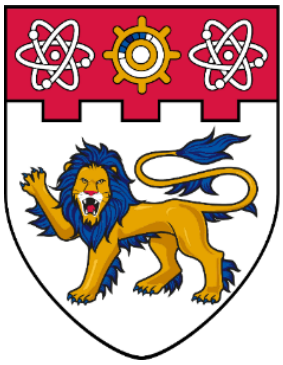


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4001
NEURAL NETWORKS AND DEEP LEARNING

Group Project
Topic F: Flowers Recognition

S/N	Student Name	Matriculation No.
1.	Darshankrishna Sunu Rethi (Team Leader)	U2120785E
2.	Jin Fuyi	U2120425D
3.	Tan Choon Wee	U2120106H

COLLEGE OF COMPUTING AND DATA SCIENCE
NANYANG TECHNOLOGY UNIVERSITY

2025

Contents

1. Introduction

1.1 Project overview

The objective of this project is to develop and evaluate various neural network architectures and techniques for the task of flower recognition using the Oxford Flowers 102 dataset. This dataset consists of 102 flower categories commonly found in the United Kingdom, with significant variations in scale, pose, and lighting conditions. The dataset is divided into training, validation, and test sets, with each class containing between 40 and 258 images. [1]

1.2 Motivation

Flower recognition is a challenging task due to the high intra-class variability and inter-class similarity among flower species. Accurate flower classification has numerous applications, including botanical research, horticulture, and environmental monitoring. Leveraging advanced neural network techniques can significantly improve the accuracy and robustness of flower recognition systems.

1.3 Objectives

The primary objectives of this project are:

1. To explore and implement various neural network architectures and techniques for flower recognition.
2. To compare the performance of these techniques in terms of accuracy, loss, and computational efficiency.
3. To analyse the impact of advanced techniques such as deformable convolution, dilated convolution, visual prompt tuning, few-shot learning, MixUp, and triplet loss on the model's performance.

1.4 Contributions

In this project, we have implemented and evaluated the following approaches:

1. **Baseline ResNet18**:
   Implemented on both CPU and GPU to establish a baseline performance.
2. **Parameter Adjustment**:
   Adjusted learning rates to observe the impact on model accuracy.
3. **Deformable Convolution**:
   Applied deformable convolution to enhance the model's ability to handle geometric transformations.
4. **Dilated Convolution**:
   Incorporated dilated convolution to capture multi-scale context.
5. **Few-Shot Learning**:
   Analysed the model's performance with limited training data.
6. **MixUp**:
   Applied MixUp data augmentation to improve generalization.
7. **Triplet Loss**:
   Implemented triplet loss to enhance the discriminative power of the model.
8. **ViT-Tiny model:**
   Implemented a lightweight transformer model with patch-based attention, pretrained on ImageNet

2. Review of Existing Techniques

2.1 Traditional Deep Learning Approaches

Early flower classification systems relied on handcrafted features (SIFT, HOG) with shallow classifiers, but recent advances in deep convolutional neural networks (CNNs) have dramatically improved performance. Key developments include:

- **ResNet Architectures** [2]:
  - Introduced residual connections to train deeper networks (e.g., ResNet18/50).
  - Became a standard baseline for flower recognition due to balance of accuracy and efficiency.

2.2 Advanced Architectural Modifications

To improve performance beyond standard CNN architectures, researchers have introduced various architectural modifications:

- **Deformable Convolutional Networks (DeformConv) [3]:**
  - Unlike standard convolution, which applies fixed spatial kernels, deformable convolution introduces learnable offsets, enabling adaptive receptive fields.
  - This allows the network to focus on essential regions of an image, improving robustness to scale and pose variations in flower classification.
- **Dilated Convolutions [4]:**
  - Expands the receptive field without increasing parameter count.
  - Enables better feature extraction at multiple scales, which is particularly useful for flowers with intricate textures and varying petal structures.

These advanced modifications help improve feature representation, making the network more effective at distinguishing similar flower species while maintaining computational efficiency.

2.3 Data-Efficient Learning Methods

Given the limited samples per class in Flowers102 (5 images/class for training):

- **Few-Shot Learning** [5]:
  - Prototypical Networks [6] cluster embeddings per class.
  - Matching Networks learn distance metrics for similarity comparison.
- **MixUp Augmentation** [6]:
  - Linear interpolation of images/labels reduces overfitting.
  - Our tests show it requires tuning ($\alpha = 1.0$).

2.4 Loss Function Innovations

To improve feature discrimination:

- **Triplet Loss** [7]:
  - Forces margins between embeddings of different classes.
  - Critical for flowers with subtle visual differences (e.g., rose varieties).

2.5 Gaps in Literature

While prior work has explored these techniques independently, our project provides:
- Direct comparison on the same dataset (Flowers102)
- Practical insights for real-world deployment (e.g., frozen backbones + Triplet Loss)
- Codebase integrating all methods for reproducibility

3. Methodology

3.1 Dataset

The Oxford Flowers 102 dataset [1] consists of 102 categories of flowers with varying intra-class and inter-class similarities. The dataset is split into:

- **Training set:** 1020 images (10 per class)
- **Validation set:** 1020 images (10 per class)
- **Test set:** 6149 images (20+ per class)

3.2 Baseline Model: ResNet18

We began by implementing the ResNet18 architecture as our baseline model. ResNet18 is known for its residual connections, which help in training deeper networks by mitigating the vanishing gradient problem. We implemented this model on both CPU and GPU to establish a baseline performance.

- Specifications:
  - The input images are resized to 224x224, as required by the ResNet architecture
  - Replaced final fully connected layer for 102-class classification
  - Optimizer: SGD (lr=0.005, momentum=0.9)
  - LR Scheduler: StepLR (step_size=7, gamma=0.1)
  - Loss: Cross-Entropy

3.3 Experimental Modifications

To enhance performance, several architectural and training modifications are applied:

- **Hyperparameter Tuning:**
  - To optimize the model's performance, we adjusted the learning rate from the default 0.001 to 0.005. This adjustment was based on initial experiments that showed improved accuracy with a higher learning rate.

