Anemoi Training

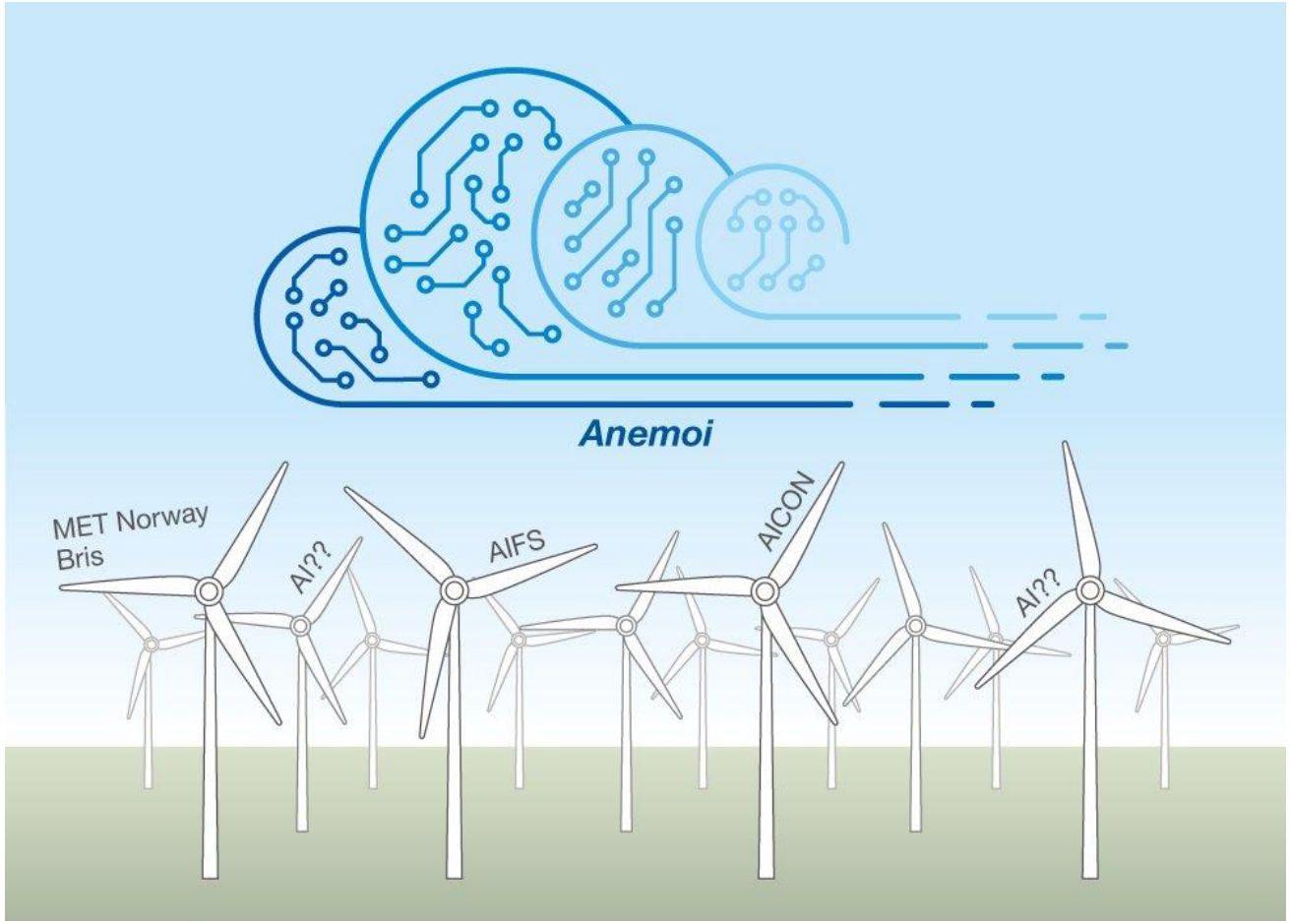
Jesper Dramsch

Scientist for Machine Learning

Jesper.Dramsch@ecmwf.int

Anemoi framework

- Develop machine learning (ML) weather forecasting models.
  - Dataset preparation tools.
  - Graph creation tools.
  - ML model training support.
  - Inference tools integrated with verification software.
  - Registry for datasets and trained models.
- **Benefits:**
  - Simplifies shared meteorological challenges.
  - Enables training of models using existing recipes and custom data.



