

**AY2024/25 SEMESTER 1**

**SC4001 - Neural Networks and Deep Learning**

| **Name** | **Matriculation Number** |
| --- | --- |
| Lam Wan Yu, Alicia | U2220458L |
| Ong Xin Ning, Trini | U2222202K |
| Yeoh Yu Yong | U2240371D |

**Table of Contents**

[**1.0 Introduction 2**](#_heading=h.gjdgxs)

[**2.0 Data Cleaning 2**](#_heading=h.1fob9te)

[2.1 Data Augmentation 2](#_heading=h.wzm75z8lfi0o)

[2.2 Mixup Augmentation 3](#_heading=h.lw12qnuyaq3b)

[2.3 Training Parameters 3](#_heading=h.61xy74zd9kwm)

[**3.0 Analysing Existing Methods and Experimentation 3**](#_heading=h.3znysh7)

[3.1 Existing Methods 3](#_heading=h.vz7qrjfvklkx)

[3.2 Our Methods 3](#_heading=h.vulftaoh1ymx)

[3.2.1 ResNet 3](#_heading=h.i5u14bu0u48y)

[3.2.2 VGG16 5](#_heading=h.pmgud4ystd08)

[3.2.3 VGG19 6](#_heading=h.m3ncs1jxwgj1)

[3.2.4 DeiT 7](#_heading=h.s28ibrqameyw)

[3.2.5 Soft Voting Ensemble 7](#_heading=h.r5mzdnfylgh2)

[**4.0 Result and Analysis 8**](#_heading=h.2et92p0)

[4.1 Individual Models 8](#_heading=h.vjto918octee)

[4.2 Soft Voting Ensemble Method 8](#_heading=h.3dlz8lc8d3x)

[**5.0 Conclusion 9**](#_heading=h.nltfna2551lz)

[6.0 References 9](#_heading=h.3dy6vkm)

# 1.0 Introduction

Deep learning has transformed how we tackle image recognition, making it possible to classify even complex images with high accuracy. In this project, we aim to classify 102 flower types using the Oxford Flowers 102 dataset, a challenging dataset with significant visual variation and similarities among classes. The goal of this project is to build a deep learning model that accurately classifies each flower type in this dataset. We experiment with popular models like ResNet, VGG and DieT, using transfer learning to take advantage of pretrained models. Our goal is to build a robust classifier that demonstrates deep learning’s potential for detailed image classification.

# 2.0 Data Cleaning

For the data cleaning process, we started with two files provided in .mat format: Imagelabel.mat and setid.mat. Since these were MATLAB files, we needed to use the loadmat function to load them into Python. The imagelabels file contained the label data for each image, while setid included information about how the images were divided into training, validation, and testing sets.

To work with the labels effectively, we converted the imagelabels data into a pandas DataFrame. This allowed us to map each image ID to its corresponding class. For the setid file, we created three separate DataFrames, one each for training, validation, and testing.

Next, we associate image filenames with their image IDs. Finally, we used pandas' merge function to bring everything together into a new DataFrame called df\_image\_labels

To prepare the dataset for training, we needed to organise the images into folders based on their classes. This meant creating separate folders for each class within each dataset type (training, validation, testing)

In order to do this, we used a for loop and the shutil package to automate this sorting process. We loop through each row in the dataframe and move each image to its corresponding folder according to its type and class.

## 2.1 Data Augmentation

To make the training data more varied and help prevent overfitting, we used a few data augmentation techniques:

* Horizontal Flip: Randomly flips images to give variety in orientation.
* Rotation: Rotates images up to 20 degrees to add variation in positioning.

We also resize all images to 224 x 224 pixels to match the input size expected by most neural networks. This resizing keeps the image dimensions consistent for batch processing. Each image is converted from a PIL image to a PyTorch tensor and then normalised to match the ImageNet dataset. Since ResNet18 was pre trained on ImageNet, this normalisation makes our images similar to what the model was originally trained on which helps it to perform better.

