Tutorial 6:
Phase picking and earthquake catalogues

OBS training workshop, VUW, April 14-16, 2025

https://github.com/seisbench/seisbench

SeisBench

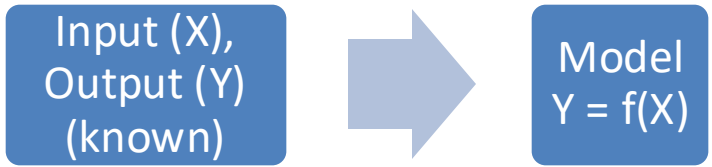
A toolbox for machine learning in seismology

A platform for the application of Deel Learning models in seismicity studies:

1. Develop DL models of seismic waveforms from human-generated picks
2. Apply DL models to annotate seismic waveforms for seismic phases and detections
3. Provides complete workflows to build earthquake catalogues from DL models

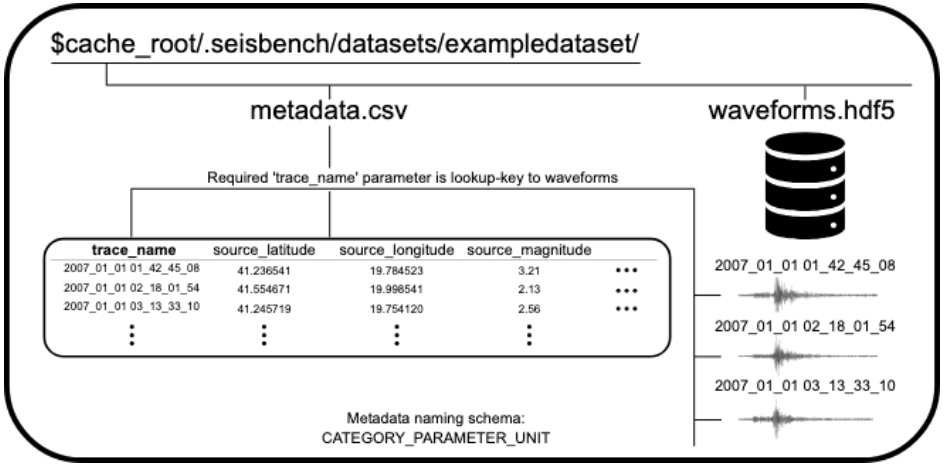
SeisBench platform

- In machine (deep) learning, a « model » is a function that maps inputs to predictions (outputs).
- For phase-picking tasks, inputs (X) are seismic waveforms, and output/predictions(Y) are seismic phase picks (and/or detections).
- The process of « learning » how the outputs are related to inputs is called « training ».



SeisBench datasets

- To train DL models, SeisBench uses specific input data for training:
  - Earthquake catalogue with seismic phase picks (.csv file)
  - Seismic waveforms for the corresponding earthquakes and picks



- Here we won't be concerned with creating datasets – in case it's useful, check here: https://seisbench.readthedocs.io/en/stable/pages/data_format.html

SeisBench datasets

- Several datasets already exist and can be used to train new DL models (e.g., new model architecture)
- A list can be found here: https://seisbench.readthedocs.io/en/stable/pages/benchmark_datasets.html
- When « applying » a DL model to new data you can select models that have been trained on any one of those datasets.
  - Performance will vary hugely due to specific source-station coverage and geometry, unique noise conditions, limitations of data set, etc.
  - For OBS data, this was traditionally a problem. There are now models trained on OBS datasets for picking/detecting.
- You can also create a new dataset and extend the picking capability of currently existing models. This is called « transfer learning ».

