AI and ML Fundamentals

Project: House Plant Species Classification
Group Members: Natig Alizada, Rustam Amirov, Hasan Alakbarzada
Report Submitted: 05.12.2024

I. Introduction

This project focused on using convolutional neural networks (CNNs) for *multi-class classification* of **houseplant species**. The dataset was split into **training**, *validation*, and **testing subsets**. The performance of **VGG16** and **ResNet-18**, both *pretrained* and *non-pretrained*, was compared to evaluate the impact of **transfer learning**. Additionally, the effects of **Adam** and **SGD** optimizers were analyzed across **8 training configurations**, with key metrics such as **accuracy** and **F1 score** used to assess results.

II. Dataset and Preparation

The dataset used in this project consists of **14,790 images** from **47 distinct houseplant species**, sourced from *Bing Images* and manually curated (not by professional biologists). The images were resized to 224×224 pixels using a Python script to ensure uniformity and reduce computational requirements. The dataset was then split into training (70%), validation (15%), and test (15%) sets.

While the dataset captures real-world variations in lighting, orientation, and background clutter, it also exhibits some **class imbalance**, which may impact model performance, favoring species with more images. This dataset diversity enhances its applicability but presents challenges for generalization.

A. Data Augmentation and Transformations

To enhance the model's generalization and reduce overfitting, several image transformations were applied. These included:

- **Random Rotation:** Each image was randomly rotated by up to 20 degrees.
- **Random Horizontal Flip:** A 50% chance of flipping images horizontally.
- **Resizing:** All images were resized to 224×224 for consistency and reduced computational load.
- **Normalization:** Images were normalized using ImageNet statistics: $[0.485, 0.456, 0.406]$ for mean and $[0.229, 0.224, 0.225]$ for standard deviation.
- **Tensor Conversion:** Images were converted to PyTorch tensors.

The training set received all augmentations, while the validation and test sets only underwent resizing and normalization. Example images were visualized to ensure the transformations were applied correctly without distorting key features.

Impact on Model Performance: The augmentations helped prevent overfitting and improved model robustness, especially with variations like rotation and flipping. Future work may explore additional transformations, such as cropping or color jitter, to further enhance the model's robustness.

Fig. 1. After augmentation and transformation

III. Model Architectures

In this work, four different deep learning architectures were implemented to evaluate the classification performance: two based on ResNet18 and two based on VGG16. Each architecture was used in both pre-trained (transfer learning) and non-pre-trained (from scratch) configurations. The details of these models are as follows:

a) ResNet18 Without Transfer Learning.: This model was built using ResNet18 without pre-trained weights. The feature extraction layers of the ResNet18 backbone were modified to include Batch Normalization layers after each convolutional layer to enhance convergence and stability. The fully connected layer at the end of the network was replaced with a new classifier containing a single linear layer with an output size equal to the number of classes (47).

b) ResNet18 With Transfer Learning.: The ResNet18 architecture with pre-trained ImageNet weights was employed for transfer learning. The final fully connected layer of the pre-trained model was replaced with a linear layer to adapt the model for our classification task. All other layers were retained, leveraging the learned features from the ImageNet dataset.

c) VGG16 Without Transfer Learning.: The VGG16 architecture without pre-trained weights was implemented. Batch Normalization layers were added after each convolutional layer to improve training efficiency. The original classifier of VGG16 was replaced with a new linear layer, which takes a flattened feature map of size 25,088 as input and outputs predictions for 47 classes.

d) VGG16 With Transfer Learning.: For transfer learning with VGG16, the pre-trained model was used, and the final layer of the classifier was replaced with a new linear layer to suit the dataset. The pre-trained layers provided a strong feature extraction base, allowing faster convergence and better performance.

e) Implementation Considerations.: For the transfer learning models, freezing or fine-tuning of layers was explored to optimize performance. The non-pretrained models required more epochs and careful regularization due to the absence of pre-learned features. Each model architecture was carefully designed to balance complexity and computational efficiency for the classification task.

A. Optimization Strategies

During the training phase, two optimization algorithms were employed: Stochastic Gradient Descent (SGD) and Adam. These optimizers were applied across both pretrained and non-pretrained versions of the ResNet18 and VGG16 architectures to evaluate their effectiveness.

