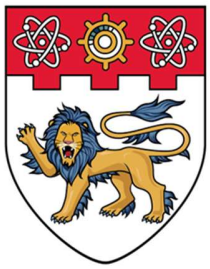

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4001 : Neural Network & Deep Learning

GROUP PROJECT

No.	Name	Matriculation Number
1	Lavanya Sharma	U2120324E
2	Mittal Arnav	U2120833H

Table of Contents

1. Introduction

We have chosen the following Task F - Flowers Recognition, for which we would be using 3 pre-trained models and experimenting with different modifications to each of them. Each of these models would have three baselines to compare our modifications to. These baselines include:

1) Freezing the whole architecture and training only the last layer (Frozen).
2) Training the entire original architecture with pre-trained weights (Unfrozen).
3) Training the triplet loss of the model (Triplet Loss).

The 3 models and their modifications are as follows:

1) EfficientNet - Architecture Modification
2) ResNet - Loss Function Modification
3) Vision Transformer - Loss Function modification

2. EfficientNet

2.1 Introduction

EfficientNet stands out among convolutional neural networks due to its innovative compound scaling method, which optimizes the model architecture in a holistic manner. Traditional approaches to improving deep learning models typically focus on scaling just one aspect, such as increasing the depth (adding more layers) or expanding the width (adding more channels). However, these methods often lead to diminishing returns, excessive computational costs, and inefficiencies in utilizing available resources.

EfficientNet's efficiency is particularly important for deep learning tasks where hardware resources may be limited, such as deploying models on mobile devices or embedded systems. This makes it an ideal choice for real-world applications where both speed and accuracy are crucial.

2.2 Compound Scaling Approach

EfficientNet introduces a more balanced and systematic approach to scaling by adjusting three critical dimensions of a neural network:

1. **Width**:
    - Adds more channels per layer, allowing the network to capture richer, diverse features.
    - Useful for handling detailed textures and fine patterns, which is crucial for tasks like flower classification.

2. **Depth**:
    - Adds more layers, helping the network learn complex feature hierarchies.
    - While deeper networks can capture intricate patterns, scaling only depth can cause issues like vanishing gradients. EfficientNet mitigates this by balancing other dimensions.

3. **Input Resolution**:
    - Enlarges the input image size, enhancing the network's ability to capture fine details.
    - Especially beneficial for flower recognition, where subtle differences in petal structure and colour matter.

2.3 EfficientNet – Baseline Model

This report centres on EfficientNetB0 as our baseline model, selected for its compatibility with 224x224 image inputs, aligning it with the input sizes used by other models in our study for a fair comparison. We intentionally kept the depth, width, and resolution settings unchanged, as they have already been rigorously optimized using a compound scaling approach.

In the next section, we will delve into architectural modifications to EfficientNetB0. We will then evaluate its performance on the Flowers102 dataset and compare the modified version to the baseline to assess improvements in classification accuracy.

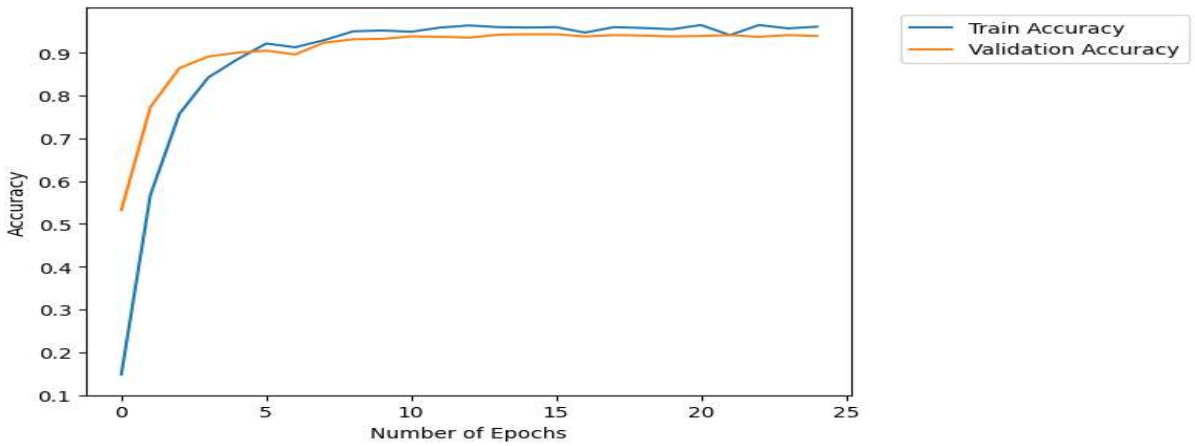

Figure 1: Epoch vs Accuracy for EfficientNet Unfrozen Model

2.4 Replacing SE Modules with ECA Modules (Architecture Modification)

To optimize EfficientNet's performance, our focus shifted toward improving the efficiency of its attention mechanism. In its default configuration, EfficientNet uses Squeeze-and-Excitation (SE) modules to enhance the network's ability to prioritize meaningful features during training.

