EQTransformer Documentation

Release 2020

S. Mostafa Mousavi

Aug 04, 2023

Contents

1 Github development page: **3**

2 Contents **5**

2.1 Overview 5
2.2 Installation 7
2.3 Tutorial 13
2.4 Copyright and License 22
2.5 Contributing 22
2.6 References 23
2.7 Indices and tables 23

EQTransformer is an AI-based earthquake signal detector and phase (P&S) picker based on a deep neural network with an attention mechanism. It has a hierarchical architecture specifically designed for earthquake signals. **EQTransformer** has been trained on global seismic data and can perform detection and arrival time picking simultaneously. In addition to the prediction probabilities, it can also provides model uncertainties.

The `EQTransformer` python 3 package includes modules for downloading continuous seismic data, preprocessing, performing earthquake signal detection, and phase (P & S) picking using pre-trained models, building and testing new models, and performing a simple phase association.

The following is the main reference of **EQTransformer**:

- Mousavi, S.M., Ellsworth, W.L., Zhu, W., Chuang, L.Y., Beroza, G.C., "Earthquake Transformer: An Attentive Deep-learning Model for Simultaneous Earthquake Detection and Phase Picking ". Nature Communications, (2020).

CHAPTER 1

Github development page:

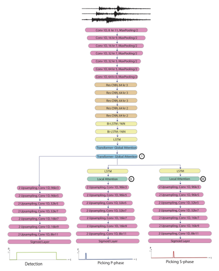
https://github.com/smousavi05/EQTransformer

CHAPTER 2

Contents

2.1 Overview

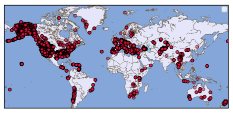
EQTransformer is a multi-task deep neural network for simultaneous earthquake detection and phase picking with a hierarchical attentive model. It mainly consists of one very deep encoder and three separate decoders (detector, P-picker, and S-picker branches) with an attention mechanism. Attention mechanisms in Neural Networks are inspired by human visual attention. Humans focus on a certain region of an image with high resolution while perceiving the surrounding image at low resolution and then adjusting the focal point over time. Our model emulates this through two levels of attention mechanism in a hierarchical structure. one at the global level for identifying an earthquake signal in the input time series, and one at the local level for identifying different seismic phases within that earthquake signal. Two levels of self-attention (global and local) help the neural network capture and exploit dependencies between local (individual phases) and global (full-waveform) features within an earthquake signal. This model has several distinctive characteristics: 1) it is the first hierarchical-attentive model specifically designed for earthquake signal; 2) with 56 activation layers, it is the deepest network that has been trained for seismic signal processing; 3) it has a multi-task architecture that simultaneously performs the detection and phase picking - using separate loss functions - while modeling the dependency of these tasks on each other through a hierarchical structure; 4) in addition to the prediction probabilities, it provides output variations based on Bayesian inference; 5) it is the first model trained using a globally distributed training set of 1.3 M local earthquake observations; 6) it consists of both convolutional and recurrent neurons. Read our paper for more details.



Architecture of **EQTransformer**

2.1.1 Dataset

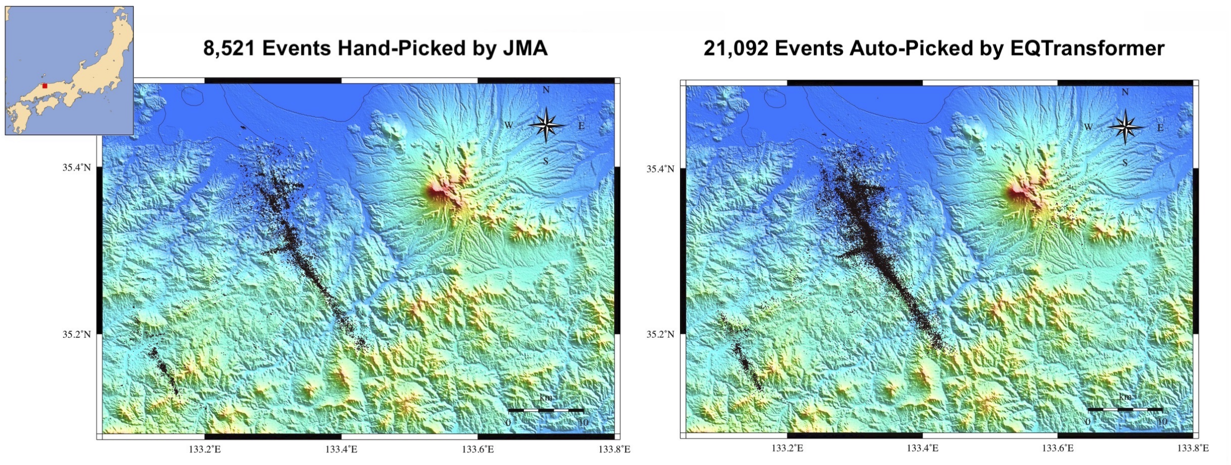
STanford EArthquake Dataset (STEAD) is used to train the **EQTransformer**. STEAD is a large-scale global dataset of labeled earthquake and non-earthquake signals. Here we used 1 M earthquake and 300 K noise waveforms (including both ambient and cultural noise) recorded by ~ 2600 seismic stations at epicentral distances up to 300 km. Earthquake waveforms are associated with about 450 K earthquakes with a diverse geographical distribution around the world. The majority of these earthquakes are smaller than M 2.5 and have been recorded within 100 km from the epicenter. A full description of properties of the dataset can be found in STEAD. Waveforms are 1 minute long with a sampling rate of 100 Hz and are causally band-passed filtered from 1.0-45.0 Hz.



STEAD contains earthquake signals from most of the seismically active countries with a few exceptions like Japan.

2.1.2 Application to Japan

However, **EQTransformer** has a high generalization ability. Applying it to 5 weeks of continuous data recorded during the 2000 Mw 6.6 western Tottori, Japan earthquake, two times more events were detected compared with the catalog of Japan Meteorological Agency (JMA).


8,521 Events Hand-Picked by JMA
21,092 Events Auto-Picked by EQTransformer
35.4°N
35.2°N
133.2°E
133.4°E
133.6°E
133.8°E
N
S
W
E

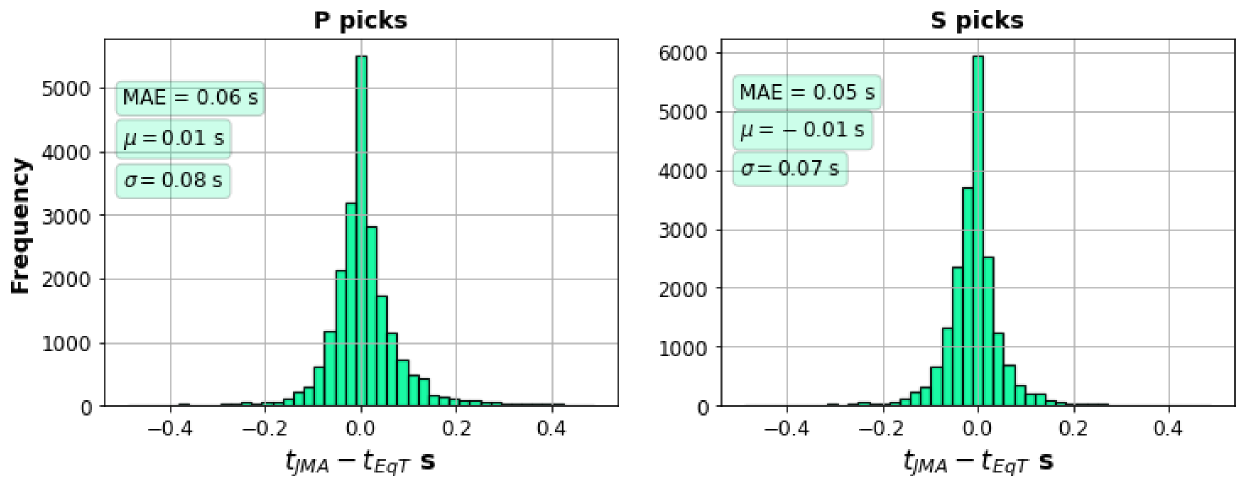
In total, JMA's analysts picked 279,104 P and S arrival times on 57 stations, while EQTransformer was able to pick 401,566 P and S arrival-time on 18 of those stations (due to unavailability of data for other stations). To compare the manual picks by JMA with our automatic picks we used about 42,000 picks on the common stations and calculated the arrival time differences. The distributions of these arrival time differences between the manual and deep-learning picks for P and S waves are shown in the following figure. The standard deviation of differences between picks is around 0.08 second with a mean absolute error of around 0.06 second or 6 samples. The mean error is only 1 sample (0.01 s).

