NANYANG TECHNOLOGICAL UNIVERSITY

SC4001 - Neural Networks and Deep Learning

Semester 1, AY2025/2026

Name	Matriculation Number
Pang Boslyn	
Sally Ngui Yu Ying	

1 Introduction

We have chosen Task F, to implement a classification model to classify flower images in the Oxford Flowers 102 dataset. This dataset poses significant challenges as flowers belonging to the same category may look very different and at the same time, flowers of different categories may appear very similar, making it difficult for models to tell them apart. Therefore, this dataset is a good testbed for exploring the strengths and limitations of different deep learning models.

This project aims to design, implement and evaluate neural network-based classification models trained and tested on the Oxford Flowers 102 dataset. We explored and discussed various existing techniques. We first implemented 2 well-known Convolutional Neural Network (CNN) architectures, ResNet18 and VGG16. Then, we introduced several enhancements such as architectural enhancements, data augmentation and use of advanced loss function to improve accuracy and generalisation. Next, we examined Vision Transformer (ViT) models, beginning with fine tuning a pretrained ViT on the Flowers data as our baseline model. We then explored model improvements with hyperparameters fine-tuning strategies. Further improvements include Visual Prompt Tuning and exploring the use of advanced loss functions.

2 Data Processing

The Oxford Flowers dataset contains 102 flower categories, with each class consisting between 40 to 258 images. The train and validation set each consist of 10 images per class while the test set consists of at least 20 images per class. Both CNN and Transformers follow the following standardised preprocessing and loading pipeline.

2.1 Dataset Preparation

Images were resized to 224 x 224 pixels to be aligned with the input dimensions used by ImageNet-pretrained networks, such as ResNet and ViT. Next, the pixel data of images are then converted into PyTorch tensors for compatibility. Normalisation was then applied using the mean and standard deviation values from the ImageNet dataset, to ensure consistency between the data input and the pretrained models, improving convergence during fine-tuning.

2.2 Data Loading Pipeline

Each split was loaded using PyTorch DataLoader objects for efficient mini-batch training. During training, data shuffling was applied to the training set to introduce randomness, reducing the risk of overfitting to specific images. As for validation and test sets, they remained unshuffled as no training will be carried out.

3 CNN

3.1 ResNet18 Models

Residual Networks (ResNet) is a deep convolutional architecture that introduces residual connections, enabling more effective training of deeper networks and allowing gradients to flow more easily through the network, improving the vanishing gradient problem. ResNet18 was selected as it achieves a strong trade-off between computational efficiency and model performance. ResNet18 consists of the stem, four main stages with 2 residual blocks each and a fully connected layer (Appendix A). The stem consists of a Conv2d layer, BatchNorm2d layer, ReLU activation function, and MaxPool2d layer.

3.1.1 Baseline ResNet18 Transfer Learning and Hyperparameter Tuning

Transfer learning initialises the backbone with pre-trained weights from ImageNet. The final fully-connected (FC) layer is then replaced to match 102 flower classes, followed by further fine-tuning and training on the Flowers102 dataset.

Methodology

We established a transfer learning baseline by fine-tuning the pre-trained model, with a learning rate of 0.001, batch size of 32, and no dropout and weight decay. Hyperparameter tuning was then conducted by experimenting with different permutations of learning rate and batch size, finding the optimal learning rate and batch size that achieves the highest validation accuracy. After hyperparameter tuning, various configurations of dropout and weight decay values were tested.

Initial Learning Rate	Batch Size	Dropout	Weight Decay (L2 Regularization)
0.0005, 0.0001, 0.001	16, 32, 64	0.0, 0.2, 0.3	0.0, 1e-5, 1e-4

Other fixed configurations were early stopping with patience of 5 epochs, Adam Optimizer, Cross Entropy Loss function.

Results

The baseline transfer learning model achieved a best validation accuracy of 0.8010 and test accuracy of 0.7603, with early stopping at epoch 10 (Figure 3.1). After hyperparameter tuning and regularization, the final baseline model with optimal configurations achieved the best validation accuracy of 0.9137 and test accuracy of 0.8707, early stopping at epoch 26 (Figure 3.2). The best configuration is: Initial Learning Rate of 0.0005, Batch Size of 32, Dropout of 0.3 and Weight Decay of 0.0001.

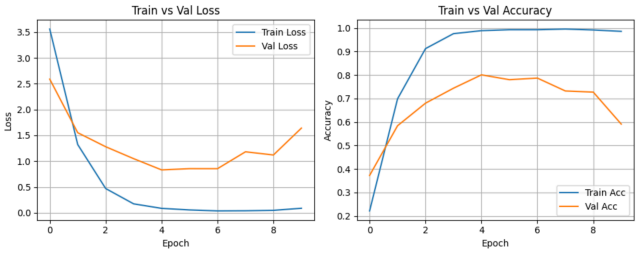


Figure 3.1: Loss and Accuracy Curves of Baseline Model

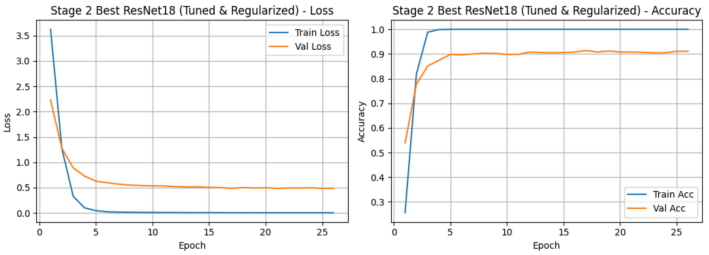


Figure 3.2: Loss and Accuracy Curves of Optimal Model

Discussion

In the initial pre-trained ResNet18 model, the train loss decreases sharply and train accuracy quickly approaches 1. However, from epoch 4, the validation accuracy peaks and declines, increasing the gap between train and validation accuracy. This shows clear signs of overfitting to train set, although already mitigated by early stopping. After conducting hyperparameter tuning and regularisation techniques, the validation accuracy improved from 0.8010 to 0.9137 and test accuracy rose from 0.7603 to 0.8707. The learning curves show a steady reduction in loss and a small generalisation gap between train and validation loss, which indicates good transfer to unseen data and model stability. The lower learning rate allowed for stable convergence, the moderate batch size provided a balance between stability and speed, and the dropout and L2 penalty prevented overfitting to the train set. This shows that transfer learning effectively provides a strong baseline for the Flowers102 dataset, which is further enhanced with hyperparameter tuning and regularisation specific to our dataset, with factors to prevent overfitting. To ensure fair comparison, all subsequent ResNet18 models use these optimal configurations unless specified.

