He suggested that this will allow the network to be more computationally efficient and enables faster processing times, while still maintaining high accuracy in object recognition tasks.

Ruder et al. [11] discussed, Gradient descent as one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. It presents a comparative analysis of batch gradient descent, stochastic gradient descent, and mini-batch gradient descent. The experiments show that batch gradient descent can lead to slow convergence and poor generalization, especially when dealing with large datasets. Stochastic gradient descent can be faster but may suffer from high variance and noisy updates. Mini-batch gradient descent, on the other hand, provides a good balance between the two, leading to fast convergence and good generalization. He also discussed how gradient descend optimization algorithms such as Adagrad, Adadelta, RMSprop, Adam, AdaMax, Nadam can be utilized to improve the performance of neural network models.

Transfer learning has been widely used with convolutional neural networks (CNNs). Transfer learning involves taking a pre-trained CNN, removing its final layer(s), and using the remaining layers as a feature extractor for a new task. Transfer learning has several benefits such as reducing training time by using a well-performing pre-trained network, improving generalization by being pre-trained on general features that are useful for a range of tasks and reducing data requirements by having learned general features from large datasets ([12], [13]). Rongxin et al [14] used a pre-training model of VGG-16 on the ImageNet dataset for the CNN based flower classification model with the purpose of reducing the overfitting of model as well as to save training time and reduce the amount of required training data. Xiaoling et al [5] published a paper on using Inception-v3 with CNN architecture for flower classification. His model demonstrated an accuracy of 95% and 94% on Oxford-I7 flower dataset and Oxford-102 flower dataset respectively. Yong et al. [7] discussed on developing an effective flower classification approach using convolution neural network and transfer learning. VGG-16, VGG-19, Inception-v3 and ResNet50 models were used to compare the network initialization model with the transfer learning model. The four models in the paper pre-trained the model on an ImageNet dataset containing more than 1.2 million natural images and more than 1,000 different categories. Experiments were carried out on Oxford-17 flower dataset and Oxford-102 flower dataset. The results show that transfer learning can effectively avoid deep convolution networks are prone to local optimal problems and overfitting problems. VGG-16, VGG-19, Inception-v3, ResNet50 transfer models demonstrated an accuracy of 83.53%, 84.71%, 94.58% and 95.29% respectively on the Oxford-17 flower dataset.

Data augmentation techniques are incorporated as a pre-processing step with CNN models to increase the amount the training data as neural network models require large training data to archive state-of-art results. Data augmentation is also useful to reduce overfitting by providing the model with diverse training samples and handle class imbalance by creating more samples of under-represented classes [6]. Busra et al. [6] developed a flower classification model with CNN and observed an increase in accuracy of results by using data augmentation as a pre-processing step. He used Oxford-17 and Oxford-102 flower datasets and doubled the size of the training data by vertically flipping all the images. Krizhevsky et al. [8] proposed two data augmentation methods for developing a CNN based classification model for ILSVRC-2010 dataset. The first technique was on generating image translations and horizontal reflections by extracting random 224×224 patches (and their horizontal reflections) from the 256×256 images and training the network on these extracted patches. The second technique was on altering the intensities of the RGB channels in training images by performing PCA on the set of RGB pixel values throughout the ImageNet training set. He demonstrated that the data augmentation techniques were able to reduce the top-1 error rate by over 1%.

4. Methodology & Experiments

4.1. Dataset

The dataset used for the project is obtained from Kaggle - Petals to the Medal - Flower Classification competition [3]. The dataset is small and comes from many disparate sources. The dataset has 104 different flower species. The dataset is created by drawing images from five different public datasets. Major challenge of the dataset is the severe class imbalance. Some species have comparatively high number of image samples that will cause the model to be biased towards them. Additionally, some classes contain many sub-types (eg- wild roses) whereas some classes contain only one particular sub-type of flower (eg – pink primrose). Figure 3 demonstrates the distribution of the classes in the dataset. It shows inverse of the distribution, meaning that the lower range bars are of classes with higher percentage of examples.