2.1.3 Application to Other Regions

Test set data from STEAD:

Ridgecrest, California:

Tottori, Japan:



West Texas, USA:

Variations in the output probability predictions (model uncertainty) can be useful to identify false-positive events (like the one shown in the above figure).

2.1.4 Comparison with Other Methods

Below are the picking errors for P and S waves. All methods have been applied to the same benchmark test set from STEAD.

1) Comparing with some deep-learning pickers:

PhaseNet, GPD, Yews, PpkNet, pickNet

1) Comparing with some traditional pickers:

Kurtosis, FilterPicker, AIC

2.2 Installation

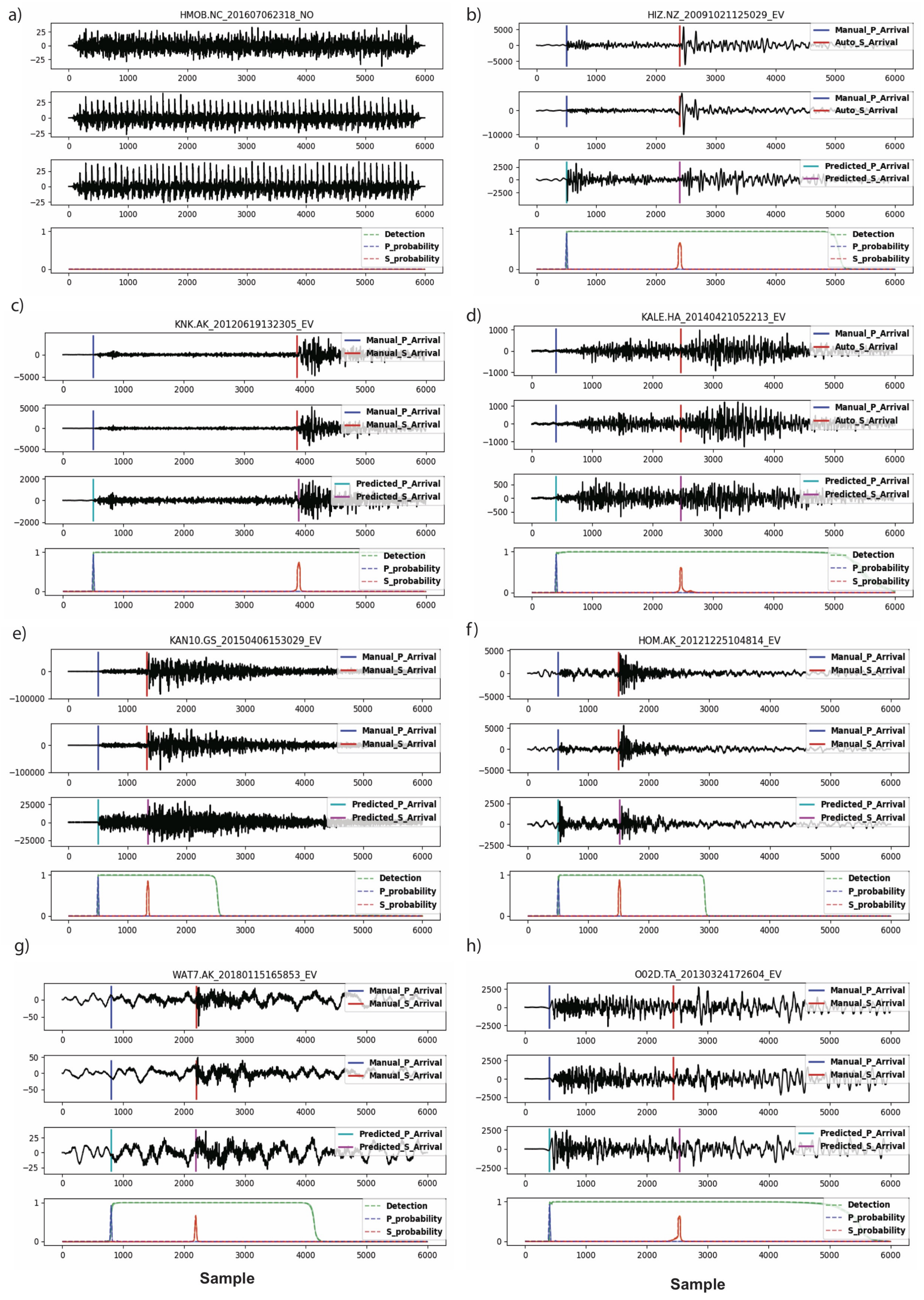
EQTransformer is a Python 3.x package that uses libraries from Tensorflow and Obspy.

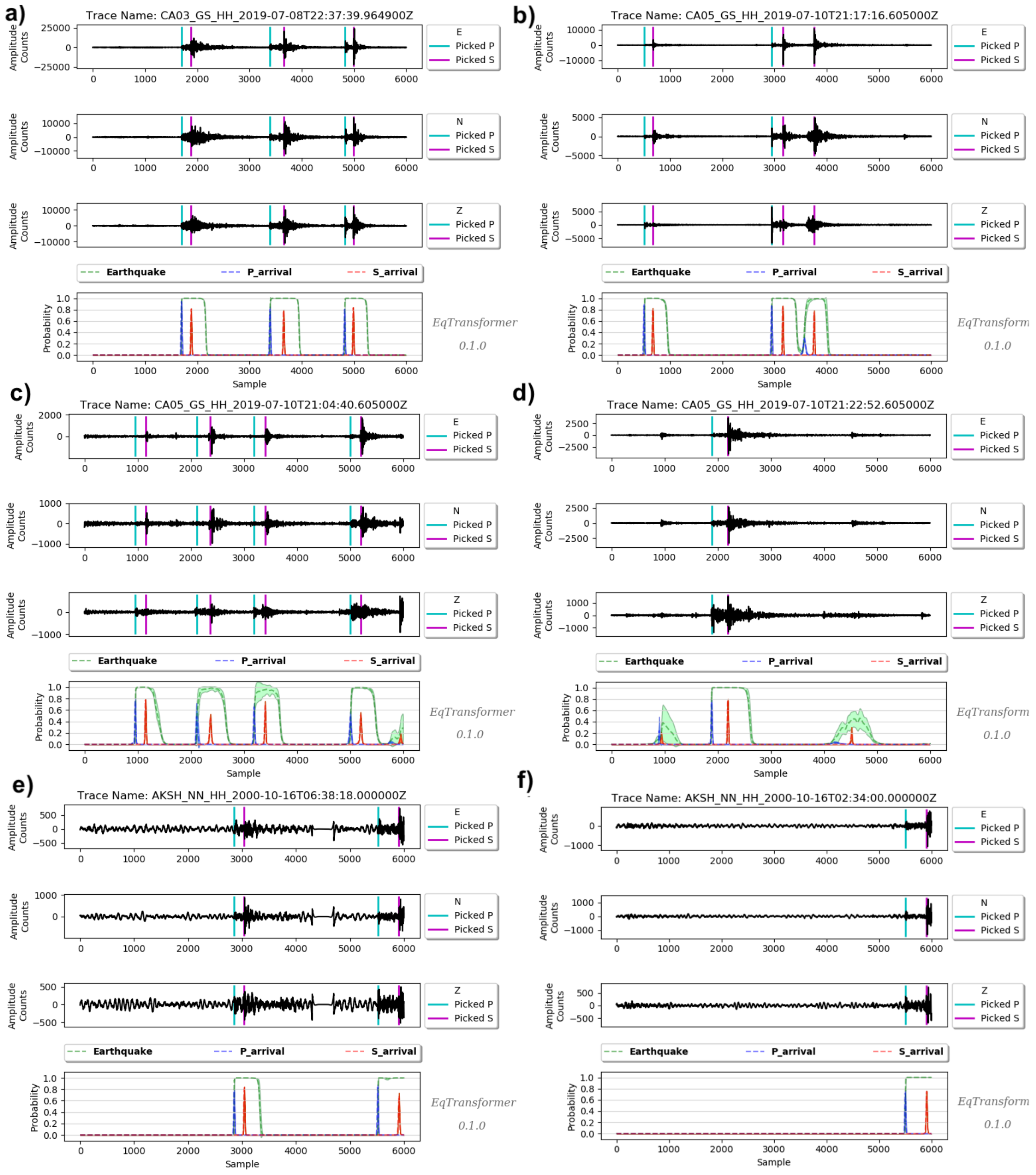
2.2.1 Installation via conda (recommended)