3.1.2 ResNet18 with Different Number of Frozen Layers

Freezing earlier layers preserve general features learned from ImageNet dataset and leave only the final task-specific layers to be trained to adapt to the Flowers102 dataset, hence mitigating overfitting. Furthermore, freezing layers improves computational efficiency by reducing the number of trainable parameters.

Methodology

Building on the final baseline model from Section 3.1.1, we progressively froze k stages in the pre-trained ResNet18 model, where k ranges from 0 to 4. When k equals to 0, all layers are trainable. When k equals to 1, the stem, with the initial layers for input processing, is frozen. As k increases from 2 to 4, one additional residual stage after the stem is frozen each time. The final FC layer remains trainable in all experiments. The Adam optimizer only updates the parameters of layers with `requires_grad = True`.

Results

The optimal model is freezing only the stem, with the highest validation accuracy of 0.9167 and test accuracy of 0.8801 (Figure 3.3) (Figure 3.4).

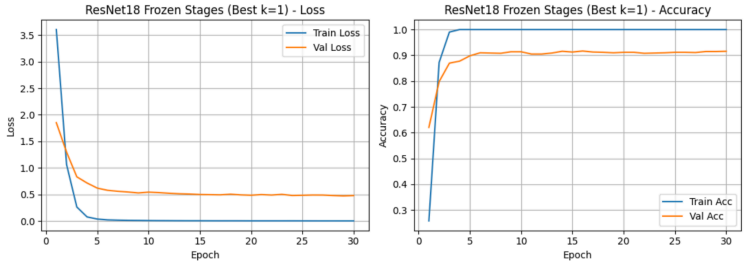


Figure 3.3: Loss and Accuracy Curves of 1 Frozen Stage

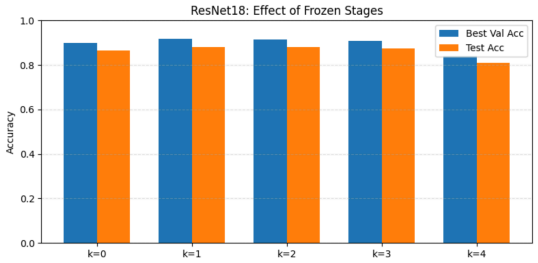


Figure 3.4: Comparison of Results of Frozen Stages

Discussion

Freezing only the stem performed better than the baseline model in Section 3.1.1, where all layers were trainable, as it provides the best balance between transfer from pre-trained model and adaptation to our dataset. As the number of stages frozen increases from 1 to 4, the validation and test accuracy decline from 0.9167 to 0.8392 and 0.8801 to 0.8102 respectively, as it limits the model's ability to adapt to the specifics of the Flowers102 dataset and hence underfits.

3.1.3 ResNet18 MixUp and CutMix

Figure 3.5: MixUp samples

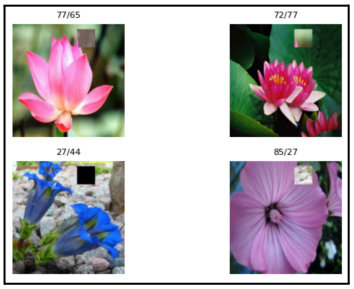


Figure 3.6: CutMix samples

MixUp and CutMix are data augmentation techniques that create new samples by combining train samples. MixUp blends two images to create a new image by taking the weighted average of their pixel values and labels (Figure 3.5). CutMix cuts a random patch from one image to replace a patch of another image and mixes labels proportional to the patch area (Figure 3.6). This reduces overfitting as it encourages the model to learn more generalisable features instead of local patterns.

Methodology

Building on the final baseline model from Section 3.1.1, the training pipeline and configurations remain the same. The only differences are the collate function that forms each mini-batch in Dataloader for each augmentation type, and the utilisation of Soft Target Cross Entropy Loss instead of Cross Entropy Loss since these augmentations result in blended and non one-hot labels. Three experiments were conducted, which were the MixUp variant, CutMix variant, and combined MixUp and CutMix variant which selects either augmentation randomly for each batch. We included Expected Calibration Error (ECE), which measures the gap between confidence and accuracy. ECE shifts the evaluation focus from simple accuracy to the reliability of the model's predictions, since the true label is a distribution rather than a single fixed class.

Results

CutMix obtained the best performance, reaching a validation accuracy of 0.7775 and a test accuracy of 0.7627 (Figure 3.7). It also produced the strongest calibration with the lowest ECE on validation and test sets at 0.1623 and 0.1769 respectively, compared to MixUp at 0.3147 and 0.3061 and MixUp + CutMix at 0.2233 and 0.2050 (Figure 3.8).

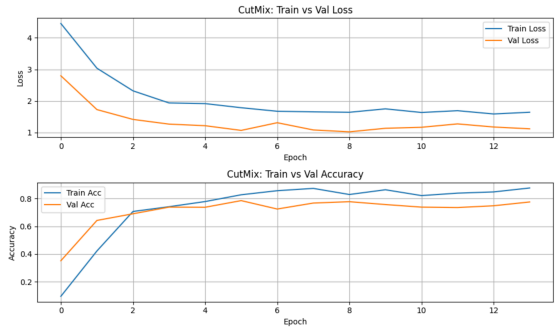


Figure 3.7: Loss and Accuracy Curves of CutMix Model

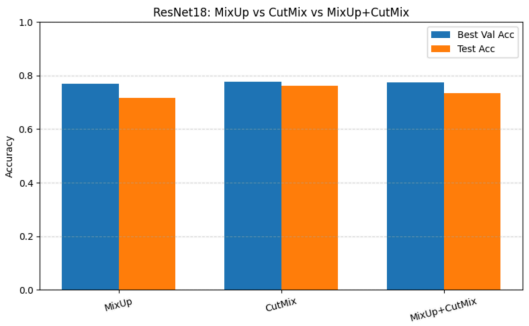


Figure 3.8: Comparison of Results of 3 Variants

Discussion

Compared to the baseline model in Section 3.1.1, these variants underperformed in terms of validation accuracy. This is expected as Flowers102 is a fine-grained task where classification relies on small details. MixUp's pixel averaging blurs subtle cues and results in greater difficulty for the model to classify correctly, causing the performance to flatten and early stopping to be triggered after only 8 epochs. CutMix variant has relatively higher accuracy and lower ECE than MixUp and CutMix + MixUp variant because CutMix still keeps large patches from each class intact, allowing the model to see these distinctive features more clearly. This implies that the CutMix variant is more accurate, reliable, and better calibrated.

