

AI-based Data Assimilation --- Pre-class Setup

This document describes the preparation required for the hands-on exercises accompanying the **AI-based Data Assimilation** lecture. The practical session is designed to connect classical data assimilation (DA) concepts with a modern deep-learning-based DA framework.

The exercises will use two open-source repositories:

1. **Classical Data Assimilation / Ensemble Kalman Filter Tutorials**

Repository: <https://github.com/nansencenter/DA-tutorials>

2. **ADAF: An Artificial Intelligence Data Assimilation Framework for Weather Forecasting**

Repository: <https://github.com/microsoft/ADAF>

1. Learning Objectives

- Understand the basic workflow of a conventional DA experiment;
- Explore how observations modify a model background in ensemble-based DA;
- Identify the roles of the background, observations, and analysis;
- Understand the input/output structure of an AI-based DA system;
- Run or inspect an ADAF inference workflow;
- Examine how observation availability affects an AI-based analysis;
- Compare the conceptual workflow of classical DA and AI-based DA.

Part I --- Classical Data Assimilation

2. Repository

We will use the interactive tutorials developed by the Nansen Environmental and Remote Sensing Center:

<https://github.com/nansencenter/DA-tutorials>

The repository provides Jupyter-notebook-based tutorials on DA and the **Ensemble Kalman Filter (EnKF)**.

3. Recommended Local Environment

The current upstream instructions recommend **Python 3.12** for running the tutorials locally.

Option A --- Conda environment

```
conda create --name da-tutorial python=3.12
conda activate da-tutorial
```

Check the Python version:

```
python --version
```

Clone the repository:

```
git clone https://github.com/nansencenter/DA-tutorials.git
cd DA-tutorials
```

Install the dependencies provided by the repository:

```
pip install -r requirements.txt
```

Launch Jupyter:

```
jupyter-notebook
```

Then open the notebooks under:

```
DA-tutorials/notebooks/
```

Option B --- Cloud environment

The DA-tutorials repository also provides links for running the notebooks through cloud notebook services. This can be used as a backup

if local installation is inconvenient.

However, a local environment is recommended when possible because it generally provides a smoother classroom experience.

4. Pre-class Check for Part I

Before class, please verify that you can:

- ☐ Activate the `da-tutorial` environment;
- ☐ Start Jupyter Notebook;
- ☐ Open a notebook in `DA-tutorials/notebooks/` ;
- ☐ Run a simple Python/NumPy cell successfully;
- ☐ Execute several initial cells without import errors.

Part II --- AI-based Data Assimilation with ADAF

5. Repository

We will use:

<https://github.com/microsoft/ADAF>

ADAF is an AI-based DA framework designed to generate km-scale near-surface atmospheric analyses from multiple sources of information.

The released implementation uses gridded inputs derived from:

- Surface weather observations
- GOES-16 satellite imagery
- HRRR forecast background fields
- Topography

The analysis targets include four near-surface variables:

- **T2M** --- 2-m temperature;
- **Q2M** --- 2-m humidity;
- **U10** --- 10-m zonal wind;
- **V10** --- 10-m meridional wind.

For the classroom exercise, we will focus primarily on the **inference workflow and sensitivity to observations**, rather than full model training.

6. Hardware Recommendation

Minimum for code/data inspection

A normal laptop is sufficient for:

- reading the source code;
- inspecting configuration files;
- examining NetCDF samples;
- understanding the model inputs and outputs;
- plotting selected variables.

Recommended for model inference

A machine with a compatible **GPU environment** is recommended if you intend to execute ADAF inference during the class.

The public ADAF example uses GPU execution. Exact PyTorch/CUDA installation depends on your operating system, GPU, and driver configuration.

Important: Please verify your GPU/PyTorch installation before class. Do not wait until the hands-on session to configure GPU drivers.

Full ADAF training is **not required** for the classroom exercise.

7. Suggested ADAF Environment

Because deep-learning dependencies can depend on the available hardware, we recommend creating a **separate environment** for ADAF rather than installing it into the classical DA environment.

For example:

```
conda create --name adaf python=3.10
conda activate adaf
```

Then clone the repository:

```
git clone https://github.com/microsoft/ADAF.git
cd ADAF
```

Inspect the repository structure and install the Python packages required by your local setup.

At minimum, please ensure that your environment can import the major packages used by the ADAF code and that PyTorch can detect your

intended
compute device.

A useful check is:

```
python -c "import torch; print(torch.__version__);  
print(torch.cuda.is_available())"
```

The Python 3.10 environment above is a **course recommendation for environment isolation**, not an upstream ADAF version requirement. If you already have a working environment for the repository, you may use it.

8. ADAF Data and Model Files

Please download the required files **before class**.

The ADAF repository provides links to pre-processed data, normalization statistics, and trained model weights through its README.

For inference, the upstream documentation identifies the following required components:

```
ADAF/  
├─ config/  
│   └─ experiment.yaml  
├─ data/  
│   ├── test/  
│   │   ├── YYYY-MM-DD_HH.nc  
│   │   └─ ...  
│   └─ stats.csv
```

```
|— model_weights/  
|   |— <trained model checkpoint>  
|— inference.py
```

You will therefore need:

- ☐ The ADAF source code;
- ☐ At least the test sample(s) specified for the class;
- ☐ Pre-computed normalization statistics;
- ☐ The trained model weights;
- ☐ The experiment configuration file.

The exact checkpoint filename may differ between the repository description and the downloaded archive. Use the checkpoint supplied with the course material or the current ADAF release.

9. Important Note on Normalization Statistics

If you use the provided pre-trained model, you must also use the corresponding **pre-computed normalization statistics**.

