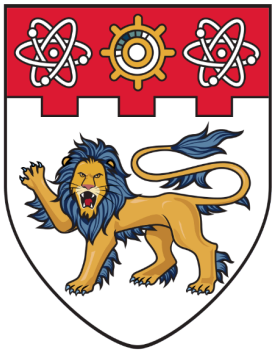

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4001 : Neural Networks & Deep Learning

Assignment 2: Flowers Recognition on Oxford Flowers 102 dataset

Isaac Chun Jun Heng (U2221389B)
J'sen Ong Jia Xuan (U2220457J)
Matthew Heng Yu Jie (U2223483D)

College of Computing and Data Science

Table of Contents

1. Introduction

The Oxford Flowers 102 dataset, created by Nilsback and Zisserman in 2008 stands as a significant and timeless benchmark in the field of computer vision, particularly for fine-grained visual categorization tasks. It is essential for the development and evaluation of image classification algorithms due to its challenging nature that tests the limits against state-of-the-art computer vision techniques. The dataset comprises **8189** images with 102 different flower categories commonly found in the United Kingdom. Unlike other datasets that focus on broadly distinguishable object classes, this dataset presents the challenge from variations in scale, pose, lighting, background clutter, and even the same label class flowers having different colours. Additionally, the dataset is split into training, validation (10 images per class, totaling 1020 images for each set), and a larger test set of 6149 images. This limited training data makes it difficult as the data is heavily unbalanced, which makes it ideal for exploring few-shot learning scenarios, advanced data augmentation and architectural improvements.

This project aims to enhance flower classification using state-of-the-art techniques, including modified CNN architectures, vision transformers with prompt tuning, dilated and deformable convolutions, MixUp augmentation, and triplet loss.

2. Existing Techniques

At the time of creation of this dataset, early approaches to flower recognition typically relied on handcrafted features. Nilsback and Zisserman had initially used a combination of shape, color and texture features with Support Vector Machines (SVMs) to achieve a **72.8% accuracy.** However, these traditional methods typically require significant domain expertise and a deep analysis of the dataset to design effective feature extractors. Modern techniques have been remarkable through several key innovations such as CNN architecture advancements, first demonstrated through AlexNet (Krizhevsky et al., 2012), the first significant leap over traditional methods by automatically learning hierarchical features directly from raw image data. This eliminated the need for manual feature engineering and superior generalization capabilities across visual domains.More recently, **Vision Transformers (ViT)** marked a paradigm shift by applying self attention mechanisms from natural language processing to image classification tasks, being able to effectively capture long-range dependencies between image patches, which was beneficial in recognizing global structures into the image.

Beyond architectural innovations, specialized techniques have emerged to address the specific challenges of fine grained object classification. Deformable convolutions (Dai et al., 2017) adaptively adjust the receptive fields to focus on distinct flower parts regardless of orientation or pose variations in the dataset. An older technique such as dilated convolutions (Yu & Koltun, 2016) is able to capture content while preserving spatial resolution and maintain fine details for classification. In recent times, visual prompt tuning (Jia et al., 2022) has enabled efficient adaptation of pre-trained models with minimal parameter updates, making it well suited for datasets with limited training samples like Oxford Flowers 102. The challenge of such limited training samples have also spurred innovation in data augmentation strategies, where techniques like MixUp (Zhang et al., 2018) and CutMix (Yun et al., 2019) create synthetic training examples through interpolation or region replacement, effectively expanding the training distribution to improve generalization across categories with limited training samples. These approaches mentioned thus far have now all become fundamental tools in modern computer vision, where data is not guaranteed to be abundant and labelled correctly.

3. Goal, Methods Used and Evaluation

The goal of this project is to systematically investigate how modern deep learning approaches can overcome the challenges of fine-grained flower classification with limited training and validation data (an unbalanced dataset) in the Oxford Flowers 102 dataset. Specifically, we aim to:

1. Compare and analyze the performance of modified CNN architectures and Vision Transformers, with particular focus on effects of advanced techniques such as **deformable convolution, dilated convolution and visual prompt tuning.**
2. Evaluate how performance degrades if we aggressively reduce parameters by converting convolution layers to **depthwise+pointwise** convolution layers.
3. Evaluate the impact of advanced data augmentation of **MixUp** on model generalization and performance stability
4. Assess whether specialized loss functions like **triplet loss** improve discriminations between visually similar flower species as compared to standard cross-entropy loss.