## 2.2 Mixup Augmentation

We added another technique called MixUp to improve generalisation of deep learning models. MixUp generates new training samples by mixing two random samples from the batch and their corresponding labels, creating a linear combination of both images and labels. Here’s a breakdown of how the code works:

## 2.3 Training Parameters

For all models, we used the same base function for training

* 2 phases, training first then validation to allow an all in one functionality increasing code readability
* Application of mixup in training phase
* Calculation of loss with mixup

# 3.0 Analysing Existing Methods and Experimentation

## 3.1 Existing Methods

In the original paper, Automated Flower Classification Over a Large Number of Classes, by Nilsback and Zisserman at the University of Oxford, an advanced multi-feature Support Vector Machine (SVM) model employing a one-vs-rest classification strategy was proposed. The authors first generated image segmentations, which we will also be using, followed by feature extraction across multiple visual aspects. They then applied a linear combination of kernels, with each kernel corresponding to a specific feature type. Chi-squared distance measures were incorporated within these kernels, and feature-specific weights were assigned, enabling the model to prioritize distinct visual cues depending on the flower class. This approach allowed for accurate classification based on characteristics such as colour and shape, achieving a maximum test accuracy of 72.8%, which was highly commendable for the time (2008) (Nilsback & Zisserman, 2008). However, with the recent advancements in computational power and the widespread availability of neural network and transformer architectures, image classification has progressed significantly, and more sophisticated methods now allow for higher accuracy in image classification tasks.

## 

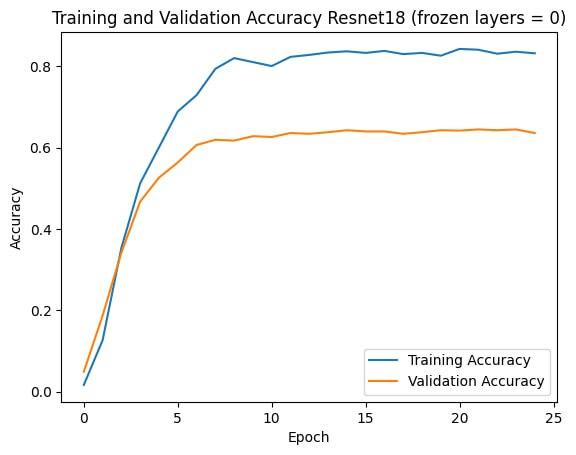
## 3.2 Our Methods

### 3.2.1 ResNet

Residual Networks (ResNet) is a CNN-based architecture that introduces residual connections, which helps mitigate the vanishing gradient problem in deep learning networks. The residual connection is a shortcut that allows the information to bypass one or more layers in the network and reach the output directly. It allows the network to learn the residual function and make small updates to the parameters, which enables the network to converge faster and achieve better performance (Banjara, 2024).

We started with ResNet50, but the 50-layer depth led to a vanishing gradient problem, meaning that the gradients became too small since they are back-propagated through the network with many layers, making it difficult for the model to effectively learn and update weights, particularly when the dataset is smaller.

To solve this, we switched to ResNet18, a simpler version of ResNet with 18 layers. For this model, we initially tested with 0 frozen layers, allowing all layers to be trained to fit our dataset. However, this led to high training accuracy and poor validation accuracy, which indicated that the model was overfitting.

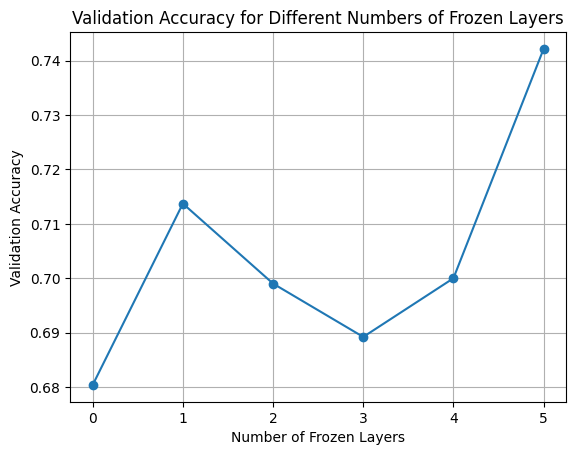


To address this, we froze the initial layers of the model. Freezing layers limits the number of trainable parameters, allowing the model to retain its general features from pretraining and reducing the risk of overfitting. Other hyperparameter settings include the choice of 20 epochs with early stopping patience of 5, as we noticed that the model does not show any significant improvement after around 10 epochs for the 0 frozen layers case. This ensured that we gave enough buffer for the different layers to train and prevented overfitting.

We started with a learning rate (lr) of 0.001, and implemented a learning rate scheduler such that the learning rate decreases by 0.1 (new\_lr = lr x 0.1) every 7 epochs, this ensures finer adjustments as the model approaches convergence and prevents the model from overshooting the optimal solution.

The optimizer of choice was Adam, an adaptive optimizer that adjusts learning rate based on first and second moments of gradients. This works well for fine-tuning pretrained models because it is less sensitive to initial learning rate and converges faster.