The Role of SE Modules

SE modules work by dynamically adjusting the importance of each feature channel. This is done by:

1. Using global average pooling to reduce the spatial dimensions, generating a compact representation of each channel.
2. Passing this compressed information through a series of fully connected layers to produce weights that rescale the original feature channels, helping the model focus on the most relevant information.

While effective, SE modules introduce additional computational complexity due to the fully connected layers and the dimensionality reduction step, which can sometimes limit the network's ability to fully capture intricate details.

Shift to Efficient Channel Attention (ECA)

We noticed that the Squeeze-and-Excitation (SE) modules in our model, while improving feature selection, were adding a lot of extra complexity with their dense layers. This increased the number of parameters and slowed down training. To make the model more efficient, we decided to try a simpler approach by replacing SE with Efficient Channel Attention (ECA) modules.

In our experiments, we found that ECA is much lighter. Instead of using dense layers, ECA uses a simple 1D convolution after global average pooling to capture important features. This method doesn't change the size of the features, so the model keeps more information while staying lightweight. The result is then multiplied with the original features, helping the model focus on the most important parts with less computational cost.

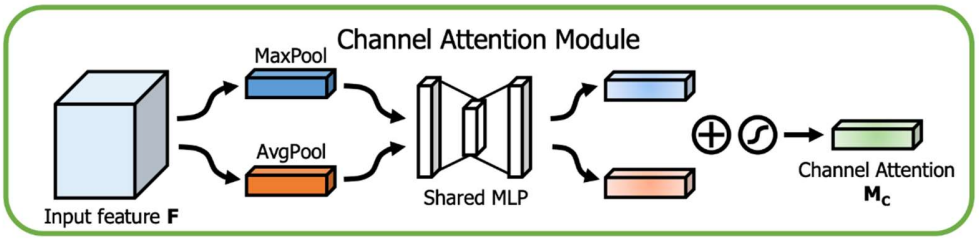

Figure 3: Efficient Channel Attention (ECA) architecture

Benefits of ECA in Our Context
We replaced all SE modules with ECA throughout the network, which reduced the number of parameters and sped up training. This also helped lower memory usage and improved the model's speed without sacrificing accuracy.

The results of this change, including the impact on classification accuracy for the Flowers102 dataset, will be detailed in the subsequent analysis.

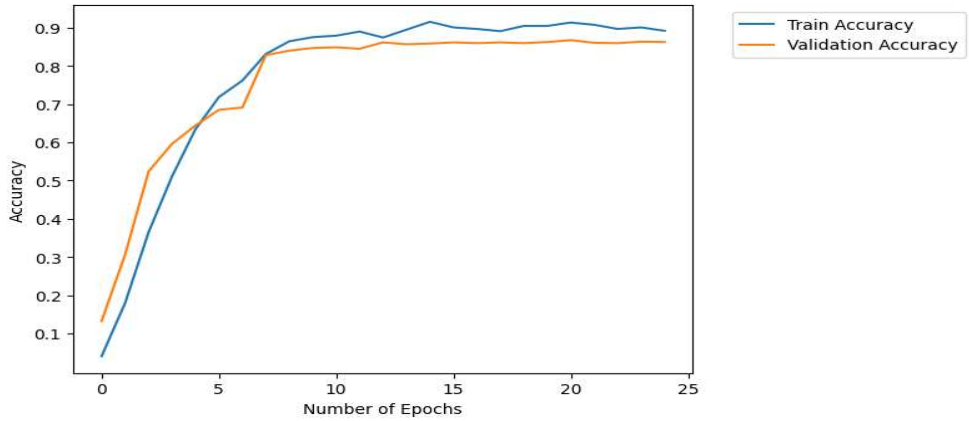

Figure 2: Epoch vs Accuracy for Efficient ECA Model

2.5 Triplet Loss with EfficientNet

After exploring architectural modifications to EfficientNet, our next step involves modifying the loss function to enhance its ability to differentiate between classes, particularly for fine grained classification tasks like recognizing diverse flower species.

Triplet Loss
Unlike traditional classification loss functions such as cross-entropy, which focus solely on maximizing class probabilities, Triplet Loss is designed to optimize the model's ability to distinguish between similar and dissimilar examples. This is particularly beneficial in tasks where classes have subtle differences, like distinguishing between different types of flowers.

It is a method used to train models to distinguish between different classes by comparing sets of three images: an anchor (a reference image), a positive (an image of the same class), and a negative (an image from a different class). The idea is to train the model so that the anchor and positive images are closer together in its learned feature space, while the anchor and negative images are pushed further apart. The loss function ensures that the distance between the anchor and positive is smaller than the distance between the anchor and negative by a set margin. This way, the model learns to recognize subtle differences and similarities more effectively.