We measure this success not only through the overall accuracy of the model, but also by analyzing class-wise performance, confusion patterns and model robustness to the variations in pose, scale and lighting conditions present in the test set. Typically, we aim to identify optimal approaches for fine-grained visual categorization tasks with constrained training data.

4. Experiments

4.1 Dataset Configuration

Our experiments utilized the Oxford Flowers 102 dataset, comprising 8,189 images across 102 flower categories. Following the standard split, we used 1,020 images (10 per class) for training, 1,020 images for validation, and the remaining 6,149 images for testing. Our dataset is transformed as seen in the following table. Since we are utilising some pre-trained models in the experiment, we use the following values as they are standard in computer vision and deep learning, particularly when working with pre-trained models. These are then kept consistent to ensure no bias among the different models we would be working with

- normalize_mean = **[0.485, 0.456, 0.406]**
- normalize_std = **[0.229, 0.224, 0.225]**

Training Set	Validation/Test Set
RandomResizedCrop(224, scale=(0.7, 1.0), ratio=(0.75, 1.33)	**Resize**(256)
RandomHorizontalFlip()	**CenterCrop**(224)
RandomVerticalFlip(p=0.2)	**Normalize**(std=[0.229, 0.224, 0.225],mean=[0.485, 0.456, 0.406])
RandomRotation(30)	-
ColorJitter(brightness,contrast,saturation,hue = 0.15**)**	-
Normalize(std=[0.229, 0.224, 0.225], mean=[0.485, 0.456, 0.406])	-

Table 1: Dataset Transforms

The transforms resulted in the following visualization of our dataset, with the top row showing the augmented data pulled from our train dataloader, the middle row showing the original image, and the last row showing an example of the “best augmented” image by comparing the largest similarity of that image to the original image (visualizes what a good augmentation would be).

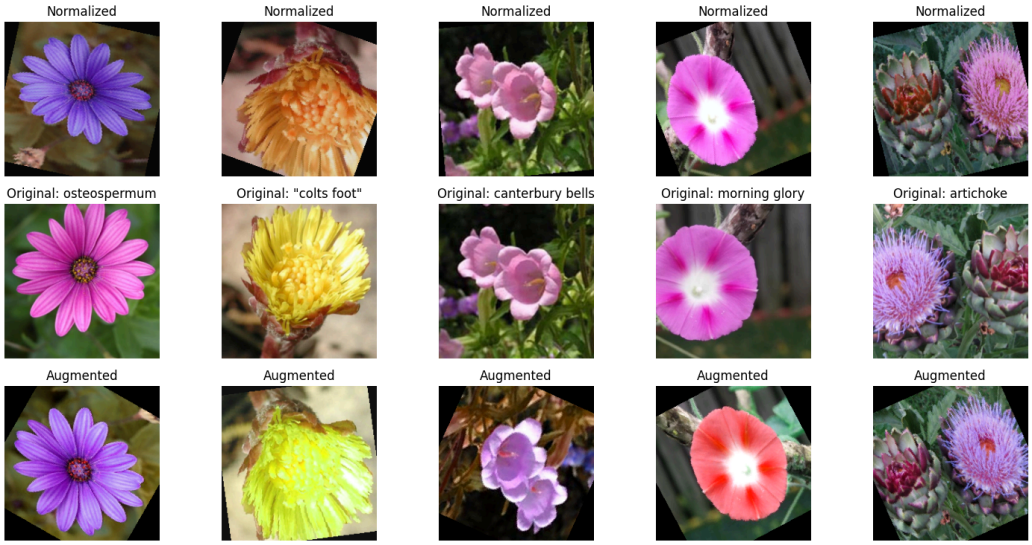

Figure 1: Visualization of Oxford 102 Train Dataset

4.2 Implementation Details

All models were implemented using **torch==2.5.1+cuda** and trained on an NVIDIA RTX 3070 GPU. Throughout the experiment, the following fine-tuned hyperparameters after our initial EDA phase were used throughout all subsequent experiments as they proved to be the best in giving us positive and good results.