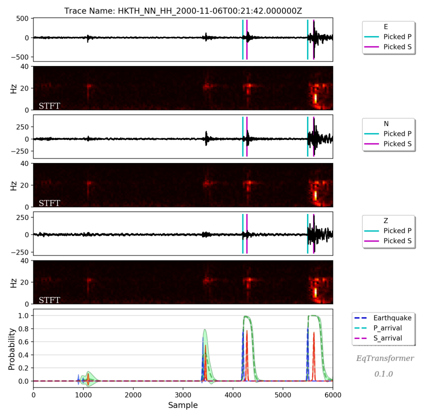
The following will download and install **EQTransformer** that supports a variety of platforms, including Windows, macOS, and Linux operating systems. Note that you will need to have Python 3.x (3.6 or 3.7) installed.

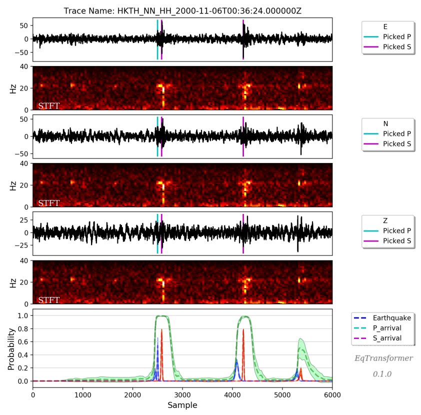
It is recommended that you use a Python virtual environment (e.g., conda) to test the **EQTransformer** package. Please follow the to install conda if you do not have either a miniconda or anaconda installed on your machine. Once you have conda installed, you can use Terminal or an Anaconda Prompt to create a Python virtual environment. Check managing for more information.

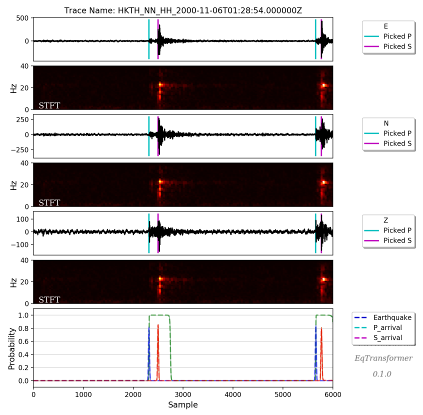
```
conda create -n eqt python=3.7

Conda activate eqt

conda install -c smousavi05 eqtransformer
```

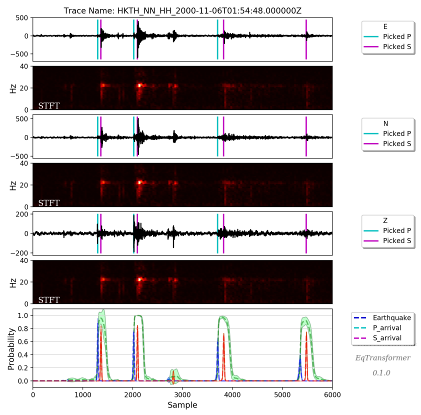

a)
HMOB.NC_201607062318_NO
b)
HIZ.NZ_20091021125029_EV
c)
KNK.AK_20120619132305_EV
d)
KALE.HA_20140421052213_EV
e)
KAN10.GS_20150406153029_EV
f)
HOM.AK_20121225104814_EV
g)
WAT7.AK_20180115165853_EV
h)
O02D.TA_20130324172604_EV
Manual_P_Arrival
Manual_S_Arrival
Auto_S_Arrival
Predicted_P_Arrival
Predicted_S_Arrival
Detection
P_probability
S_probability
Sample

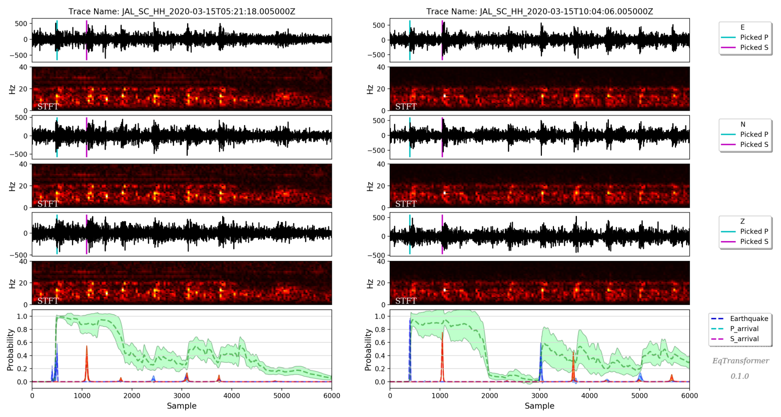

a)
Trace Name: CA03_GS_HH_2019-07-08T22:37:39.964900Z
b)
Trace Name: CA05_GS_HH_2019-07-10T21:17:16.605000Z
c)
Trace Name: CA05_GS_HH_2019-07-10T21:04:40.605000Z
d)
Trace Name: CA05_GS_HH_2019-07-10T21:22:52.605000Z
e)
Trace Name: AKSH_NN_HH_2000-10-16T06:38:18.000000Z
f)
Trace Name: AKSH_NN_HH_2000-10-16T02:34:00.000000Z
Amplitude Counts
E
N
Z
Picked P
Picked S
Earthquake
P_arrival
S_arrival
Probability
Sample
EqTransformer
0.1.0