3.1.4 ResNet18 Triplet Loss

Triplet loss is a loss function that encourages images from the same class to converge closer together and images from different classes to be further apart. It operates on anchor, positive and negative samples, reducing the distance between anchor and positive samples and increasing distance between anchor and negative samples (Appendix B).This improves class separability and fine-grained recognition, which is important for Flowers102 dataset where flower species might only have subtle inter-class visual differences but large intra-class variation (Figure 3.9).

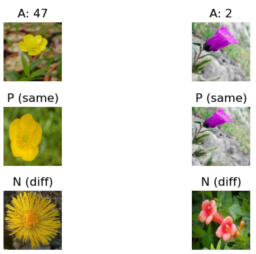


Figure 3.9: Anchor (A), Positive (P) and Negative (N) samples

Methodology

Building on the final baseline model from Section 3.1.1, a custom triplet data sampling is used to form anchor-positive-negative tuples for the mini-batches. Instead of only logits, the network creates a low-dimensional embedding. Two variations were tested. The triplet loss variant uses only Triplet Margin Loss (with a fixed margin of 1.0) as the loss function, and classification accuracy is determined by fitting a k Nearest Neighbours classifier (with k = 5) on

the learned embeddings. The triplet loss plus cross-entropy variant uses and computes a weighted sum of both Triplet Margin Loss (with a fixed margin of 1.0) and Cross Entropy Loss as the loss function for optimization. The test accuracy is computed directly using the classifier's predictions. Since this section focuses on geometry in the feature space, we used t-SNE of embeddings to visualise their performance on test set, by projecting the high-dimensional embeddings into a 2D space to inspect structures like clusters and overlaps.

Results

The triplet loss variant achieved a higher validation accuracy of 0.9029, but the test accuracy collapsed to 0.3511. On the other hand, while the triplet loss plus cross-entropy variant achieved a lower validation accuracy of 0.8059, it has a decent test accuracy of 0.7525.

Discussion

The triplet loss variant tends to overfit, as shown from the higher validation accuracy but much lower test accuracy. This shows the difficulty and brittleness of using kNN on our dataset, due to its small size and subtle differences between classes. The triplet loss plus cross-entropy variant achieved more stable generalisation and lower tendency to overfit. Looking at the t-SNE based on test set predictions, the triplet loss plus cross-entropy variant generated clearer and more separated color clusters, while the triplet variant produced more overlaps, indicating difficulty to separate the different species (Figure 3.10 & 3.11). While these variants produced more class-structured embeddings, they underperformed as compared to the baseline cross-entropy model in Section 3.1.1. This shows cross-entropy is still most effective in maximising accuracy, but metric learning can be useful with enhancements like margin and neighbourhood tuning.

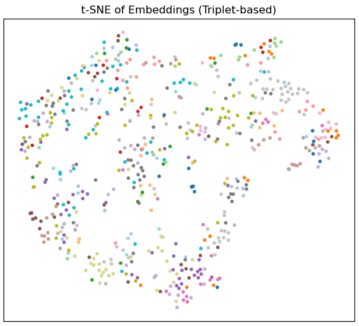


Figure 3.10: t-SNE of Embeddings for Triplet variant

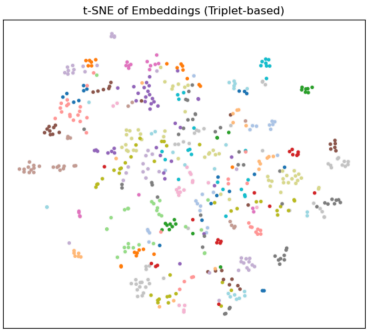


Figure 3.11: t-SNE of Embeddings for Triplet + CE variant

3.1.5 ResNet18 FewShot CNN Prototypical Classification

This model studies Few-Shot Learning using Prototypical Networks, which classifies images by measuring their distances to class prototypes and assigning images to the closest one. During meta-training, class prototypes are computed as the mean of the embeddings of support examples (Appendix C). This is useful for fine-grained classification and ensures robustness to noise in scenarios with limited data, by averaging across support samples.

Methodology

Building on the final baseline model from Section 3.1.1, the final FC layer is replaced with a linear projection and embeddings are L2-normalised. Episodic sampling with n-way k-shot episodes was used during training. During each episode, query samples are classified to class prototype with closest squared Euclidean distance, where the negative distances are passed to Cross Entropy Loss function. Validation episodes are evaluated every 20 episodes, with early stopping with patience of 10 epochs. After training, the best model is used to build K-shot prototypes (with K in [1, 3, 5]) from the train split. We then compute test accuracy using prototypical classification on the test split for each K.

Results

The episodic training produced validation accuracy of 0.9040, and the K-shot test accuracy was 0.5872, 0.5925 and 0.5970 for K = 1, 3 and 5 respectively (Figure 3.13). The curves show rapid early gains in the first 50-100 episodes, followed by small oscillations as the curves plateau (Figure 3.12).

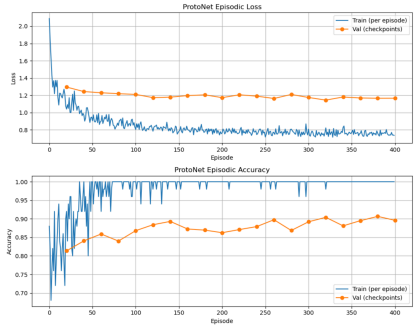


Figure 3.12: Loss and Accuracy Curves of ProtoNet

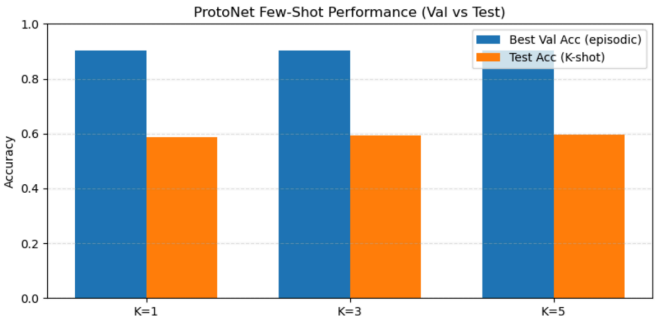


Figure 3.13: Comparison of Results of K=1,3,5

Discussion

Despite the high validation accuracy coming close to the baseline model performance, the K-shot test accuracy remained low at around 0.59. This could be because ProtoNet only relies on small sets of support samples and a fixed distance metric, causing overfitting to occur. This suggests ProtoNet might not be as effective on the Flowers102 dataset where there is limited labelled data and there are only subtle differences between classes.