Item	Value	Description/Notes
NUM_EPOCHS	25	25 epochs was usually more than enough
BATCH_SIZE	32	This is best given small training data size
LEARNING_RATE	0.001	Standard learning rate for experiments
NUM_CLASSES	102	The number of classes in our dataset
MIXUP_ALPHA	0.2	Good enough parameter to acquire a good distribution of mixed images when MixUp
Optimizer	SGD w/ momentum 0.9	Initial experiments with **Adam** did not allow us to tune our hyperparameters efficiently due to more tolerance on lazy hyperparameters initialisation
Scheduler	ReduceLROnPlateau	Scheduler was used to track validation loss, and reduce LR if the validation loss was plateau-ing to ensure we have good performance
Criterion	CrossEntropyLoss	Loss function for multi class classification

Table 2: Hyperparameters and Training Setup

4.3 Evaluation Metrics

- **Accuracy**: Percentage of test images correctly classified
- **F1-score**: Harmonic mean of precision and recall
- **Confusion matrix**: analysis of error patterns between similar flower species
- **Model loss**: How well model generalizes to new, unseen data

4.4 Base Models Without Architecture Modification

We first established baseline performance using standard architectures without modifications. Base CNN is a baseline model built by the team with just enough layers to acquire a decent accuracy. We utilize pre-trained models like **ResNet-18, ResNet-50**, and pre-trained ViT model **vit_base_patch16_224.** When training, **all layers are frozen and we just train the classifier for transfer learning**. Additionally, we added visual prompt tuning to the pre-trained ViT model to have **five** models for comparison. By using transfer learning, we can evaluate the baseline performance without any specialized techniques on "modern" models on our limited flower dataset.

Our **Base CNN model** follows design principles similar to VGG but incorporates modern practices like batch normalization. It has progressive channel expansions from **32 → 64 → 128 → 256 → 512**, and systematic spatial reduction via **MaxPooling.** This design balances depth and width to efficiently extract features without much training time required. We also note that it would be insufficient to capture all the details in the flower dataset, but it serves as a good benchmark to the problem for future improvements. The following diagram and table summarizes the results of these experiments.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 224, 224]	896
BatchNorm2d-2	[-1, 32, 224, 224]	64
ReLU-3	[-1, 32, 224, 224]	0
Conv2d-4	[-1, 64, 224, 224]	18,496
BatchNorm2d-5	[-1, 64, 224, 224]	128
ReLU-6	[-1, 64, 224, 224]	0
MaxPool2d-7	[-1, 64, 112, 112]	0
Conv2d-8	[-1, 128, 112, 112]	73,856
BatchNorm2d-9	[-1, 128, 112, 112]	256
ReLU-10	[-1, 128, 112, 112]	0
MaxPool2d-11	[-1, 128, 56, 56]	0
Conv2d-12	[-1, 256, 56, 56]	295,168
BatchNorm2d-13	[-1, 256, 56, 56]	512
ReLU-14	[-1, 256, 56, 56]	0
Conv2d-15	[-1, 512, 56, 56]	1,180,160
BatchNorm2d-16	[-1, 512, 56, 56]	1,024
ReLU-17	[-1, 512, 56, 56]	0
MaxPool2d-18	[-1, 512, 28, 28]	0
Conv2d-19	[-1, 512, 28, 28]	2,359,808
BatchNorm2d-20	[-1, 512, 28, 28]	1,024
ReLU-21	[-1, 512, 28, 28]	0
MaxPool2d-22	[-1, 512, 14, 14]	0
MaxPool2d-23	[-1, 512, 7, 7]	0
AdaptiveAvgPool2d-24	[-1, 512, 1, 1]	0
Linear-25	[-1, 102]	52,326

Figure 2: Base CNN Model Architecture

Model	Parameters(M)	Accuracy (Test)	F1-Score	Loss
Base CNN	3,983,718	0.2691	0.24	3.1135
Base ResNet-18	11,228,838	0.7896	0.79	1.0315
Base ResNet-50	23,717,030	0.8356	0.84	0.7285
Base ViT	85,725,030	0.9673	0.97	0.2486
Base ViT + Prompt Tuning	85,915,494	0.9681	0.97	0.2116

Table 3: Performance of Base Models

We can observe that there is a correlation between model complexity and performance correlation. As the size of the model increases, performance metrics like accuracy and f1-score improve and loss decreases. However, we can observe that **ResNet-50** still achieves impressive results with only about **28%** of the parameter of ViT, which makes it a more efficient choice for resource constrained

applications. Overall, through the power of transfer learning, we can see how using large pre-trained models proves to be more efficient and successful than training our CNN from the ground up.