Anemoi framework

- Develop machine learning (ML) weather forecasting models.
  - Dataset preparation tools.
  - Graph creation tools.
  - ML model training support.
  - Inference tools integrated with verification software.
  - Registry for datasets and trained models.
- **Benefits:**
  - Simplifies shared meteorological challenges.
  - Enables training of models using existing recipes and custom data.

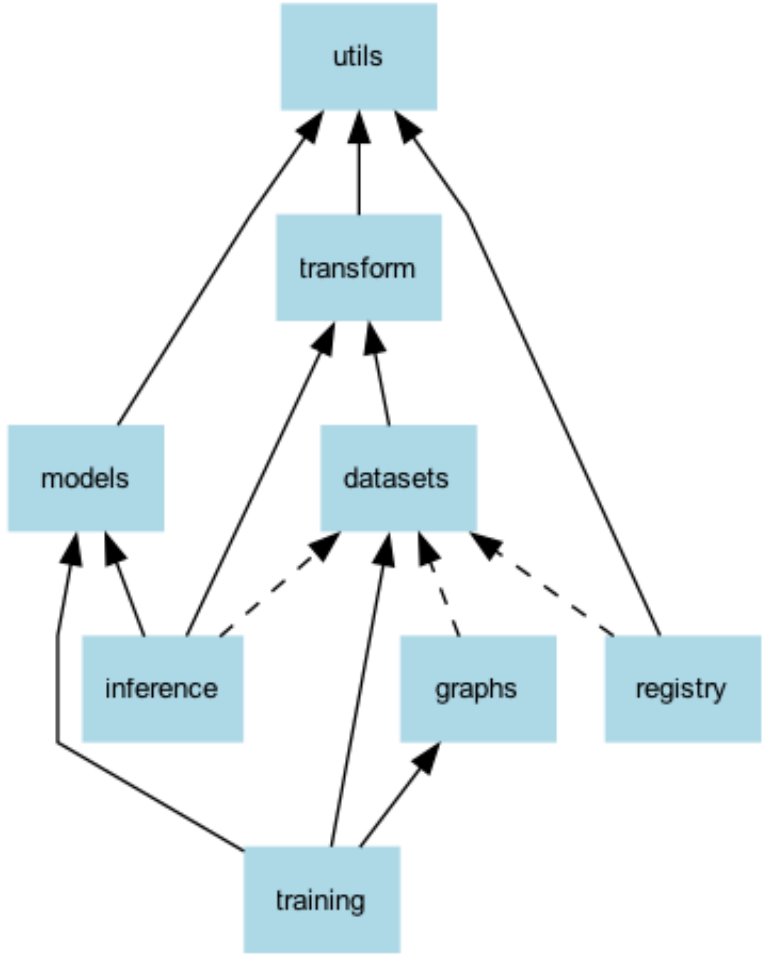
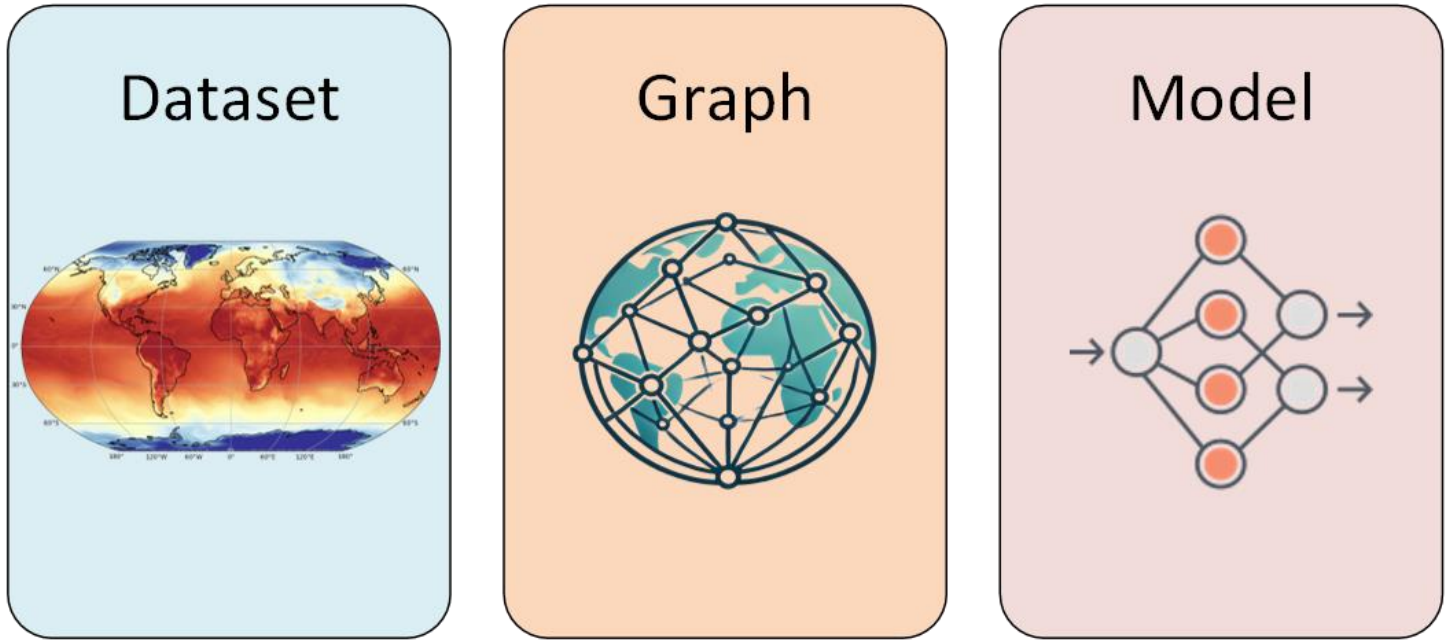