3.2 VGG16

The Visual Geometry Group (VGG) family is a set of convolutional networks that stacks many small 3x3 filters to build deep networks with a simple and uniform design. We selected VGG16 as it is a strong transfer-learning baseline. It has an input layer, followed by 5 convolutional blocks containing stacked 3x3 convolutional layers with increasing number of filters and a max-pooling layer, followed by 3 FC layers (Appendix D). In the VGG16 Batch Normalisation (VGG16-BN) variant, batch normalisation layers are inserted after each convolutional layer to improve stability.

3.2.1 Baseline VGG16-BN Transfer Learning and Fine-Tuning

Building on the VGG16-BN transfer learning baseline, this section evaluates the effect of freezing layers and changing classifier heads on model performance. Freezing layers help to reduce computational cost and prevent overfitting. The original FC classifier head allows learning of complex relationships but might cause overfitting and high computational cost. The GAP classifier head can significantly reduce model size and improves generalisation but might not capture intricate patterns.

Methodology

Transfer learning initialises the backbone with pre-trained weights from ImageNet. We explored permutations of 2 classifier heads and 3 freezing strategies. For the 2 classifier heads, the last FC layer is replaced to match 102 classes in the FC head baseline, while features are pooled and passed through a dropout layer and linear layer in the GAP baseline. For the 3 freezing strategies, where the Full variant leaves all layers trainable, the Partial variant freezes the last convolutional block, and the Frozen variant freezes the entire convolutional backbone and leaves only the classifier trainable. Training uses these fixed configurations: initial learning rate of 0.0005, batch size of 32, L2 regularization of 0.0001, Adam Optimizer and early stopping with a patience of 5 epochs.

Classifier Heads	Fine-tuning Strategies
FC head, GAP head	Full, Partial, Frozen

Results

Across the 6 variants, the GAP head with full fine-tuning achieved the highest validation accuracy of 0.8637 and test accuracy of 0.8149, while the FC head with full fine-tuning followed closely behind with validation accuracy of 0.8520 and a higher test accuracy of 0.8445. Overall, full fine-tuning consistently outperforms, followed by partial then frozen fine-tuning (Figure 3.14).

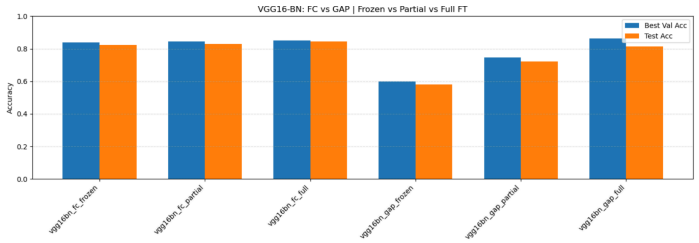


Figure 3.14: Comparison of Results of Heads and Fine-Tuning

Discussion

While GAP head achieved the highest validation accuracy, its lower test accuracy suggests that it results in loss of spatially important details to differentiate between classes. On the other hand, FC head produced higher test accuracies as it helps to learn and preserve discriminative patterns, allowing better transfer to unseen data while ensuring decent validation score. FC head was therefore chosen as the configuration in Section 3.2.2 instead of GAP head. While full fine-tuning tends to outperform as it allows learning of complicated patterns, partial fine-tuning also offers a competitive accuracy score. Partial fine-tuning can be used as a compromise when efficiency is required as it retains most of the full fine-tuning gains while reducing training time.

3.2.2 VGG16-BN with Attention Blocks

This section integrates attention mechanisms into VGG16-BN to pay attention to the most relevant features, hence improving feature extraction and reducing overfitting. The Squeeze-and-Excitation (SE) block focuses on channel attention, while the Convolutional Block Attention Module (CBAM) is a hybrid attention model combining channel and spatial attention. This is crucial for the Flowers102 dataset where classes might only differ by subtle cues that might be missed out, such as petal shape and leaf vein patterns.

Methodology

Using the FC classifier head on the baseline VGG16 model, we evaluated two attention modules by inserting them after the last 3 convolutional blocks. SE variant uses global average pooling followed by a multilayer perceptron, while CBAM

variant sequentially uses a channel attention module followed by a spatial attention module. We also explored 3 fine-tuning variants, full, partial and frozen, for each attention module type.

Attention Module	Fine-tuning Strategy
SE, CBAM	Full, Partial, Frozen

Results

The best model is the SE-Frozen variant, with the best validation accuracy of 0.8118 and test accuracy of 0.7749 (Figure 3.15) (Figure 3.16).

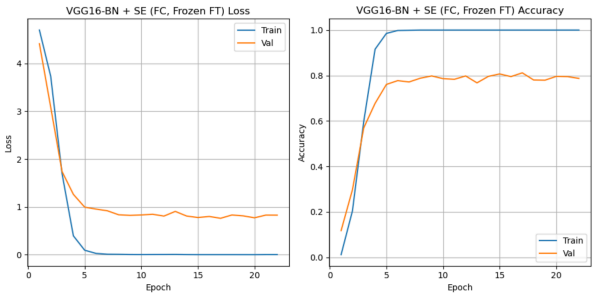


Figure 3.15: Loss and Accuracy Curves of SE-Frozen

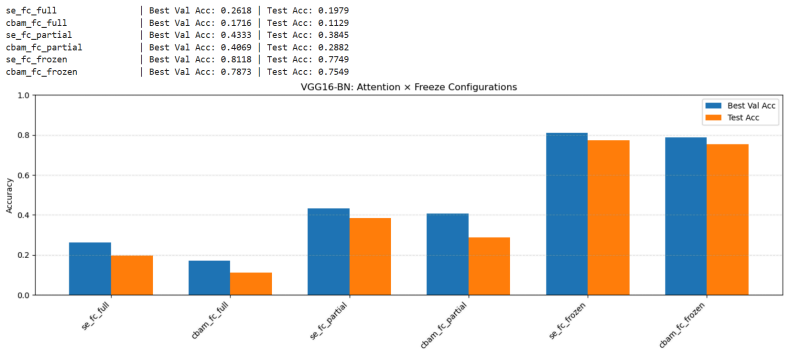


Figure 3.16: Comparison of Results of Attention and Fine-Tuning

Discussion

Both attention variants underperformed as compared to the baseline VGG16-BN model in Section 3.2.1. Across fine-tuning settings, SE consistently outperformed CBAM. This suggests that channel-only attention in SE block ensured stable optimization of the model, while the additional spatial attention in CBAM only introduced more parameters and hence greater noise, causing overfitting and a large gap between train and validation.

For fine-tuning strategy on VGG16-BN with attention, unlike the baseline model, frozen variants performed the best as attention only provided slight re-weighting following the pre-trained frozen backbone. Full and partial fine-tuning only destabilised the model and resulted in overfitting before the trainable convolutional filters could form meaningful adaptations. This is likely because of the small dataset size.

