Ben-Gurion University of the Negev

Machine Learning Course

Assignment 4 – CNN Flower Classifiers

Lecturer: Dr Lior Rokah

Students: Jacob Yaacubov 312028970

Barak Milshtein 209309954

1. Introduction

This report presents the implementation and evaluation of two Convolutional Neural Network (CNN) architectures for flower classification using transfer learning. The objective is to classify images from the Oxford 102 Flowers dataset into 102 different flower categories.

Transfer learning is a machine learning technique where a model trained on a large dataset (such as ImageNet) is adapted to a new, related task. This approach is particularly effective when the target dataset is relatively small, as it leverages the learned features from the source domain.

Two architectures were selected for comparison:

- VGG19: A deep convolutional network known for its simplicity and effectiveness
- YOLOv5: A modern, efficient architecture adapted for classification

2. Dataset

The Oxford 102 Flowers dataset contains 8,189 images of flowers belonging to 102 different categories. The flowers were chosen to be commonly occurring in the United Kingdom.

2.1 Data Split

The dataset was split as follows:

- Training set: 50% (4,094 images)
- Validation set: 25% (2,047 images)
- Test set: 25% (2,048 images)

Two random splits were created using different random seeds (42 and 123) to analyze the effect of data distribution on model performance.

3. Image Preprocessing

Image preprocessing is crucial for CNN training. The following preprocessing steps were created:

3.1 Preprocessing for VGG19

Step	Description
Resize	Resize all images to 224x224 pixels
ToTensor	Convert PIL images to PyTorch tensors and scale pixel values to [0, 1]
Normalize	Normalize using ImageNet mean [0.485, 0.456, 0.406] and std [0.229, 0.224, 0.225]

Data Augmentation (Training Only)

- RandomHorizontalFlip: 50% probability of horizontal flip
- RandomRotation: Random rotation up to 10 degrees

3.2 Preprocessing for YOLOv5

Step	Description
Resize	Resize all images to 640x640 pixels
ToTensor	Convert PIL images to PyTorch tensors and scale pixel values to [0, 1]
Normalize	Normalize using ImageNet mean [0.485, 0.456, 0.406] and std [0.229, 0.224, 0.225]

Same data augmentation techniques (RandomHorizontalFlip and RandomRotation) were applied during training.

4. Network Architectures

4.1 VGG19 Architecture

VGG19 is a deep convolutional neural network developed by the Visual Geometry Group at Oxford. It consists of 19 layers with learnable weights (16 convolutional layers and 3 fully connected layers).

Original VGG19 Structure

- Features (Convolutional Layers): 16 convolutional layers with 3x3 kernels
- 5 Max Pooling layers (2x2, stride 2)
- Classifier: 3 fully connected layers (4096 -> 4096 -> 1000)

Transfer Learning Modifications

Component	Status	Description
Features (Conv layers)	FROZEN	Pretrained on ImageNet, weights not updated
Classifier [0-5]	TRAINABLE	Fully connected layers fine-tuned
Classifier [6]	REPLACED	New head: Linear(4096->512)->ReLU->Dropout(0.3)->Linear(512->102)

Parameter Count

- Total parameters: ~143 million
- Frozen parameters (features): ~20 million
- Trainable parameters: ~123 million

4.2 YOLOv5 Classification Architecture

YOLOv5 is a modern, efficient architecture originally designed for object detection. The classification variant (YOLOv5s-cls) was used for this task, pretrained on ImageNet.

YOLOv5s-cls Structure

- Backbone (Layers 0-8): CSPDarknet with C3 modules for feature extraction
- Layer 9: SPPF (Spatial Pyramid Pooling Fast) + Classification head
- Final layer: Linear(1280 -> 1000) for ImageNet classes

Transfer Learning Modifications

Component	Status	Description
Backbone (Layers 0-9)	FROZEN	Pretrained feature extractors, weights not updated
model.9.linear	REPLACED	Changed from Linear(1280->1000) to Linear(1280->102)

Parameter Count

- Total parameters: ~5.4 million
- Frozen parameters: ~5.3 million
- Trainable parameters: ~130,000 (final linear layer)

5. Training Configuration

5.1 Hyperparameters

Hyperparameter	VGG19	YOLOv5
Learning Rate	0.00001	0.001
Batch Size	256	256
Epochs	20	20
Optimizer	Adam	Adam
Loss Function	CrossEntropyLoss	CrossEntropyLoss
Image Size	224 x 224	224 x 224

5.2 Training Environment

- Framework: PyTorch 2.x
- GPU: NVIDIA Tesla T4 (15GB VRAM)
- Mixed Precision Training: FP16 with GradScaler
- Data Loading: 4 workers with pin_memory enabled

6. Results

6.1 Test Accuracy Summary

Model / Split	Split 42	Split 123
VGG19	[XX.XX]%	[XX.XX]%
YOLOv5	[XX.XX]%	[XX.XX]%

[TODO: Fill in your actual accuracy results from training]

6.2 Training Curves

The following graphs show the training and validation accuracy/loss over epochs:

VGG19 - Split 42

TODO:
[Insert vgg_split_42_accuracy.png and vgg_split_42_loss.png here]

VGG19 - Split 123

TODO:
[Insert vgg_split_123_accuracy.png and vgg_split_123_loss.png here]

YOLOv5 - Split 42

TODO:
[Insert yolo_split_42_accuracy.png and yolo_split_42_loss.png here]

YOLOv5 - Split 123

TODO:
[Insert yolo_split_123_accuracy.png and yolo_split_123_loss.png here]

7. Analysis and Discussion

7.1 Model Comparison

TODO:
Analyze which model performed better and why. TAKE INTO ACCOUTN THOSE :

- Architecture differences (depth, parameter count)
- Number of trainable parameters
- Pretrained features quality
- Convergence speed

7.2 Effect of Random Seeds

[ADD the Discussion how different data splits (seed 42 vs 123) affected the results. Did both splits produce similar accuracy? What does this tell us about model robustness accuracy?

7.3 Transfer Learning Benefits

benefits of transfer learning observed in this experiment

    - Faster convergence compared to training from scratch
- Better generalization with limited data
- Reduced computational requirements

8. Conclusion

- Which model achieved the best performance
- Whether the minimum accuracy requirement (>70%) was met
- Key takeaways about transfer learning for image classification
- Potential improvements for future work

9. References

1. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556
2. Ultralytics YOLOv5. https://github.com/ultralytics/yolov5
3. Nilsback, M-E. & Zisserman, A. (2008). Automated Flower Classification over a Large Number of Classes. Indian Conference on Computer Vision, Graphics and Image Processing.
4. Oxford 102 Flowers Dataset. https://www.robots.ox.ac.uk/~vgg/data/flowers/102/

10. Appendix

10.1 GitHub Repository

https://github.com/Jacob170/flower_classification

10.2 Dataset Link

https://www.robots.ox.ac.uk/~vgg/data/flowers/102/