Diagram of dependencies within anemoi packages.

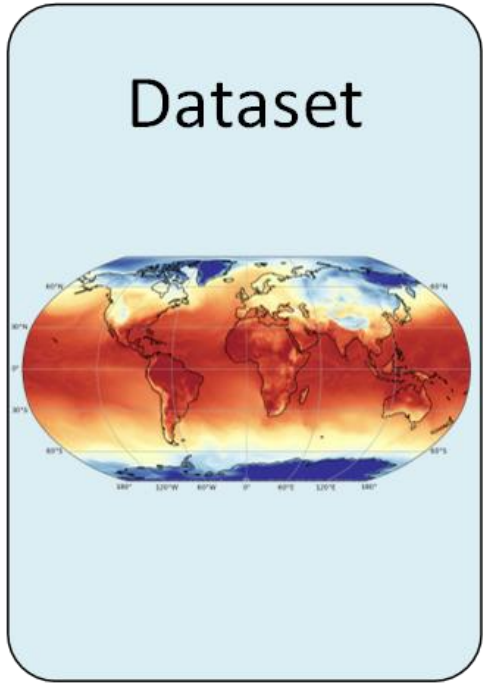
Getting started

To train a data-driven weather forecasting model in **Anemoi**, three main components are needed:

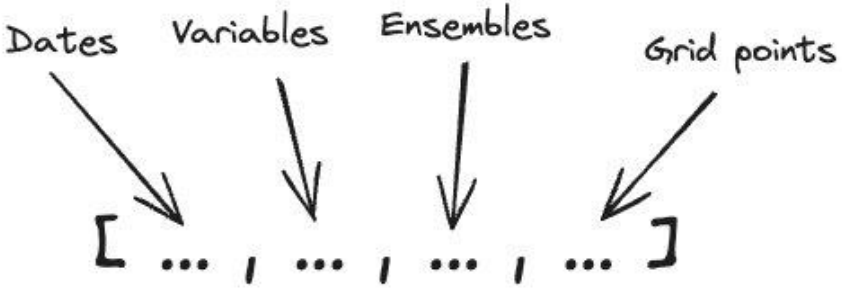


Zarr PyG PyTorch

anemoi-datasets

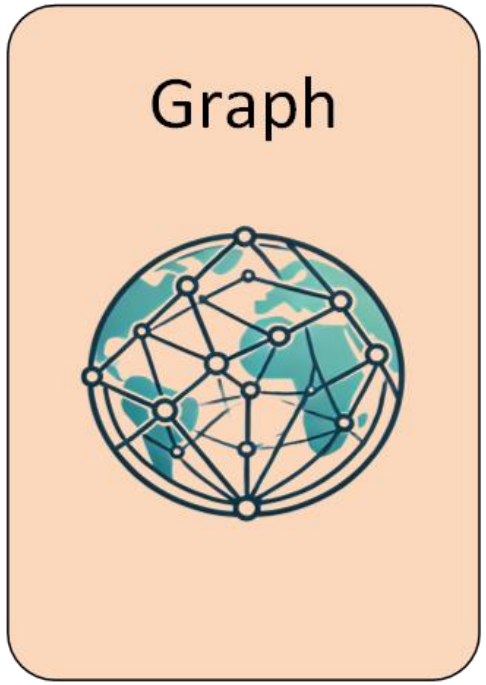


anemoi-datasets.readthedocs.io



- Create "machine-learning ready" datasets for training data-driven weather forecasts.
- Make the loading of data samples as efficient as possible
  - I/O operations are minimised
- Zarr
  - Offers an array-like view on chunks
  - Each file is a single date
- Using datasets
  - Subsetting datasets (time, variables, member, …)
  - Combining datasets (join, concat, cutout, ensemble, ...)