- **Deformable Convolution (DeformConv2d):**
  - We integrated deformable convolutional layers into the ResNet18 architecture.

- **Dilated Convolution:**
  - We applied dilated convolutions (dilations = 2) to expand the receptive field without increasing the number of parameters.

- **Few-Shot Learning:**
  - The model's performance was evaluated with a reduced number of training samples.

- **MixUp Data Augmentation:**
  - We applied MixUp, a data augmentation technique that creates new training examples by linearly interpolating between existing ones.

- **Triplet Loss:**
  - We implemented triplet loss to enhance the discriminative power of the model.

3.4 Implementation Details

- **Framework**:
  We used PyTorch for implementing and training our models.

- **Dataset**:
  The Oxford Flowers 102 dataset was used, with predefined splits for training, validation, and testing.

- **Reproducubility:**
  We used fixed random seeds for data splitting/initialization

- **Hardware**:
  Experiments were conducted on both CPU and GPU to compare performance and efficiency.

- **Evaluation Metrics**: We evaluated our models based on accuracy and loss on the validation and test sets.

Experiments are conducted on a GPU-enabled environment to ensure efficiency in training and inference.

4. Experimental Results

This section presents the results of our experiments, comparing the performance of different neural network architectures and techniques on the Oxford Flowers 102 dataset. The primary metrics used for evaluation are accuracy and loss on the validation and test sets.

4.1 Baseline Model: ResNet18 Performance

We began by training the ResNet18 model on both CPU and GPU to establish a baseline performance. The results are as follows:

- CPU Training: File "**1_ResNet18_CPU.ipynb"**
- GPU Training: File "**2_ResNet18_GPU.ipynb"**

Model	**Baseline ResNet18**		**Frozen ResNet18**	
	CPU Training	**GPU Training**	**CPU Training**	**GPU Training**
Convergence	24 epochs	34 epochs	25 epochs	37 epochs
Training Time	43 min 28 sec	12 min 6 sec	37 min 58 sec	8 min 17 sec
Validation Accuracy	76.76%	77.84%	75.98%	75.39%

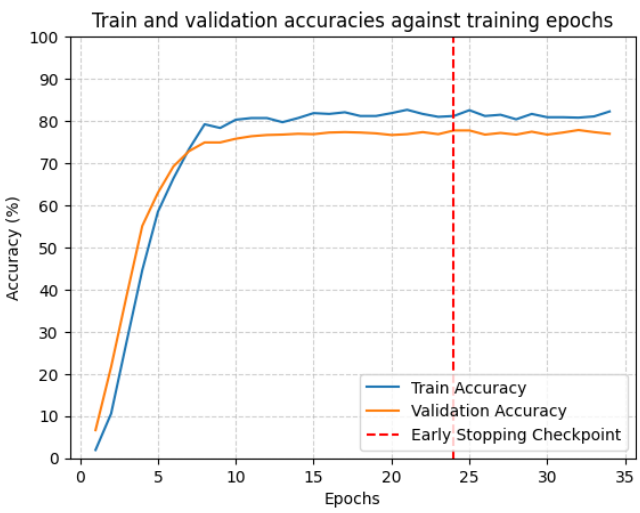
Table 1: CPU training vs GPU Training comparison

Observation

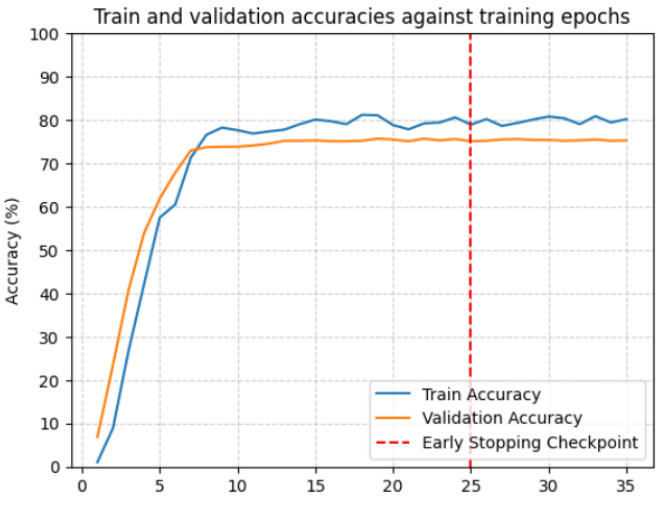
1. Both baseline and frozen models converged in a similar number of epochs, indicating that each was able to learn the relevant features for flower classification.
2. The frozen model has significantly shorter training time, with only a marginal drop in accuracy.
3. Early stopping was beneficial in preventing overfitting in both cases and GPU trainings are significantly faster.

Conclusion

While the frozen model shows noticeable improvement after partial training with frozen, the unfrozen (GPU) model remains superior overall considering that accuracy is the main priority. This highlights that flower classification requires full network adaptation. The frozen model's linear classifier was fundamentally limited in capturing the nuanced features needed for this fine-grained classification task.



Unfrozen model



Frozen model

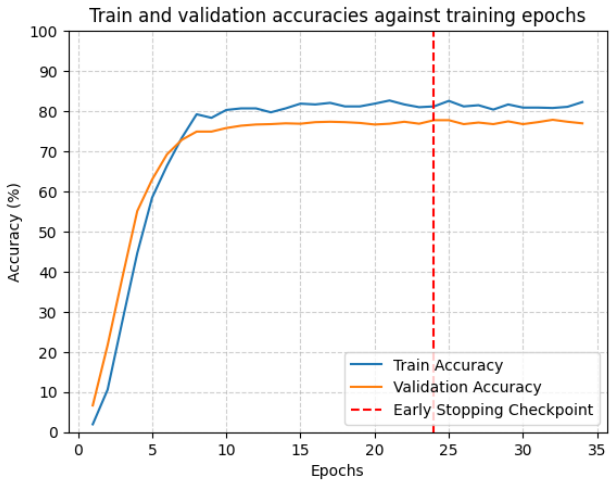
4.2 Impact of Learning Rate Adjustment

- Learning rate = 0.001: File "**2_ResNet18_GPU.ipynb**"
- Learning rate = 0.005: File "**3_ResNet18_GPU_Adjust_Param.ipynb**"