4.5 Dilated and Deformable Convolutions

1. **Dilated Convolution** introduces gaps between kernel elements, expanding the receptive field without increasing parameters. A dilation rate ***d*** spreads the kernel over a wider area, allowing the model to capture more context. This is especially useful in tasks like semantic segmentation where global information is important.

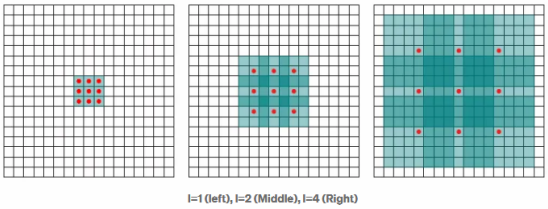

Figure 3: Dilated Convolutions Receptive Field (Medium)

2. **Deformed Convolution** adds learnable offsets to kernel sampling points, letting the model adjust its receptive field dynamically. It improves flexibility in capturing geometric variations, making it effective for images with irregular shapes.

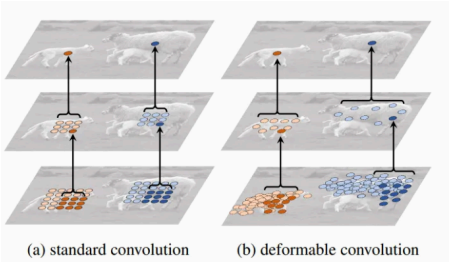

Figure 4: Standard vs Deformable Convolutions (Medium)

We implement ***dilated and deformable convolutions*** by altering our initial base CNN model. **For dilated convolutions,** we increase the dilation to **2** in the later layers to improve the receptive field of the model. We note that it is possible that increasing the receptive field from dilation did not benefit in this task as we had **loss of local detail, and that the nature of this task primarily relies on fine-grained local features and textures.** For **deformable convolutions**, we add an offset, mask and DeformConv2D layer to perform our calculations. We note a good increase in accuracy to **0.2935**, with the fact that deformable convolutions can learn to adjust their sampling locations based on input, allowing them to **focus on the most discriminative parts of the flower regardless of shape or position**. In a classification done on this dataset, deformable CNN proves to be very **useful** in solving and better handling of irregular shapes for more accurate classification.

Model	Parameters(M)	Accuracy	F1-Score	Loss
Base CNN	3,983,718	0.2691	0.24	3.1135
Dilated CNN	3,983,718	0.2693	0.25	3.0428
Deformable CNN	4,170,396	0.2935	0.28	2.9499

Table 4: Performance of Dilated and Deformable Convolution

4.6 Base Models With Reduced Parameters

We also attempt to visualize how reduction in parameters would affect accuracy. We swap out the

last two non 1x1 Conv2D layers to **Depthwise+Pointwise** layers, effectively removing a large amount of parameters. We note that while accuracy generally drops based on the parameters, the earlier layers of pre-trained models help provide a good foundation, allowing accuracy to not drop as much for pre-trained models. However, we can observe that the lightweight modification affects **ResNet-50** more severely, possibly due to the usage of Bottleneck blocks that causes it to be more sensitive to the replacement. Still, the accuracy should increase given more epochs which can be done in an alternate experiment.

Model	Parameters(M)	Reduction (%)	Accuracy	F1-Score	Loss
Base CNN	3,983,718	-	0.2691	0.24	3.1135
Lightweight CNN	845,670	78.77	0.2064	0.17	3.3455
Base ResNet-18	11,228,838	-	0.7896	0.79	1.0315
Lightweight ResNet-18	7,043,750	37.27	0.6502	0.65	1.5087
Base ResNet-50	23,717,030	-	0.8356	0.84	0.7285
Lightweight ResNet-50	19,531,942	17.65	0.6408	0.62	1.6384