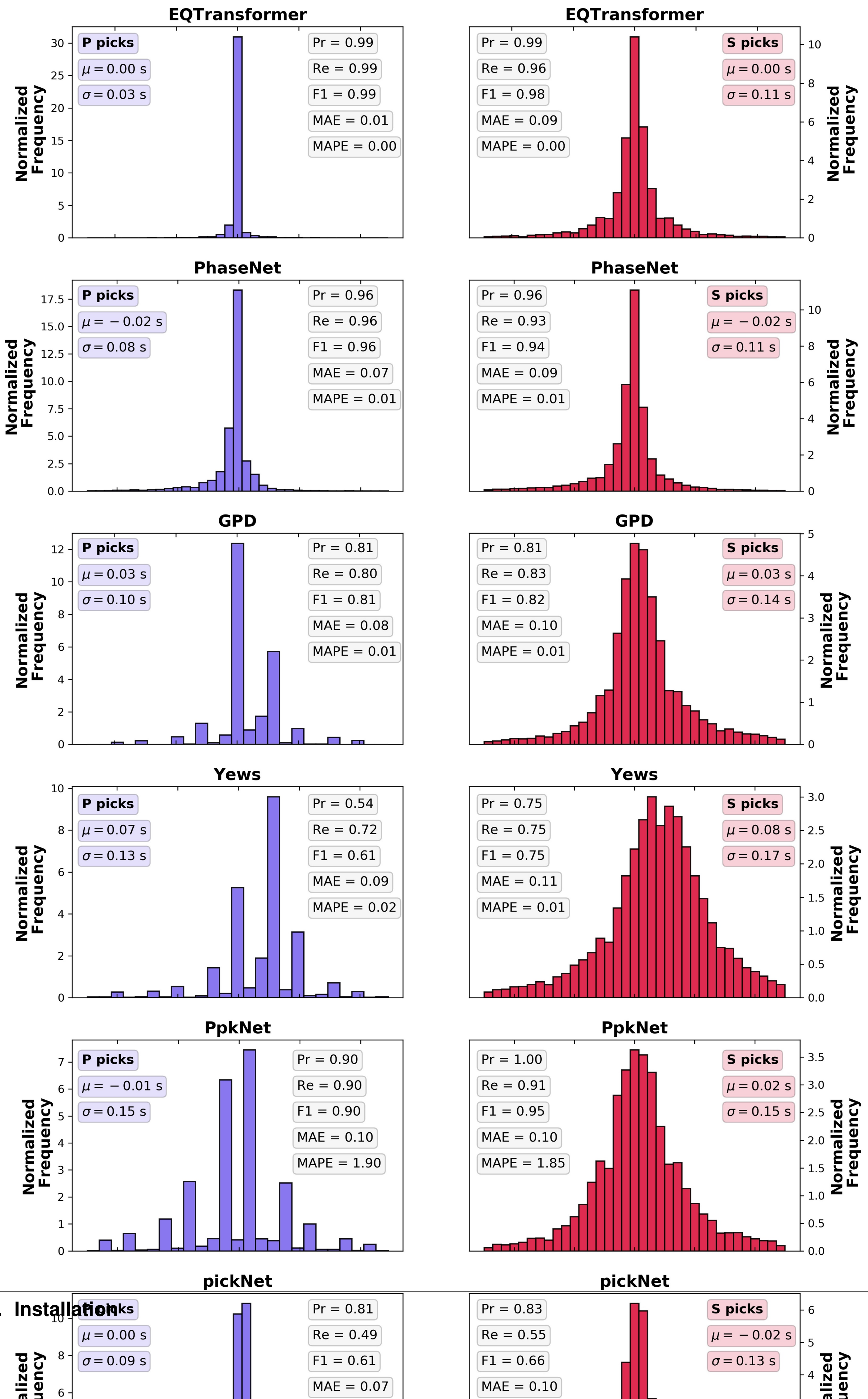

Trace Name: HKTH_NN_HH_2000-11-06T00:21:42.000000Z
EqTransformer
0.1.0

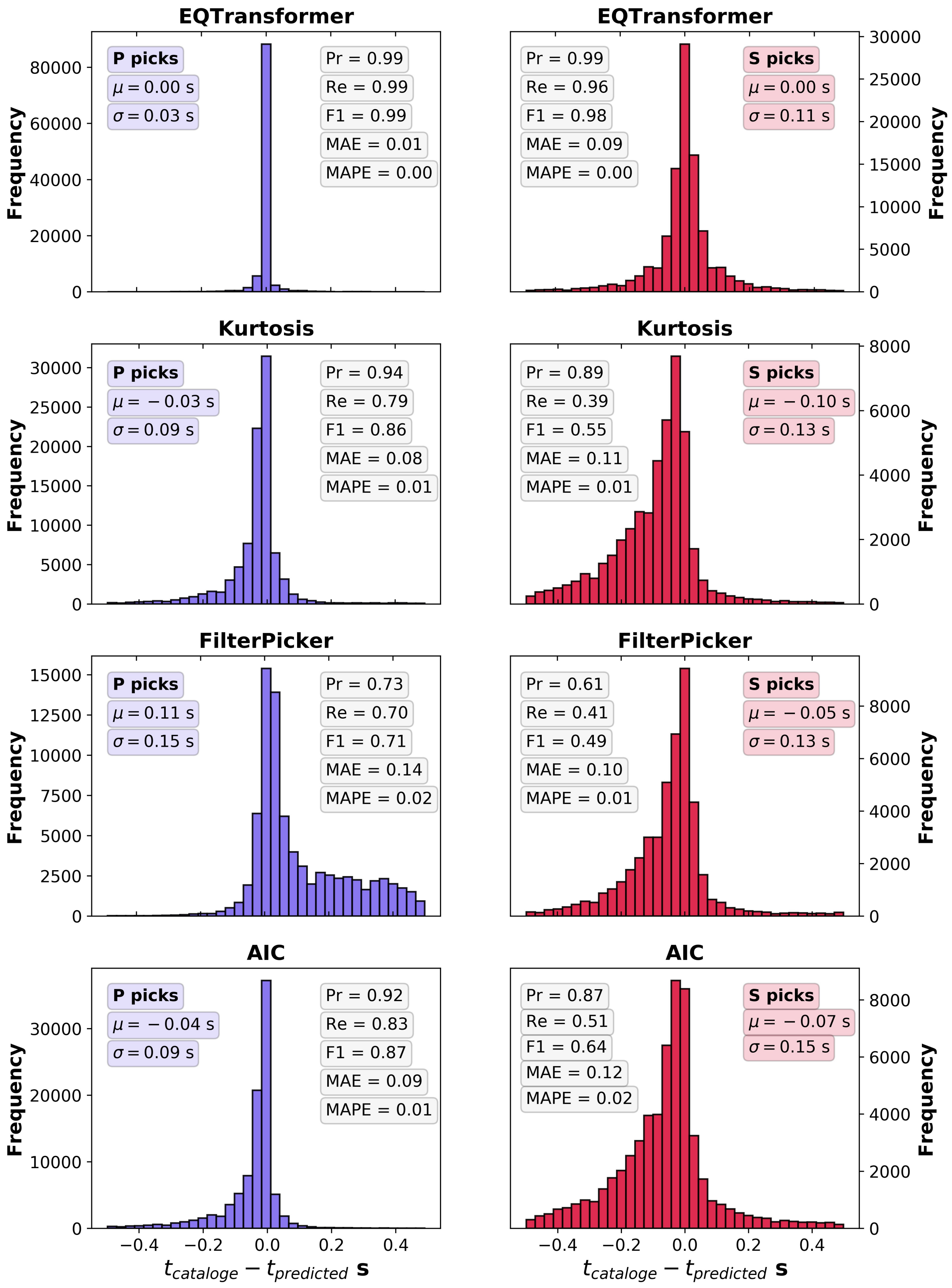

Trace Name: HKTH_NN_HH_2000-11-06T00:36:24.000000Z
EqTransformer
0.1.0


Trace Name: HKTH_NN_HH_2000-11-06T01:28:54.000000Z
EqTransformer
0.1.0


Trace Name: HKTH_NN_HH_2000-11-06T01:54:48.000000Z
EqTransformer
0.1.0


Trace Name: JAL_SC_HH_2020-03-15T05:21:18.005000Z
Trace Name: JAL_SC_HH_2020-03-15T10:04:06.005000Z
EqTransformer
0.1.0


EQTransformer
P picks
μ = 0.00 s
σ = 0.03 s
Pr = 0.99
Re = 0.99
F1 = 0.99
MAE = 0.01
MAPE = 0.00
EQTransformer
Pr = 0.99
Re = 0.96
F1 = 0.98
MAE = 0.09
MAPE = 0.00
S picks
μ = 0.00 s
σ = 0.11 s
PhaseNet
P picks
μ = −0.02 s
σ = 0.08 s
Pr = 0.96
Re = 0.96
F1 = 0.96
MAE = 0.07
MAPE = 0.01
PhaseNet
Pr = 0.96
Re = 0.93
F1 = 0.94
MAE = 0.09
MAPE = 0.01
S picks
μ = −0.02 s
σ = 0.11 s
GPD
P picks
μ = 0.03 s
σ = 0.10 s
Pr = 0.81
Re = 0.80
F1 = 0.81
MAE = 0.08
MAPE = 0.01
GPD
Pr = 0.81
Re = 0.83
F1 = 0.82
MAE = 0.10
MAPE = 0.01
S picks
μ = 0.03 s
σ = 0.14 s
Yews
P picks
μ = 0.07 s
σ = 0.13 s
Pr = 0.54
Re = 0.72
F1 = 0.61
MAE = 0.09
MAPE = 0.02
Yews
Pr = 0.75
Re = 0.75
F1 = 0.75
MAE = 0.11
MAPE = 0.01
S picks
μ = 0.08 s
σ = 0.17 s
PpkNet
P picks
μ = −0.01 s
σ = 0.15 s
Pr = 0.90
Re = 0.90
F1 = 0.90
MAE = 0.10
MAPE = 1.90
PpkNet
Pr = 1.00
Re = 0.91
F1 = 0.95
MAE = 0.10
MAPE = 1.85
S picks
μ = 0.02 s
σ = 0.15 s
pickNet
μ = 0.00 s
σ = 0.09 s
Pr = 0.81
Re = 0.49
F1 = 0.61
MAE = 0.07
pickNet
Pr = 0.83
Re = 0.55
F1 = 0.66
MAE = 0.10
S picks
μ = −0.02 s
σ = 0.13 s
Normalized Frequency


