Comparative Analysis of Custom CNNs vs Pretrained Image Models Using Federated Learning on CIFAR-10 Dataset

Abstract

This project explores the comparative analysis between custom Convolutional Neural Network (CNN) models and pretrained image models for image classification tasks using the CIFAR-10 dataset. The models were evaluated using federated learning, a decentralized machine learning technique where training is distributed across multiple devices without sharing raw data.

The custom CNN models were compared against pretrained models such as VGG16, ResNet50, and InceptionV3, with the goal of determining the effectiveness and efficiency of each approach in the context of federated learning.

Project Overview

This project focuses on performing a comparative analysis between custom CNN architectures and pretrained models (VGG16, ResNet50, InceptionV3) on the CIFAR-10 dataset. The goal is to evaluate each model's performance in terms of accuracy and efficiency, when trained using federated learning.

Federated learning was employed to train the models in a decentralized way across multiple clients, maintaining data privacy by training locally on client devices and sharing only model updates.

Key Features

- **Custom CNN Models**: Three custom CNN architectures designed for image classification.
- **Pretrained Models**: Use of VGG16, ResNet50, and InceptionV3 pretrained on ImageNet, adapted for CIFAR-10.
- **Federated Learning**: Distributed training setup simulating a client-server architecture for decentralized learning.
- **CIFAR-10 Dataset**: A standard dataset for image classification tasks with 60,000 32x32 color images across 10 classes.

Technologies Used

1. Python
2. TensorFlow
3. Keras
4. NumPy
5. Google Colab
6. Jupyter Notebook

Installation Instructions

1. Download the `instructions.ipynb` file from the location: `https://github.com/DarshiAshish/DIP_Group_3.git`.
2. Upload this notebook (`instructions.ipynb`) to your Google Drive under the folder `Colab Notebooks`.
3. Run the file `instructions.ipynb`.
4. After successful execution, three folders will be created in your Drive under the `Colab Notebooks` folder:
   - **demo**: Path in Colab: `/content/drive/My Drive/Colab Notebooks/demo`.
   - **DIP_proj**: Path in Colab: `/content/drive/My Drive/Colab Notebooks/DIP_proj`.
   - **DIP_Group_3**: Path in Colab: `/content/drive/My Drive/Colab Notebooks/DIP_Group_`

Note: The project is developed in Jupyter Notebook and is designed to run in a Colab environment. Do not attempt to run it locally.

Demo

To run a demo:

1. Navigate to the `demo` folder.
2. Run the `final_demo.ipynb` file.

How to Use

1. Run `pretrained_model_1.ipynb`, `pretrained_model_2.ipynb`, and `pretrained_model_3.ipynb` to retrieve embeddings. These will be automatically stored in your Drive. Precomputed embeddings (`.npz` files) are already provided.
2. Run `local_model_train_custom.ipynb` and `local_model_train_pretrained.ipynb` to perform federated learning.

3. Observe the metrics for each model in the above notebooks.

4. For a demo, navigate to the `demo` folder and run the `final_demo.ipynb` file.

Code Structure

- `read_data.ipynb` – Load and explore the CIFAR-10 dataset.

- `preprocess_data_1.ipynb` – Preprocess the CIFAR-10 dataset for random distribution.

- `preprocess_data_2.ipynb` – Preprocess the CIFAR-10 dataset for categorical distribution.

- `custom_model_1.ipynb` – Defines the first custom CNN model.

- `custom_model_2.ipynb` – Defines the second custom CNN model.

- `custom_model_3.ipynb` – Defines the third custom CNN model.

- `pretrained_model_1.ipynb` – Loads and retrieves embeddings using ResNet50.

- `pretrained_model_2.ipynb` – Loads and retrieves embeddings using VGG16.

- `pretrained_model_3.ipynb` – Loads and retrieves embeddings using InceptionV3.

- `pretrained_second_layer.ipynb` – Defines the second layer of a pretrained model for further training.

- `local_model_train_custom.ipynb` – Trains custom models using federated learning and aggregates updates.

- `local_model_train_pretrained.ipynb` – Trains embeddings retrieved by pretrained models using federated learning and aggregates updates.