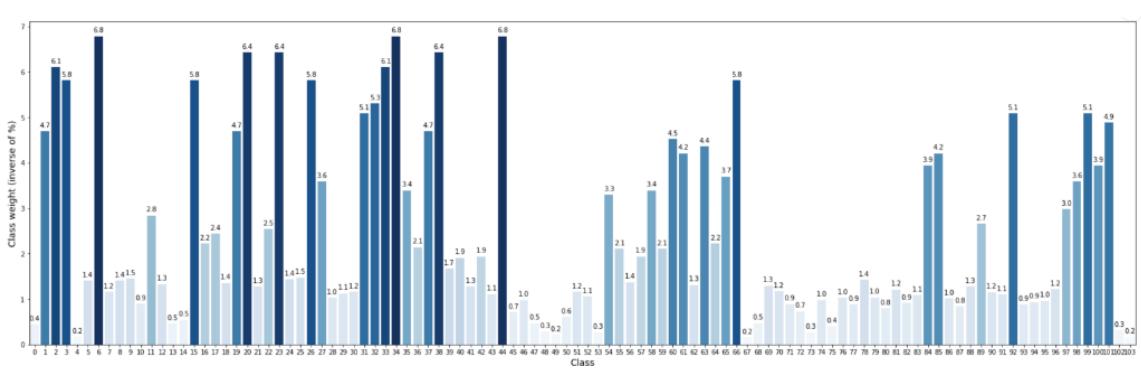

Figure 3. Class imbalance of the dataset

Also there are few issues in the clarity of images such as Images having large scale, exposure and light variations. There are also many classes that show great differences in the dataset as well as many classes with similarities to other classes. Additionally, images of the flowers are often present as a backdrop of a model, or a modern machinery, or in odd places. For the experimental purposes the dataset has been divides into 3 part – Training, validation and test data.

4.2. CNN Architecture and Optimization Techniques

CNN models developed for this project consist of a combination of convolutional layers, pooling layers, and fully connected layers. The convolutional layers extract the features from the input images and apply a set of learnable filters to the input image and produce feature maps, which represent the presence of certain visual patterns or features in the input image. The pooling layers reduces the spatial size of the feature maps by down sampling, which helps in reducing the number of parameters and computational complexity of the model. The fully connected layers take the flattened feature maps and learn the relationship between them to produce the final output, which is the predicted class of the input image. We experimented with combinations of 3 to 6 layers for our models.

To avoid substantial overfitting in the fully-connected layers, we employed the dropout regularization technique. Dropout randomly removes a fraction of the neurons from the layer, which forces the remaining neurons to learn the features that were previously learned by the removed neurons. By doing so, dropout can also improve the generalization of the model. We experimented with Dropout with rates 0.25 and 0.5. We also experimented with Batch normalization to standardize the inputs to a layer for each mini-batch. This is expected to

stabilize the learning process and effectively reduce the number of training epochs required to train deep networks.

We used Softmax and Relu activation function for different layers. Relu has the particular advantage of being computationally efficient and helpful in preventing the vanishing gradient problem. We also experimented with Adam and Nadam optimization algorithms to change the attributes of the neural network such as weights and learning rate in order to reduce the losses. We implemented a learning rate callback function that will decrease the learning rate when the validation loss stops improving to prevent the model from overshooting the minimum and also allowing the model to converge more quickly and accurately.

4.3. Transfer Learning

Transfer learning is a technique used to save significant time and computational resources by allowing the use of pre-trained weights and architecture of a model that has already learned to recognize basic features of images. Transfer learning not only can learn, colours, texture and other low-level features (e.g. edges, blobs etc.) on the pre-training dataset, but it can also learn to help the classification of the target dataset's advanced semantic features. Our models suffer from data sparsity due to the small size of the dataset, and transfer learning is utilized to address it. We also intend this approach will help the model to generalize better to new data and also improves the model's performance by avoiding the overfitting that occurs when training a deep learning model with a small amount of data.

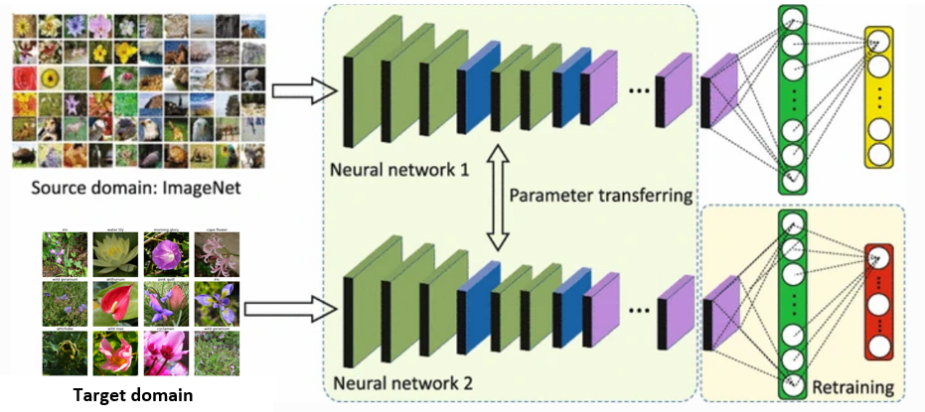

Figure 4. Transfer learning based on CNN model (Source:[15])

We experimented with three transfer learning models - ResNet50, DenseNet201 and Xception ([12], [16], [17]). ResNet50 consists of 50 layers and was designed to address the vanishing gradient problem that occurs in very deep neural networks. ResNet50 uses a residual block structure in which each block contains two or more convolutional layers along with shortcut connections that bypass the block and feed the input directly to the output. DenseNet201 has over 20 layers, and it is similar to ResNet, and uses skip connections to address the vanishing gradient problem. Xception that uses depth wise separable convolutions, which are computationally efficient variants of standard convolutions. It has over 36 convolutional layers. All three models are trained on the ImageNet dataset that has over 1.2 million images and 1000 classes.