Table 5: Performance of Lightweight Models

4.7 MixUp Data Augmentation

MixUp is a data augmentation technique that synthesizes a new training sample through combining two existing data points. We represent *x* represents the data value and *y* represents the corresponding layer. We do a linear combination of the two original data points x_i and x□ with the weight λ as well as combining the label from the two original labels y_i and y□.The equations can be seen as below:

$$\hat{x} = \lambda x_i + (1 - \lambda)x$$

$$\hat{y} = \lambda y_i + (1 - \lambda)y$$

This augmentation results in the following result:

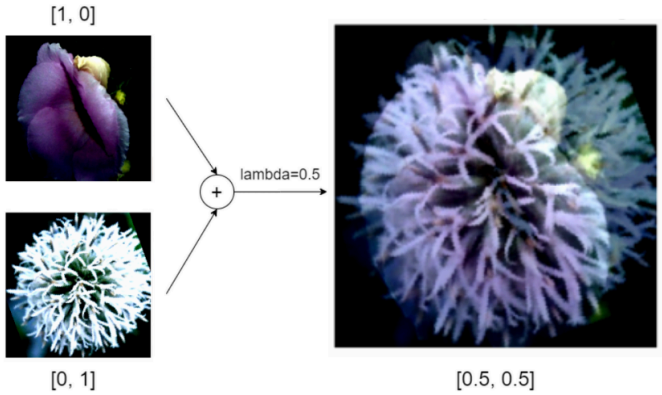

Figure 5: MixUp Augmentation on Oxford102 Flower Dataset

In this experiment, we add **MixUp data augmentation** due to the limited amount of training data. We utilized the best models from previous experiments separated into categories (CNN, ResNet and ViT). We observe a **consistent improvement across** all three model types in terms of accuracy and loss. Notably, Deformable CNN accuracy increased the most **(0.2935 to 0.3494)** on the test set. Due to the adaptive nature of deformable convolutions and when combined with synthetic examples from MixUp, it possibly allowed the model **to experience a wider variety of spatial configurations given our**

limited data, increasing the accuracy greatly despite its. On the other hand, when combining our initial ViT + Prompt Tuning model, the accuracy increased and loss decreased**, proving that data augmentation techniques like MixUp + Prompt Tuning yields exceptional results even with limited training data.**

Model	Parameters(M)	Accuracy (Test)	F1-Score	Loss
Deformable CNN	4,170,396	0.3494	0.35	2.751
Base ResNet-50	23,717,030	0.8559	0.86	0.6162
Base ViT + Prompt Tuning	85,725,030	0.9712	0.97	0.1949

Table 6: Performance of Best Models With MixUp

4.8 Using Triplet Loss On ResNet-50 Model

Lastly, we experiment with **Triplet loss** on our **ResNet-50** model to compare how using a different loss function helps in generalizing. **Triplet loss** (Schroff et al., 2015) is a distance-based loss function designed to learn embeddings where images of the same class are close together in the embedding space, and other images are far apart. We operate on three images (anchor, positive and negative) with the following formulae:

$$L=\max(0,d(a,p)-d(a,n)+\text{margin})$$

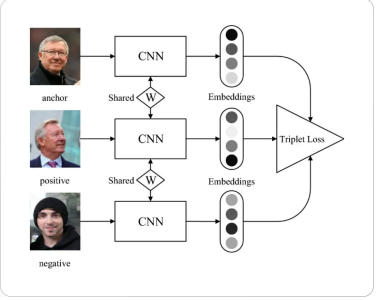

Figure 6: Triplet Loss

From this experiment, we note a significant improvement of the **ResNet-50 model with Triplet Loss** by resulting in a higher accuracy and lower loss when compared with the other methods. Interestingly, when comparing to MixUp results, they both acquire higher accuracy, **which suggests that improving data variety through augmentation may just be as effective as refining the feature space**. These two approaches address different aspects of the learning problem. Triplet loss was particularly suitable for this problem due to our initial limited training data, and that several flower categories in the dataset looked too similar. Triplet loss addressed this challenge by pushing the embedding of similar-looking but different classes apart, causing the model to improve on its accuracy throughout the test set.

