EFDOPara

🚀 Stop waiting, start training! Slash training times for geophysical neural operators on multi-GPU setups with our dynamic training strategy, without sacrificing accuracy.

1. 🛠️ Installation

You can set up your environment in two ways: manually with Conda/Mamba or using our pre-built Singularity container for maximum reproducibility.

1.1. Option 1: Manual Installation with Mamba/Conda

> 💡 Pro tip: We recommend using `Anaconda` with `Mamba` for lightning-fast package installation!

1.1.1. Step 1: Get Mamba Up and Running

First, grab Mamba from the Mambaforge download page:

```
bash Miniforge3-Linux-x86_64.sh -b -p ${HOME}/miniforge
```

1.1.2. Step 2: Set Up Your Environment

Add these magic lines to your `~/.bashrc`:

```
# conda
if [ -f "${HOME}/miniforge/etc/profile.d/conda.sh" ]; then
    source "${HOME}/miniforge/etc/profile.d/conda.sh"
fi
# mamba
if [ -f "${HOME}/miniforge/etc/profile.d/mamba.sh" ]; then
    source "${HOME}/miniforge/etc/profile.d/mamba.sh"
fi

alias conda=mamba
```

1.1.3. Step 3: Create Your EFDO Environment

```
conda create -n torch python=3.11
conda activate torch
```

1.1.4. Step 4: Install Dependencies

```
# Install PyTorch with CUDA support
conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c nvidia

# Install other required packages
conda install torchinfo pyyaml numpy scipy pandas matplotlib jupyter notebook
pip install ray
```

1.1.5. Step 5: Get the Code

```
git clone https://github.com/CUG-EMI/EFDOPara
```

1.2. Option 2: Using the Singularity Container (Recommended for HPC)

1.2.1. Download the Container:

For maximum reproducibility and to avoid complex environment setups, we provide two pre-built Singularity images. Please choose the one that matches your training needs.

1. **Standard PyTorch Container**
   This container is suitable for standard single-node, multi-GPU training (using torch.distributed.DistributedDataParallel).
2. **Horovod Container**
   This container is specifically designed for multi-node distributed training experiments using the Horovod framework.

Important: The Horovod environment, with its specific MPI and NCCL dependencies, is complex and error-prone to configure from scratch. To ensure a correct and stable environment, we strongly recommend using this pre-built container for all Horovod-related tests.

- **Singularity Image Link**: Download Singularity Image (*.sif)

1.2.2. Run the Container:

Place the downloaded `.sif` file in your `EFDOPara` project directory. Then, run the following command in your terminal to start an interactive shell:

```
# To start the standard PyTorch container
singularity shell --nv pytorchv2.1_2.7.0.sif

# To start the Horovod container
singularity shell --nv horovodv2.sif
```

> **Note**: The `--nv` flag is crucial for enabling NVIDIA GPU support.

Your project directory will be automatically mounted inside the container, and you can start running experiments immediately!

2. 💾 Get the DataSet

Our pre-processed training and testing datasets are required to run the examples.

You can download the dataset from the link below. After downloading, please unzip the file and place the contents into the `data` directory within the project's root folder.

- **Dataset Link**: Download Dataset (*.mat)

Based on the file structure you've provided, here is a "Usage" or "How to Run" section for your `README.md` file. This guide explains how to run the experiments, focusing on the Slurm scripts for easy reproduction of results.

---

3. 🚀 How to Run

This project contains the code for three different training strategies, located in their respective directories:

- `code_fixed/`: For the fixed-batch-size strategy.
- `code_floating/`: For the floating-batch-size strategy.
- `code_dynamic/`: For our proposed dynamic-batch-size strategy.

Each directory contains the necessary scripts to run the experiments for that specific strategy.

3.1. File Structure Overview

Inside each `code_*` directory, you will find:

- `.py` **scripts**: These contain the core Python logic.
  - `EFDO_main.py`: The main script for single-GPU (non-parallel) execution.
  - `EFDO_main_parallel.py`: The main script for multi-GPU training using `torch.distributed`.
  - `EFDO_main_parallel_HVD.py`: The main script for multi-GPU training using Horovod.
  - `EFDO_main_parallel_MPI.py`: The main script for multi-GPU training using MPI.
- `config_EFDO.yml`: The configuration file where all hyperparameters and settings are defined. Experiments are organized by `item` names within this file.
- `.slurm` **scripts**: These are job submission scripts for the Slurm workload manager, designed to run the experiments without needing to modify the Python code directly.