4.4. Data Augmentation

Data augmentation is a technique used to increase the size of a training dataset by creating new variations of the existing data, thus addressing data sparsity issue. Data augmentation also helps to prevent overfitting by providing the CNN model with more training examples and reducing its sensitivity to the exact details of the training dataset. It also helps to improve the generalization ability of the model, as it learns to recognize the important features of the images even when they are presented in different orientations and scales. For our experiments we are using data augmentation specially to address the class imbalance. Instances of classes which are below selected threshold were augmented multiple times comparatively to the classes with high instance counts.

Figure 5. Sample data

Figure 6. Augmented - Flipping

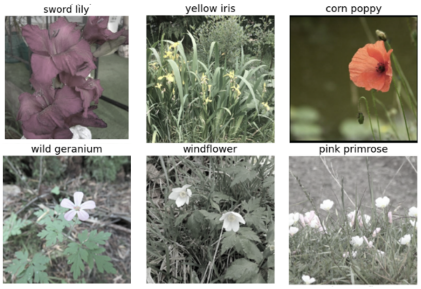

Figure 7. Augmented - Colour Saturation

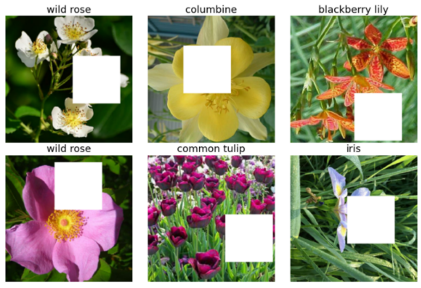

Figure 8. Augmented - Cropping

We used three data augmentation techniques. First Random flip where images were randomly flipped horizontally or vertically. Second technique was applying random saturation for images, to prevent the model high dependency of the perfect colors. As real test images can be of different sizes, may not also contain the entire flower, as the final technique we randomly cropped out small segment of the images. Figure 5 shows a sample of initial dataset and Figures 6, 7, 8 shows data samples from the three augmented techniques.

5. Experimental Setup

Our experiments were performed on Kaggle notebooks with the Petals to the Medal - Flower Classification competition dataset. Kaggle notebooks are cloud-based environments which provides access to a variety of resources, including CPU, GPU, and TPU (Tensor Processing Unit) for running machine learning and deep learning models, as well as storage resources for storing datasets, models, and other files. We used Tensor Processing Unit which is an AI accelerator application-specific integrated circuit (ASIC) designed by Google specifically to accelerate machine learning workloads. They are designed to handle the large matrix computations that are involved in deep learning models, and are optimized

for training and inference tasks. TPUs have a higher computational power and memory bandwidth compared to GPUs and can be more cost-effective when training large models.

We incrementally applied the discussed techniques and observed the performance of models (Fig - 9). First set of models were developed by incorporating various layer optimization techniques (Baseline models). For the next set of models, we incorporated different pre-trained models to the Baseline architecture. Finally, we identified the best performing model and applied combinations of discussed data augmentation techniques as a pre-processing step. To measure the performance of the models we are evaluating several matrices – Accuracy, F1 score, Specificity, Sensitivity of the test dataset. We are also observing on the accuracy and loss of training and validation datasets.

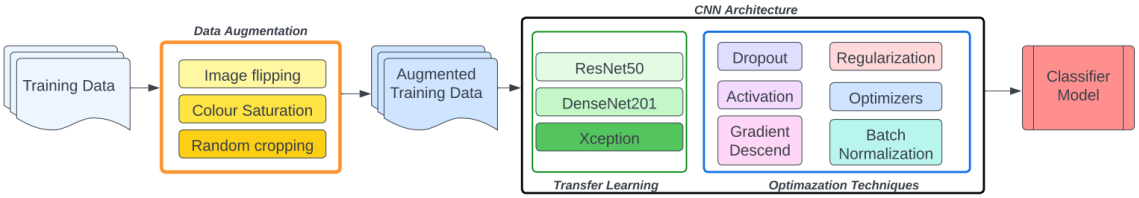

Figure 9. Available enhancement techniques in the proposed model

6. Result and Discussion

We have experimented with a total of eight models. We started with a base model and incrementally improved the performance of the model by applying the techniques discussed in the methodology. After fine tuning the baseline model, we incorperated the three transfer learning models. We use the best model from there and experimented with the data augmentation techniques. Performance of all models on training and validation dataset and on test dataset is present in Annex Table - 2 and Table - 1.