3.3 ResNet18 and VGG16-BN Hybrid

We created a hybrid model combining a pre-trained VGG16-BN backbone with a pre-trained ResNet18 backbone, initialised with ImageNet weights. This combines the strengths of VGG16's ability to capture complex patterns and ResNet18's efficiency and ability to build deeper networks with residual connections.

Methodology

The model runs both pre-trained backbones in parallel. For the ResNet18 branch, the final average pooling layer and FC layer is removed. For the VGG16-BN branch, the classifier stack is removed to keep only features. Each branch produces a 512 channel activation map that is passed through a GAP layer to produce a 512-dimensional vector. These 2 vectors are concatenated to form a 1024-dimensional hybrid feature, which is passed through a hybrid multi-layer classifier consisting a linear layer with ReLU activation, dropout layer and final linear layer to map to 102 classes.

Similar to Section 3.2.1, 3 fine-tuning freezing variants are evaluated. The Frozen variant freezes both backbones and trains only the hybrid classifier, the Partial variant additionally unfreezes the last ResNet18 stage and last VGG16 convolutional block, while the Full variant leaves both backbones and hybrid classifier trainable.

Results

The best hybrid model is the Full variant, with the best validation accuracy of 0.9186 and test accuracy of 0.8805 (Figure 3.17 & 3.18).

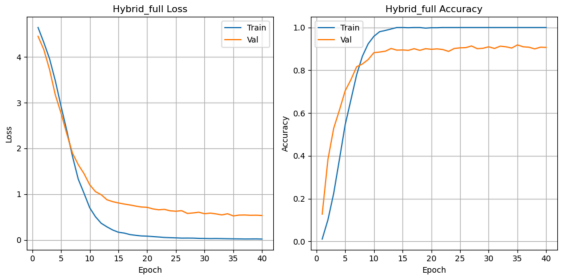


Figure 3.17: Loss and Accuracy Curves of Hybrid-Full variant

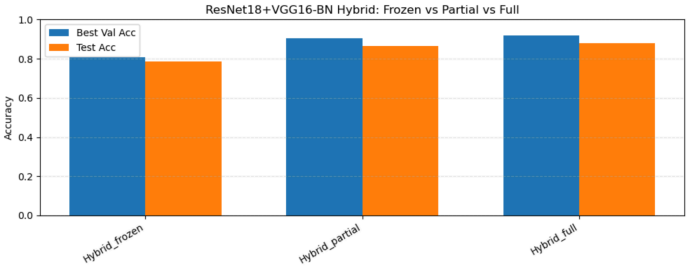


Figure 3.18: Comparison of Results of Fine-Tuning

Discussion

The hybrid model with full fine-tuning outperformed partial and frozen variants, with validation accuracy improving as more layers are unfrozen. This suggests that full fine-tuning allows the model to learn more complex features and adapt to Flowers102 dataset. Compared to both tuned ResNet18 and VGG16 baselines, the hybrid model achieved the highest validation and test accuracy across all CNN models. This shows concatenating the pooled features allowed us to reap the benefits of both networks, where ResNet18 branch contributed residual-refined semantics that stabilised optimisation and prevented overfitting, and VGG16 branch contributed to its dense local pattern sensitivity to allow better differentiation between flower classes. However, if resources are constrained, tuned ResNet18 baseline can be justified, with a competitive validation accuracy.

	Fine-tuned ResNet18	Fine-tuned Pre-trained VGG16	Hybrid with Full Fine-Tuning
Validation Accuracy	0.9137	0.8520	0.9186
Test Accuracy	0.8707	0.8445	0.8805

4 Vision Transformers (ViT)

ViT is a model that applies the transformer architecture adapted for computer vision tasks. However, unlike transformers that consist of encoder and decoder, ViT only uses an encoder (Appendix E) (Dosovitskiy et al., 2021). Instead of using convolutional layers like CNN, ViTs treat an image as a sequence of patches, like how words are treated as tokens in Natural Language Processing (NLP) models. ViT models are often pre-trained on large datasets, such as ImageNet-21K and JFT 300M and fine tuned using smaller datasets, achieving results that are on par with traditional convolutional networks while using considerably lesser computational resources during training.

4.1 Fine-tune Pretrained Vision Transformer: ViT-B/16

Methodology

We start with a ViT-B/16 model pretrained on the ImageNet 1K dataset and fine-tuned by training it on the Oxford Flowers-102 dataset to establish a transfer learning baseline. The final classification head of the pretrained model was replaced with a fully connected output layer producing 102 output classes, to align with the number of flower categories. Training was carried out with a limit of 20 epochs, Adam optimizer with a fixed learning rate of 0.0003 (Touvron et al., 2022) and a weight decay of 0.05. As this is a multi-class classification model, Cross Entropy is the loss function employed. Gradient clipping was applied to prevent the exploding gradients problem to ensure stable model updates. To avoid overfitting to the training data, early stopping with a patience of 3 epochs and a minimum loss improvement threshold of 0.001 was implemented, automatically halting the training process when validation loss stops improving.

Results

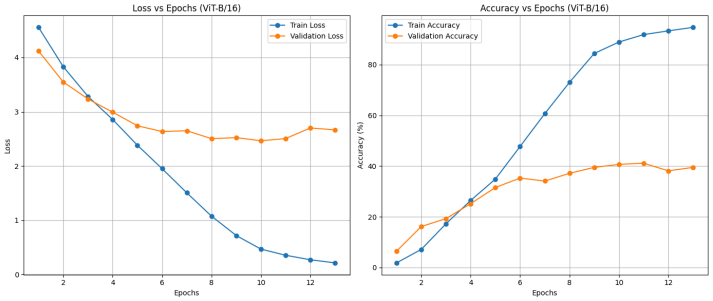


Figure 4.1: ViT-B/16 Loss Curve (left) and Accuracy Curve (right)

As seen in Fig 4.1, the model showed steady improvements in training performance across epochs, with training loss decreasing consistently. On the other hand, validation loss declined before stabilising at around epoch 10, and validation accuracy increased to about 0.4069 before plateauing. Early stopping was triggered at epoch 13 to prevent overfitting,

with the best validation loss of 2.4641 and validation accuracy of 0.4069. The fine-tuned model achieved a test accuracy of 0.3721 and a test loss of 2.6836, reflecting moderate to poor performance on unseen Oxford Flowers 102 test data.