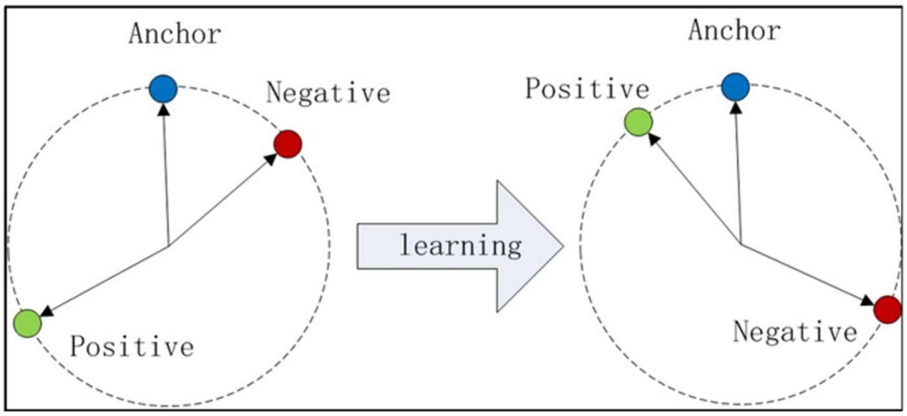

Figure 3. Visual Representation of Triplet Loss Learning

Triplet Loss for EfficientNet

While EfficientNet, with its compound scaling and attention mechanisms, already performs well on classification tasks, integrating Triplet Loss can significantly enhance its discriminative power.

This is especially useful for the Flowers102 dataset, where many flower species have subtle visual distinctions that a traditional cross-entropy loss might not capture effectively.

Triplet Loss focuses on the relative distances between feature embeddings, making it more robust for recognizing minor differences in flower petals, shapes, and textures. By learning better feature embeddings, the model can more accurately cluster similar flower types in the feature space while pushing dissimilar ones further apart.

Integrating Triplet Loss with Entropy Loss for EfficientNet

To effectively train our model using Triplet Loss, we prepared a custom dataset to generate triplets, each consisting of an anchor, a positive, and a negative image. The anchor and positive are from the same class (e.g., two roses), while the negative is from a different class (e.g., a daisy). This setup helps the model minimize the distance between similar images and maximize it between different ones, enhancing fine-grained classification.

However, while Triplet Loss improved feature separation, it didn't optimize for correct label prediction, leading to lower classification accuracy. To address this, we combined Triplet Loss with Cross-Entropy Loss which directly focused on improving the model's ability to assign the correct class to each image.. This combination maintained the benefits of Triplet Loss while directly enhancing the model's ability to classify flower species, thereby improving overall performance

2.6 Experiments & Results

We observed that the Unfrozen EfficientNetB0 model had the best performance overall, with a validation accuracy of 94.31% and a test accuracy of 92.11%, along with a low test loss of 0.2957. By allowing all layers to be trainable, we enabled the model to adapt better to the dataset, capturing detailed patterns and generalizing effectively to new data.

On the other hand, with the Frozen EfficientNetB0, where we only trained the last layer, we noticed some limitations. It achieved a validation accuracy of 87.55% and a lower test accuracy of 84.09%, with a higher test loss of 1.0002. This suggests that freezing most layers restricted the model's ability to adjust, making it less effective at distinguishing between similar flower classes.

When we incorporated Efficient Channel Attention (ECA), we saw some improvement over the frozen model, reaching a validation accuracy of 88.63% and a test accuracy of 87.87%, with a test loss of 0.4610. Although ECA helped the network focus on important features, it didn't quite reach the performance of the fully unfrozen model, indicating that while ECA enhances feature extraction, it cannot fully replace the benefits of fine-tuning all layers.

Lastly, when we used Triplet Loss, it lagged behind, with a validation accuracy of 63.53% and a training accuracy of 59.61%**.** This showed us that, although Triplet Loss helped in learning better feature embeddings, it struggled to generalize for classification. This was likely because it focused more on separating similar and dissimilar samples rather than optimizing for accurate class labels.