SGD was implemented with a fixed learning rate and no momentum, providing a straightforward approach to optimization. Adam, on the other hand, was used for its adaptive learning rate capabilities, which can accelerate convergence, especially in scenarios with high variability in gradients.

For each model, experiments were conducted using both optimizers to compare their performance. This allowed for a comprehensive analysis of how the choice of optimizer influenced the convergence behavior and final accuracy, regardless of whether transfer learning was applied.

IV. Training and Validation of Models

The training process is managed by the `train_model` function, which iterates through epochs and batches to train the model. For each batch, the model computes the loss using the provided criterion, updates the model parameters, and logs the training loss. After each epoch, the function evaluates the model on the validation set. If a checkpoint exists, it is loaded to resume training from the last saved epoch.

The training loop involves setting the model to training mode, zeroing gradients, performing the forward pass, calculating the loss, and updating the model's parameters with the optimizer. The training loss for each batch is accumulated and averaged over the epoch, then logged using TensorBoard.

The validation loop evaluates the model's performance without updating gradients. The loss and F1 score are computed for the validation set. The softmax function is applied to the model outputs to compute probabilities, which are used to calculate the F1 score. These values are also logged using TensorBoard.

At the end of each epoch, the model's state and optimizer are saved to a checkpoint file to allow for resuming training if interrupted.

Optimizer configurations:

- ResNet18 pretrained with SGD and Adam Learning rate = 0.001
- ResNet18 not pretrained with SGD: Learning rate = 0.01, with Adam : Learning rate = 0.008
- Vgg16 pretrained Lr = 0.001 , not pretrained = 0.01

The models were trained for the following number of epochs:

- ResNet18 models: 30 epochs
- VGG16 model: 30 epochs

V. Model Comparison and Evaluation

For the evaluation of the models, TensorBoard was used to visualize and compare the training and validation metrics. The following graphs were analyzed:

- Training Loss
- Validation Loss
- Validation F1 Score

The models evaluated in this stage include ResNet18 and VGG16, with two optimizer configurations (SGD and Adam), both with and without transfer learning. The key observations and results are summarized as follows:

Key Observations

- **Transfer Learning vs. Training from Scratch:** Models trained using transfer learning consistently outperformed those trained from scratch across all metrics, highlighting the advantage of using pre-trained weights.
- **SGD vs. Adam Optimizer:**
  - **ResNet18 with Transfer Learning:** The model trained with the SGD optimizer achieved the fastest convergence, the lowest validation loss, and the highest validation F1 score, indicating strong generalization to the validation set. The Adam optimizer also performed well but slightly lagged behind SGD in convergence speed and generalization.
  - **VGG16 with Transfer Learning:** The SGD optimizer delivered the best performance, achieving the lowest training and validation losses and the highest validation F1 score. The Adam optimizer struggled to converge, with flat or inconsistent performance across all metrics.
  - **Models without Transfer Learning:** Both ResNet18 and VGG16 without transfer learning showed slower convergence, higher final losses, and lower F1 scores compared to their transfer learning setups, emphasizing the challenges of training from scratch on limited data.
- **Training Stability:** Models trained with SGD demonstrated more consistent performance improvements across epochs, while Adam showed slower or stagnant convergence, particularly in VGG16 configurations.