Discussion

The performance of the model can be attributed to several factors. Firstly the Flowers dataset is relatively small as compared to the high capacity ViT model, causing it to be prone to overfitting, as seen in the divergence between training and validation loss and accuracy rates. Furthermore, hyperparameters such as the learning rate and weight decay are arbitrary values that are not finetuned, which may not be optimal values for this dataset. Therefore, these results suggest that the baseline configuration is not sufficient and hyperparameters tuning should be carried out.

4.2 Hyperparameters Tuning of ViT Model

Methodology

Hyperparameter tuning was done to improve the performance of the baseline ViT-B/16 model on the flowers dataset. As training of ViT is computationally demanding, running a full grid search over a large hyperparameter space is impractical. Furthermore, the Flowers dataset is relatively small, running an extensive grid search would be unnecessary and computationally inefficient. Therefore, to balance practicality and potential performance improvements, the search space was deliberately kept small to focus on the following hyperparameters and a simple grid search without cross validation was performed.

Fine-tuning strategy	Learning Rate	Weight Decay (L2 Regularisation)
`["head_only", "last_block", "full"]`	`[5e-3 ,1e-3, 5e-4, 1e-4]`	`[0.01, 0.05]`

These combinations were enumerated into parameter pairs and evaluated separately, resulting in a total of 24 configurations. For each configuration, the training and validation procedure followed the same setup as in section 4.1, using the Oxford Flowers-102 training and validation splits. Upon finding the optimal hyperparameters, we trained the pretrained ViT-B/16 model on the Flowers Dataset with the optimal hyperparameters found.

The fine-tuning strategy controls how much of the model is updated during training, ranging from updating only the classification head (`head_only`), partially updating the final transformer block (`last_block`) and updating all layers (`full`). As for learning rate, it determines how quickly the model adapts to the task, controlling the extent of adjustments the model makes to its parameters at each step of the optimisation algorithm. When deciding on the learning rate, it is important to acknowledge the trade off between speed of training and the stability of the final model. Lastly, weight decay acts as a regularizer to prevent overfitting, which is crucial for a model that is trained on a small dataset.

Results

The best performing configuration is as shown below:

Fine-tuning strategy	Learning Rate	Weight Decay (L2 Regularisation)
full	1e-4	0.05

Amongst the other configurations, this configuration achieved the highest validation accuracy of 0.9392. The ViT-B/16 model trained using this configuration achieved a test accuracy of 0.9024, with a test loss of 0.4248. All subsequent ViT models will adopt these hyperparameters unless specified.

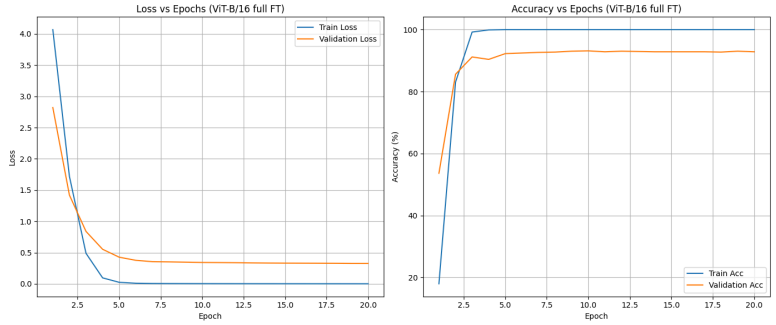


Figure 4.2 : Optimal Hyperparameters ViT-B/16 Loss Curve (left) and Accuracy Curve (right)

Discussion

Compared to the baseline model in 4.1, optimal hyperparameter configuration led to a substantial improvement in both accuracy and loss, with test accuracy increasing from 0.3721 to 0.9024. This significant increase shows the importance of proper hyperparameter choices. Full fine-tuning enabled the model to refine all layers to the fine-grained flower categories, while the low learning rate of 1e-4 ensured that the training process was stable, and the higher weight decay allowed for effective regularisation to prevent overfitting. As seen in Fig 4.2, the smooth curves and alignment between validation and training curves indicate that the model did not overfit despite full fine-tuning. In conclusion, these results highlight how by using optimal hyperparameters in training, it would give rise to an accurate and well-generalised model.

4.3 Visual Prompt Tuning (VPT)

We also explored more advanced techniques such as VPT, first introduced in the paper “Visual Prompt Tuning” by Jia et al. in 2022. VPT is a parameter-efficient alternative to full fine-tuning by learning a small set of trainable “prompts” inserted into the input space (Appendix F), while keeping the ViT backbone frozen. This approach significantly reduces the number of trainable parameters, lowering the risk of overfitting on small datasets and accelerates training while not compromising on the model’s performance.

Methodology

For this experiment, we adopted the VPT implementation provided in an open source repository on GitHub (TooTouch, 2023), which was based on the original paper’s implementation of VPT. We evaluated two variants introduced in the original work:

1. VPT-Shallow: Inserts prompt tokens only at input layer of transformer
2. VPT-Deep: Inserts prompts at every transformer block

Both variants were trained with the same configuration, instantiating a ViT-B/16 backbone with 5 tokens as length of prompt and 0 prompt dropout, together with the hyperparameters we have defined in the previous sections.

Results

VPT Type	Test Accuracy	Test Loss
Shallow	0.9798	0.1114
Deep	0.9776	0.0988

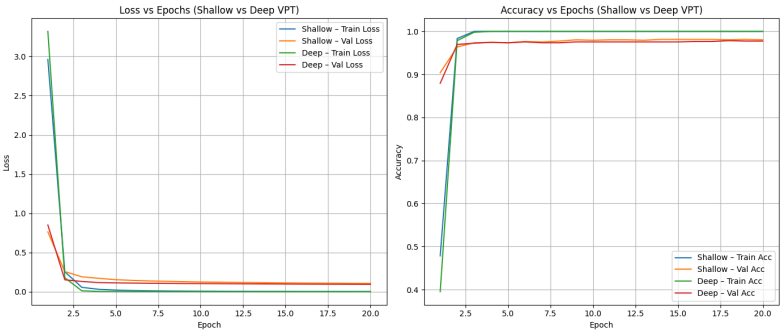


Figure 4.3: VPT ViT-B/16 Loss Curve (left) and Accuracy Curve (right)

Both shallow and deep VPT models converged within the first few epochs, with training and validation losses dropping sharply, stabilising at very low values. Both the shallow and deep models achieved similar test accuracies, at 0.9798 and 0.9776 respectively. The graphs in Fig 4.3 shows no signs of overfitting, as train and validation curves of each model do not diverge and are closely aligned throughout

Discussion

The strong performance of both variants of VPT highlights the effectiveness of VPT as a parameter-efficient method for the ViT-B/16 model on the Flowers-102 dataset. Despite freezing the entire backbone, high accuracies of almost 0.98 was achieved, surpassing the performance of the baseline model. This shows that a small number of learnable prompt tokens is sufficient to guide the ViT-B/16 model toward the downstream task.