Model	Parameters(M)	Accuracy (Test)	F1-Score	Loss
Base ResNet-50	23,717,030	0.8356	0.84	0.7285
Lightweight ResNet-50	19,531,942	0.6408	0.62	1.6384
Base ResNet-50 + MixUp	23,717,030	0.8559	0.86	0.6162
Base ResNet-50 + Triplet-Loss	23,717,030	**0.8514**	0.85	0.6981

Table 7: Performance of ResNet-50 with Triplet Loss Against Other ResNet-50 Models

4.9 Conclusion

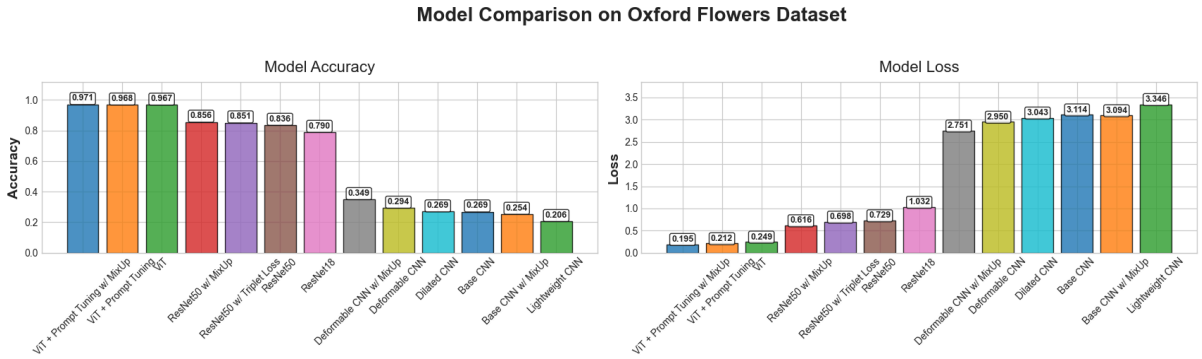

Figure 7: Best Model from Experiments

After conducting extensive experiments with various model architectures and techniques on the Oxford Flowers 102 dataset, we gain valuable insights into effective approaches in the task of fine-grained flower classification. Our results demonstrate a clear hierarchy of model performance and the impact of different enhancement techniques. We conclude that **ViT + Prompt Tuning w/ Mixup** performed the best on the dataset, achieving a remarkable 97.1% accuracy with lowest loss 0.915. In the following sections, we discuss **ViTs and Prompt Tuning** and how it acquires its superior performance especially in modern computer vision architectures.

5. Discussion on ViTs and Prompt Tuning

Prompt tuning for ViT represents a parameter-efficient transfer learning approach that has been adapted for vision tasks through Visual Prompt Tuning (VPT) (Jia et al., 2022). Instead of updating all model parameters during fine-tuning, VPT **introduces a small set of learnable tokens**, "prompts", that are prepended to the sequence of patch embeddings while keeping the pre-trained model frozen. This approach offers significant advantages over various fine-tuning methods:

5.1 Prompt Tuning vs Full Fine-Tuning vs Classification-Head Tuning vs Adapter-Based Tuning vs LoRA on Vision Transformer (ViT)

Compared to full fine-tuning, VPT requires updating only **0.1-1%** of total parameters, dramatically reducing computational costs and memory requirements while achieving competitive performance. This efficiency makes VPT particularly attractive for deployment on edge devices or in **resource-constrained environments.**

Unlike classification-head tuning **(linear probing)**, which is limited to optimizing only the final classification layer on fixed feature representations, VPT introduces learnable tokens that interact with all layers of the transformer architecture through self-attention mechanisms. This cross-layer interaction enables the model to adaptively modify feature representations at all levels of abstraction.

In contrast to **adapter-based methods,** which interrupt the original network flow by inserting new bottleneck modules between existing transformer blocks, VPT preserves the native transformer architecture and leverages its inherent attention mechanism. This architectural elegance eliminates potential bottlenecks and gradient flow issues that can arise with adapter insertions.