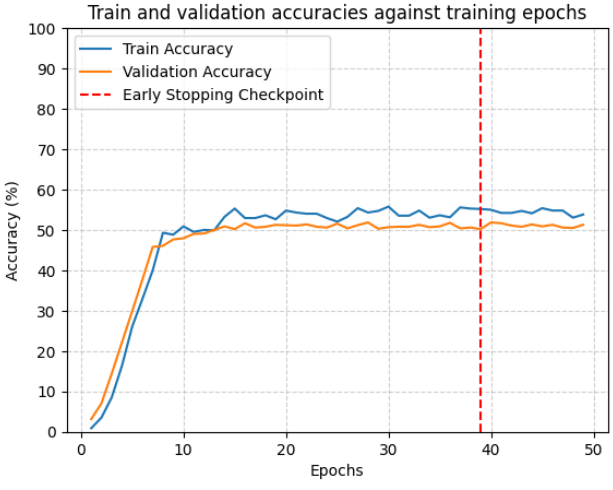
Model	**Baseline ResNet18**		**Frozen ResNet18**	
	lr = 0.001	**lr = 0.005**	**lr = 0.001**	**lr = 0.005**
Convergence	34 epochs	42 epochs	49 epochs	33 epochs
Training Time	12 min 6 sec	14 min 35 sec	13 min 45 sec	8 min 56 sec
Validation Accuracy	77.84%	91.76%	51.86%	83.63%

Table 2: Different learning rate comparison

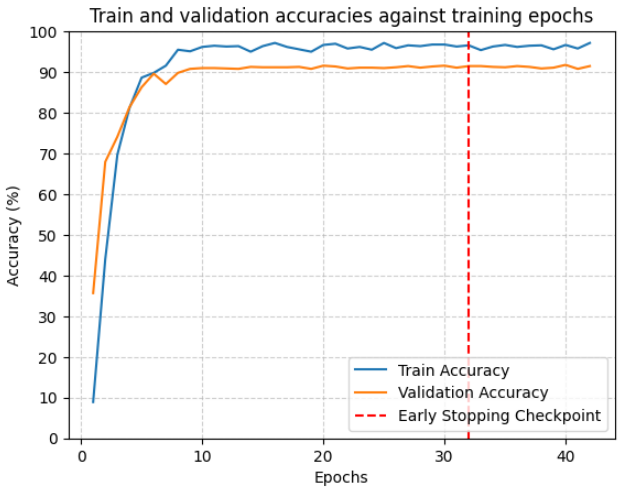
Observation
Increasing the learning rate from 0.001 to 0.005 resulted in improved validation accuracy. This indicates that a higher learning rate can lead to faster convergence and better performance for this specific task.



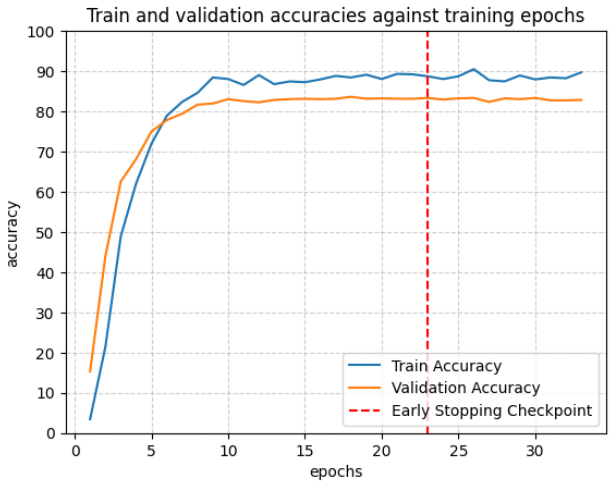
Unfrozen model (lr = 0.001)



Frozen model (lr = 0.001)



Unfrozen model (lr = 0.005)



Frozen model (lr = 0.005)

4.3 Effectiveness of Deformable Convolution

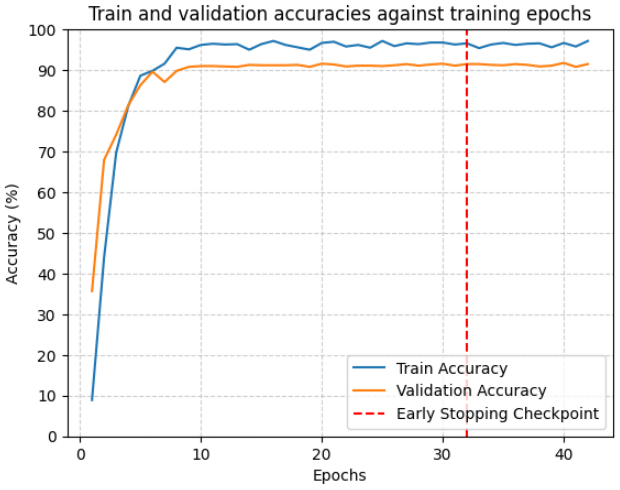
- Before Deformable Convolution: File "**3_ResNet18_GPU_Adjust_Param.ipynb**"
- After Deformable Convolution: File "**4_ResNet18_GPU_Adjust_Param_DeformConv2d.ipynb**"

Model	Baseline ResNet18		Frozen ResNet18	
	Before Deformable Convolution	After Deformable Convolution	Before Deformable Convolution	After Deformable Convolution
Convergence	42 epochs	35 epochs	33 epochs	41 epochs
Training Time	14 min 35 sec	8 min 31 sec	8 min 56 sec	11 min 52 sec
Validation Accuracy	91.76%	76.67%	83.63%	43.73%