EQTransformer
P picks
$\mu = 0.00$ s
$\sigma = 0.03$ s
Pr = 0.99
Re = 0.99
F1 = 0.99
MAE = 0.01
MAPE = 0.00
EQTransformer
S picks
$\mu = 0.00$ s
$\sigma = 0.11$ s
Pr = 0.99
Re = 0.96
F1 = 0.98
MAE = 0.09
MAPE = 0.00
Kurtosis
P picks
$\mu = -0.03$ s
$\sigma = 0.09$ s
Pr = 0.94
Re = 0.79
F1 = 0.86
MAE = 0.08
MAPE = 0.01
Kurtosis
S picks
$\mu = -0.10$ s
$\sigma = 0.13$ s
Pr = 0.89
Re = 0.39
F1 = 0.55
MAE = 0.11
MAPE = 0.01
FilterPicker
P picks
$\mu = 0.11$ s
$\sigma = 0.15$ s
Pr = 0.73
Re = 0.70
F1 = 0.71
MAE = 0.14
MAPE = 0.02
FilterPicker
S picks
$\mu = -0.05$ s
$\sigma = 0.13$ s
Pr = 0.61
Re = 0.41
F1 = 0.49
MAE = 0.10
MAPE = 0.01
AIC
P picks
$\mu = -0.04$ s
$\sigma = 0.09$ s
Pr = 0.92
Re = 0.83
F1 = 0.87
MAE = 0.09
MAPE = 0.01
AIC
S picks
$\mu = -0.07$ s
$\sigma = 0.15$ s
Pr = 0.87
Re = 0.51
F1 = 0.64
MAE = 0.12
MAPE = 0.02
Frequency
$t_{cataloge} - t_{predicted}$ s
−0.4
−0.2
0.0
0.2
0.4

This will download and install **EQTransformer** and all required packages (including Tensorflow and Obspy) into your machine. *Note:* Keep executing the last line if it did not succeed in the first try.

2.2.2 Installation via PyPI

If you already have Obspy installed on your machine, you can get EQTransformer through PyPI:

```
pip install EQTransformer
```

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2.3 Installation from Source

The sources for EQTransformer can be downloaded from the Github repo.

You can either clone the public repository:

```
git clone git://github.com/smousavi05/EQTransformer
```

Once you have a copy of the source, you can cd to EQTransformer directory and install it with:

```
python setup.py install
```

2.3 Tutorial

EQTransformer package is divided into two main sub-modules, the `core` and `utils` sub-modules.

The `core` sub-module contains the main, high-level functions:

trainer It can be used to generate and train new **EQTransformer** models with different encoder depths.

tester It is used to test a trained model using ground truth data.

predictor It is used to apply a pre-trained model to pre-processed continuous data.

mseed_predictor It is used to perform a fast detection & picking directly on continuous data in MiniSeed format.

The `utils` sub-module contains the main, high-level functions:

downloader It can be used to download continuous data from seismic networks.

hdf5_maker It is used to pre-process the continuous data and slice it to 1-minute windows used by predictor module.

plot It contains a few different methods to visualize the detection and downloading results.

associator Performs a simple phase association and output phase information for the associated events in HypoInverse input format.

2.3.1 Downloading Continuous Data

The following will download the information on the stations that are available based on your search criteria:

```
import os
json_basepath = os.path.join(os.getcwd(),"json/station_list.json")

from EQTransformer.utils.downloader import makeStationList

makeStationList(json_path=json_basepath, client_list=["SCEDC"], min_lat=35.50, max_
↪lat=35.60, min_lon=-117.80, max_lon=-117.40, start_time="2019-09-01 00:00:00.00",␣
↪end_time="2019-09-03 00:00:00.00", channel_list=["HH[ZNE]", "HH[Z21]", "BH[ZNE]"],␣
↪filter_network=["SY"], filter_station=[])
```

The above function will generate `station_list.json` file containing the station information. Next, you can use this file and download 1 day of data for the available stations at Ridgecrest, California from Southern California Earthquake Data Center or IRIS using the following:

```
from EQTransformer.utils.downloader import downloadMseeds

downloadMseeds(client_list=["SCEDC", "IRIS"], stations_json=json_basepath, output_dir=
↪"downloads_mseeds", min_lat=35.50, max_lat=35.60, min_lon=-117.80, max_lon=-117.40,␣
↪start_time="2019-09-01 00:00:00.00", end_time="2019-09-03 00:00:00.00", chunk_
↪size=1, channel_list=[], n_processor=2)
```

This will download the continous data (in MiniSeed) and save them into individual folders for each station insider your defined output directory (i.e. downloads_mseeds).

Check the downloading.ipynb or API Documentations for more details.

2.3.2 Detection and Picking

To perform detection & picking you need a pre-trained model of **EQTransformer** which you can get from Model-sAndSampleData.

EQTransformer provides two different option for performing the detection & picking on the continuous data:

- Option (I) using pre-processed data (hdf5 files):

This option is recommended for smaller periods (a few days to a month). This allows you to test the performance and explore the effects of different parameters while the provided hdf5 file makes it easy to access the waveforms.

For this option, you first need to convert your MiniSeed files for each station into 1-min long Numpy arrays in a single hdf5 file and generated a CSV file containing the list of traces in the hdf5 file. You can do this using the following command:

```
from EQTransformer.utils.hdf5_maker import preprocessor

preprocessor(preproc_dir="preproc", mseed_dir='downloads_mseeds', stations_json=json_
↪basepath, overlap=0.3, n_processor=2)
```

This will generate one `station_name.hdf5` and one `station_name.csv` file for each of your station's data and put them into a directory named `mseed_dir+_hdfs`. Then you need to pass the name of this directory (which contains all of your hdf5 & CSV files) and a model to the following command:

```
from EQTransformer.core.predictor import predictor

predictor(input_dir= 'downloads_mseeds_processed_hdfs', input_model='EqT_model.h5',␣
↪output_dir='detections', detection_threshold=0.3, P_threshold=0.1, S_threshold=0.1,␣
↪number_of_plots=100, plot_mode='time')
```

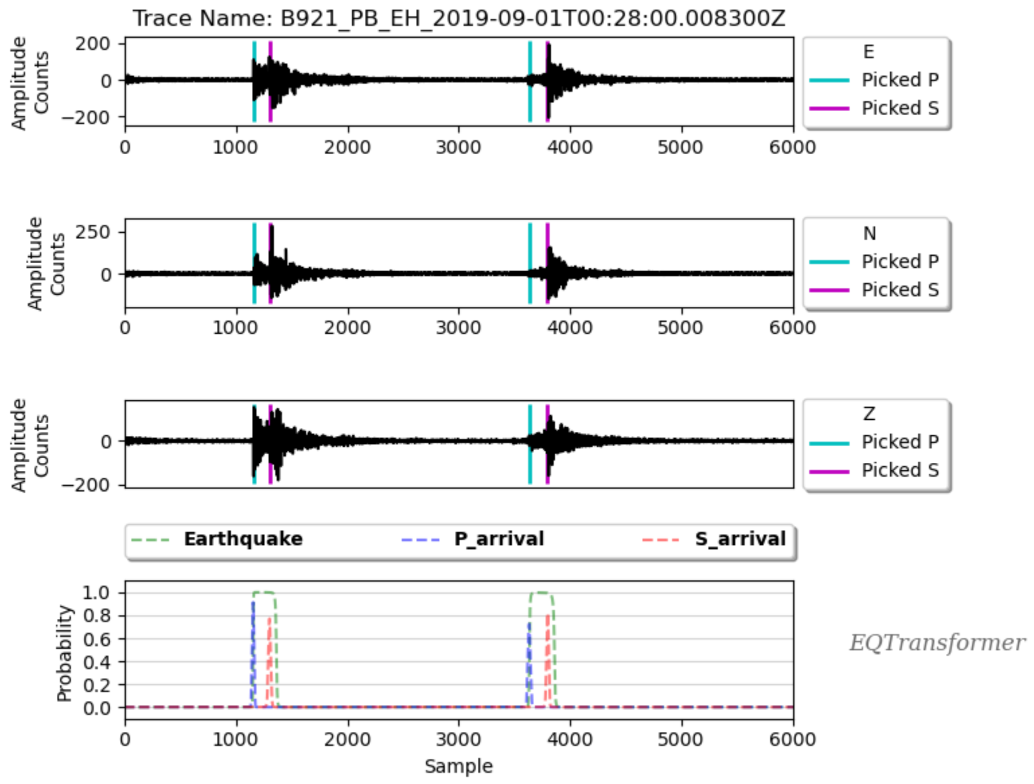
You can use relatively low threshold values for the detection and picking since **EQTransformer** is robust to false positives. Note that enabling uncertainty estimation, outputting probabilities, or plotting all the detected events will slow down the process.