3.2. Quick Start: Reproducing Results with Slurm

The easiest way to reproduce the results from our paper is to use the provided Slurm scripts.

3.2.1. Step 1: Navigate to a Strategy Directory

Choose the strategy you want to run and navigate into its directory.

```
# Example for the fixed strategy
cd code_fixed/
```

3.2.2. Step 2: Configure the Slurm Script

3.2.2.1. Method 1: Running with a Conda Environment (Bare-Metal)

This method runs the Python scripts directly on the cluster nodes. It requires you to have a fully configured Conda environment as described in the "Manual Installation" section. This approach is suitable for simpler setups like the MPI-based training.

Anatomy of a Conda-based Slurm Script (`EFDOMPI_\*.slurm`):

1. **Activate Environment**: The script first activates your Conda environment.

```
# Activate Conda Environment
source /path/to/your/mambaforge/etc/profile.d/conda.sh
conda activate torch
```

2. **Define Script Path**: It defines the absolute path to the Python script to be executed.

```
# Define the script path
SCRIPT_PATH="/path/to/your/code_fixed/EFDO_main_parallel_MPI.py"
```

3. **Execute**: The script is launched using an appropriate runner, such as `mpiexec`.

```
# Excute the script
mpiexec -np 1 python "$SCRIPT_PATH" --item EFDO_config_20250505_MPI_1GPU
```

> **Note**: `-np` is an abbreviation for GPU Number of Processes.

3.2.2.2. Method 2: Running with a Singularity Container (Recommended)

This is the **highly recommended** method, especially for complex environments like Horovod, as it guarantees a correct and reproducible setup.

Anatomy of a Singularity-based Slurm Script (`EFDOHVD_\*.slurm`):

1. **Load Modules**: The script loads the necessary `singularity` and `cuda` modules on the host system.

```
# Load singularity and cuda modules
module load singularity
module load nvidia/cuda/12.2
```

2. **Execute Container**: The core of the script is the `singularity exec` command, which runs a command inside the container. Let's break it down:

   Bash

```
# Excute the script by singularity
singularity exec --nv \
    -B /path/to/your/EFDOPara:/workspace \
    /path/to/your/horovodv2.sif \
    bash -c "cd /workspace/code_fixed && horovodrun --gloo -np 1 python
EFDO_main_parallel_HVD.py --item ..."
```

   - `--nv`: Enables NVIDIA GPU support inside the container. **This is essential.**
   - `-B /host/path:/container/path`: This mounts your project directory on the host machine (`/path/to/your/EFDOPara`) to a directory inside the container (`/workspace`). **You must update the host path to match your system.**
   - `/path/to/image.sif`: The absolute path to the Singularity image file.
   - `bash -c "..."`: The command to execute inside the container. Here, we first `cd` into the project directory (`/workspace/code_fixed`) and then launch the script with `horovodrun`.

3.2.3. Step 3: Submit the Job

Once the Slurm script is configured, submit it to the queue using `sbatch`.

```
# Submit the Job
sbatch EFDOPara_8GPUs.slurm
```

The job will now be scheduled, and the output and error logs will be saved to files (e.g., `job_12345.out` and `job_12345.err`).

4. 🏆 Pre-trained Models

To help you get started quickly and for easy comparison with the results in our paper, we provide the final, trained model checkpoints (`.pkl` files).

You can download the models from the link below. After downloading, please place the `.pkl` files into the `model/` directory in the project's root folder.

- **Model Checkpoints Link**: Download Models (*.pkl)

Citation:

```
@article{liao2025fast,
  title={Fast forward modeling of magnetotelluric data in complex continuous media using an extended Fourier DeepONet architecture},
  author={Liao, Weiyang and Peng, Ronghua and Hu, Xiangyun and Zhang, Yue and Zhou, Wenlong and Fu, Xiaonian and Lin, Haikun},
  journal={Geophysics},
  volume={90},
  number={2},
  pages={F11--F25},
  year={2025},
  publisher={Society of Exploration Geophysicists}
}
```