Model	Epochs	Accuracy	Specificity	Sensitivity	PPV	NPV
Baseline	20	0.389	0.405	0.429	0.75	0.386
ResNet50	20	0.699	0.911	0.858	0.667	0.889
DenseNet201	20	0.185	0.456	0.0	NAN	0.434
Xception	20	0.386	0.165	0.0	NAN	1.0
Flip & De-colorization	20	0.896	0.962	1.0	1.0	0.962
Random White Squares & De-colorization	20	0.888	0.950	0.858	1.0	0.962
All data augmnetation	20	0.883	0.975	0.858	0.975	0.95
Final - RestNet50 with Flip & De-colorization	35	0.937	0.962	1.0	0.875	0.962

Table 1. Experiment Result

6.1. Base Model - CNN

The architecture of the our basic CNN model comprises of Conv2D, MaxPool2D, Dropout, Flatten, and Dense layers. The Conv2D layers are used to apply convolution operation on the input image with 32 and 64 filters and a kernel size of 5x5, along with 'relu' activation for introducing non-linearity. 'Same' padding is used to maintain spatial dimensions of the feature maps. The MaxPool2D layers are used for down-sampling the feature maps with a pool size of 3x3 to reduce computational cost and retain important features. Dropout layers with a rate of 0.25 are used for regularization to reduce overfitting by randomly setting 25% of neurons to 0 during training. The Flatten layer is used to convert the 3D feature maps into a 1D vector for further processing. The Dense layer with 'softmax' activation function is used in the output layer for multi-class classification, and the number of neurons in the

output layer is set to the number of classes in the problem. We made this design choice by following popular practices in literature, and also by experimenting with different values for the dropout percentage, and pool size of the MaxPool2D.

The performance of the our basic CNN is not the best. The model achieved an accuracy of 0.3887, indicating the overall correctness of its predictions. The specificity, which measures the model's ability to correctly identify negative cases, was observed to be 0.4051, suggesting less than moderate ability to correctly identify true negative cases.

The sensitivity, which measures the model's ability to correctly identify positive cases, was found to be 0.4286, indicating that the model is not great in correctly identifying true positive cases. The positive predictive value (PPV) was calculated to be 0.75, suggesting that 75% of the predicted positive cases were true positives. The negative predictive value (NPV) was found to be 0.3855, indicating the model's ability to correctly predict true negative cases among the predicted negative cases is low.

From epochs vs accuracy and epochs vs loss graph in Fig - 15 and Fig - 14 we can see that initially the accuracy increased sharply as the epoch number increases, but after around 10 epochs the validation accuracy line is becoming flatter. It indicates that the validation accuracy is not improving much with the epochs increase. Thus we stopped the training after 20 epochs.

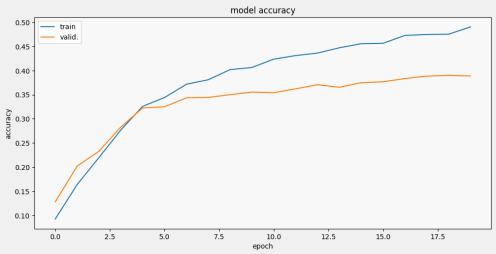

Figure 10. Performance of Baseline Model - Accuracy

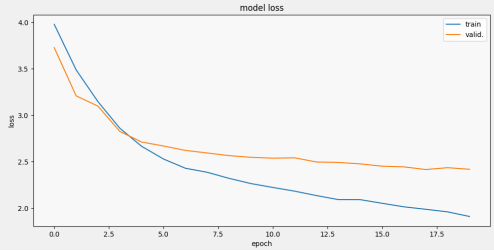

Figure 11. Performance of the Baseline Model - Loss

6.2. Transfer Learning

We used three transfer learning modles - RestNET50, DenseNet201, and Xception. From our experiment we found that ResNet50 performed the best than Densenet201 and Xception. ResNet's skip connections allow for easy flow of gradients which may enable the model to learn the patterns in data effectively even in the presence of class imbalance. The flow of gradients across different layers is facilliated by skip connections which in turn helps mitigate the fewer samples issue.