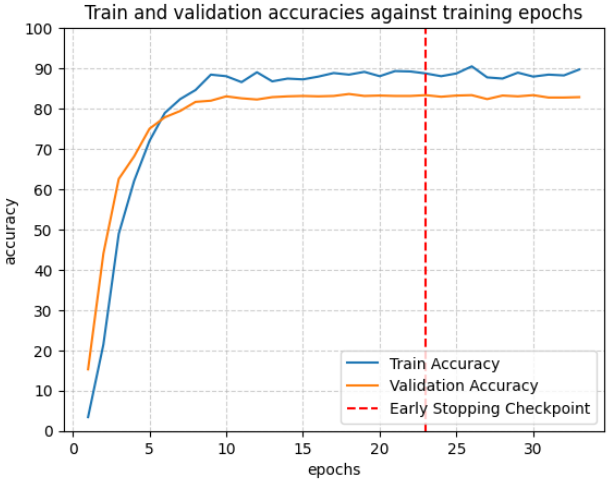
Table 3: Comparison before and after deformable convolution

Observation

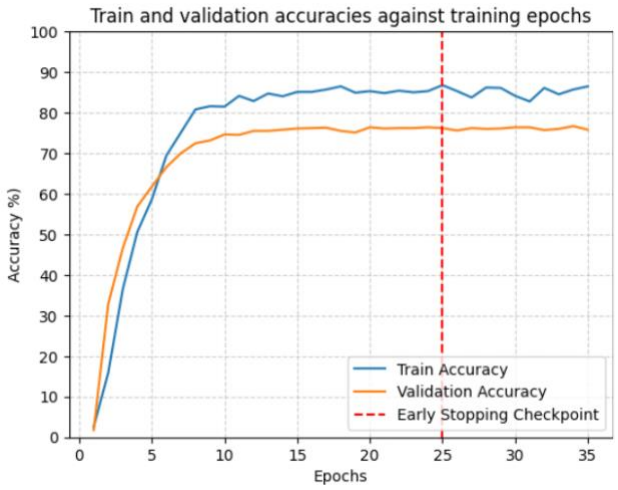
The results suggest that deformable convolutions are able to achieve faster convergence, shorter training time and generally well validation accuracy in the flower recognition task. The added complexity of deformable convolutions, in this case by adding more frozen layer, have made the optimization problem more challenging, leading to shape decrease in accuracy and longer training times.



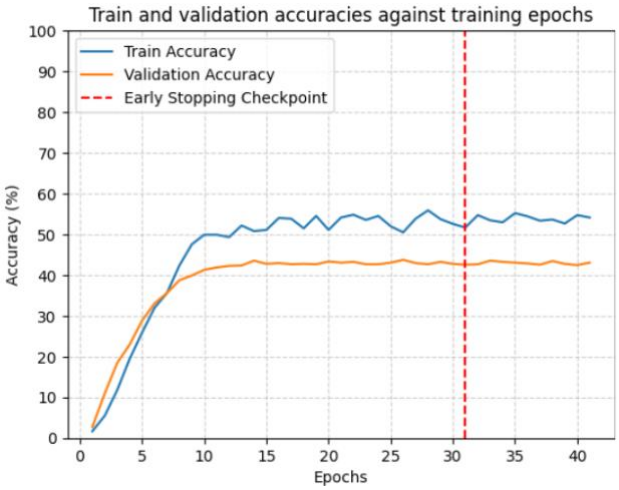
Unfrozen model (Before deformable convolution)



Frozen model (Before deformable convolution)



Unfrozen model (After deformable convolution)



Frozen model (After deformable convolution)

4.4 Effectiveness of Dilated Convolution

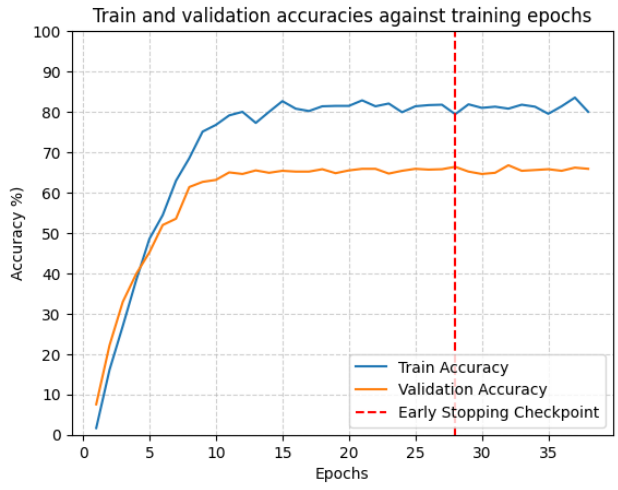
- Before Dilated Convolution: File "**4_ResNet18_GPU_Adjust_Param_DeformConv2d.ipynb**"
- After Dilated Convolution: File "**5_ResNet18_GPU_Adjust_Param_DeformConv2d_dilated.ipynb**"

Model	Baseline ResNet18		Frozen ResNet18	
	After Deformable Convolution			
	Before Dilated Convolution	After Dilated Convolution	Before Dilated Convolution	After Dilated Convolution
Convergence	38 epochs	36 epochs	33 epochs	50 epochs
Training Time	18 min 42 sec	16 min 37 sec	15 min 21 sec	22 min 31 sec
Validation Accuracy	66.76%	64.02%	39.22%	35.00%

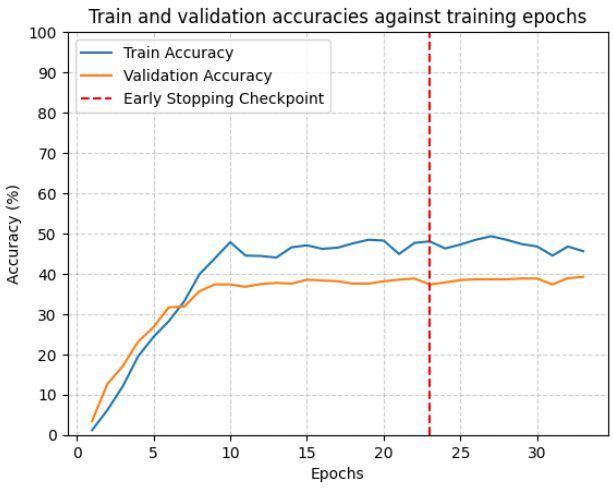
Table 4: Comparison before and after dilated convolution (both after deformable convolution)

