



**Module 2 - Customize a Pre-trained Model for CV Classification**

**Sangayya Rudrayya Hiremath**

**Northeastern University: College of Professional Studies**

**ALY6080: Integrated Experiential Learning**

**Professor: Sergiy Shevchenko**

**Week 2: 3rd June, 2025**

# Introduction

Transfer learning using pre-trained convolutional neural networks (CNNs) has become a widely adopted strategy in computer vision (CV) tasks. In this assignment, I used the **Oxford 102 Flowers dataset, which includes 8,189 flower images classified into 102 categories**. I employed the ResNet-18 model pre-trained on ImageNet and fine-tuned it for flower classification using PyTorch. This report documents a case of overtraining, preventive strategies, optimal stopping point, and class-wise performance analysis.

1. **Overtraining Scenario**

To observe overfitting, I trained the model for 100 epochs without implementing early stopping or regularization. During this trial, training accuracy reached 99.8%, while validation accuracy plateaued around 70%. The training loss continually decreased, while the validation loss started increasing after approximately epoch 30. This gap between training and validation performance clearly illustrated overtraining. The model began to memorize the training data, including noise and irrelevant patterns, instead of learning generalizable features.

This was further evidenced by erratic predictions on the validation set and misclassification of visually similar flower species. Although the training metrics continued to improve, the degradation in validation performance signaled that the model had lost the ability to generalize. Additionally, visualizing the learned feature maps revealed that the model was focusing on background textures rather than the discriminative parts of flowers. These observations underscore the importance of monitoring validation loss and other generalization metrics throughout training, particularly when working with fine-grained classification tasks like the Oxford 102 Flowers dataset.

1. **Methods to Prevent Overtraining**

Several effective methods were implemented in follow-up experiments to combat overtraining:

* **Early Stopping:** A custom callback monitored validation loss and stopped training if there was no improvement for 10 consecutive epochs. This prevented unnecessary training beyond the point of generalization, typically halting at epoch 35–40. Early stopping not only conserved computational resources but also served as a safety mechanism to avoid performance degradation.
* **Data Augmentation:** Using PyTorch’s transforms, I applied RandomHorizontalFlip, RandomRotation, and ColorJitter. These transformations simulated realistic image variability, enhancing the model's robustness to unseen test images. Augmentation helped expose the model to a wider feature distribution, which improved generalization and reduced overfitting to specific camera angles or lighting conditions found in the original dataset.
* **Dropout and Regularization:** A dropout layer with a rate of 0.5 was added to the classifier head, and L2 regularization (weight decay = 1e-4) was applied to the optimizer. These techniques reduced the likelihood of the model overfitting to specific patterns in the training set. Dropout helped prevent co-adaptation of neurons, while L2 regularization discouraged the model from assigning excessive importance to any single feature.
* **Layer Freezing (Transfer Learning):** Initially, I froze all convolutional layers and trained only the final fully connected layers. Later, selective unfreezing allowed fine-tuning of deeper layers, which improved class-specific performance while minimizing overfitting risk. This gradual unfreezing strategy enabled a balance between leveraging pre-trained knowledge and adapting to the unique features of flower images.

These methods collectively improved validation accuracy to 85.6% while maintaining a relatively small gap from training accuracy. More importantly, they helped stabilize model performance across epochs, enabling consistent and reliable results across both training and validation datasets.

1. **Identifying the Optimal Stop Point**

The point at which the model should stop training became evident by monitoring the validation loss. After epoch 38, validation loss either plateaued or slightly increased, even as training loss continued to fall. Using early stopping with a patience value of 10 helped automatically identify this convergence point. Additionally, the learning rate scheduler ReduceLROnPlateau slowed the learning rate when improvements stalled, reinforcing the model’s readiness to stop.

This approach was particularly useful in avoiding the trap of overfitting while ensuring that the model had sufficient training time to capture complex patterns. Without this stopping strategy, the model risked deviating further from the optimal solution due to overspecialization on the training set. The convergence pattern was also confirmed through graphical visualization of loss curves, where the validation curve leveled off while the training curve continued downward. Moreover, monitoring metrics like validation accuracy and F1 score revealed diminishing returns after this point, indicating that further training would not yield meaningful improvements. Identifying this optimal stop point ensured both model generalizability and efficient training resource usage.

1. **Class-wise Performance Analysis**

The Oxford 102 Flowers dataset features fine-grained flower species, many of which are visually similar, increasing classification difficulty. A class-wise F1 score and confusion matrix revealed performance trends:

* **Best Performing Class:** *Sunflower (Class 65)* showed over 95% precision and recall. Its distinct color and structure contributed to superior performance.
* **Worst Performing Class:** *Wild Geranium (Class 41)* scored below 60% F1. It was often misclassified as similar-looking species such as *Common Dandelion* and *Cowslip*.

**Recommendations for Worst-Performing Classes:**

* **High-Resolution Inputs:** The original images were downsampled to 224x224. Training with higher-resolution variants could help capture more fine-grained features.
* **Class-Aware Sampling:** Implementing sampling strategies that give more weight to underperforming classes during batch selection can improve minority class learning.
* **Data Augmentation Targeting Low-Performing Classes:** Applying heavier augmentation specifically to underperforming classes can artificially increase diversity and reduce overfitting to dominant patterns.
* **Ensemble Learning:** Combining predictions from multiple models trained with different augmentations or architectures could increase robustness and accuracy for difficult classes.

In addition to these strategies, misclassified examples often exhibited poor lighting, background clutter, or partial occlusion, which blurred class boundaries and confused the model. Incorporating domain-specific preprocessing techniques, such as background subtraction or flower segmentation, could improve feature extraction. Fine-tuning deeper models like ResNet-50 or Vision Transformers (ViTs) may also help the network focus on critical floral structures and enhance class discrimination.

# Conclusion

This project demonstrated the application of transfer learning using a pre-trained ResNet-18 model on the Oxford 102 Flowers dataset. Overtraining was intentionally induced and observed, then mitigated through early stopping, regularization, and data augmentation. Model performance was best for visually distinctive flowers, while it lagged for species with high visual similarity. Fine-tuning strategies, higher-resolution inputs, and class-specific augmentation are recommended for future work to address these challenges. This assignment underlined the importance of thoughtful training strategies and evaluation in fine-grained image classification.

**References**

* He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.
* Nilsback, M. E., & Zisserman, A. (2008). Automated flower classification over a large number of classes. *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*.
* Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.