6.2.1. ResNet50

With the ResNet50 model, trained with 20 Epochs we achieved the accuracy of 0.6985. The ability to correctly identify position cases, the sensitivity, of the model was 0.8571. The specificity, which measures the ability to correctly identify negative cases, was 0.9114. The result shows that the model's performance was better in terms of specificity relative to sensitivity. Additionally, the positive prediction value (PPV) was 0.6667 while the negative predictive value (NPV) was 0.889. The lower PPV indicates that the model have a higher percentage of false positive predictions. The higher NPV value indicates that the model is slightly better in correctly predicting true negative cases. Thus the model is less error prone to missclassifying negative cases as positive. But again, the lower PPV value indicates

that the model is misclassifying positive cases as negative. The accuracy and loss graph is presented in Fig - 12 and Fig - 13.

6.2.2. DenseNet201

From our experiment we found that DenseNet201 performed poorly in performing flower classifying task. After 25 epochs it achieved only 0.1853 in accuracy, which indicates overall low correctness of the model's predictions. The specificity was 0.4557 and the NPV was 0.4337, both indicates the the model is able to predict some true negative cases correctly. But the value also indicates that the model predicts a considerable negative cases as positives. To investigate the sensitivity value we found that it is 0.0 which indicates that the model fails to identify any positive cases. This is a concern because failing to correctly identify any class means that the model is basically of no use. The accuracy and loss graph is presented in Annex Fig -18 and Fig - 19.

DenseNet201's architecture is a densely connected architecture which requires huge amount of data for learning patterns related to every class present. The dense connectivity of the network evidently could not capture relevant featues to properpy classify flower images.

6.2.3. Xception

Our experiment with Xception resulted in a low accuracy of 0.3858. The specificity was observed to be 0.1646 only, suggesting a high false positive rate. Another metrics, sensitivity is found to be 0.0, which is of concern as it means that the model is unable to correctly identify any positive cases. We suspect that it can due to the class imbalance presents in the dataset, and also probably the Xception architecture is not suitable for our dataset. As the model fails to predict any positive cases the PPV was found to be "nan". However, on the more positive side the model achieved 1.0 in NPV, indicating the it is able to correctly predict true negative cases among the predicted negative cases. Further investigation is necessary to quantify the model's performance by fine-tuning various parameters such as learning rate, epochs. The accuracy and loss graph is presented in Annex Fig - 20 and Fig - 21.

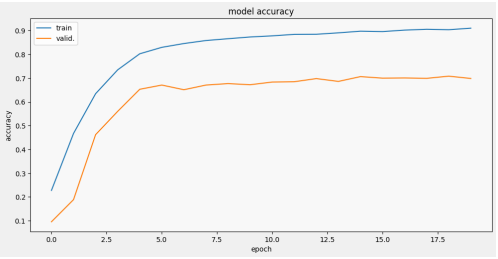

Figure 12. Best Transfer Learning Model ResNet50's Performance - Accuracy

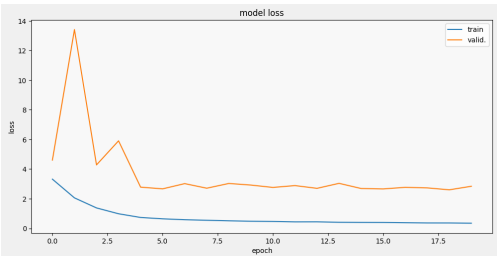

Figure 13. Best Transfer Learning Model ResNet50's Performance - Loss

6.3. Augmentation

In order to address data sparsity and the class imbalance present in the dataset, we employed data augmentations. We used three types of data augmentations - Flip, Decolorization & Random White squares. The detailed result of the experiments is discussed in the following sections. From our experiments with three transfer learning models we found that ResNet50 performed the best, so we continue with ResNet50 to further optimize its performance.

6.3.1. Flip and De-colorization

This dataset has class imbalance problem which we addressed by performing data augmentations. We experimented with flipping the images both horizontally and vertically. Additionally, we use de-colorization because images outside of the dataset may not always have the brightest colored image available. By doing de-colorization we tried to reduce the model's overfitting to specific colors. The motive behind it is to force the model to learn the features of the flowers as well as the colors. The experimented resulted in an accuracy of 0.896 which will dramatically better than any of the transfer learning approaches. The specificity is 0.962 and the sensitivity is 1.0 indicates that the model has a low rate of false positive and it can correctly identify all positive cases. The PPV value is also 1.0 means that all predicted positive values are true positives. The 0.952 NPV value demonstrates that it has a low rate of false negatives. In terms of all of the metrics the models performance has notably increased. The accuracy and loss graph is presented in Annex Fig - 22 and Fig - 23.