After testing by freezing 0 to 5 layers, we found the best validation accuracy (0.7422) came from freezing the first five layers.



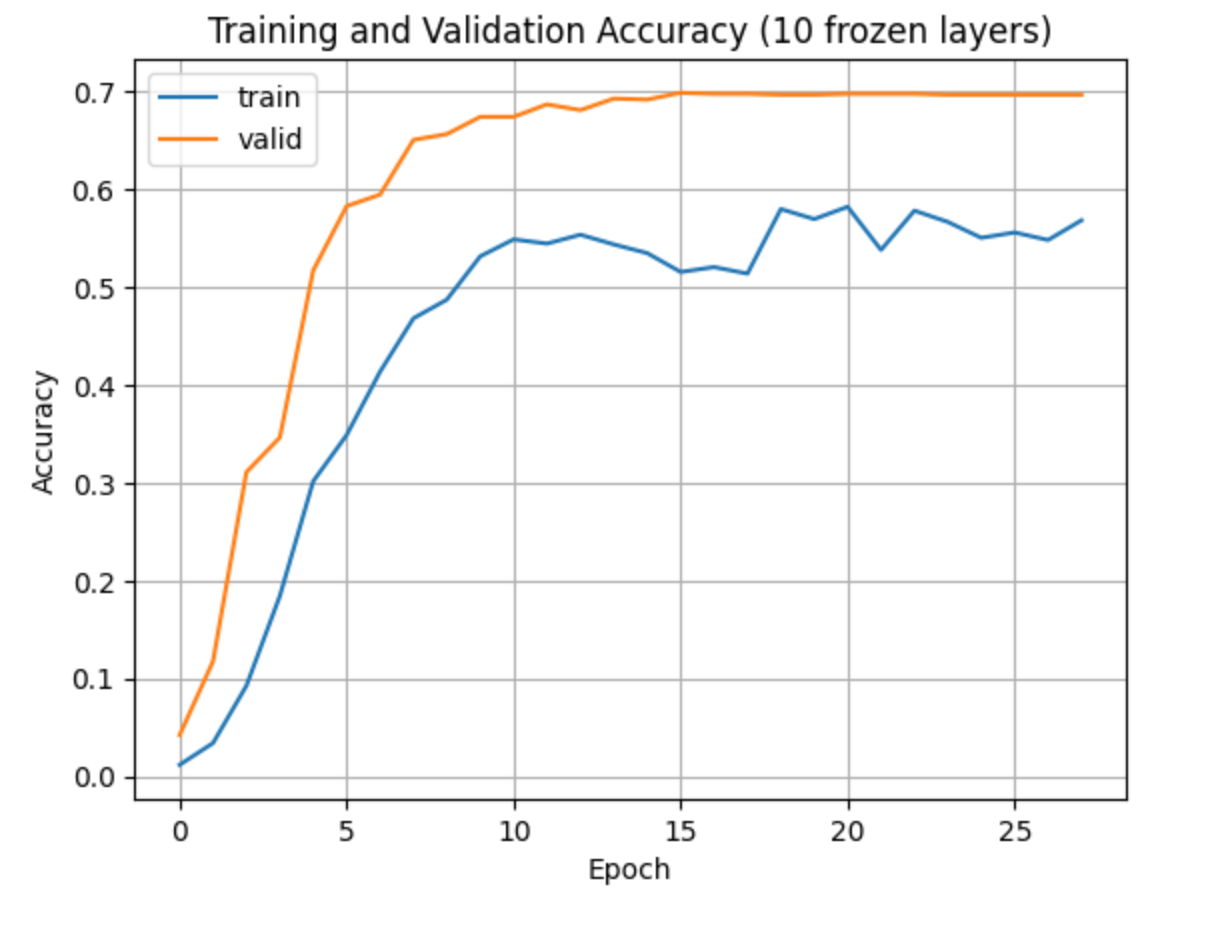
### 3.2.2 VGG16

Visual Geometry Group (VGG) is also a CNN-based architecture. The VGG architecture maintains a consistent arrangement of convolutional (with 3x3 filter) and max-pooling layers throughout, allowing it to effectively capture hierarchical features. At the end of the network, two fully connected layers are used to consolidate these features, followed by a soft max layer for output classification. Similarly to ResNet18, it was trained on ImageNet.