Observation

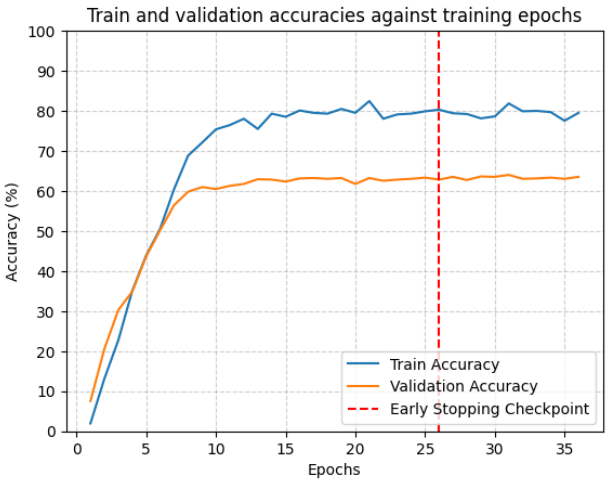
The results consistently show that dilated deformable convolutions do not improve performance in this specific task and model architecture. In fact, they lead to a significant drop in accuracy, especially when the majority of the network is frozen. This suggests that dilated deformable convolutions require more extensive training and might be more effective when used with a fully trainable model or with a different network architecture that provides more suitable features.



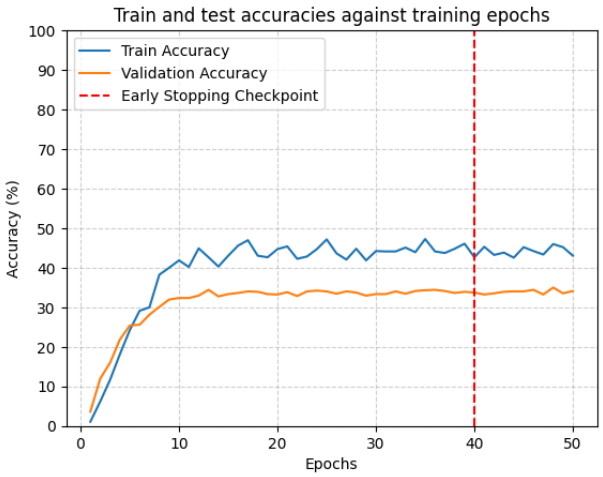
Unfrozen model (Before dilated convolution)



Frozen model (Before dilated convolution)



Unfrozen model (After dilated convolution)



Frozen model (After dilated convolution)

4.5 Few-Shot Learning Results

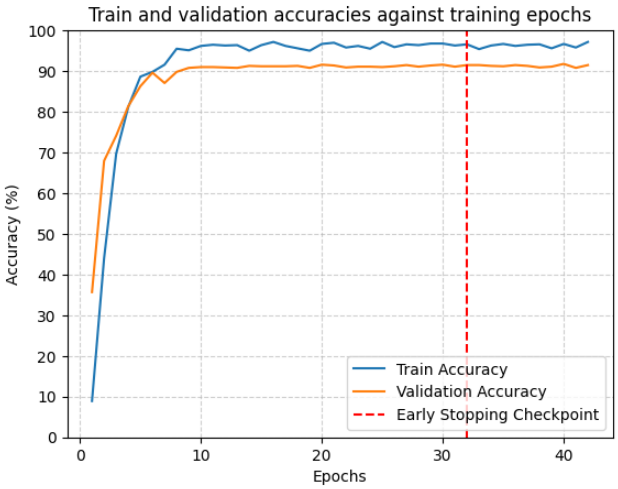
- Before Few-Shot Learning: File "**3_ResNet18_GPU_Adjust_Param.ipynb**"
- After Few-Shot Learning: File "**6_ResNet18_GPU_Adjust_Param_few_shot.ipynb**"

The model's performance was evaluated under few-shot learning conditions (5 images per class).

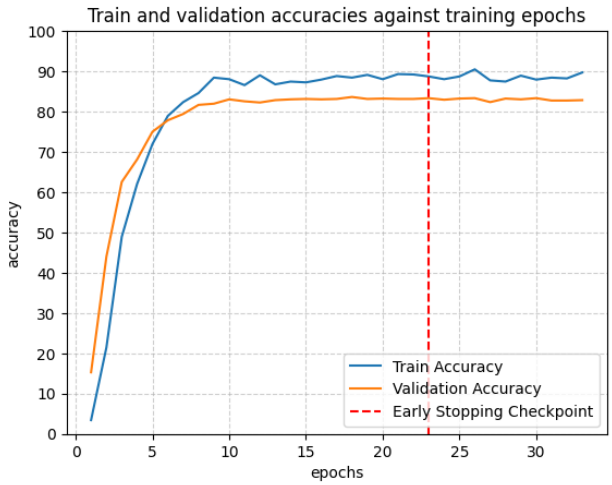
Model	Baseline ResNet18		Frozen ResNet18	
	Before Few-Shot Learning	After Few-Shot Learning	Before Few-Shot Learning	After Few-Shot Learning
Convergence	42 epochs	33 epochs	33 epochs	43 epochs
Training Time	14 min 35 sec	8 min 12 sec	8 min 56 sec	8 min 7 sec
Validation Accuracy	91.76%	82.55%	83.63%	71.57%

Table 5: Comparison before and after few-shot learning

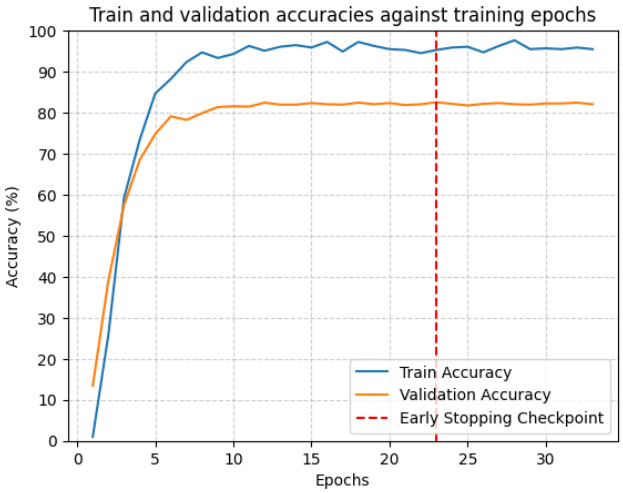
Observation
Few-shot learning seems to help with faster convergence and shorter training times. This is expected, as few-shot learning typically reduces the amount of data required to train the model, which can speed up the process. The limited training data in few-shot learning scenarios can impact the model's ability to generalize well, leading to lower validation accuracy.