6.3.2. White Squares and De-colorization

We continued with our experiment with data augmentation, and we selected random white squares to add on the images. Erasing or hiding random part of the images forces the model to learn various aspects of the flowers and it reduces overfitting as well. We kept the de-colorization we wanted to make the model robust to unseen data. The combinations of these two data augmentation with ResNet50 resulted in an accuracy of 0.8882 which is a bit lower than the first augmentation experiment. From table 1 we can see that the in terms of other metrics this model performed similar to the first data augmentation combination. Which means overall the model has very good performance in predicting all predicted positive values but there is room for improvement in terms of Sensitivity and NPV. The accuracy and loss graph is presented in Annex Fig - 24 and Fig - 25. .

6.3.3. Combination of Flip, White Squares, and De-colorization

As both the combinations separately performed in a similar manner we further extend our experiments by combining all three data augmentation techniques - Flip, Random White Squares and De-colorization. The accuracy we got by combining all data augmentations is 0.8825 which is similar to the other two experiments. But the specificity value is 0.9746 which is better than any other models performance. So it demonstrates that combination of augmentations improved the models ability in correctly identifying true positives.

We observed that including the white square augmentation method slightly lowers the accuracy of the model. It can be due to the fact that most images in the dataset is quite clear and the while squares significantly added noise to the training dataset, which caused the model to under fit.

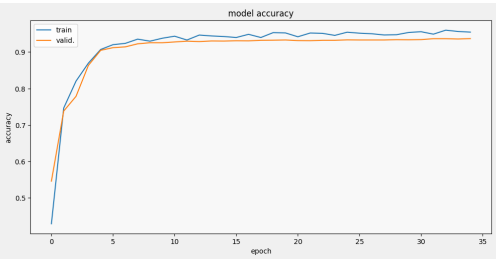

Figure 14. Flip + De-colorization Model's Performance - Accuracy

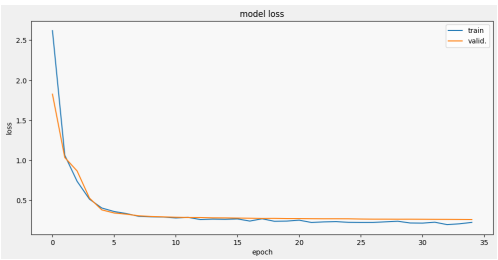

Figure 15. Flip + De-colorization Model's Performance - Loss

6.4. Final Model: ResNet50 with Flip and De-colorization Data Augmentation

From the output of the experiments with data augmentation we observe the ResNet50 with Flip and De-colorization data augmentation has the highest accuracy. Also the sensitivity and PPV was 1.0 for that model. Thus we trained this model further by increasing the epoch number from 20 to 35. Our result shows that indeed increasing epochs helped the model to perform better. This final model achieved an accuracy of 93.7% which is the best so far. We were focusing on increasing the sensitivity value because the flower classification problem requires a higher rate of true positive rate. Thus developing this model we were able to achieve our goal to gaining moderate accuracy with a high true positives prediction rate.

From Figure - 16 and Figure - 17 we can observe that the model reached highest accuracy around 6 epochs and afterwards the increase was slow. It is also noticeable that both training and validation accuracy and loss graphs are close to each other, which indicates that the model has good generalization capability.

The final model demonstrated a significant improvement from the 38.9% accuracy of the baseline model. Our experiments and the result of the final model show that by using data augmentations in combination with transfer learning we were successfully able to address the class imbalance and data sparsity problems present in the flower dataset and prevent overfitting to a considerable degree.

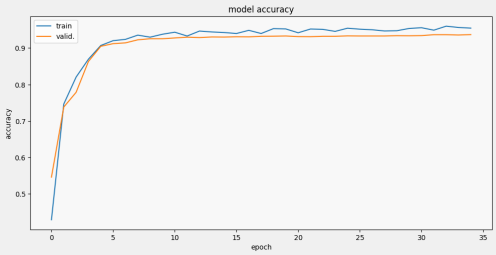

Figure 16. Final Modelś Performance - Accuracy

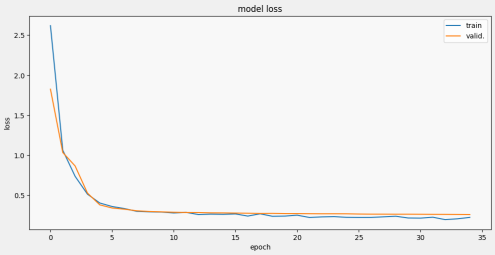

Figure 17. Final Modelś Performance - Loss

7. Conclusion

In this paper, we have experimented with a flower image dataset from Kaggle containing more than 140 flower classes. We presented our experiment result with a base model, then we leveraged transfer learning techniques which achieved better results by allowing the model to generalize well in addition to being pre-trained on a very large dataset. To address the class imbalance problem we used data augmentations that also increased the robustness of the models by a large degree. Our final model is ResNet50 with the combination of Flip + De-colourization augmentations. From running the training for 35 epochs we achieved the best result among all the experiments which is 93.7 accuracy%.