SeisBench predictions

- In SeisBench, we can either « annotate » or « classify » seismic waveforms.
- The « annotate » function takes an obspy stream object as input and returns annotations as stream again. For example, for picking models the output would be the characteristic functions, i.e., the pick probabilities over time.

```
stream = obspy.read("my_waveforms.mseed")
annotations = model.annotate(stream)  # Returns obspy stream object with annotations
```

- The « classify » function also takes an obspy stream as input, but in contrast to the annotate function returns discrete results. The structure of these results might be model dependent.
- For example, a pure picking model will return a list of picks, while a picking and detection model might return a list of picks and a list of detections.

```
stream = obspy.read("my_waveforms.mseed")
outputs = model.classify(stream)  # Returns a list of picks
print(outputs)
```

SeisBench models

- SeisBench offers a range of pretrained model weights through a common interface. Model weights are downloaded on the first use and cached locally afterwards.
- For some models, multiple versions are available (i.e., trained on different datasets).
- For details on accessing these, check the documentation: https://seisbench.readthedocs.io/en/stable/pages/models.html#models-integrated-into-seisbench.

```
import seisbench.models as sbm
sbm.PhaseNet.list_pretrained()                  # Get available models
model = sbm.PhaseNet.from_pretrained("geofon")  # Load the model trained on GEOFON
```

From picks to catalogues

- Once phase picks have been obtained, several external python packages can use the output from SeisBench (or slightly reformatted) and turn them into preliminary earthquake source locations to build a catalogue.
- Examples include GaMMA, PyOcto, REAL.
- Here, we will examine simple workflows to annotate seismic waveforms and build preliminary catalogues using the HOBITSS data.

GaMMA: tuning

Hyperparameters:

- `use_amplitude` (default = True): If using amplitude information.
- `vel` (default = {"p": 6.0, "s": 6.0 / 1.75}): velocity for P and S phases. Homogeneous velocity model
- `use_dbscan`: If using dbscan to cut a long sequence of picks into segments. Using DBSCAN can significantly speed up association using small windows.
- `dbscan_eps` (default = 10.0s): The maximum time between two picks for one to be considered as a neighbor of the other. See details in DBSCAN
- `dbscan_min_samples` (default = 3): The number of samples in a neighborhood for a point to be considered as a core point. See details in DBSCAN
- `oversampling_factor` (default = 10): The initial number of clusters is determined by (Number of picks)/(Number of stations)/(Inital points) * (oversampling factor). Clustering
- `initial_points` (default=[1,1,1] for (x, y, z) directions): Initial earthquake locations (cluster centers). For a large area over 10 degrees, more initial points are helpful, such as [2,2,1].
- `covariance_prior` (default = (5, 5)): covariance prior of time and amplitude residuals. Because current code only uses an uniform velocity model, a large covariance prior can be used to avoid splitting one event into multiple events.
- Filtering low quality association
  - `min_picks_per_eq`: Minimum picks for associated earthquakes. We can also specify minimum P or S picks:
  - `min_p_picks_per_eq`: Minimum P-picks for associated earthquakes.
  - `min_s_picks_per_eq`: Minimum S-picks for associated earthquakes.
  - `max_sigma11`: Max phase time residual (s)
  - `max_sigma22`: Max phase amplitude residual (in log scale)
  - `max_sigma12`: Max covariance term. (Usually not used)

Filtering

GaMMA: geographic configuration

- GaMMA works in cartesian coordinates: we need a projection that will convert geographic coordinates to cartesian and vice versa
- We need to use the package pyproj to help out with this
  - Challenge: find the coordinate system (EPSG) for your area: https://epsg.io/
- You will then need to specify the box coordinates (in km) to search for earthquake locations.

PyOcto: Configuration

- Parameters: https://pyocto.readthedocs.io/en/latest/pages/associator.html
- See also: https://pyocto.readthedocs.io/en/latest/pages/parameters.html#parameters
- Noteworthy: PyOcto can use homogeneous or 1D seismic velocity models!
- Simplified handling of geographic configuration: `pyocto.OctoAssociator.from_area()` (https://pyocto.readthedocs.io/en/latest/pages/associator.html#pyocto.associator.OctoAssociator.from_area)