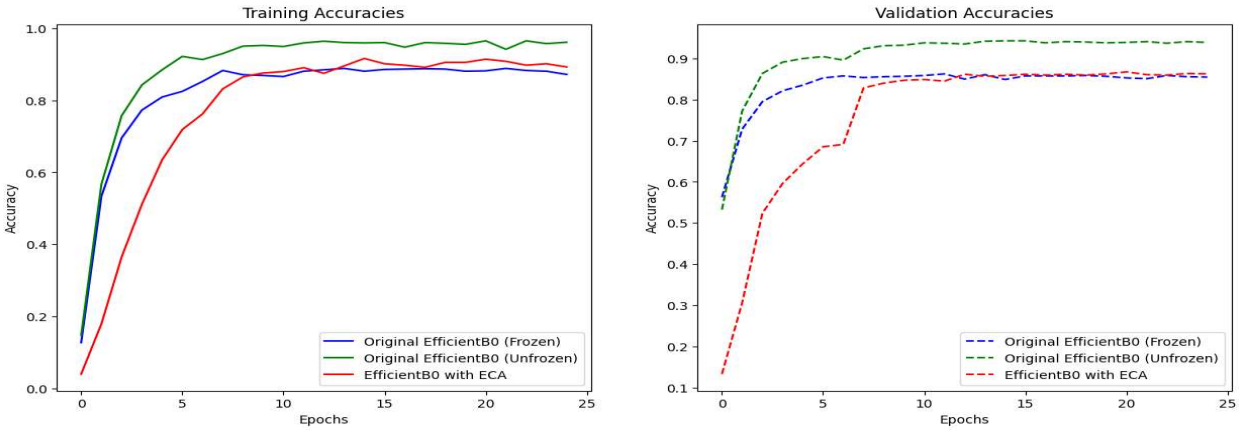

Figure 4. EfficientNet Train and Validation Accuracies for Different Models against Epochs

2.7 Future Improvements

For future improvements, we can try combining Efficient Channel Attention (ECA) with a fully unfrozen model to boost accuracy by leveraging both flexible attention and fine-tuning all layers. We could also explore better balancing Triplet Loss and Cross-Entropy Loss to improve both class separation and prediction accuracy. Using more advanced data augmentation or transfer learning from larger datasets could help the model generalize better.

3. ResNet

3.1 Introduction

ResNet (Residual Networks) is known for its ability to train deep neural networks effectively using residual connections, which help the model avoid issues like vanishing gradients. This allows ResNet to learn complex patterns by passing information efficiently through deep layers. For the Flowers102 dataset, ResNet is particularly useful due to its ability to capture subtle details in flower species, such as variations in petal shapes, textures, and colors. Its deep architecture helps the model differentiate between visually similar classes, while the residual connections improve generalization.

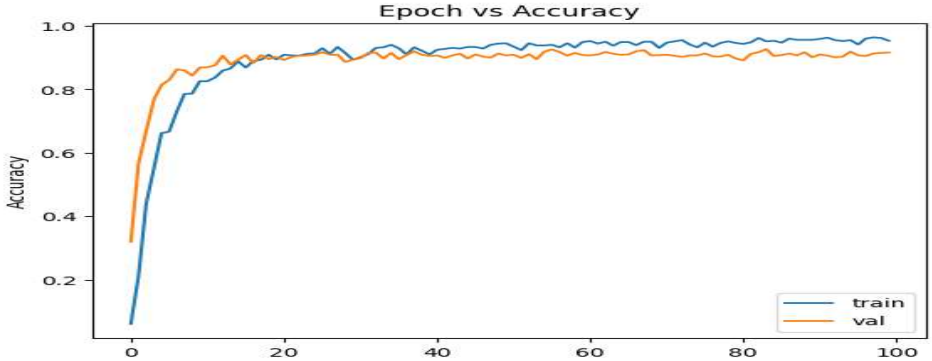

Figure 5: Epoch vs Accuracy for ResNet Unfrozen Model

3.2 Deformed Convulation with ResNet (Architecture Modification)

Baseline ResNet18:

- Started with the original ResNet18 architecture as a control.

Adding Deformable Convolution After conv1:

- In this method, we took the original ResNet18 model and added a deformable convolution right after the first layer (conv1).
- The idea here was to let the model start adapting to complex patterns as early as possible. By allowing the filters to change shape right after the initial convolution, the model could adjust to variations in the input images early on.

Combining conv1 with Deformable Convolution:

- For the second approach, we tried something different. Instead of just adding the deformable convolution after conv1, we merged the output of conv1 with the output of a deformable convolution applied to the same input.
- This means that we fed the input image into both the standard conv1 layer and a deformable convolution, then combined their results. This allowed the model to use both the fixed features from the standard convolution and the flexible features from the deformable convolution at the same time.
- The goal here was to let the model learn both stable patterns (using conv1) and adaptable patterns (using deformable convolution) right from the start.

The key idea behind using deformable convolutions is that they enable the filters to dynamically adjust their shape to better capture complex patterns. Unlike standard convolutions where filter shapes are fixed, deformable convolutions allow the filters to flexibly adapt to changes in the input features, handling variations and transformations more effectively.

To implement this, we utilized the deform_conv2d function from torchvision for incorporating deformable convolutions into our model. This setup allowed us to test whether introducing deformable convolutions early in the network improves the model's ability to capture diverse input features while maintaining stable learning.