Although LoRA (Hu et al., 2022) is also an parameter-efficient transfer learning method, the additional parameters that LoRA added only influence locally in the layer, whereas the additional **prompt tokens** introduced by VPT are able to **flow through the all layers of transformers.**

5.2 Model Architecture - Linear Probe ViT vs Prompt Tuning on ViT

- **Why is prompt tuned ViT better than linear probe ViT?**

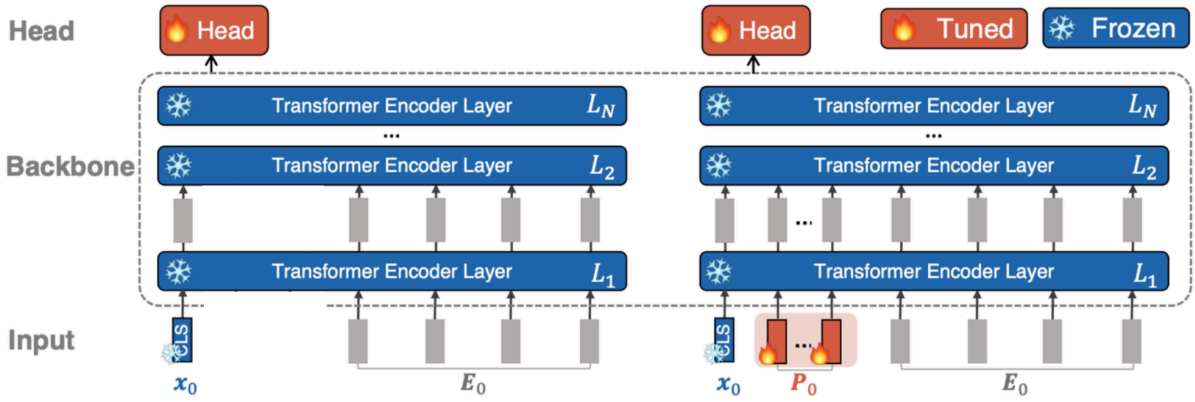

Figure 8: ViTB16 Model Architecture Information —
Linear Probe (Left) vs. Prompt Tuning: Shallow (Right)
Image credit: Adapted from "Visual Prompt Tuning" (Jia et al., ECCV 2022).

This figure illustrates the Visual Prompt Tuning (VPT) architecture compared to the standard linear probe approach. On the left, we see the traditional linear probe method where only the classification head is trained while the transformer backbone remains frozen. On the right is the Visual Prompt Tuning approach, which introduces trainable prompt tokens (P_0, shown in red) that are prepended to the embedded input sequence. The key innovation of VPT is that it keeps the backbone model parameters frozen while only training these prompt tokens and the classification head. This allows for efficient adaptation to downstream tasks with minimal parameter updates, as the prompt tokens learn to modulate the frozen model's behavior through the attention mechanism of the transformer layers. The approach maintains the pre-trained model's knowledge while enabling task-specific fine-tuning with significantly fewer trainable parameters. Also, prompt-tuned ViT has more trainable parameters than linear probe ViT (Dosovitskiy et al., 2021).

5.3 Result of Linear Probe ViT vs Prompt-tuning ViT

Model	Parameters(M)	Accuracy (Test)	F1-Score	Loss
Base ViT	85,725,030	0.9673	0.97	0.2486
Base ViT + Prompt Tuning	85,915,494	0.9681	0.97	0.2116

Insights

- Prompt Tuning slightly outperforms Linear Probe (**96.81% vs. 96.73%** accuracy)
- Linear Probe: **96.73%** accuracy, **0.2486** loss
- Prompt Tuning: **96.81%** accuracy, **0.2116** loss
- Difference: **+0.12%** accuracy improvement with Prompt Tuning

5.4 Discussion on Linear Probe ViT vs Prompt-tuning ViT

- **Why does prompt tuning with ViT offer only marginal improvements over linear probing?**

The modest performance delta between prompt tuning and linear probing for certain datasets can be attributed to the already high-quality representational capacity of frozen ViT embeddings (**~97% performance ceiling on standard benchmarks**). This indicates near-saturation of the representational

space for simple classification tasks. However, as evidenced in the Visual Prompt Tuning (VPT) research by Jia et al. (ECCV 2022), fine-grained classification tasks with higher intra-class variation and subtler discriminative features, such as Stanford Cars classification, demonstrate substantial performance improvements **with prompt tuning compared to linear probing (68.7% vs. 51.3% accuracy)**. This underscores prompt tuning's efficacy in domains requiring more nuanced feature extraction capabilities.

The following figure is a reference on the performance of ViT-B/16 on Oxford Flowers via fine-tuning vs prompt tuning in the original VPT paper.