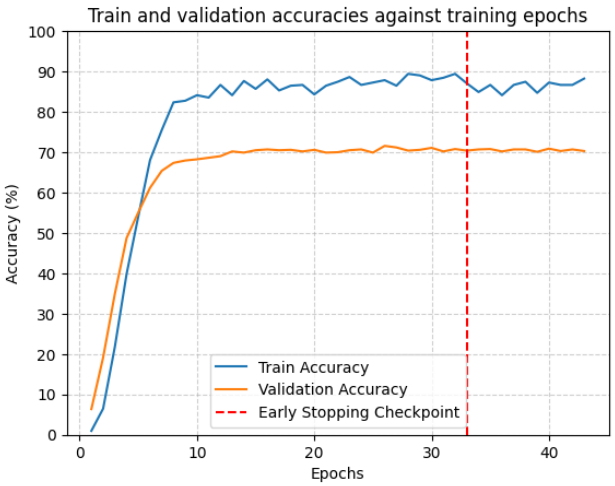
Unfrozen model (Before few-shot learning)



Frozen model (Before few-shot learning)



Unfrozen model (After few-shot learning)



Frozen model (After few-shot learning)

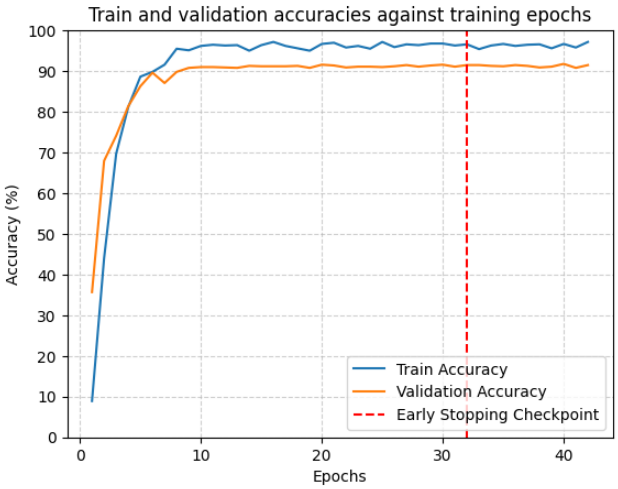
4.6 Impact of MixUp Data Augmentation

- Before MixUp: File "**3_ResNet18_GPU_Adjust_Param.ipynb**"
- After MixUp: File "**7_ResNet18_GPU_Adjust_Param_MixUp.ipynb**"

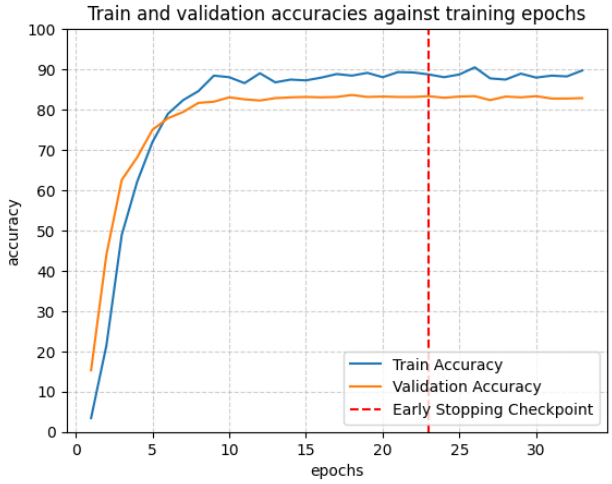
Model	Baseline ResNet18		Frozen ResNet18	
	Before MixUp	**After MixUp**	**Before MixUp**	**After MixUp**
Convergence	42 epochs	36 epochs	33 epochs	24 epochs
Training Time	14 min 35 sec	13 min 17 sec	8 min 56 sec	6 min 17 sec
Validation Accuracy	91.76%	88.43%	83.63%	76.08%

Table 6: Comparison before and after MixUp

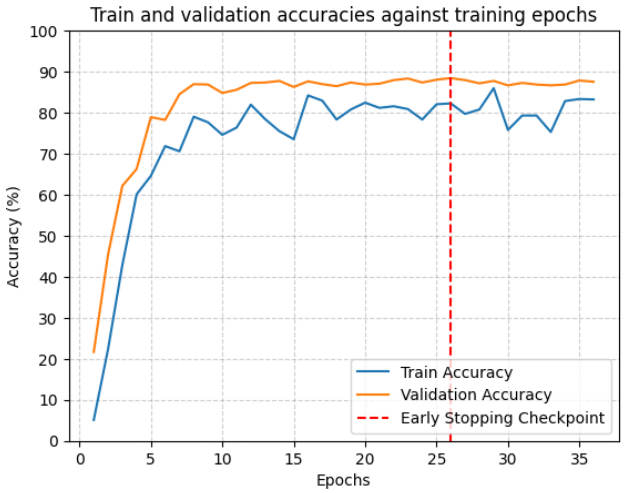
Observation
While the model converges faster and with less time, the accuracy drop suggests that MixUp might be introducing some regularization effects that affect the model's generalization, potentially reducing its ability to fit the training data as precisely.