Outputs for each station will be written in your output directory (i.e. detections).

`X_report.txt` contains the processing info on input parameters used for the detection &picking and final results such as running time, the total number of detected events (these are unique events and duplicated ones have been already removed).

`X_prediction_results.csv` contains detection & picking results.

In the figures folder, you can find the plots for some detected events:



These plots are helpful to check if you are getting too many false positives (non-earthquake signals) and get a better sense that if your selected threshold values for the detection and picking is too high or too low.

If you are using local MiniSeed files you can generate a station_list.json by supplying an absolute path to a directory containing Miniseed files and a station location dictionary using the stationListFromMseed function like the following:

- Option (II) directly from mseed files:

You can perform the detection & phase picking directly on downloaded MiniSeed files. This saves both preprocessing time and the extra space needed for the hdf5 file and is recommended for larger (longer) datasets. However, it can be more memory intensive. So it is better to have your MiniSeed fils being shorter than one month or so.

This option also does not allow you to estimate the uncertainties, save the prediction probabilities, or use the advantages of having hdf5 files which makes it easy to access the raw event waveforms based on detection results.

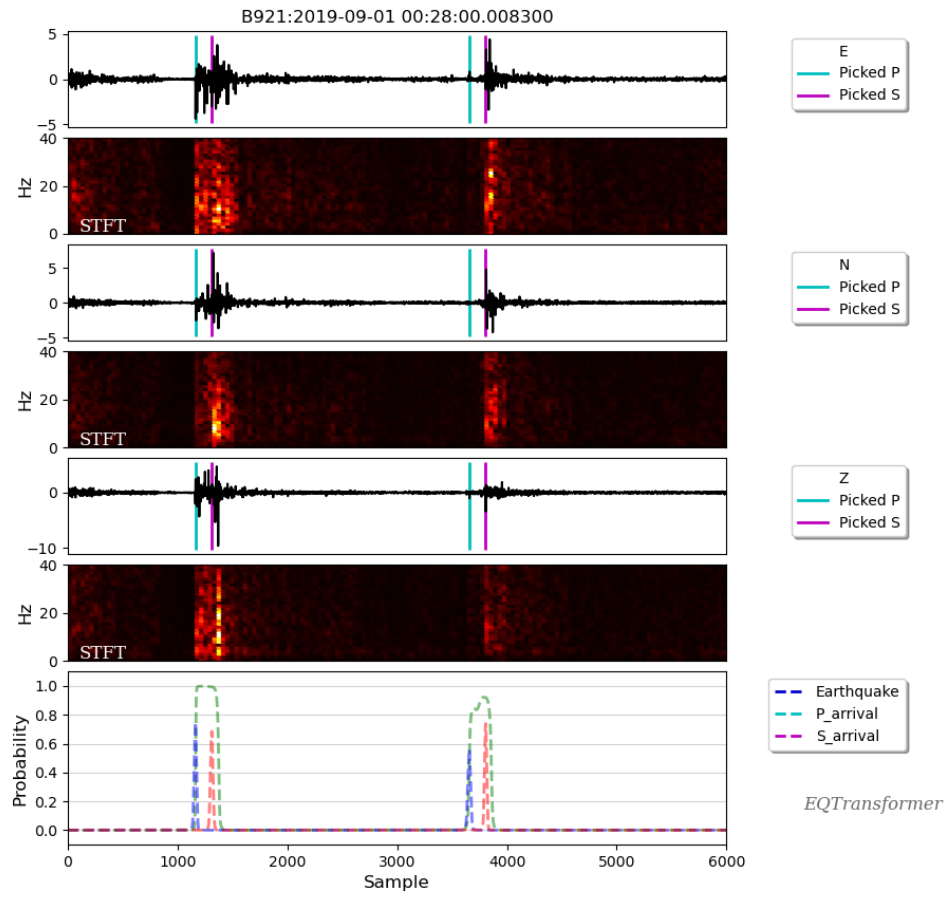
```
from EQTransformer.core.mseed_predictor import mseed_predictor

mseed_predictor(input_dir='downloads_mseeds', input_model='EqT_model.h5', stations_
↪json=json_basepath, output_dir='detections', detection_threshold=0.3, P_threshold=0.
↪1, S_threshold=0.1, number_of_plots=100, plot_mode='time_frequency', overlap=0.3,␣
↪batch_size=500)
```

(continues on next page)

(continued from previous page)

```

```

As you can see from the above example, you can choose between two different modes for your plots. The selected time_frequency mode will output following plots that can be useful to identify non-earthquake signals from earthquake ones based on their frequency contents:



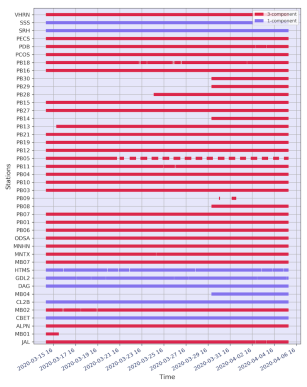
Check the detection.ipynb or API Documentations for more details.

2.3.3 Visualizing the Results

- Continouty of the Seismic Data Being Processed:

Both `prepocessor` and `mseed_predictor` output a `time_tracks.pkl` file that contains the time info of original data and their number of components. You can use this file to visualize the continuity and type of your data using the following module:

```
from EQTransformer.utils.plot import plot_data_chart

plot_data_chart('time_tracks.pkl', time_interval=10)
```

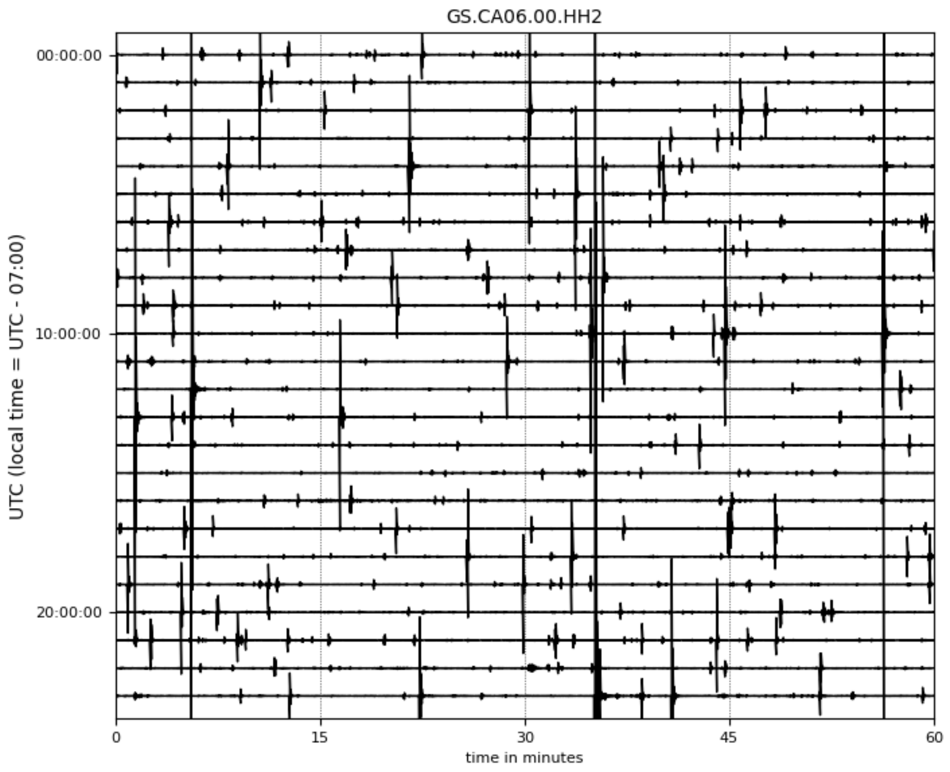


- Helicorder Plots:

To check if you are missing too many events (high false negative) in the continuous data or catch most of them, it is always a good idea to check out the raw data (the most important lesson in observational seismology). You can do it using these commands:

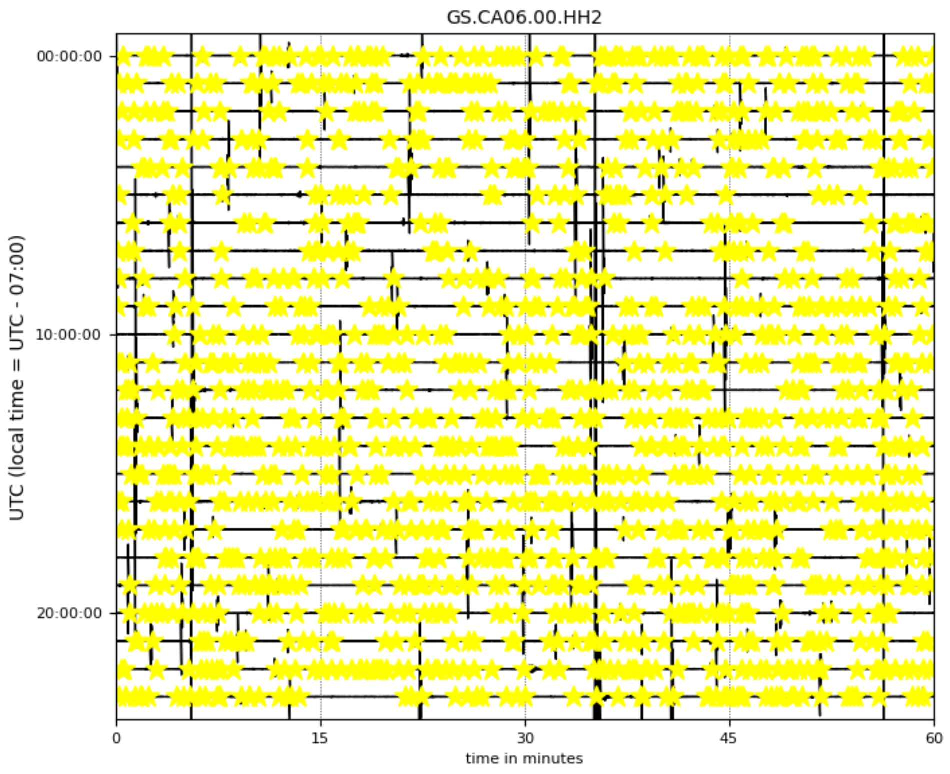
First, you can check one particular day of (raw) data:

```
from EQTransformer.utils.plot import plot_detections, plot_helicorder

plot_helicorder(input_mseed='downloads_mseeds/CA06/GS.CA06.00.HHZ__20190902T000000Z__
↪20190903T000000Z.mseed', input_csv=None)
```



Now the following command will mark those events that you have detected on your helicorder plot:

```
plot_helicorder(input_mseed='downloads_mseeds/CA06/GS.CA06.00.HHZ__20190902T000000Z__
↪20190903T000000Z.mseed', input_csv='detections/CA06_outputs/X_prediction_results.csv
↪')
```

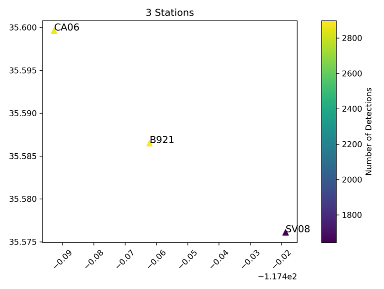


This together with the events plots can give you a sense that if you are using too high or too low threshold levels.

- (III) Map Plot:

You can also visualize the number of detections over stations using this:

```
plot_detections(input_dir="detections", input_json="station_list.json", plot_type=
↪'station_map', marker_size=50)
```

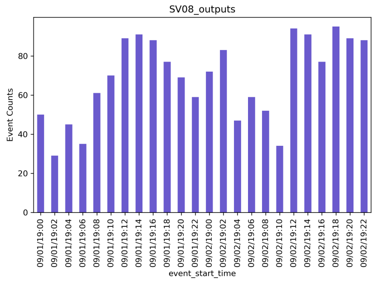


This is sometimes helpful to identify problematic stations (e.g. those that are closer to anthropogenic sources) and exclude them from you're further analyses.

- (IV) Histograms:

And the following command will generate histograms of the detected events for each station in your detections folder:

```
plot_detections(input_dir="detections", input_json="station_list.json", plot_type=
↪'hist', time_window=120)
```



Check the visualization.ipynb or API Documentations for more details.

2.3.4 Phase Association

After detection, the following performs a simple and fast association and writes down the results in HypoInverse format (`Y2000.phs`) and ObsPy QuakeML format (`associations.xml`) which can directly be used to locate the detected earthquakes using conventional location algorithms like HypoInverse or NonLinLoc. This also outputs `traceName_dic.json`, a dictionary where the trace names for source waveforms of all the detections associated with an event are listed. This can be used later to access the original waveform traces for calculating the cross-correlations during the relocation process or magnitude estimation.

```
import shutil
import os
from EQTransformer.utils.associator import run_associator

out_dir = "asociation"
try:
        shutil.rmtree(out_dir)
except Exception:
        pass
os.makedirs(out_dir)

run_associator(input_dir='detections', start_time="2019-09-01 00:00:00.00", end_time=
↪"2019-09-03 00:00:00.00",  moving_window=15, pair_n=3)
```

Note that unlike the `predictor`, `mseed_predictor`, and `downloader` modules the `associator` does not automatically generate the output directory and you need to create it first. Otherwise, it will write the output files in the current directory.

Check the association.ipynb or API Documentations for more details.

2.3.5 Building and Testing a New Model

You can also generate your own **EQTransformer** network (e.g. with different encoder depths, augmentation, label type, etc) and train it on your data. The only prerequisite is that your data need to be in our data format (STEAD).

```
from EQTransformer.core.trainer import trainer

trainer(input_hdf5='waveforms.hdf5', input_csv='metadata.csv', output_name='test_
↪trainer', cnn_blocks=2, lstm_blocks=1, padding='same', activation='relu', drop_
↪rate=0.2, label_type='gaussian', add_event_r=0.6, add_gap_r=0.2, shift_event_r=0.9,␣
↪add_noise_r=0.5, mode='generator', train_valid_test_split=[0.60, 0.20, 0.20], batch_
↪size=20, epochs=10, patience=2, gpuid=None, gpu_limit=None)
```

(continues on next page)

(continued from previous page)

After you built your model you can also test it using your ground truth data:

```
from EQTransformer.core.tester import tester

tester(input_hdf5='waveforms.hdf5', input_testset='test.npy', input_model='test_
↪trainer_001.h5', output_name='test_tester', detection_threshold=0.20, P_threshold=0.
↪1, S_threshold=0.1, number_of_plots=3, estimate_uncertainty=True, number_of_
↪sampling=2, input_dimention=(6000, 3), normalization_mode='std', mode='generator',␣
↪batch_size=10, gpuid=None, gpu_limit=None)
```

Check the training.ipynb or API Documentations for more details.