anemoi-graphs



anemoi-graphs.readthedocs.io

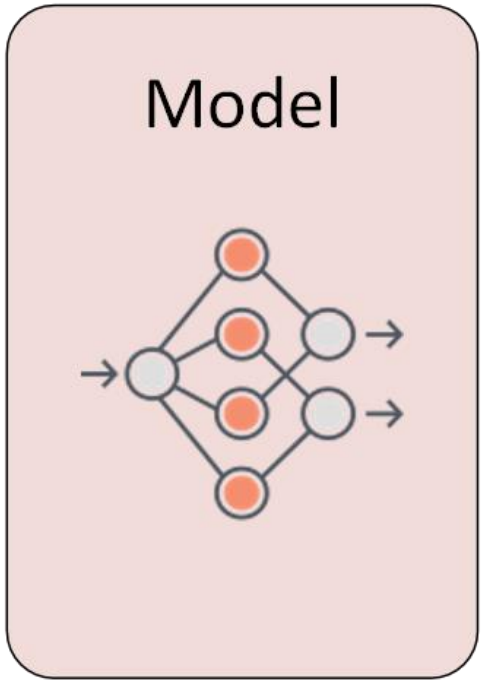
In Anemoi, it is represented by a *torch_geometric.data.**HeteroData*** object, and stored in a .PT file.

- A graph is a structure composed of:
  - **Nodes**: Represent locations in space.
  - **Edges**: Define how information flows between nodes.

```
HeteroData(
  data={
    x=[40320, 2],  # coordinates in radians (lat in [-pi/2, pi/2], lon in [0, 2pi])
    node_type='ZarrDatasetNodes',
    area_weight=[40320, 1],
  },
  hidden={
    x=[10242, 2],  # coordinates in radians (lat in [-pi/2, pi/2], lon in [0, 2pi])
    node_type='TriNodes',
    area_weight=[10242, 1],
  },
  (data, to, hidden)={
    edge_index=[2, 62980],
    edge_type='CutOffEdges',
    edge_length=[62980, 1],
    edge_dirs=[62980, 2],
  },
  (hidden, to, hidden)={
    edge_index=[2, 81900],
    edge_type='MultiScaleEdges',
    edge_length=[81900, 1],
    edge_dirs=[81900, 2],
  },
  (hidden, to, data)={
    edge_index=[2, 120960],
    edge_type='KNNEdges',
    edge_length=[120960, 1],
    edge_dirs=[120960, 2],
  }
)
```

Console log of a graph created with anemoi-graphs.

anemoi-models



anemoi-models.readthedocs.io

- Graph Neural Networks (GNNs) are a type of neural network designed to operate on **graph-structured** data.
  - GNN (G) = G', where G, G' are graphs
- Node features are update based on *message passing*:
  - Message creation (from node and edge features)
  - Message aggregation
  - Node update
- Permutation-invariance

anemoi-training

- Code to train models, using torch-lightning and Hydra
  - Multi-node/multi-GPU training support
  - Deterministic training with probabilistic training coming soon…
  - Callbacks for profiling evaluating, plotting and logging intermediate results
  - Implement various losses, more can be easily added
  - Interfaces with trackers such as mlflow

- Highly configurable

- Interfaces with:
  - anemoi-dataset via data loaders
  - anemoi-models via Hydra configuration
  - anemoi-inference via metadata-rich checkpoints

anemoi-training (cont.)

