Training AI Models

Our platform facilitates the creation and training of machine learning models. We offer comprehensive instructions for establishing a development environment and initiating initial training sessions. Additionally, we demonstrate the process of testing and validating trained models through metric logging into our AI Workbench. This guide is designed to assist you in configuring your environment on an HPC cluster and executing training tasks on extensive datasets utilizing multiple GPUs.

Training AI models

Our Platform supports the user with training their own weights for predefined models and even to create their own models. In the first case this could help to improve the quality of the model by training in on data coming from a particular seismic networks (see e.g. Johnson et al. 2021).

To start with training we recommend to use the our HPC clusters which provide enough computing power and access to GPU. The prerequisites are following:

1. **Active account on PLGrid Portal**. To create an account see the official guide available in English and Polish.
2. **Computing grant on PLGrid Infrastructure**. The resources needed for training are highly depended on the size of a training dataset. For a dataset of about 10^4 samples a single training session lasts about 10 minutes on a single GPU for models like GPD or PhaseNet. Please keep in mind that fine tuning of the model hyper-parameters usually requires tu run training sessions multiple times. Therefore applying for a grant, please use this numbers only as a starting point.
   We recommend to apply for resources on a cluster **Athena** or **Ares**, but it is possible to run the training on other clusters as well. Please apply for the CPU and GPU computing time, as the GPU significantly accelerates the training process.
3. **Miniforge installation**.
   Miniforge is a Python distribution based on Conda. This is a preferred way to run scripts and notebooks distributed on our Platform. It is possible to install and run on other python distributions, but we provide support only for Miniforge/Anaconda.

   The installation starts with downloading the Miniforge binary from the official project site for your platform. Then follow the instructions in the official guide.
4. **Environment for training AI models**. Please install the `epos-ai-train` environment, by downloading `epos-ai-train.yml` from our repository and running:

   ```
   mamba env create -f epos-ai-train.yml
   ```

5. **Activate the environment it**. is necessary to run for each new shell session:

   ```
   conda activate epos-ai-train
   ```

   To check if you have enabled the environment correctly you should see the name of the environment in the shell prompt.

Running jobs on an HPC cluster with SLURM

To submit a job on an HPC cluster use the following job script scheme:

```
#!/bin/bash

## Set you JOB_NAME to to make it easier to see in the job queue
#SBATCH --job-name=JOB_NAME

## Max task execution time (format is HH:MM:SS)
#SBATCH --time=00:15:00

## Name of grant to which resource usage will be charged
#SBATCH --account=GRANT_ID

## Name of partition
#SBATCH --partition=plgrid

## Number of allocated nodes
#SBATCH --nodes=1

## Number of tasks per node (by default this corresponds to the number of cores allocated per node)
#SBATCH --cpus-per-task=1

## Change to sbatch working directory
cd $SLURM_SUBMIT_DIR

## Activate Python environment
source PATH_TO_MINIFORGE_INSTALLATION/miniforge/bin/activate
conda activate epos-ai-train

## Your usual invoke method e.g.
python train.py
```

To enable GPU access for the job please add to the top:

```
#SBATCH --gres=gpu:1
```

where the number :1 states the number of GPU devices, can be changed to higher numbers. To find which partitions provide access to GPU, please run:

```
sinfo -o '%P || %N || %G' | column -t
```

if in the last column is not null then, it indicates that the partition have access to GPU devices. Please setup the correct partition in the job script.

To find more about submitting jobs please see in corresponding documentation for Ares or Athena.

Running Jupyter notebooks on HPC cluster

To run an interactive session of Jupyter notebooks using jupterLab please follow the official PLGrid documentation.

The exemplar SLURM Jobs script can be as follows:

```
#!/bin/bash

## Set you JOB_NAME to to make it easier to see in the job queue
#SBATCH --job-name=JOB_NAME

## Max task execution time (format is HH:MM:SS)
#SBATCH --time=00:15:00

## Name of grant to which resource usage will be charged
#SBATCH --account=GRANT_ID

## Name of partition
#SBATCH --partition=plgrid

## Number of allocated nodes
#SBATCH --nodes=1

## Number of tasks per node (by default this corresponds to the number of cores allocated per node)
#SBATCH --cpus-per-task=1

## Number of GPUs
#SBATCH --gres=gpu:1


## get tunneling info
XDG_RUNTIME_DIR=""
ipnport=$(shuf -i8000-9999 -n1)
ipnip=$(hostname -i)
user=$USER

## print tunneling instructions to jupyter-log-{jobid}.txt
echo -e "
    Copy/Paste this in your local terminal to ssh tunnel with remote
    -----------------------------------------------------------------
    ssh -o ServerAliveInterval=300 -N -L $ipnport:$ipnip:$ipnport ${user}@ares.cyfronet.pl
    -----------------------------------------------------------------

    Then open a browser on your local machine to the following address
    -----------------------------------------------------------------
    localhost:$ipnport  (prefix w/ https:// if using password)
    -----------------------------------------------------------------

## Activate Python environment
source PATH_TO_MINIFORGE_INSTALLATION/miniforge/bin/activate
conda activate epos-ai-train

## Change to sbatch working directory
cd $SLURM_SUBMIT_DIR

jupyterlab --no-browser --port=$ipnport --ip=$ipnip
```

For accessing the started JupyterLab instance please follow the official guide.

Available Tools

- Seisbench - an open-source python toolbox for machine learning in seismology,
- Weight & Biases SDK - a library to track experiments, version and iterate on datasets, evaluate model performance, reproduce models, and manage your ML workflows end-to-end,
- ObsPy - an open-source project dedicated to provide a Python framework for processing seismological data,
- Jupyter Notebook and JupyterLab - a web services for interactive computing,
- IPython - a powerful interactive shell,
- PyTorch - an optimized tensor library for deep learning using GPUs and CPUs,
- CUDA - provides a development environment for creating high performance GPU-accelerated applications, a backend library for PyTorch

Back to top

Related Documents

- EPOS AI Workbench