DAT301m Lab 1: Image Classification with CNN using TensorFlow

Overview

In this assignment, you will implement convolutional neural networks (CNNs) for image classification a dataset. Using Tensorflow, you will develop custom CNN architectures, implement data augmentation techniques, and analyze model performance through various visualization methods.

Learning Objectives

- Design and implement CNNs for image classification
- Apply effective data augmentation and preprocessing techniques
- Visualize and interpret model performance
- Optimize CNN architectures for better performance

Requirements: Jupyter Notebook or Python Scripts, using Tensorflow.

Part 1: Core Requirements (8 points)

Completing Part 1 can give you a maximum of 8 points.

The **Intel Image Classification Dataset** is a dataset of Natural Scenes around the world. It contains around 25000 images of size 150x150 distributed under 6 categories: buildings, forest, glacier, mountain, sea, and street. The Train, Test and Prediction data is separated in each zip files. There are around 14000 images in Train, 3000 in Test and 7000 in Prediction. The dataset can be downloaded here: https://www.kaggle.com/datasets/puneet6060/intel-image-classification/data

Task 1.1: Dataset Exploration and Preprocessing (2 points)

The first task is to do data preprocessing and augmentation.

- Load the dataset using OpenCV or ImageDataGenerator.
- Implement data preprocessing, including normalization and resizing.
- Implement appropriate data augmentation techniques, such as using rotation, flipping, zooming, rescaling, or shifting.
- Analyze and visualize the dataset distribution and characteristics with interesting and insightful visualizations.

Task 1.2: Custom CNN Architecture (4 points)

Now, we build a custom CNN to train on the Train data. We will use Adam Optimizer with learning rate 0.01 and Cross Entropy loss to compile our model.

- Design a basic CNN architecture with at least 4 convolutional layers, using ReLU activation layer and no padding, with MaxPooling2D layer in between.
- For the classification layers, include at least 2 hidden fully-connected layers before the output layer.
- Train our model for around 10-20 epochs.

After that, we will explore extra ways that can help improve the model's performances, such as adding more layers or using more supporting layers like Batch Normalization, extra padding, or Dropout layers.

- Improve our model by adding more layers or implementing batch normalization and dropout for regularization, or anything else that you think is helpful.
- Tune hyperpameters and implement learning rate scheduling and early stopping.
- Train your model for at least 30 epochs.

Task 1.3: Performance Analysis (2 points)

For each model:

- Record and plot training and validation metrics.
- Create a confusion matrix for your test set predictions.
- Calculate precision, recall, and F1-score for each class.
- Compare the metrics to interpret your model's performance.

Answer the following question (Q1) in your report: How do your extra additions help improve the model’s performances? Justify the architecture design decisions and hyperparameter choices. If there are no improvements at all, explain why as well.

Part 2: Advanced Challenges (2 points)

To earn the remaining 2 points, we will explore more advanced options. Things we can do include:

- **Explore advanced classification architectures**, such as VGG, ResNet, DenseNet... We can replicate the architecture of those network, or create another custom CNN architecture with residual connections or dense blocks. Train this network for the same number of epochs, as in Part 1, tune any hyperparameters as you need, and use the metrics compare the performances of the models. Analyze the benefits and drawbacks of the advanced architecture.
- **Explore and use custom loss functions and optimizers:** Implement and experiment with custom loss functions, and explore other optimizers beside Adam and compare its performance with our standard implementation.
- Any other advanced option that can show considerable differences in the performances.

Answer the following question (Q2) in your report: How did your advanced architecture perform compared to Part 1's model? What are some possible advantages and disadvantages of such architecture/loss function/optimizer? Justify your answers. If there are no improvements at all, explain why as well.

What to submit:

1. Your well-documented Jupyter notebooks or Python scripts containing all implementations.
2. A report (2-6 pages) detailing:
   - Your approach to each task.
   - Analysis of results with supporting visualizations.
   - Justification for architecture and hyperparameter choices of each part.

Your Grades will be based on completing the required tasks and submitting a report. It is mandatory and directly influences the points awarded for each task. **Failure to submit a report can result in a maximum of 30% deduction.** You can zip all files and submit a single .zip or .rar file. If you reach maximum file size when submitting your work, you can either submit a Google Drive or GitHub repo link.

Here's a breakdown of each part:

Part 1: Core Requirements (8 points total)

- **Task 1.1: Dataset Exploration and Preprocessing (2 points)**
  - Loading and Preprocessing – 0.5 pt
  - Data augmentation techniques – 0.5 pt
  - Dataset visualization and distribution analysis – 1 pt
- **Task 1.2: Custom CNN Architecture (4 points)**
  - Initial CNN architecture and training with initial configuration – 1 pt
  - Model architecture improvement – 1 pt
  - Hyperparameter tuning and training the improved model – 1.5 pt
  - Proper documentation and justification in report – 0.5 pt
- **Task 1.3: Performance Analysis (2 points)**
  - Training/validation plots – 0.5 pt
  - Confusion matrix, precision, recall, F1-score – 0.5 pt
  - Interpretation, comparison and answer Q1 in the report – 1 pt

Part 2: Advanced Challenges (2 points total)

- **Exploration of advanced models or techniques**
  - Implementing advanced architectures or custom loss functions – 1 pt
  - Evaluation and comparison with baseline model – 0.5 pt
  - Explanation and justification and answer Q2 in the report – 0.5 pt