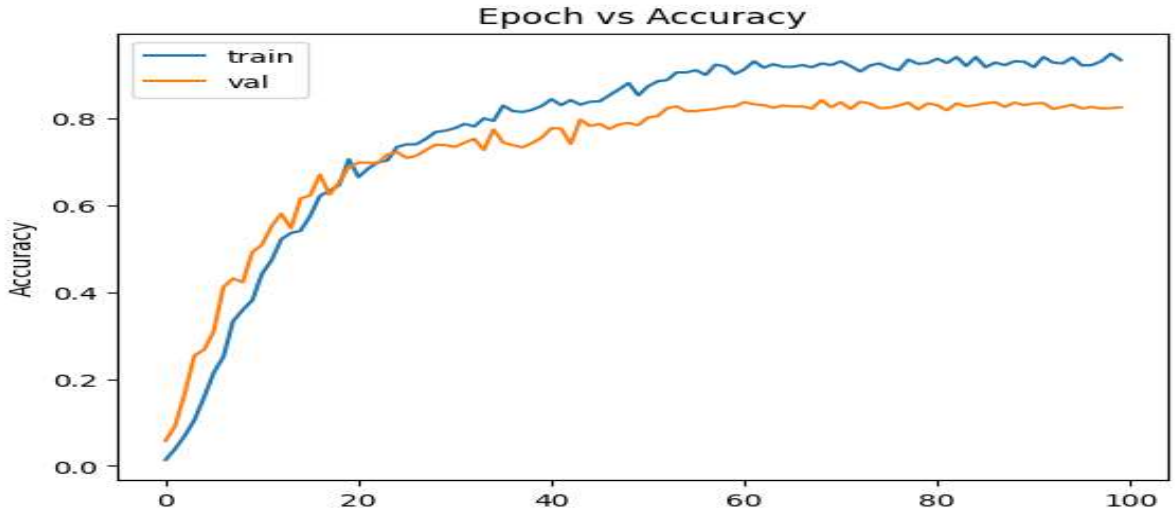

Figure 6: Epoch vs Accuracy for Combining conv1with Deformable ResNet Convolution

3.3 Triplet Loss with ResNet

We wanted to use both Triplet Loss and Cross-Entropy Loss because just relying on Cross-Entropy wasn't enough for our task. While Cross-Entropy helps the model classify flowers correctly, it doesn't ensure that similar-looking flowers are well separated in the feature space.

By adding Triplet Loss, we aimed to make the model better at distinguishing between closely related flower species by pushing apart different classes and clustering similar ones together. This way, we could improve both the accuracy and the ability of ResNet to handle the subtle differences in the Flowers102 dataset.

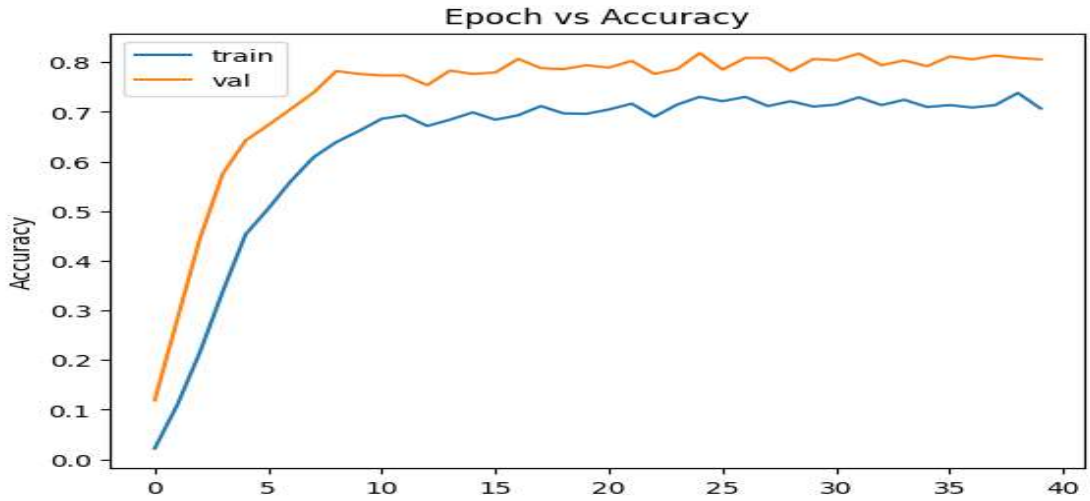

Figure 7: Epoch vs Accuracy for ResNet18 Triplet Loss

3.4 Results

We observed that the Unfrozen ResNet18 model had the best performance overall, with a validation accuracy of 92.65% and a test accuracy of 89.23%, along with a low test loss of 0.4874. By allowing all layers to be trainable, we enabled the model to adapt better to the dataset, capturing detailed patterns and generalizing effectively to new data.

On the other hand, with the Frozen Resnet18, where we only trained the last layer, we noticed some limitations. It achieved a validation accuracy of 87.25% and a lower test accuracy of 84.44%, with a higher test loss of 0.6159. This suggests that freezing most layers restricted the model's ability to adjust, making it less effective at distinguishing between similar flower classes.

When we incorporated the Deformed Convulations with ResNet, we observed the following –

i. Adding deformable convolution after conv1 achieved a validation accuracy of 75.69% and a test accuracy of 68.91% along with a test loss of 1.7265
ii. Combining conv1 with deformable convolution achieved a validation accuracy of 84.22% and a test accuracy of 78.52% along with a test loss of 0.9640.

This shows that combining conv1 with deformable convolution yielded better results though still below Frozen and Unfrozen ResNet18 Models.