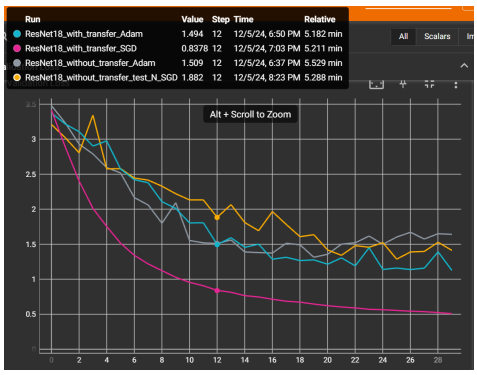

Fig. 4. Resnet18 Validation Loss graph

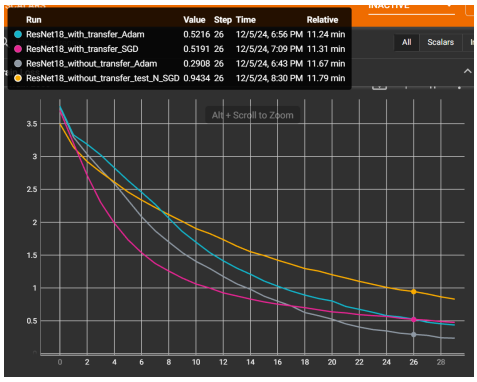

Fig. 2. Resnet18 Train Loss graph

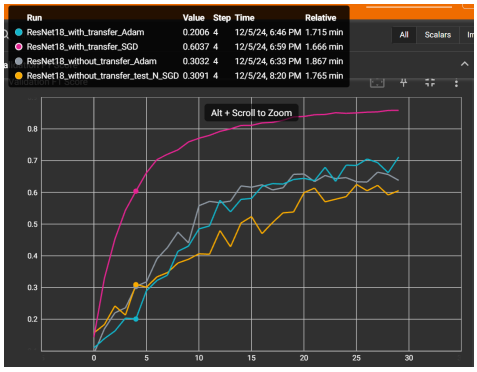

Fig. 3. Resnet18 Validation F1 scores graph

Model Performance Analysis

- **Training Loss:**
  - For ResNet18, the steepest decline was observed with transfer learning and the SGD optimizer, reflecting efficient optimization.
  - For VGG16, the best results were achieved with transfer learning and SGD, whereas the Adam optimizer struggled to converge, particularly with transfer learning.
- **Validation Loss:**
  - ResNet18 with transfer learning and SGD achieved the lowest validation loss, signifying effective generalization.
  - VGG16 with transfer learning and SGD also achieved the lowest validation loss. Models trained without transfer learning exhibited higher and more fluctuating validation loss.
- **Validation F1 Score:**
  - ResNet18 with transfer learning and SGD reached the highest F1 score, stabilizing quickly and indicating better precision-recall balance compared to other configurations.
  - VGG16 with transfer learning and SGD reached the highest F1 score among all VGG16 configurations. Models with the Adam optimizer performed poorly, particularly with transfer learning.

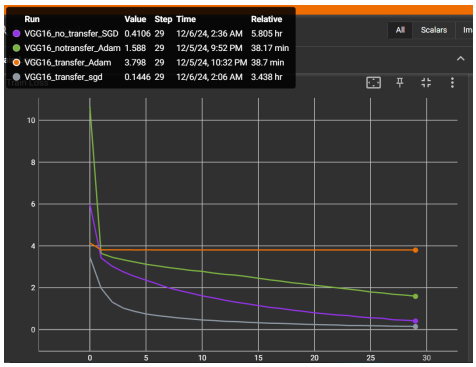

Fig. 5. VGG16 Train Loss graph

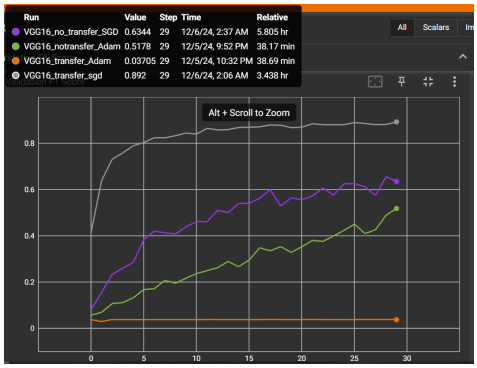

Fig. 6. VGG16 Validation F1 scores graph

Conclusion

Based on the results:

- The optimal configuration for ResNet18 and VGG16 was identified as **transfer learning with the SGD optimizer**. This setup provided the fastest convergence, lowest validation loss, and highest F1 score, demonstrating superior performance and generalization.
- Models without transfer learning struggled to match the performance of their transfer learning counterparts,

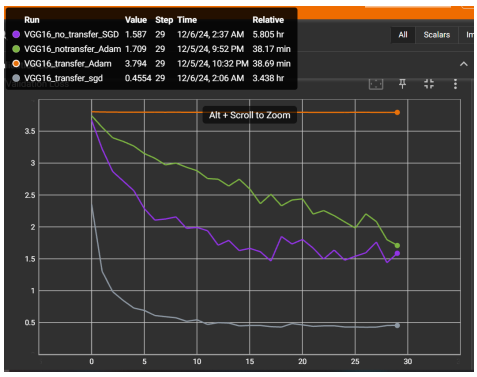

Fig. 7. VGG16 Validation Loss graph

underscoring the importance of leveraging pre-trained weights.