As future work we propose to experiment more with fine tuning transfer learning models. We also plan to apply data augmentation techniques to the test dataset in order to analyse whether it can improve the accuracy of the predictions. Finally, we propose to test this CNN architecture on other similar classification problems such plant classification or fruit classifications.

References

[1] Y. Wu, X. Qin, Y. Pan, and C. Yuan, "Convolution neural network based transfer learning for classification of flowers," in *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*, pp. 562–566, 2018.

[2] H. Hiary, H. Saadeh, M. Saadeh, and M. Yaqub, "Flower classification using deep convolutional neural networks," *IET Computer Vision*, vol. 12, no. 6, pp. 855–862, 2018.

[3] "Petals to the metal - flower classification on tpu = https://www.kaggle.com/competitions/tpu-getting-started/overview, note = Accessed: 2023-03-25."

[4] C. Narvekar and M. Rao, "Flower classification using cnn and transfer learning in cnn-agriculture perspective," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 660–664, IEEE, 2020.

[5] X. Xia, C. Xu, and B. Nan, "Inception-v3 for flower classification," in *2017 2nd international conference on image, vision and computing (ICIVC)*, pp. 783–787, IEEE, 2017.

[6] B. R. Mete and T. Ensari, "Flower classification with deep cnn and machine learning algorithms," in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 1–5, IEEE, 2019.

[7] Y. Wu, X. Qin, Y. Pan, and C. Yuan, "Convolution neural network based transfer learning for classification of flowers," in *2018 IEEE 3rd international conference on signal and image processing (ICSIP)*, pp. 562–566, IEEE, 2018.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[9] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8609–8613, IEEE, 2013.

[10] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *2013 IEEE International Conference on Image Processing*, pp. 4034–4038, IEEE, 2013.

[11] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[13] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[14] R. Lv, Z. Li, J. Zuo, and J. Liu, "Flower classification and recognition based on significance test and transfer learning," in *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, pp. 649–652, IEEE, 2021.

[15] Z. Yang, W. Yu, P. Liang, H. Guo, L. Xia, F. Zhang, Y. Ma, and J. Ma, "Deep transfer learning for military object recognition under small training set condition," *Neural Computing and Applications*, vol. 31, pp. 6469–6478, 2019.

[16] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[17] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

