README

Overview

This repository hosts three Python scripts focusing on deep learning models, with applications in data security and federated learning. Below is an outline of each file:

1. **Data_Security_Level1.py**
    - Implements a convolutional neural network (CNN) for image classification using the MNIST or CIFAR-10 datasets.
    - Contains utilities for training, validation, and plotting model performance.
2. **Federated_Learning_Level2.py**
    - Demonstrates a federated learning setup where local models are trained by multiple clients and aggregated on a central server using the FedAvg algorithm.
    - Supports experiments with MNIST and CIFAR-10 datasets.
3. **Data_Security_Level3.py**
    - Builds on the federated learning structure with:
        - Simulations of malicious client behavior.
        - Mechanisms to detect suspicious updates.
        - Adaptive sampling for balanced client participation.

Requirements

- Python 3.8 or newer
- GPU compatibility (CUDA 11.0 or higher recommended)
- Required Python packages:
    - torch (PyTorch library)
    - torch vision
    - NumPy
    - matplotlib

Install these dependencies by running:

pip install torch torch vision matplotlib NumPy

Usage Instructions

Executing the Scripts

1. Data_Security_Level1.py

- This script trains a CNN model using either the MNIST or CIFAR-10 dataset.
- Run the script with:

    python Data_security_level1.py
- By default, it processes the MNIST dataset. To use CIFAR-10, adjust the dataset parameter in the main() function:
  main('CIFAR10')

2. Federated_Learning_Level2.py

- Simulates federated learning with multiple distributed clients.
- To execute:

  python federated_learning_level2.py
- Defaults include 10 clients and MNIST. Modify settings like dataset or number of clients in the federated_learning() function.

3. Data_Security_Level3.py

- Simulates federated learning with added security features like malicious client detection.
- Run this script using:

  python Data_security_Level3.py
- The default configuration supports 10 clients and MNIST. You can customize parameters like client count or detection thresholds in the train_main() function.

Outputs

- Model checkpoints are stored as .pth files.
- Training metrics, such as accuracy and loss, are visualized with matplotlib plots.
- Detection results for malicious clients are logged in the output.

Additional Notes

- Ensure adequate disk space for downloading datasets like MNIST or CIFAR-10.
- A GPU-enabled system significantly accelerates training.
- Modify batch sizes or learning rates to optimize for your specific hardware.

License

Attribution

The structure of the federated learning process in this repository draws inspiration from established implementations of the FedAvg algorithm (plagiarism retained).

Key Outputs

- Model checkpoints are saved as `.pth` files.
- Training accuracy and loss metrics are visualized using `matplotlib`.
- Federated learning scripts output detection results for malicious clients.