Lastly, when we used Triplet Loss, it lagged behind, with a validation accuracy of 63.53% and a training accuracy of 59.61%**.** This showed us that, although Triplet Loss helped in learning better feature embeddings, it struggled to generalize for classification. This was likely because it focused more on separating similar and dissimilar samples rather than optimizing for accurate class labels.

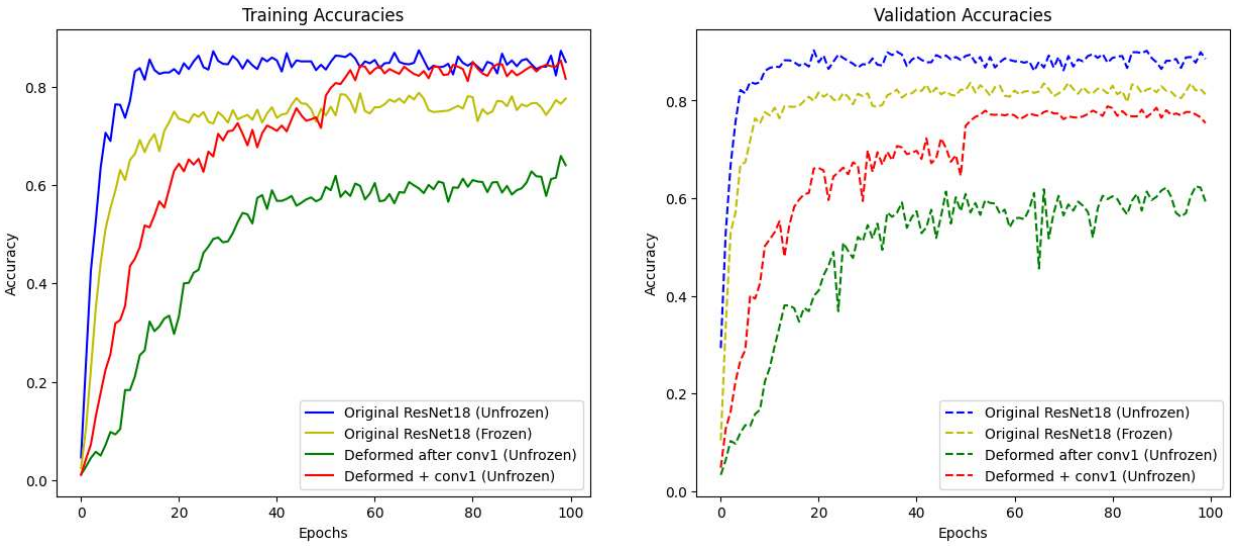

Figure 8. ResNet18 Train and Validation Accuracies for Different Models against Epochs

3.5 Future Improvements

For future improvements with ResNet18, we can explore combining deformable convolutions with a fully unfrozen model to leverage both flexibility in capturing spatial deformations and the adaptability of fine-tuning all layers. Additionally, integrating deformable convolutions in deeper layers rather than after conv1 might yield better performance. Further enhancements could involve dynamically balancing Triplet Loss and Cross-Entropy Loss to optimize both feature separation and classification accuracy, allowing the model to generalize better. Lastly, experimenting with advanced data augmentation techniques or leveraging transfer learning from larger, diverse datasets could further boost performance.

4. Vision Transformers

4.1 Introduction

ViTs have been a revolutionary shift in computer vision, utilizing transformer architectures, originally designed for natural language processing, to process images. Unlike traditional CNNs that rely on convolutions to capture spatial hierarchies, the internal self-attention mechanisms within ViTs model global relationships across an entire image. This allows ViTs to capture long-range dependencies and intricate patterns that could be easily overlooked by the localized receptive fields of CNNs.

By treating an image as a set of non-overlapping patches, Vision Transformers can directly attend to any part of the image. This capability enhances their effectiveness in recognizing complex structures and fine-grained details. ViTs excel at tasks that require nuanced understanding, such as distinguishing subtle differences between similar objects. Moreover, the scalability of ViT models improves significantly with pre-training on large datasets, boosting their generalization abilities across a wide range of image classification tasks.

In this section, we discuss how different loss functions impact the performance of ViTs, specifically comparing the Triplet Loss (TL) with the standard Cross-Entropy Loss (CEL) for image classification tasks. While CEL focuses on maximizing accuracy by directly optimizing for correct class predictions, TL takes a different route by learning discriminative embeddings to optimize the separation between similar and dissimilar samples. We analyze how both loss functions influence ViTs' performance in capturing meaningful feature representations and improving classification accuracy.