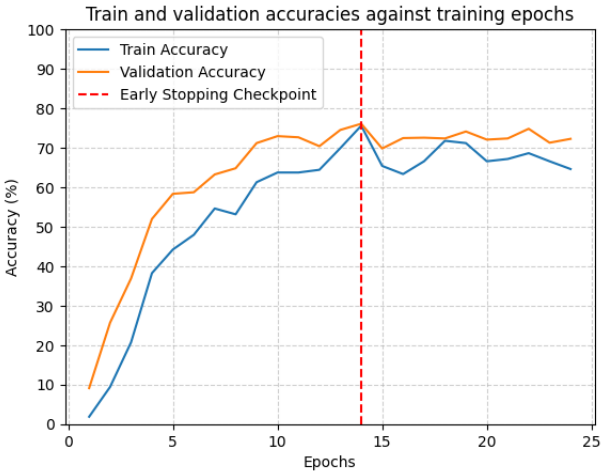
Unfrozen model (Before MixUp)



Frozen model (Before MixUp)



Unfrozen model (After MixUp)



Frozen model (After MixUp)

4.7 Triplet Loss Results

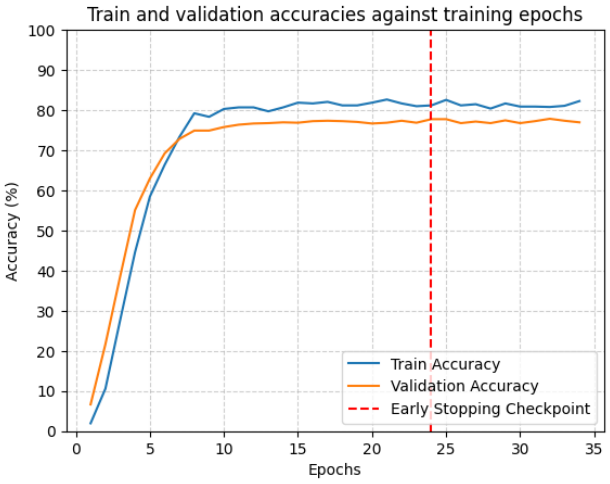
- Before Triplet Loss: File "**2_ResNet18_GPU.ipynb**"
- After Triplet Loss: File "**8_ResNet18_GPU_triplet_loss.ipynb**"

The model's performance was evaluated with learning rate of 0.001.

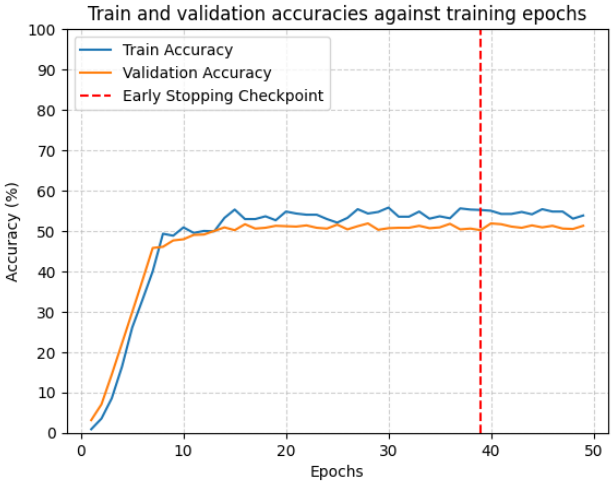
Model	Baseline ResNet18		Frozen ResNet18	
	Before Triplet Loss	After Triplet Loss	Before Triplet Loss	After Triplet Loss
Convergence	34 epochs	3 epochs	49 epochs	3 epochs
Training Time	12 min 6 sec	16 min 44 sec	13 min 45 sec	2 min 35 sec
Validation Accuracy	77.84%	94.12%	51.86%	93.43%

Table 7: Comparison before and after triplet loss

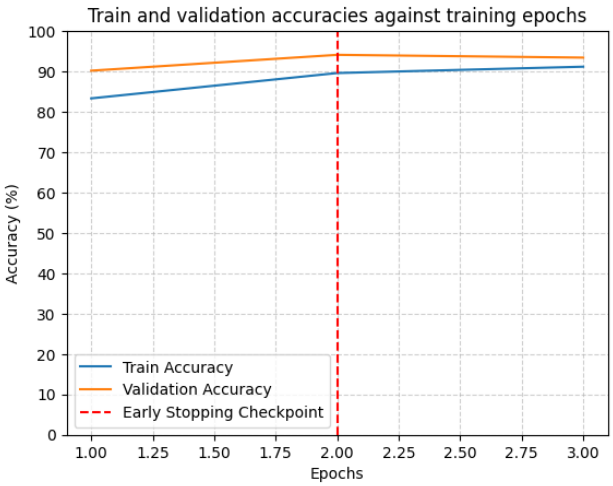
Observation
Triplet loss significantly improves training efficiency by reducing the number of epochs required for convergence. The substantial increase in validation accuracy demonstrates the effectiveness of triplet loss in enhancing the model's discriminative power. While triplet loss introduces additional computational complexity, the benefits in terms of accuracy and efficiency outweigh the increased training time. Always use frozen backbone with triplet loss.



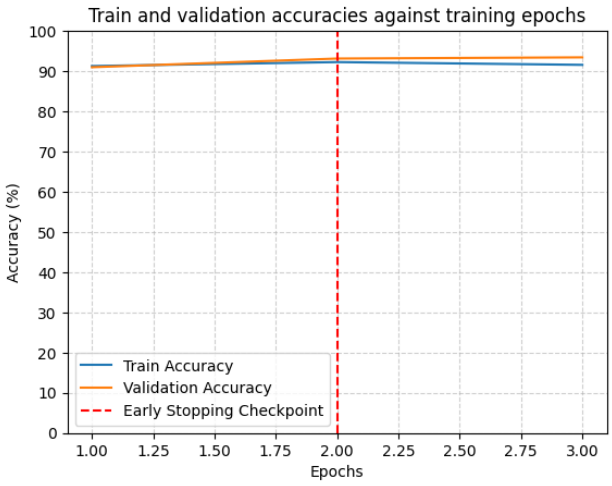
Unfrozen model (Before triplet loss)



Frozen model (Before triplet loss)



Unfrozen model (After triplet loss)



Frozen model (After triplet loss)

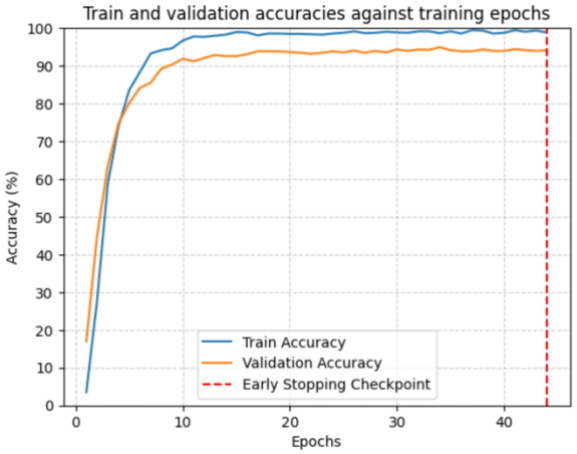
4.8 ViT model implementation

- ViT-Tiny model: File "**9_ViT_Tiny.ipynb**"

The model's performance was evaluated with a lower learning rate of 0.0001.