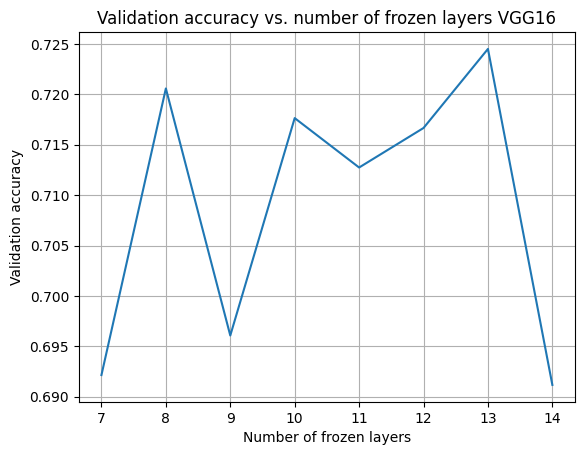
For this model, we switched to using an SGD optimizer instead of Adam because VGG16 is a larger pre-trained model than ResNet18, and hence SGD with momentum encourages a stable and flatter optimization path compared to Adam. Furthermore, VGG16 includes batch normalization layers which works well with SGD while Adam might interfere with the internal scaling dynamics of batch normalization.

The learning rate scheduler remained the same, but we increased the maximum number of epochs to 30, taking into account the increased depth of the model, but kept early stopping with patience of 5 to prevent overtraining.

We initially froze 10 layers, but the validation accuracy was low at less than 0.60.

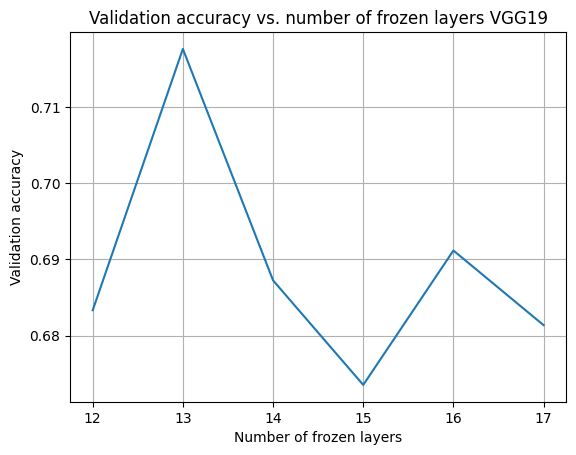


After experimenting with different numbers of frozen layers, we discovered that freezing 8 layers gave us the best validation accuracy for this model.



### 3.2.3 VGG19

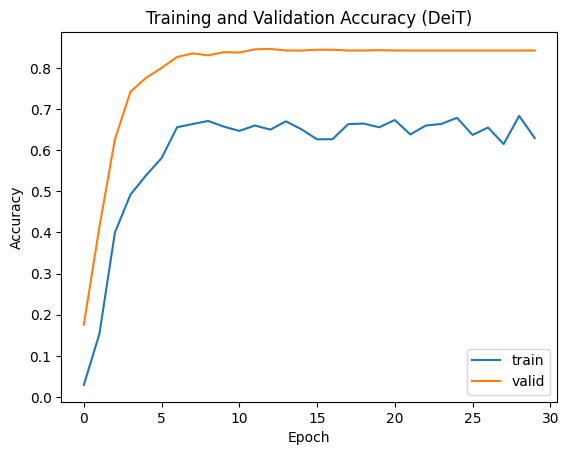
For VGG19, we used a similar hyperparameter set up as with VGG16. Similarly to VGG16, we tested the validation accuracies of freezing a range of layers to determine the optimal layers to be frozen. We achieved the best validation accuracy by freezing 14 layers.



### 3.2.4 DeiT

Data-Efficient Image Transformer (DieT) is a transformer-based architecture designed to achieve high performance in image classification tasks with significantly less data compared to traditional vision transformers using knowledge distillation and strong data augmentation.

We used the same maximum epochs, early stopping patience and learning rate as the VGG model.



### 3.2.5 Soft Voting Ensemble

To further improve prediction accuracy, we implemented the soft voting ensemble method. Soft voting combines the probabilities of individual models to make a prediction. Each model provides the probabilities, and the ensemble will aggregate them, predicting the class with the highest average probabilities.

For our experiment, we made use of the above 4 models - ResNet18, VGG16, VGG19 and DieT, using different combinations for the ensemble. This will be discussed in the next section.

# 4.0 Result and Analysis

## 4.1 Individual Models

| **Model** | **Test Accuracy** |
| --- | --- |
| ResNet18(4 Frozen Layers) | 64.32% |
| VGG16 | 73.23% |
| VGG19 | 71.05% |
| DeiT | 80.37% |

Test Accuracy Ranking: DeiT > VGG16 > VGG19 > ResNet18

The significantly lower test accuracy for ResNet18 indicates the model struggles with the complexity of the classification since it is more suited for tasks where computational efficiency is a priority (Product Teacher, n.d.), which is a trade-off to maximising accuracy by learning more details.