4.4 Advanced Loss Functions (Label Smoothing Cross Entropy & ArcFace)

Methodology

To explore whether advanced loss functions could improve the model’s performance, we experimented with Label Smoothing Cross Entropy (CE) and ArcFace with the same ViT-B/16 architecture. Label smoothing tackles overconfidence and overfitting by slightly modifying the target labels. Instead of a one-hot encoded target label used in standard training, label smoothing puts a small amount of probability mass on all classes (Wong, 2019).

ArcFace, originally proposed for facial recognition, introduces Additive Angular Margin Loss in the classification layer, allowing the model to learn more discriminative feature embeddings, by increasing inter-class separability and tightening intra-class clusters (Deng et al., 2018). This technique is useful on the Flowers dataset as classes of flower have similar features and the added angular margin can help sharpen class boundaries and produce more discriminative embeddings.

Results

Loss	Test Accuracy	Test Loss
Label Smoothing CE	0.967312	1.086011
ArcFace	0.758660	2.281477

During the training of the model with Label smoothing CE, early stopping was triggered at epoch 10. The trained model achieved a test accuracy of 0.9673 with a test loss of 1.086. In contrast, ArcFace performance was significantly lower on this dataset, achieving a lower test accuracy of 0.7587, with early stopping occurring at epoch 9 during the training.

Discussion

With label smoothing applied to the cross entropy loss function, the model achieved higher test accuracy compared to training with the same hyperparameters but without label smoothing (4.2), implying the effectiveness of label smoothing. One reason for the subpar performance of the model employing Additive Angular Margin Loss is that the hyperparameters used in this experiment may not have been optimal for this configuration. Future works could involve tuning additive angular margin loss specific hyperparameters and experimenting with hybrid losses.

5 Conclusion

The table below summarises the best-performing models from each section and their respective test accuracies:

Model Family	Model	Test Accuracy
CNN	Baseline ResNet18 Transfer Learning and Hyperparameter Tuning	0.8707
	Pre-trained ResNet18 with 1 Frozen Layer	0.8801
	Pre-trained ResNet18 with CutMix	0.7627
	Pre-trained ResNet18 with Triplet Loss + Cross Entropy Loss	0.7525
	ResNet18 with Prototypical Networks and Few-Shot Learning	0.5970
	Baseline VGG16-BN Transfer Learning with FC Head and Full Fine-Tuning	0.8445
	Pre-trained VGG16-BN with SE Attention Blocks and Frozen Fine-Tuning	0.7749
	ResNet18 VGG16-BN Hybrid with Full Fine-Tuning	0.8805
Transformer	Fine-tuned ViT-B/16	0.3721
	Optimal Hyperparameter Fine-tuned ViT-B/16	0.9024
	VPT-Shallow	**0.9798**
	VPT-Deep	0.9776
	ViT-B/16 with Label Smoothing CE	0.9673
	ViT-B/16 with Additive Angular Margin Loss (ArcFace)	0.7587

Across all models, Visual Prompt Tuning models performed best on unseen data, with VPT-Shallow achieving the highest test accuracy of 0.9798 and having clean convergence while training only prompt tokens.

Among CNNs, the hybrid ResNet18 + VGG16-BN model with full fine-tuning produced the best results of 0.8805 test accuracy, with hyperparameter-tuned pre-trained ResNet18 following closely behind at 0.8707. Augmentation techniques and metric learning caused the CNN models to underperform, especially on the test set, reflecting CNN's difficulty to capture subtle cues.

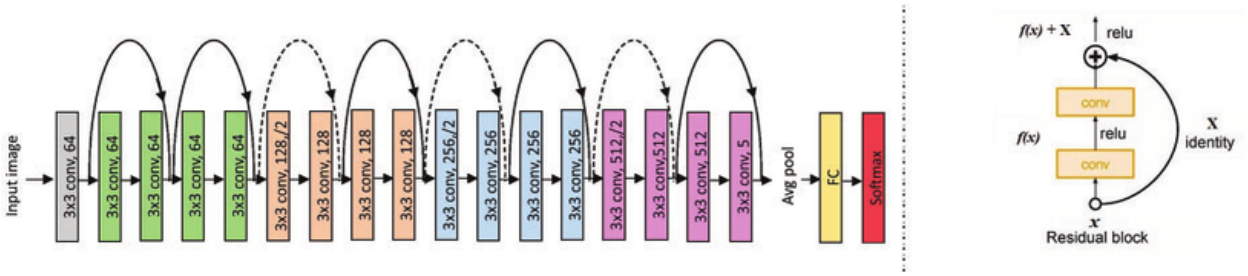
Architecturally, ViTs' self-attention mechanism can capture global long-range dependencies across the entire flower image, so augmentations still allow the model to generate coherent token-level context and produce higher accuracies with sufficient data (Mekal, 2025). On the other hand, CNNs rely on local features processed through filters, so augmentations like MixUp and CutMix can disrupt cues and limit the model's ability to capture distant relationships that differentiate classes as depth increases. This is only mitigated using ResNet's residual connections.

While ViTs seem to perform better on Flowers102 dataset, they might underperform compared to CNNs when dataset sizes are small, as ViTs usually require large datasets to learn effectively. Furthermore, ViT's self-attention mechanism causes higher computational costs. In the case of the Flowers102 dataset, VPT-Shallow is the most practical model as it achieves high accuracy while ensuring computational efficiency.

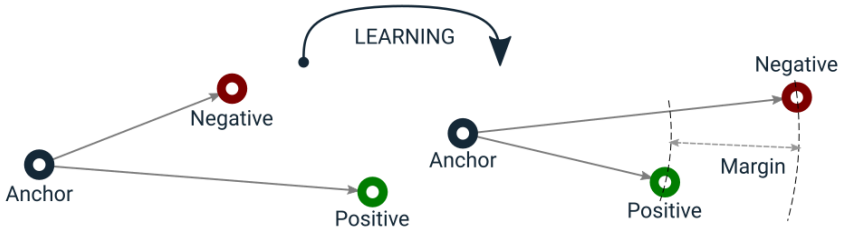
To further improve the performance of VPT-shallow, data augmentation techniques such as CutMix and MixUp can be applied to enhance the robustness of the model. Another promising extension is to combine VPT-Shallow with Label Smoothing Cross Entropy as the models that employed the 2 techniques separately were the strongest-performing. By applying label smoothing on top of VPT, it may further improve generalisation by reducing overconfidence when retaining the parameter-efficient benefits of prompt tuning done in VPT.