The model parameters and normalization statistics were used together during training. Replacing the statistics with independently calculated values may lead to incorrect inference results.

10. Pre-class Check for Part II

Before class, please verify the following.

Repository

```
cd ADAF
```

Confirm that you can see files/directories such as:

```
config/  
data/  
models/  
utils/  
inference.py  
train.py
```

Python environment

Run:

```
python -c "import torch; print(torch.__version__)"
```

If you plan to use a GPU:

```
python -c "import torch; print(torch.cuda.is_available())"
```

Data

Confirm that the required test NetCDF file(s) can be found under your chosen test-data directory.

Model

Confirm that the trained model checkpoint has been downloaded.

Configuration

Open:

```
config/experiment.yaml
```

and familiarize yourself with the paths used for data and experiment output.

11. What We Will Do in Class

The hands-on session will contain two connected exercises.

Exercise 1 --- Classical DA: From Background to Analysis

Using the DA tutorials, we will explore a simplified assimilation problem and examine:

```
Background
  +
Observations
  ↓
Data Assimilation
  ↓
Analysis
```

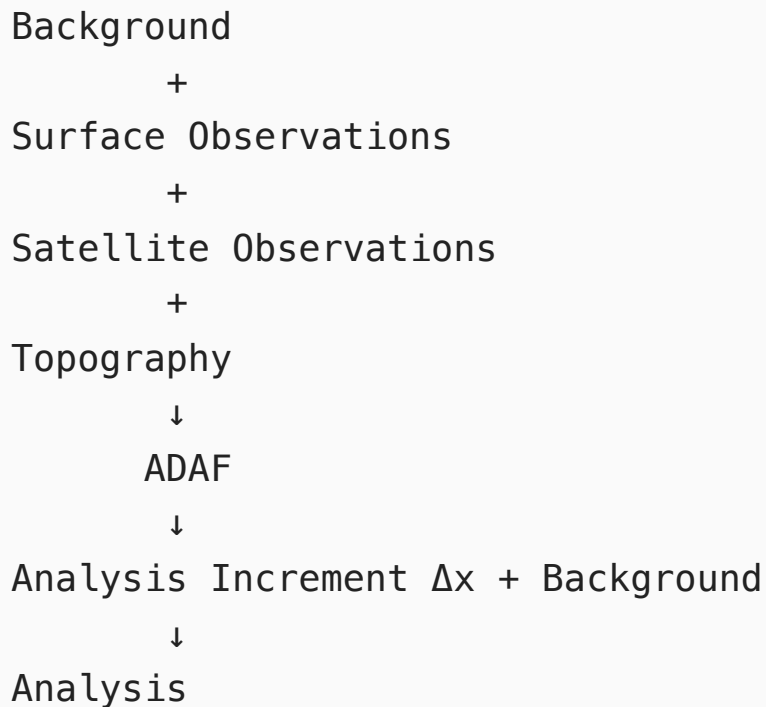
We will discuss questions such as:

- How does an observation modify the background?
- How is observational information propagated to unobserved locations?
- What controls the magnitude of the analysis increment?
- How does observation uncertainty affect the analysis?
- What role does the ensemble play in an EnKF?

The emphasis will be on the **mechanism of information propagation in classical DA**.

Exercise 2 --- AI-based DA: ADAF Inference

We will then examine the corresponding workflow in ADAF:



Depending on available computing resources, participants will either run the provided inference example or work with pre-computed outputs.

We will examine:

- the structure of an ADAF input sample;
- background and observational inputs;
- normalization and preprocessing;
- the model configuration;
- the inference pipeline;
- analysis increments;
- the resulting atmospheric analysis.

12. Observation Sensitivity Experiment

One particularly useful classroom experiment is to change the fraction of surface observations provided to ADAF.

The inference script exposes a parameter similar to:

```
--hold_out_obs_ratio
```

We will use this experiment to ask:

What happens to the analysis when less observational information is available?

For example, we may compare several observation configurations and examine changes in analysis quality.

This experiment connects directly to an important DA question:

How does the value of observational information change when the observing network becomes sparse?

13. Preparation Checklist

Please complete this checklist before attending the practical session.

General

- ☐ Conda/Miniconda/Anaconda or another Python environment manager is available.
- ☐ Jupyter Notebook can be launched.
- ☐ You are comfortable running basic Python commands.

Classical DA

- ☐ Clone `nansencenter/DA-tutorials` .
- ☐ Create the Python 3.12 environment.
- ☐ Install `requirements.txt` .
- ☐ Open at least one tutorial notebook.
- ☐ Run several cells successfully.

ADAF

- ☐ Clone `microsoft/ADAF` .
- ☐ Create a separate Python environment.
- ☐ Verify that PyTorch imports successfully.
- ☐ If applicable, verify GPU availability.
- ☐ Download the assigned ADAF test data.
- ☐ Download the normalization statistics.
- ☐ Download the trained model checkpoint.
- ☐ Confirm that `config/experiment.yaml` is accessible.
- ☐ Confirm that the required data and model paths are known.

14. Recommended Preparation Strategy

To avoid installation problems during class, please prepare in the following order:

1. Clone both repositories
↓
2. Create separate Python environments
↓
3. Test the classical DA notebooks
↓
4. Download ADAF data and model files
↓
5. Verify PyTorch / GPU environment

↓

6. Inspect one ADAF NetCDF sample

↓

7. Keep both environments ready for class

Please do not spend significant time attempting to train ADAF before the class. The practical session is designed around **understanding the DA workflow, running/inspecting inference, and interpreting the results.**