8. Appendix

8.1. Additional Tables & Diagrams

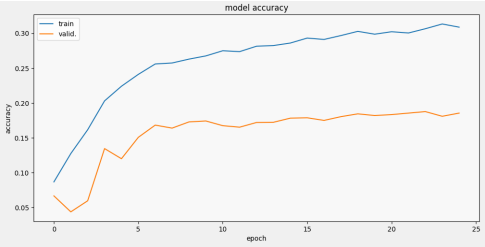

Figure 18. Densenet201's Performance - Accuracy

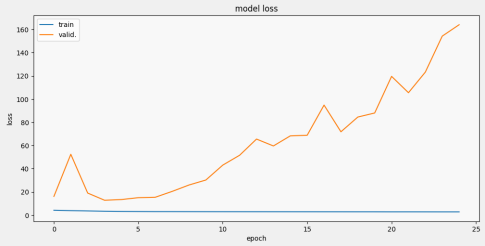

Figure 19. Densenet201's Performance - Loss

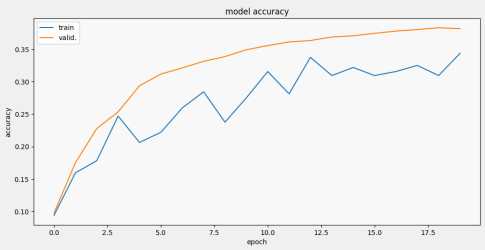

Figure 20. Xception's Performance - Accuracy

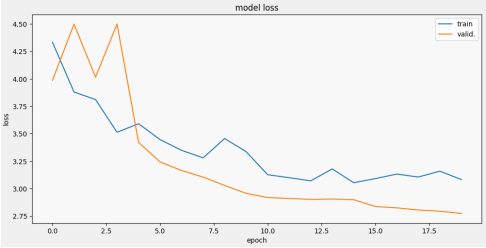

Figure 21. Xception's Performance - Loss

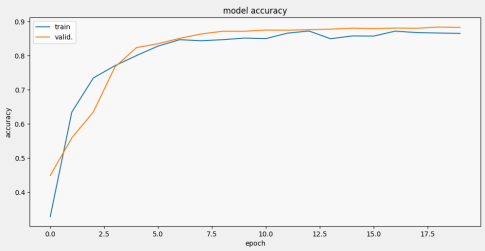

Figure 22. Augmentation: Flip + De-colorization's Performance - Accuracy

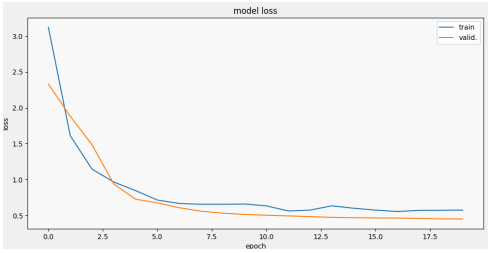

Figure 23. Augmentation: Flip + De-colorization's Performance - Loss

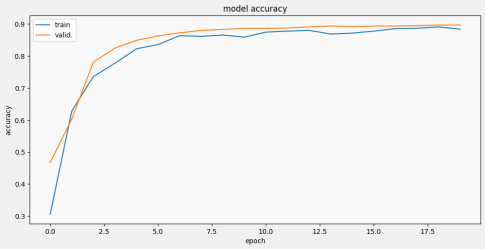

Figure 24. Augmentation: White Square + De-colorization's Performance - Accuracy

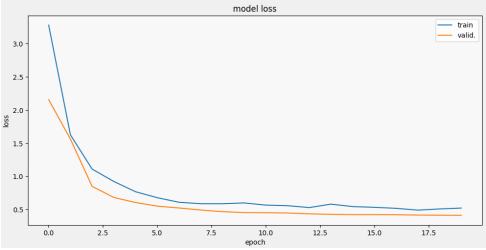

Figure 25. Augmentation: White Square + De-colorization's Performance - Loss

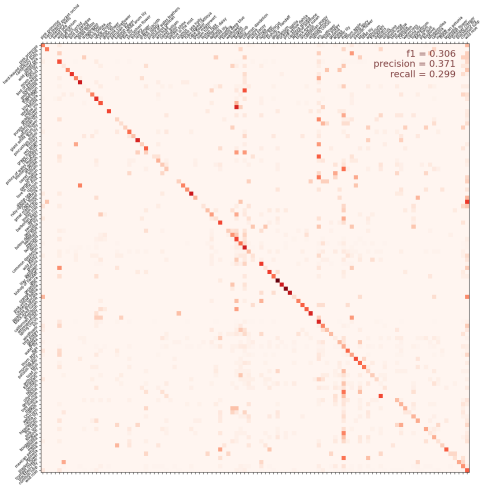

Figure 26. Confusion Matrix - Baseline Model

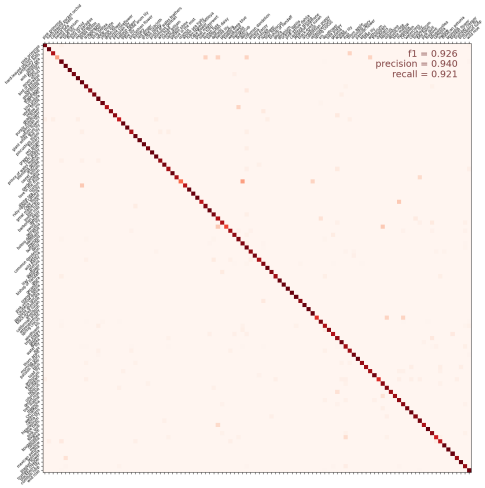

Figure 27. Confusion Matrix of the Final Model

	Training Data		Validation Data	
Model	**Accuracy**	**Loss**	**Accuracy**	**Loss**
Baseline	0.4904	1.9108	0.3887	2.4188
ResNet50	0.9103	0.3488	0.6985	2.8399
DenseNet201	0.3087	2.7189	0.1853	164.0934
Xception	0.3438	3.0831	0.3817	2.7741
Flip & De-colorization	0.8652	0.5769	0.8828	0.4533
Random White Squares & De-colorization	0.8832	0.5212	0.8960	0.4122
All data augmnetation	0.8652	0.5796	0.8828	0.4533
Final - RestNet50 with Flip & De-colorization	0.9545	0.2220	0.9367	0.2574

Table 2. All Models Performance on Training & Validation Data

8.2. Implementation & Work Breakdown

The complete version of the code is available at: https://www.kaggle.com/code/achinih/flower-classification

Both group members contributed equally to the research, discussions & coding. Report sections were divided as-

- **Achini** - Problem statement, Literature review, Methodology, Experimental setup
- **Nourin** - Abstract, Introduction, Results & Discussion, Conclusion