- The Adam optimizer underperformed in most scenarios, suggesting the need for further hyperparameter tuning to enhance its effectiveness.

For future experiments, the findings recommend prioritizing transfer learning approaches, optimizing SGD learning rates, and revisiting Adam optimizer configurations to address its limitations.

VI. Evaluating the Performance of Models

The performance of the trained models was evaluated on a separate test dataset to ensure their generalization capability to unseen data. Two primary metrics were used for evaluation:

- **Accuracy**: The proportion of correctly predicted samples out of the total samples in the test set.
- **F1 Score**: A metric combining precision and recall, particularly useful for datasets with class imbalances. The macro-average F1 score was computed to give equal importance to all classes.

A. Testing Process

1) **Model Checkpoints**: The models were saved as checkpoints corresponding to each configuration (e.g., ResNet18 with/without pretraining, VGG16 with/without pretraining, optimized with SGD or Adam). These checkpoints were loaded for testing.
2) **Test Dataset**: The test dataset, distinct from the training and validation datasets, contained diverse samples representing all classes.
3) **Evaluation Function**: A custom evaluation function was used to load the models, process the test data, and compute the accuracy and F1 score for each model. Additionally, class-wise accuracy was analyzed to assess performance across different categories.

B. Results

The evaluation results for each model configuration are summarized in Table I.

Model	Pretraining	Optimizer	Test Accuracy	F1 Score
ResNet18	Pretrained	SGD	0.8474	0.82
ResNet18	Pretrained	Adam	0.70	0.69
ResNet18	Not Pretrained	SGD	0.5985	0.58
ResNet18	Not Pretrained	Adam	0.6441	0.65
VGG16	Pretrained	SGD	0.8899	0.88
VGG16	Pretrained	Adam	0.0367	0.01
VGG16	Not Pretrained	SGD	0.6267	0.63
VGG16	Not Pretrained	Adam	0.4928	0.48

TABLE I
Test Accuracy and F1 Score for Each Model Configuration.

VII. Conclusion

The evaluation of models on the test dataset provides valuable insights into the effects of pretraining and optimizer selection on performance:

1. Effect of Pretraining: Models initialized with **pre-trained weights** significantly outperformed those trained from scratch. Pretraining provided a substantial boost in both accuracy and F1 score by leveraging feature representations learned from large-scale datasets. In contrast, models trained from scratch exhibited weaker performance, with lower accuracy and F1 scores, highlighting the challenges of learning features from limited data.

2. Impact of Optimizers: The **SGD optimizer** outperformed the Adam optimizer in most scenarios. Models trained with SGD achieved higher accuracy and F1 scores, particularly when combined with transfer learning. Conversely, Adam struggled in several configurations, with inconsistent convergence and significantly lower performance, especially for VGG16 with transfer learning, which resulted in near-zero F1 score and accuracy.

3. Model Comparison: Both **ResNet18** and **VGG16** showed competitive performance when transfer learning and SGD were used. VGG16 with transfer learning and SGD achieved the highest test accuracy (88.99%) and F1 score (0.88), making it the best-performing model overall. ResNet18 with transfer learning and SGD followed closely with 84.74% accuracy and an F1 score of 0.82. Without transfer learning, both models saw significant performance drops, with ResNet18 slightly outperforming VGG16.

Key Results Summary:

- **Best Model:** VGG16 with transfer learning and SGD (**88.99% Accuracy**, **0.88 F1 Score**).
- **Impact of Optimizers:** SGD consistently provided better generalization and stability compared to Adam.
- **Effect of Transfer Learning:** Transfer learning proved essential for achieving high performance, with pretrained models vastly outperforming those trained from scratch.

These findings emphasize the importance of using **transfer learning** and robust optimizers like **SGD** for training deep models on limited datasets. Future experiments could explore hyperparameter tuning for Adam to mitigate its limitations and further optimize VGG16 and ResNet18 configurations.