VGG models have a better accuracy which could be due to the deeper architectures compared to ResNet (Jain, 2023), allowing it to capture and differentiate more complex patterns in the images. VGG16 performed better than VGG19, indicating that the additional 3 layers in VGG19 may have caused overfitting to occur, and the model cannot generalise on unseen data as well.

DeiT significantly outperforms the other models which could be attributed to it being suitable for smaller training datasets (Zubair, 2024), like ours with 10 images per class (1020 images total). The transformer-based approach is likely better suited for capturing complex dependencies and relationships within the images, which may be critical in distinguishing between 102 classes.

## 4.2 Soft Voting Ensemble Method

|  | **Combinations** | **Test Accuracy** |
| --- | --- | --- |
| Model X | VGG16, VGG19, ResNet18 | 76.66% |
| Model Y | VGG16, VGG19, DeiT | 80.01% |
| Model Z | VGG16, VGG19, ResNet18, DeiT | 80.71% |

Test Accuracy Ranking: Model Z > Model Y > Model X

Overall, there is a slight improvement in the test accuracies for the ensembles compared to their respective individual models by themselves. This could be attributed to the fact that all 4 models have different architectures and may be more confident in predicting certain classes over others. Taking the average probabilities allows the ensemble model to cover wider ranges of classes, which is especially important in our case with 102 classes.

For Model X, it leverages on the strengths of convolutional networks, which excel at capturing localised features. CNNs treat images as structured pixel arrays and process them with convolutions. Through these trainable convolutional filters, CNNs create feature maps, which are representations of the original image. This means that the convolution operation affects only a small patch of an image at a time, and can be considered a local operation (Poulinakis, 2022). The lack of a transformer-based model, such as DeiT, might limit its ability to capture complex global patterns.

For Model Y, the accuracy is much higher after replacing ResNet18 with DeiT. DeiT, being a transformer, operates on a more global scale (Poulinakis, 2022). This allows the ensemble method to capture both smaller details of features and global pattern recognition, boosting the performance of the ensemble.

For Model Z, we observe the addition of ResNet18 to the previous (Model Y) ensemble results in minimal improvement in accuracy (increase of 0.70%). This suggests that ResNet18 adds little value to the ensemble and can be excluded from the ensemble. Hence, we prefer Model Y as it is simpler than Model Z.

# 5.0 Conclusion

In this study, we explored various models and ensemble techniques to classify images into 102 distinct categories for flower image classification. By examining the performance of individual models such as ResNet18, VGG16, VGG19, and DeiT, we observed that transformer-based models like DeiT outperformed convolutional networks on our relatively small dataset. Soft voting ensemble strategy combined the strengths of both convolutional networks and transformers, allowing for learning on both fine-grained details of images and broader contextual information, achieving improved results over individual models. Overall, we chose the ensemble of VGG16, VGG19 and DieT (Model Y) to be the best combination.

Future work could explore additional transformer methods or ensemble methods to push performance even further.

# 6.0 References

M. -E. Nilsback and A. Zisserman (2008). “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. keywords: {Shape;Kernel;Distributed computing;Support vector machines;Support vector machine classification;object classification;segmentation},

Product Teacher (n.d.) ResNet18 & ResNet50 in Computer Vision

Retrieved from <https://www.productteacher.com/quick-product-tips/resnet18-and-resnet50#:~:text=ResNet18%20consists%20of%2018%20layers,where%20computational%20resources%20are%20limited>

Jain, Dr & Bora, Mayank & Chandnani, Sameer & Grover, Sanidhay & Sadwal, Shivank. (2023). Comparison of VGG-16, VGG-19, and ResNet-101 CNN Models for the purpose of Suspicious Activity Detection. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 121-130. 10.32628/CSEIT2390124.

Zubair (2024, July 12). DeiT: Data-efficient Image Transformer Overview

Retrieved from <https://medium.com/@zakhtar2020/deit-data-efficient-image-transformer-overview-acd1cb3b1dcf>

[Konstantinos Poulinakis](https://www.linkedin.com/in/konstantinos-poulinakis-4554821a3/) (2022, August 22). Are Transformers replacing CNNs in Object Detection?

Retrieved from <https://www.picsellia.com/post/are-transformers-replacing-cnns-in-object-detection#5-How-and-Why-Are-ViTs-Different-From-CNNs->