MT NHF Documentation

Project Documentation: Real-World Image Classifier with Small Datasets

Project Description

The objective of this project is to gather a real-world image dataset containing various types of objects and to train an image classifier to solve the classification task. The dataset consists of at least three distinct classes, each with a minimum of 50 images, ensuring diversity in view angles, backgrounds, and lighting. Examples of classification tasks include differentiating between books, shoes, and furniture or categorizing plants into houseplants, outdoor plants, and trees. The chosen classes must present a non-trivial classification challenge.

Given the limited dataset size, overcoming overfitting and improving generalization were primary challenges. Leveraging image augmentation and iterative neural network development, the project builds a robust image classification pipeline. Multiple models were trained and evaluated to compare their performance, with a focus on inspecting failure cases to guide improvements.

Related Works

GitHub Repository

- **CIFAR-10 Baseline with PyTorch Lightning**:GitHub Link

Research Paper

- **Deep Residual Learning for Image Recognition**:
  He, K., Zhang, X., Ren, S., & Sun, J. (2015).
  Research Paper

Function of Files

1. `Dockerfile`:

Facilitates building and running a Docker container for the project environment.

2. `requirements.txt`:

   Lists all required Python packages and their versions for replicating the project environment.

3. `ReadMe`:

   Provides concise documentation about the project setup and usage.

4. `mt_nhf.ipynb`:

   Contains the core implementation of the project, including data processing, model training, and evaluation.

Dataset

Data Collection

The dataset was created by gathering images from two sources, **unsplash.com** and **kaggle.com**. Each class was carefully curated to ensure diversity. For example:
-
Books: Images taken from multiple angles, such as front views, side views, and stacked arrangements.
-
Shoes: Features different footwear styles, such as sneakers, boots, and loafers, with varying backgrounds.
-
Furniture: Includes chairs, tables, and fridges in diverse environments.

This diversity ensures robust model generalization by providing features transferable to unseen scenarios. This step was labor-intensive, involving manual selection and verification of images for relevance and quality.

Data Augmentation

Data augmentation artificially increases dataset diversity by applying transformations to existing images. This helps improve generalization and reduces overfitting, especially with small datasets.

Common augmentations include:
-
Rotation: Simulates different viewing angles.
-
Flipping: Increases orientation diversity.
-
Scaling and Cropping: Focuses on different image regions.
-
Color Adjustments: Mimics different lighting conditions.
-
Adding Noise: Introduces random variations to make models robust.

In this project, rotation, flipping, and color adjustments were employed, ensuring enriched dataset variability to enable the neural networks to learn robust and generalized features.

Pipeline Overview

The project pipeline includes the following key components:

1. **Data Preprocessing**:
    - Resizing images to a standard input size for neural networks.
    - Applying data augmentation.
2. **Model Training**:
    - Training various neural networks with the processed dataset.
    - Implementing iterative development to optimize performance.
3. **Evaluation**:
    - Assessing model performance using metrics such as accuracy, precision, recall, and confusion matrices.

Models and Evaluations

Baseline Models

1. **Random Guessing**:

   This model serves as the simplest baseline, randomly assigning a label to each input image. While it provides no meaningful predictive power, it establishes a lower bound for performance comparison.

2. **Basic Neural Network**:

   A fully connected feedforward neural network with a small number of layers. This model provides an initial benchmark for performance and highlights the limitations of simple architectures on complex datasets.

Iterative Neural Network Development

1. **ResNet Model (Model 1)**:

   This model employs a Residual Network (ResNet) architecture made from scratch, which uses skip connections to mitigate the vanishing gradient problem. The idea was to use an architecture optimized for image classification. However, due to the small dataset size, this model severely overfitted.

2. **Simplified ResNet Model (Model 2)**:

   A streamlined version of the ResNet architecture, reducing the number of layers to decrease computational complexity. While this addressed overfitting partially, it did not achieve the desired outcomes.

3. **Overly Simplified ResNet Model (Model 3)**:

   A minimalistic ResNet approach with reduced depth and complexity. Despite applying class weights and data augmentation, this model still overfitted. As a result, we pivoted to a basic neural network architecture.

4. **Improved Neural Network (Model 4)**:

   This custom architecture builds upon the insights gained from earlier models. It improves the baseline model by introducing class weights to balance class dominance and applying data augmentation for better generalization.

Hyperparameter Optimization

We manually tuned hyperparameters to improve model evaluation metrics. Strategies included reducing the number of layers, introducing class weights to handle imbalanced data, and fine-tuning learning rates. These adjustments enhanced generalization and improved test performance.

Early Stopping and Model Checkpointing

To enhance training efficiency and prevent overfitting, we implemented:

-

EarlyStopping: Monitors validation loss and halts training if no improvement is observed over a set number of epochs.

-

ModelCheckpoint: Saves the best-performing model based on validation performance, ensuring optimal weights.

Evaluation

Each model's performance was evaluated using metrics such as accuracy, precision, recall, and ROC-AUC curves. Confusion matrices provided a visual understanding of misclassifications. For example, cluttered backgrounds in the "Books" class often led to misclassification as "Furniture," prompting the use of class weights.

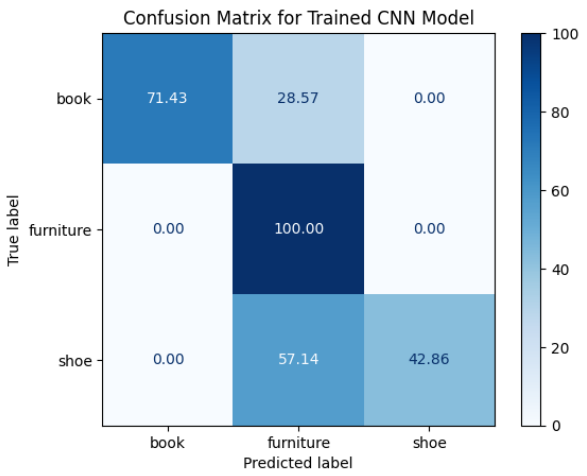
Best Model Performance:

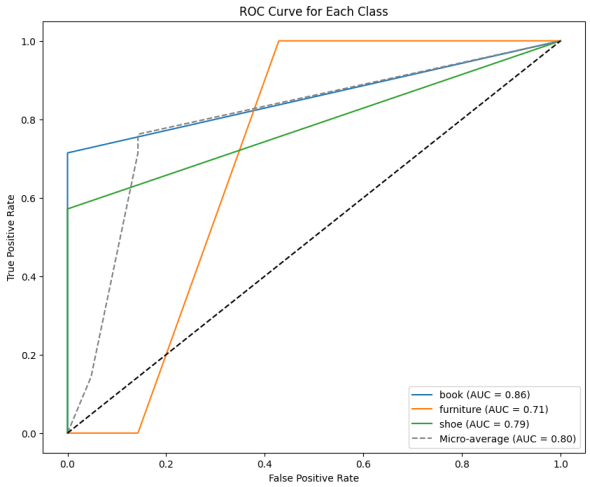
-

Accuracy: 0.71

- **Precision**: 0.85

- **Recall**: 0.71





LLM Usage

Large Language Models (LLMs) supported key aspects of our project. They provided ideas for neural network designs, assisted with Gradio implementation, and offered a solid base for creating professional-grade documentation. By leveraging LLMs, we streamlined development, resolved challenges efficiently, and maintained high-quality outputs across tasks.

Conclusion

This project demonstrates the feasibility of building an image classifier with a small dataset. By iterating on the model, using data augmentation, class weights, and analyzing model predictions, we achieved reasonably good results despite limited data resources.