Model / GNN.yml

```yaml
activation: GELU
num_channels: 512

model:
  _target_:
anemoi.models.models.encoder_processor_decoder
.AnemoiModelEncProcDec

processor:
  _target_:
anemoi.models.layers.processor.GNNProcessor
  _convert_: all
  activation: ${model.activation}
  trainable_size:
${model.trainable_parameters.hidden2hidden}
  sub_graph_edge_attributes:
${model.attributes.edges}
  num_layers: 16
  num_chunks: 2
  mlp_extra_layers: 0

encoder:
  _target_:
anemoi.models.layers.mapper.GNNForwardMapper
```

```
anemoi-training train model=gnn
```

Model / GraphTransformer.yml

```yaml
activation: GELU
num_channels: 1024

model:
  _target_:
anemoi.models.models.encoder_processor_decoder.Ane
moiModelEncProcDec

processor:
  _target_:
anemoi.models.layers.processor.GraphTransformerPro
cessor
  _convert_: all
  activation: ${model.activation}
  trainable_size:
${model.trainable_parameters.hidden2hidden}
  sub_graph_edge_attributes:
${model.attributes.edges}
  num_layers: 16
  num_chunks: 2
  mlp_hidden_ratio: 4 # GraphTransformer
  num_heads: 16 # GraphTransformer

encoder:
```

```
anemoi-training train model=graphtransformer
```

Model / Transformer.yml

```yaml
activation: GELU
num_channels: 1024

model:
  _target_:
anemoi.models.models.encoder_processor_decoder.AnemoiModelEn
cProcDec

processor:
  _target_:
anemoi.models.layers.processor.TransformerProcessor
  _convert_: all
  activation: ${model.activation}
  num_layers: 16
  num_chunks: 2
  mlp_hidden_ratio: 4 # Transformer only
  num_heads: 16 # Transformer only
  window_size: 512
  dropout_p: 0.0

encoder:
 _target_: anemoi.models.layers.mapper.GraphTransformerFor
```

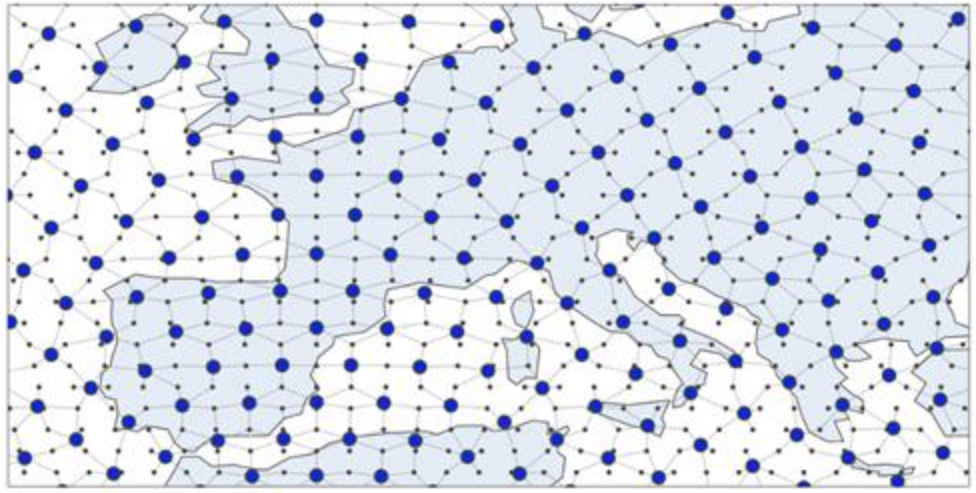
```
anemoi-training train model=transformer
```

- Make it easy to switch components
- Allow for reproduceable training
- Easy to extend with new models and components

