Report on a Denoising and Classification Pipeline

Introduction

This report outlines the design, implementation, and key decisions behind a machine learning pipeline developed to denoise and classify a dataset of flower images. The pipeline was designed with a strong emphasis on memory efficiency, modularity, and data integrity, ensuring robustness and scalability.

Denoising Method

The first stage of the pipeline aimed to remove noise from the input images and restore their quality. This was accomplished using a custom-built Denoising UNet Autoencoder.

- Model Choice: The UNet architecture was selected due to its effectiveness in image restoration tasks. Its skip connections enable the model to capture high-resolution details and effectively transfer them from noisy inputs to clean outputs.

- Performance: The model achieved high denoising accuracy, demonstrating a clear ability to distinguish between noise and underlying image features.

Memory Efficiency

Given the dataset's size, memory management was a central concern. The following strategies were applied:

1. Images were processed in small batches using torch.utils.data.DataLoader.
2. A TensorDataset wrapped the tensor of noisy images for DataLoader compatibility.
3. Each batch was passed through the UNet model for denoising.
4. Inference operations were enclosed within torch.no_grad() to disable gradient tracking and reduce memory usage.

Classification Approach

After denoising, the clean images were used as input for a Convolutional Neural Network (CNN) trained to classify different flower types.

- Architecture Enhancements: The CNN was fine-tuned with batch normalization and weight decay for stability and regularization.

- Hyperparameter Tuning: An extensive search was conducted to optimize learning rate, batch size, and other training parameters.

- Output Format: Predictions were saved in a two-column CSV (Image, Predicted_Class). A remapping step was included to align numeric predictions with user-defined class labels.

Key Design Decisions

1. Training the Classifier on Denoised Data

Instead of training directly on noisy inputs, the entire training set was first denoised and then used for classification. This separation improved the classifier’s ability to focus on true features (petal shape, color, texture) rather than noise artifacts, resulting in higher classification accuracy and better generalization.

2. Data Augmentation

To address dataset size limitations, a comprehensive augmentation strategy was implemented. Drawing on research from IEEE Xplore, transformations such as flipping, rotation, and color adjustments were applied, effectively tripling the dataset size. With a local GPU, this augmentation was computationally efficient and significantly improved model robustness.

3. Ensuring Data Integrity with Numerical Sorting

A key challenge arose from glob.glob, which sorts file paths as strings (e.g., 'Image_Test_10.png' before 'Image_Test_2.png'). This led to misalignment between images and labels. The issue was resolved by explicitly sorting image paths numerically before tensor conversion, ensuring correct mapping throughout the pipeline.

4. Modular Pipeline

The decision to treat denoising and classification as two distinct phases ensured modularity. Each model was specialized for its task, and improvements could be made independently to one component without disrupting the other. This modular design enhanced both flexibility and maintainability.

Conclusion

The denoising and classification pipeline successfully combined a Denoising UNet Autoencoder with a CNN classifier to achieve high-quality image restoration and accurate flower classification. Key design decisions—such as training on denoised data, applying effective data augmentation, ensuring numerical sorting, and maintaining a modular structure—were central to the pipeline’s robustness and scalability.

The result is a memory-efficient, flexible, and accurate pipeline capable of handling noisy datasets while delivering reliable classification results.