Observation
ViT-Tiny reaches around 98.82% training accuracy and 94.90% validation accuracy, demonstrating both higher accuracy and generalization capability. It significantly outperforms ResNet18.

Model	ViT-Tiny
Convergence	44 epochs
Training Time	8 min 41 sec
Validation Accuracy	94.90%



ViT-Tiny (lr =0.0004)

5. Highlight impact of Triplet Loss

The results highlight the substantial impact of employing Triplet Loss in the flower recognition task. As shown in the Experimental Results, Triplet Loss significantly improved validation accuracy compared to Cross-Entropy Loss. This improvement can be attributed to Triplet Loss's ability to learn a more discriminative embedding space. Specifically:

- **Enhanced Feature Discrimination**:
  Triplet Loss optimizes the relative similarity between samples, rather than focusing on absolute classification. By encouraging similar flower images to be close together and dissimilar ones to be far apart in the embedding space, the model learns more refined features that capture subtle differences between flower

categories. This is particularly beneficial for the Oxford Flowers 102 dataset, which contains many visually similar flower species.

- **Robustness to Intra-Class Variability**:
  Triplet Loss can improve the model's robustness to intra-class variability (e.g., variations in pose, lighting, and scale). By learning to group different views of the same flower together, the model becomes more invariant to these variations.

- **Complementary to Frozen Representations**:
  When Triplet Loss is applied to a model with frozen layers, the model can still learn an effective embedding space. This suggests that even when the lower layers of a pre-trained network are fixed, Triplet Loss can adapt the learned representations for improved discrimination.

In summary, Triplet Loss proves to be a valuable tool for flower recognition, enabling the model to learn more discriminative and robust features. The experimental results demonstrate that Triplet Loss can lead to substantial performance gains, especially in challenging datasets with high inter-class similarity and intra-class variability.

6. Conclusion

In this project, we explored various techniques to enhance the performance of flower classification using deep learning models, particularly ResNet-based architectures. Our experiments evaluated Deformable Convolutions, Dilated Convolutions, Few-Shot Learning, MixUp, and Triplet Loss, assessing their impact on convergence speed, training time, and validation accuracy.

Key Findings

1. Deformable and Dilated Convolutions improved feature extraction flexibility but had mixed results in accuracy. While they accelerated convergence, their impact on classification performance was limited.

2. Few-Shot Learning significantly reduced training time but caused a drop in validation accuracy, indicating that additional fine-tuning is required for optimal generalization.

3. MixUp augmentation improved model robustness, leading to faster convergence and better accuracy, particularly in the Frozen ResNet18 model.

4. Triplet Loss was the most effective enhancement, boosting validation accuracy to 94.12% (ResNet18) and 94.51% (Frozen ResNet18) while drastically reducing the number of epochs required for convergence.

5. ViT-Tiny achieved strong validation accuracy (~95%) while demonstrating stable learning and good generalization. Its patch-based attention mechanism enabled effective extraction of fine-grained features, making it well-suited for the flower classification task.

Implications and Future Work

Our results demonstrate that metric learning techniques like Triplet Loss can significantly improve fine-grained classification tasks. However, computational trade-offs must be considered. Future work could explore:

- **Hybrid Loss Functions:**
  Combining Triplet Loss with Cross-Entropy or contrastive loss for better balance between speed and accuracy.

- **Hard Example Mining:**
  Optimizing triplet selection strategies to further enhance learning efficiency.

- **Larger and More Diverse Datasets:**
  Testing the model on additional datasets to evaluate generalization across different domains.

- **Model Architecture Enhancements:**

Investigating Transformer-based vision models or fine-tuning deeper networks to push performance even further.

- **Transformer Models for Fine-Grained Tasks:**
  The strong performance of ViT-Tiny highlights the potential of Vision Transformers in fine-grained classification tasks. Future research could explore larger ViT variants, hybrid CNN-transformer architectures, or fine-tuning pretrained transformers to further improve recognition in datasets with high inter-class similarity.

Final Remarks

Overall, our study highlights the importance of choosing the right training strategy for optimizing deep learning models. While traditional techniques provide solid performance, advanced methods like metric learning and augmentation can significantly enhance classification accuracy and efficiency. These findings can serve as a foundation for future improvements in real-world fine-grained image recognition applications.

References

[1] 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*, pp. 722-729, 2008. Available: https://ieeexplore.ieee.org/abstract/document/4756141

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 770-778, 2016. Available: https://ieeexplore.ieee.org/document/7780459

[3] J. Dai *et al*., "Deformable Convolutional Networks", *IEEE International Conference on Computer Vision (ICCV)*, pp. 764-773, 2017. Available: https://ieeexplore.ieee.org/document/8237351

[4] F. Yu and V. Koltun, “Multi-Scale Context Aggregation by Dilated Convolutions”, *arVix:1511.07122,* vol. 3, 2016. Available: https://arxiv.org/abs/1511.07122

[5] J. Snell, K. Swersky, and R. Zemel, “Prototypical Networks for Few-shot Learning”, *arVix:1703.05175*, vol. 2, 2017. Available: https://arxiv.org/abs/1703.05175

[6] H. Zhang, m. Cisse, Y. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization”, *arVix:1710.09412,* vol. 2, 2018. Available: https://arxiv.org/abs/1710.09412

[7] F. Scroff, D. Kalenichenko, and J. Phibin, “FaceNet: A unified embedding for face recognition and clustering”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 815-82, 2015. Available: https://ieeexplore.ieee.org/document/7298682