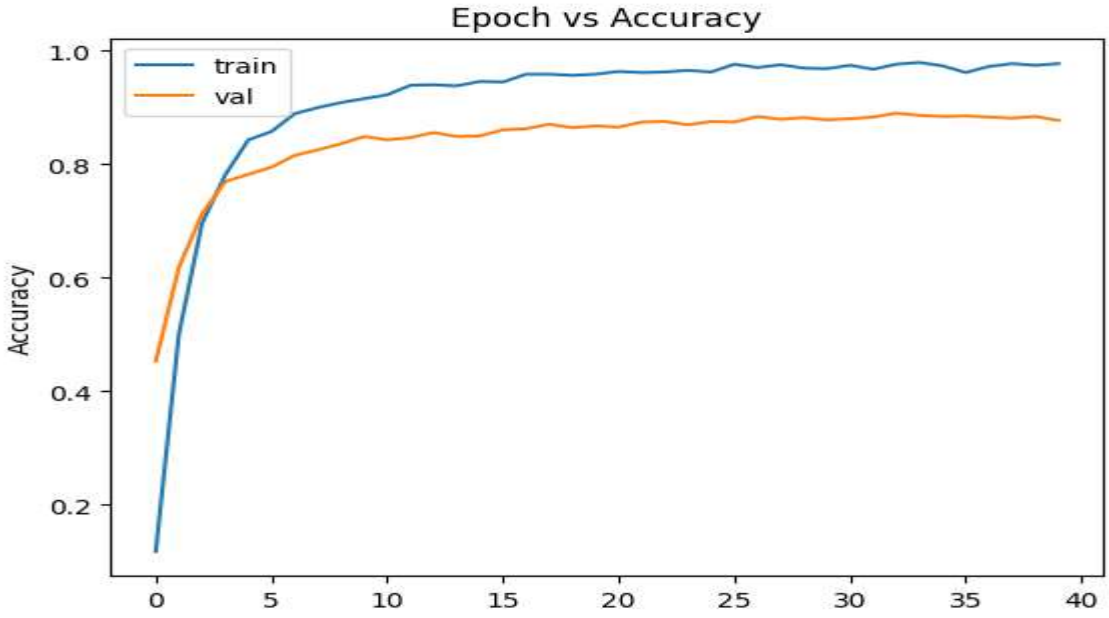

Figure 9: Epoch vs Accuracy for Vision Transformer Frozen Model

4.2 Triplet Loss with Triplet Dataset

We wanted to use both Triplet Loss and Cross-Entropy Loss because just relying on Cross-Entropy was not enough for our task. While Cross-Entropy helps the model classify flowers correctly, it does not ensure that similar-looking flowers are well separated in the feature space.

By adding Triplet Loss, we aimed to make the model better at distinguishing between closely related flower species by pushing apart different classes and clustering similar ones together. This way, we could improve both the accuracy and the ability of ResNet to handle the subtle differences in the Flowers102 dataset

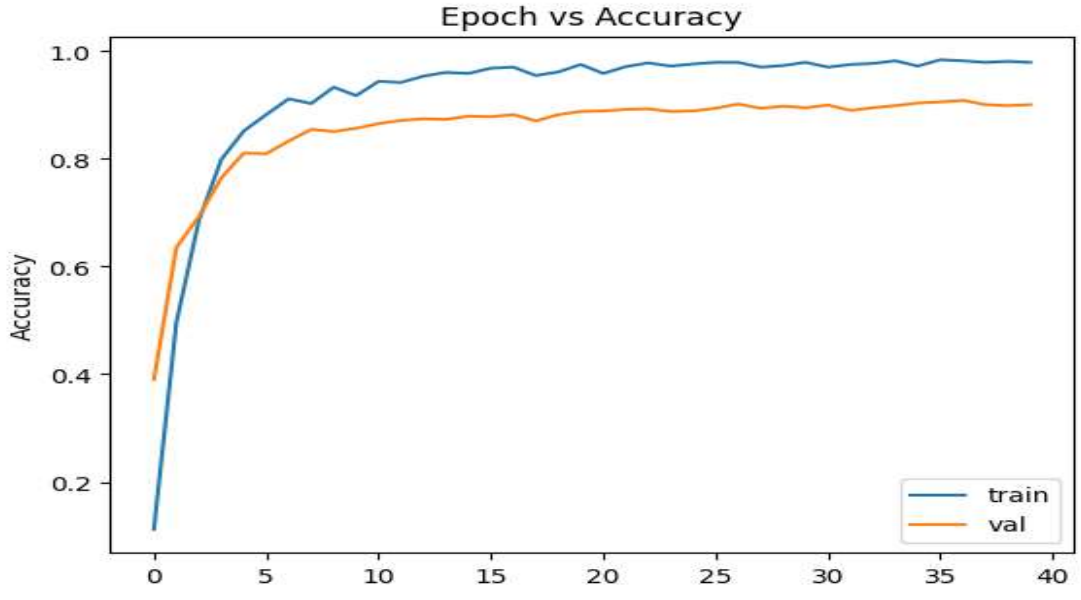

Figure10: Epoch vs Accuracy for Triplet Loss Vision Transformer

4.3 Experiments and Results

We observed that the Vision Transformer (ViT) model, when trained with Triplet Loss, yielded the best performance overall, achieving a validation accuracy of 90.88% and a test accuracy of 88.31%. By leveraging Triplet Loss, the model learned robust feature embeddings, effectively capturing similarities and differences between classes, which resulted in improved generalization and classification accuracy. This approach proved superior in distinguishing between complex patterns in the dataset, leading to better overall performance.