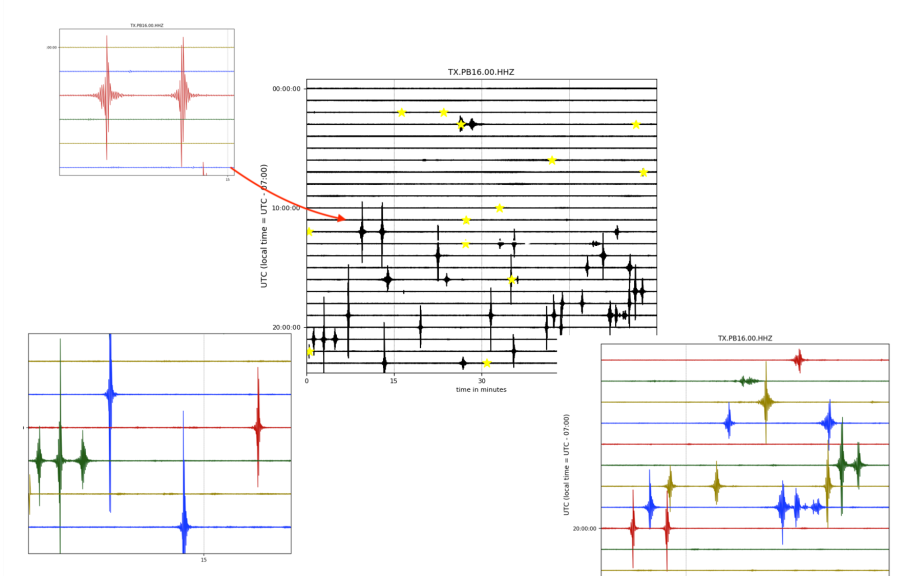
2.3.6 Warnings and Recommendations

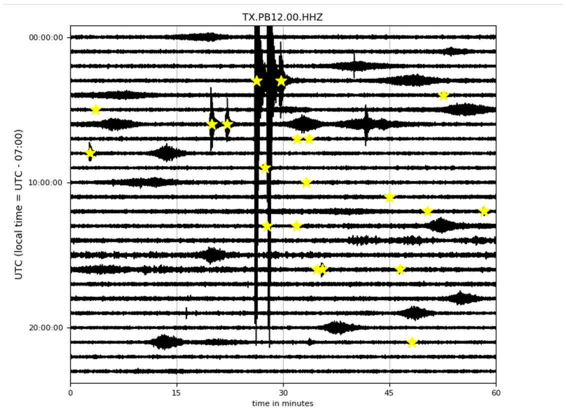
- Notice the main requirement is that your MiniSeed files names follow the IRIS format (e.g. `GS.CA06.00.HHZ__20190902T000000Z__20190903T000000Z.mseed`). If your mseed files have different name format you just need to change their names.
- The appropriate choice of values for parameters like detection and picking thresholds, batch_size, and the overlap values can affect the number of detected events. A recommended workflow is to first apply the predictor modules on a small portion of your data (1 or 2 days) with different parameter values and after hyperparameter tuning apply the model to your whole dataset.
- `downloader`, `preprocessor`, `predictor`, and `mseed_predictor` will erase the previous folders and generate an empty directory for writing the outputs. They will give you a warning ask your permission if a folder with the given name of ouput_dir already exists. So be careful if you don't want to erase your previous results.
- The provided associator module is a very simple algorithm mainly based on the detection times. It is appropriate for a small number of stations located relatively close to each other and to the source. For larger or regional networks or cases with a high seismicity rate, you may need to use a more sophisticated and accurate associator.
- The examples subfolder in the GitHub repository contains small and quick examples for each module. As a quick start, you can run them one by one after you installed the package.
- The provided models (e.g. `EqT_model.h5`, and `EqT_model2.h5`) have been trained using different settings and have different attributes. While `EqT_model.h5` has been trained to minimize false positives, `EqT_model2.h5` has been trained to minimize false-negative rate.
- EqT models use the dropout sampling technique. At each inference, a different set of neurons are randomly used. Thus the output prediction values and as a result, the number of detected events might differ from one prediction run to another. This is what we use to estimate the model uncertainties.
- And finaly these are a few interesting cases:

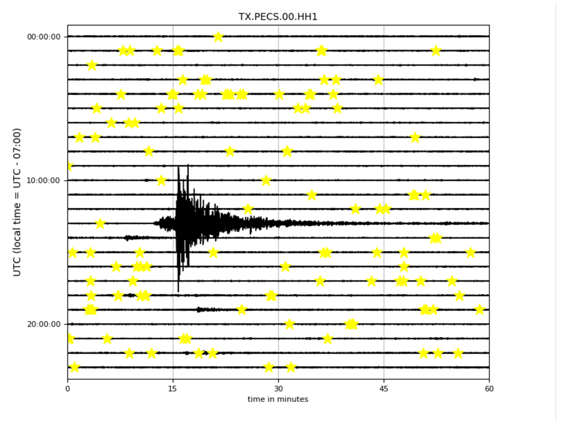
In the following figures, EqT detected some small earthquakes with weaker signals while it was insensitive to non-earthquake signals with strong impulsive energies.

Here, EqT detected many smaller earthquakes while ignoring a large teleseismic event. This is an inherent characteristic of EQTransformer to be only sensitive to local events (mainly within 150 km) and filter out regional and teleseismic ones.

- Good Luck


TX.PB16.00.HHZ
00:00:00
10:00:00
20:00:00
UTC (local time = UTC - 07:00)
time in minutes
0
15
30
TX.PB16.00.HHZ
20:00:00
UTC (local time = UTC - 07:00)


TX.PB12.00.HHZ
00:00:00
10:00:00
20:00:00
UTC (local time = UTC - 07:00)
0
15
30
45
60
time in minutes


TX.PECS.00.HH1
00:00:00
10:00:00
20:00:00
UTC (local time = UTC - 07:00)
0
15
30
45
60
time in minutes

2.4 Copyright and License

2.4.1 Copyright

```
Copyright 2020, S. Mostafa Mousavi
```

2.4.2 The MIT License

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

2.4.3 Contact

Question? Bug repots? Please contact Mostafa Mousavi vi mmousavi@stanford.edu

2.5 Contributing

We welcome all contributions including bug fixes, feature enhancements, and documentation improvments.

2.5.1 Coding Standards

- 4 space indentation (no tabs) and PEP8 conformance
- No use of __author__
- Documentation must be in Sphinx-compliant format.
- Submitted code should follow standard practices for documentation and testing.
- Automated testing (using `pytest`) must pass prior to merging.

2.6 References

Wang, J., Xiao, Z., Liu, C., Zhao, D. & Yao, Z. Deep-Learning for Picking Seismic Arrival Times. Journal of Geophysical Research: Solid Earth (2019).

Zhu, L. et al. Deep learning for seismic phase detection and picking in the aftershock zone of 2008 Mw7. 9 Wenchuan Earthquake. Physics of the Earth and Planetary Interiors (2019).

Zhou, Y., Yue, H., Kong, Q. & Zhou, S. Hybrid Event Detection and Phase-Picking Algorithm Using Convolutional and Recurrent Neural Networks. Seismological Research Letters 90, 1079–1087 (2019).

Mousavi, S. M., Zhu, W., Sheng, Y. & Beroza, G. C. CRED: A deep residual network of convolutional and recurrent units for earthquake signal detection. Scientific reports 9, 10267 (2019).

Zhu,W. & Beroza, G. C. PhaseNet: a deep-neural-network-based seismic arrival-time picking method. Geophysical Journal International 216, 261–273 (2018).

Ross, Z. E., Meier, M.-A., Hauksson, E. & Heaton, T. H. Generalized seismic phase detection with deep learning. Bulletin of the Seismological Society of America 108, 2894–2901 (2018).

Mousavi, S. M., Sheng, Y., Zhu,W. & Beroza, G. C. STanford EArthquake Dataset (STEAD): A Global Data Set of Seismic Signals for AI. IEEE Access (2019).

Lomax, A., Satriano, C. & Vassallo, M. Automatic picker developments and optimization: FilterPicker—A robust, broadband picker for real-time seismic monitoring and earthquake early warning. Seismological Research Letters 83, 531–540 (2012).

Maeda, N. A method for reading and checking phase times in autoprocessing system of seismic wave data. Zisin 38, 365–379 (1985).

Saragiotis, C. D., Hadjileontiadis, L. J. & Panas, S. M. PAI-S/K: A robust automatic seismic P phase arrival identification scheme. IEEE Transactions on Geoscience and Remote Sensing 40, 1395–1404 (2002).

Fukuyama, E., Ellsworth, W. L., Waldhauser, F. & Kubo, A. Detailed fault structure of the 2000 western Tottori, Japan, earthquake sequence. Bulletin of the Seismological Society of America 93, 1468–1478 (2003).

Earthquake Transformer: An Attentive Deep-learning Model for Simultaneous Earthquake Detection and Phase Picking

2.7 Indices and tables

- genindex
- modindex
- search