6 Appendix

Appendix A: ResNet18 Architecture (Ghassemi & Magli, 2019)

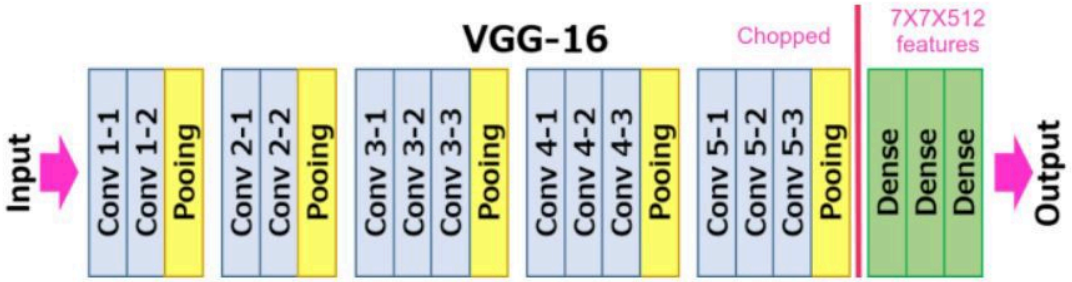


Appendix B: Triplet Loss Mechanism (Sarıgöz, 2022)

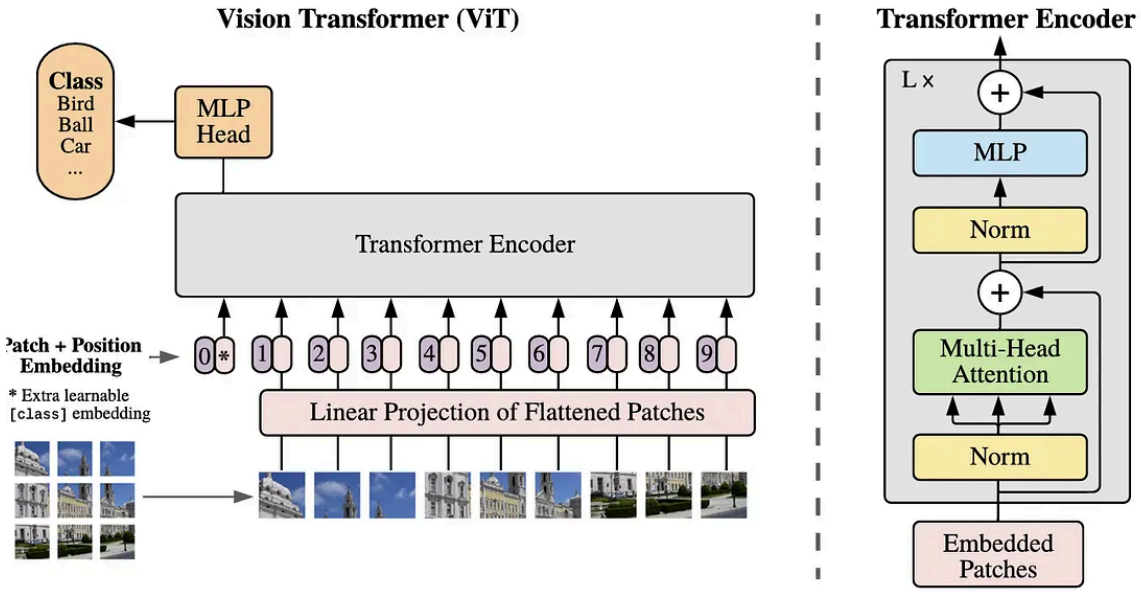


Appendix C: ProtoNet Support Examples

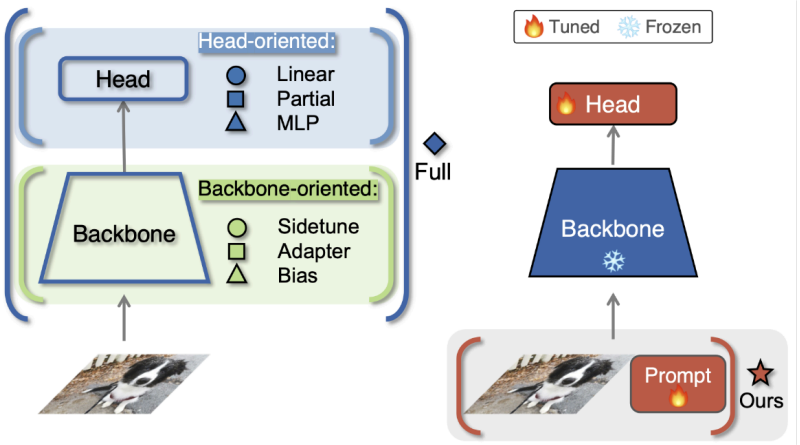
Appendix D: VGG16 Architecture (Tanaya, 2021)



Appendix E : Architecture of Vision Transformer



Appendix F: Existing Tuning Protocols vs VPT



(a) Existing tuning protocols (b) Visual-Prompt Tuning (VPT)

7 References

Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2018). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. ArXiv:1801.07698 [Cs]. https://arxiv.org/abs/1801.07698

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. https://arxiv.org/pdf/2010.11929

Ghassemi, S., & Magli, E. (2019, June 14). Convolutional Neural Networks for On-Board Cloud Screening. MDPI. https://www.mdpi.com/2072-4292/11/12/1417

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., & Lim, S.-N. (2022, March 23). Visual Prompt Tuning. ArXiv.org. https://arxiv.org/abs/2203.12119

Mekal, P. M. (2025, February 9). Transformers vs. CNNs: The Battle for Image Classification Supremacy. Medium. https://medium.com/@pmekal25/transformers-vs-cnns-the-battle-for-image-classification-supremacy-a4be1ef6e0f8

Sarıgöz, Y. (2022, March 24). Triplet Loss - Advanced Intro. Qdrant. https://qdrant.tech/articles/triplet-loss/

Tanaya, T. (2021, June 24). Transfer learning using VGG16 in Pytorch. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/transfer-learning-using-vgg16-in-pytorch/

TooTouch. (2023, May 15). GitHub - TooTouch/VPT: Unofficial Visual Prompt Tuning implementation. GitHub. https://github.com/TooTouch/VPT

Touvron, H., Cord, M., El-Nouby, A., Verbeek, J., Jégou, H., & Ai, M. (2022). Three things everyone should know about Vision Transformers. https://arxiv.org/pdf/2203.09795

Wong, W. (2019, December 17). What is Label Smoothing? Towards Data Science. https://towardsdatascience.com/what-is-label-smoothing-108debd7ef06/