On the other hand, when using the Frozen Vision Transformer, where we only trained the classification head while keeping most of the transformer layers frozen, we observed some limitations. It achieved a validation accuracy of 89.02 and a test accuracy of 87.41%, with a slightly higher test loss of 0.5740. Freezing most layers restricted the model's adaptability, preventing it from fully utilizing the rich representation capabilities of the transformer architecture, although it still performed reasonably well.

Lastly, with the Unfrozen Vision Transformer, where all layers were allowed to be trainable, we noticed that the model struggled to generalize effectively on new data. It reached a validation accuracy of 61.47% and a test accuracy of 55.52%, with a test loss of 2.1290. Despite having the flexibility to adjust all layers, the unfrozen version appeared to overfit the training set, failing to maintain the same level of performance on unseen samples.

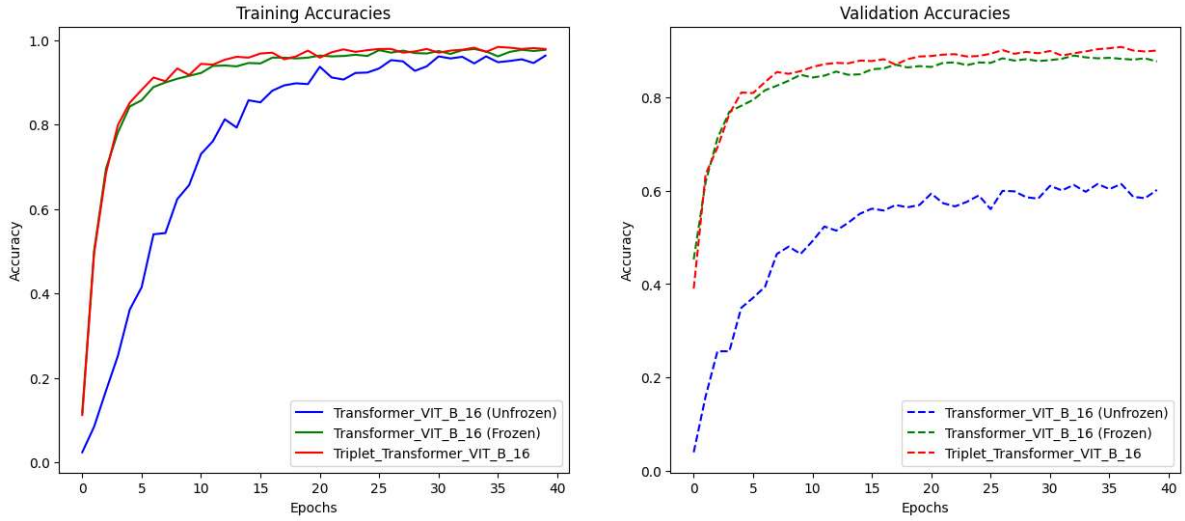

Figure 6. Vision Transformer Train and Validation Accuracies for Different Models against Epochs

4.4 Future Improvements

For future improvements with the Vision Transformer (ViT), we can explore combining Triplet Loss with a hybrid loss function that includes Cross-Entropy Loss to balance feature embedding quality with classification accuracy, potentially improving generalization. Additionally, instead of fully unfreezing all layers, selectively fine-tuning specific layers could help mitigate overfitting while still allowing the model to adapt to the dataset. Incorporating techniques such as knowledge distillation or advanced regularization strategies may also help stabilize training for the unfrozen model. Finally, leveraging data augmentation and transfer learning from pre-trained ViT models on larger datasets could further enhance performance, especially in distinguishing complex patterns.

References

5, 2023, from https://pytorch.org/vision/main/generated/torchvision.ops.deform_conv2d.html

hirotaka0122. (2020, January 11). *Triplet loss with pytorch*. Kaggle. https://www.kaggle.com/code/hirotaka0122/triplet-loss-with-pytorch

Raza, A. (2019, July 7). *Type of convolutions: Deformable and Transformable Convolution | by Ali Raza*. Towards Data Science. Retrieved November 5, 2023, from https://towardsdatascience.com/type-of-convolutions-deformable-and-transformable-con volution-1f660571eb91

Tan, M., Le, Q. V. (2020, September 11). *EfficientNet: Rethinking model scaling for Convolutional Neural Networks*. arXiv.org. https://arxiv.org/abs/1905.11946v5

Van Der Merwe, R. (2020, December 3). *Triplet entropy loss: Improving the generalisation of Short Speech Language Identification Systems.* arXiv.org. https://arxiv.org/abs/2012.03775v1

Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., & Hu, Q. (2020). ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. ArXiv:1910.03151 [Cs]. https://arxiv.org/abs/1910.03151