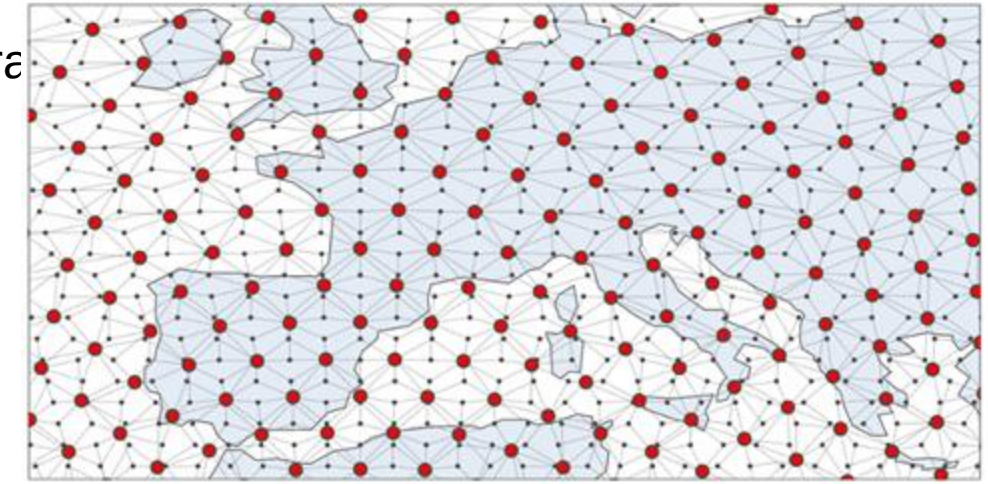
anemoi-training

Example: current AIFS configuration

Graph Encoder

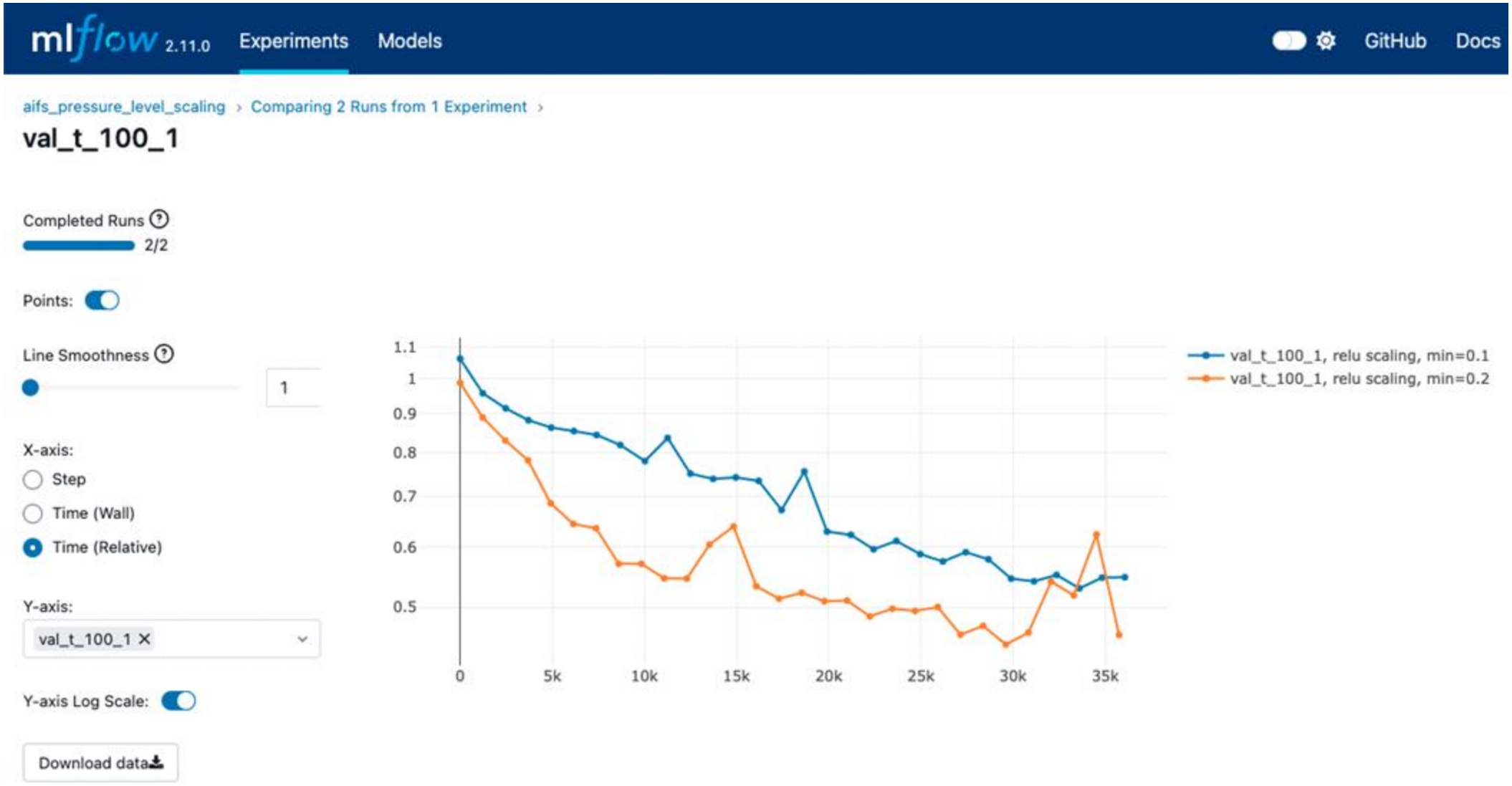


Gra



Transformer processor

Anemoi.training – MLFlow servers for Experiment Tracking



Demo

Collaboration, without aggregation around a single model