		CUB-200-2011	NABirds	Oxford Flowers	Stanford Dogs	Stanford Cars	Mean
(a)	FULL	87.3	82.7	98.8	89.4	84.5	88.54
Head-oriented							
(a)	LINEAR	85.3	75.9	97.9	86.2	51.3	79.32 (0)
	PARTIAL-1	85.6	77.8	98.2	85.5	66.2	82.63 (0)
	MLP-2	85.7	77.2	98.2	85.4	54.9	80.28 (0)
	MLP-3	85.1	77.3	97.9	84.9	53.8	79.80 (0)
	MLP-5	84.2	76.7	97.6	84.8	50.2	78.71 (0)
	MLP-9	83.2	76.0	96.2	83.7	47.6	77.31 (0)
Backbone-oriented							
(b)	SIDETUNE	84.7	75.8	96.9	85.8	48.6	78.35 (0)
	BIAS	88.4	84.2	98.8	91.2	79.4	88.41 (3)
	ADAPTER-256	87.2	84.3	98.5	89.9	68.6	85.70 (2)
	ADAPTER-64	87.1	84.3	98.5	89.8	68.6	85.67 (2)
	ADAPTER-8	87.3	84.3	98.4	88.8	68.4	85.46 (1)
Visual-Prompt Tuning							
(ours)	VPT-SHALLOW	86.7	78.8	98.4	90.7	68.7	84.62 (1)
	Prompt length (p)	100	50	100	100	100	90
	Tuned / Total (%)	0.31	0.54	0.23	0.20	0.26	0.31
	VPT-DEEP	88.5	84.2	99.0	90.2	83.6	89.11 (4)
	Prompt length (p)	10	50	5	100	200	73
	Tuned / Total (%)	0.29	1.02	0.14	1.17	2.27	0.98

Figure 9: A reference on the performance of ViT-B/16 on Oxford Flowers and others via different types of fine-tuning vs prompt tuning. Credit: This result is obtained from the original paper of VPT, namely "Visual Prompt Tuning" (Jia et al., ECCV 2022).

- **Why is ViT better than ResNet and other convolution based architecture in general?**

A simple answer for this would be that ViT has a better “global view” than convolution based models. ViT offers architectural advantages over ResNet **through their self-attention mechanism**, which captures global dependencies across the entire image in a single operation from the very first layer. Unlike ResNet's convolutional layers that build receptive fields hierarchically through stacked operations, ViT processes relationships between all image patches simultaneously, enabling more effective modeling of long-range dependencies critical for understanding complex visual patterns. This global context awareness particularly benefits tasks requiring holistic image understanding where discriminative features may appear in spatially distant regions.

6. Conclusion

Building on this study’s foundation, we have identified several promising aspects that could further enhance our flower classification performance. While we have only implemented MixUp in this iteration, implementing CutMix augmentation alongside MixUp could provide more benefits by creating even more diverse synthetic training examples. Other advanced loss functions such as ArcFace or Focal Loss might better address the fine-grained nature of flowers, even if ArcFace is more suited towards facial recognition. Also, although few-shot learning was not directly implemented in these experiments, it could serve to help us understand model degradation better when we are faced with extremely low samples. Finally, implementing even more complex architectures like CNN with transformers could offer the optimal balance between local feature detection and global context awareness. Overall, these future implementations could potentially surpass our ViT + Prompt Tuning model, ultimately leading to more robust and efficient flower classification models.

References

1. Nilsback, M-E. and Zisserman, A., “Automated flower classification over a large number of
classes,” in British Machine Vision Conference (BMVC), 2008

2. Jia, M., Tang, L., Chen, B., Cardie, C., Belongie, S., Hariharan, B., & Lim, S. N. (2022). Visual Prompt Tuning. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 709-727). Springer, Cham.

3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 25, pp. 1097-1105). Curran Associates, Inc.

4. Yu, F., & Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*.

5. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., & Wei, Y. (2017). Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 764-773).

6. Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

8. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 815-823).

9. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

10. Tsang, S.-H. (2018). Review: DilatedNet — Dilated Convolution (Semantic Segmentation). *Medium.*

11. Ito Aramendia, A. (2024). Deformable Convolutional Networks: A Complete Guide. *Medium.*

12. Shah, D. (2023). Triplet Loss: Intro, Implementation, Use Cases. *V7 Labs.*