Neural_Network_Dataset_Processing

Introduction

Image classification has become one of the cornerstone tasks of computer vision, with applications ranging from medical diagnosis to autonomous navigation. In this project, we tackle the **Intel Image Classification** problem, where the objective is to categorize natural scene images into six distinct classes: **buildings, forest, glacier, mountain, sea, and street**. The dataset is provided by Kaggle and contains over 25,000 labeled RGB images, making it a well-suited benchmark for evaluating deep learning architectures.

The primary goal of this work is not only to build a classifier but also to adopt a **research-oriented iterative approach**. This means starting with a baseline CNN to understand the fundamentals, then progressively moving toward more sophisticated models such as **transfer learning with ResNet18**. At each stage, architectural decisions, evaluation metrics, and performance outcomes are documented to provide transparency and reproducibility.

Dataset

The dataset is sourced from the **Intel Image Classification** dataset on Kaggle. It is organized into three primary folders:

- **Train Set**: Used for model training.
- **Validation Set**: Derived from a split of the training data to monitor performance during training.
- **Test Set**: Used exclusively for final evaluation.

Each class contains thousands of images captured in diverse conditions, which introduces natural variability such as lighting, perspective, and scale. This variability enhances the robustness of the model by forcing it to generalize across different visual contexts.

Before training, all images are **resized to 224×224 pixels**, which ensures uniformity and compatibility with pretrained models like ResNet18. **Data augmentation** techniques such as random horizontal flipping are applied to increase diversity and reduce overfitting. Finally, **normalization** based on ImageNet statistics is used, aligning our dataset with the distribution expected by pretrained models.

Methodology

1. Baseline CNN

The baseline CNN serves as a simple yet effective architecture. It consists of three convolutional layers interleaved with ReLU activations and pooling layers, followed by a fully connected layer for classification. This design captures progressively complex features while maintaining computational efficiency. Although lightweight, this model helps establish a performance baseline against which more advanced models can be compared.

2. Transfer Learning with ResNet18

To improve performance, we employ **transfer learning** with ResNet18, a residual neural network pretrained on the ImageNet dataset. Residual connections help mitigate the vanishing gradient problem, enabling deeper networks to train effectively. In our implementation, we freeze the pretrained backbone layers to retain general image features while fine-tuning the final fully connected layer for our six-class problem. This approach leverages previously learned features, drastically reducing training time and improving accuracy compared to training from scratch.

3. Training Setup

- **Optimizer**: Adam with a learning rate of 0.001.
- **Loss Function**: Cross-entropy loss.
- **Metrics**: Training accuracy and **macro-averaged F1 score** on the validation set.
- **Device**: GPU (CUDA) where available, otherwise CPU.

We train models for multiple epochs, saving the **best-performing model checkpoint** based on validation F1 score. This ensures that the optimal model is preserved even if subsequent epochs lead to overfitting.

Results and Evaluation

The evaluation is carried out on the test set using **classification reports and confusion matrices**. These tools provide a detailed understanding of per-class performance.

The **baseline CNN** demonstrates reasonable accuracy, capturing broad visual patterns but struggling with classes that have subtle inter-class similarities, such as glaciers and mountains. In contrast, **ResNet18 significantly outperforms the baseline**, achieving higher accuracy and balanced performance across all six categories. This improvement validates the strength of transfer learning and pretrained models for real-world classification tasks.

The **confusion matrix** provides further insights into misclassifications. For example, some overlap is observed between glacier and mountain classes, likely due to their shared snowy textures. Similarly, urban categories such as buildings and streets occasionally overlap. These observations suggest potential improvements through more advanced augmentation techniques or fine-tuning additional layers of ResNet18.

Discussion

This project illustrates the **iterative nature of deep learning experimentation**. Starting from a basic CNN architecture provides intuition about convolutional filters and pooling operations, while transfer learning demonstrates the efficiency of leveraging pretrained models. The consistent use of evaluation metrics such as accuracy and F1 score ensures that performance comparisons remain objective.

Moreover, the workflow highlights practical considerations such as checkpoint saving, GPU acceleration in Colab, and reproducibility through version-controlled code and documentation. Such practices are essential in modern machine learning research, where transparency and reliability are as important as raw accuracy.

Conclusion

In summary, this project successfully develops an image classification system for the Intel dataset using both a baseline CNN and a transfer learning approach with ResNet18. The results confirm that transfer learning offers superior performance, particularly when datasets share characteristics with ImageNet.

Beyond model accuracy, this project emphasizes **best practices in research and experimentation**, such as iterative refinement, rigorous evaluation, and thorough documentation. These principles ensure that the work is not just a coding exercise but also a meaningful contribution to understanding how CNNs and transfer learning can be applied to scene classification.

Future extensions could involve experimenting with deeper architectures like ResNet50, applying advanced augmentation strategies such as color jitter or random crops, or even integrating attention mechanisms. Such modifications have the potential to further improve performance and robustness, making the model applicable to even more complex real-world scenarios.