Deep Learning Earthquake Monitoring

Herwig Höhenberger
Tobias M. Rahn
Yanis Tournier

Group 2
ETH Zürich
Switzerland

Dr. Men-Andrin Meier
Nikolaj L. Dahmen

Supervisor
Institut for Geophysics
ETH Zürich
Switzerland

***Abstract*—Event detection and phase picking form the core of seismological workflows, crucial for precise seismic analysis. Traditional annotation of time series data for these tasks involves labour-intensive efforts by experts, contributing to a processing bottleneck. This study addresses the challenge by harnessing the transformative potential of deep learning approaches, surpassing classical methodologies and achieving human-like precision.**

Motivated by the need to enhance seismic data analysis efficiency and introduce robust uncertainty estimation, we propose implementing Bayesian uncertainty methods atop deep learning models for comprehensive uncertainty estimates in event detection and phase picking. This enhances result reliability and quantifies uncertainty, often overlooked in traditional seismological analyses. Our methodology enables seamless information transfer to subsequent tasks, such as precise earthquake-origin determination or spatial reconstruction. This work signifies a step forward in integrating deep learning into seismological workflows, providing a more comprehensive and informed approach to seismic data analysis.

I. Introduction

Seismic data, captured by seismographs measuring ground motions over time along one, two, or three dimensions and with a specific frequency, plays a pivotal role in understanding Earth's subsurface dynamics. These recordings manifest as time series of varying lengths, comprising a mix of ambient noise and seismic events, such as earthquakes. The seismic workflow involves two essential tasks: firstly, classifying a signal as either noise or indicative of a seismic event and secondly, a regression task to precisely locate primary (P) and secondary (S) waves within the seismic signal.

Recent advancements in deep learning have significantly impacted the field of seismology. Notably, studies such as [12] have demonstrated that sophisticated deep learning models consistently outperform traditional statistical models, as exemplified by the work of Baer and Kradolfer in 1987 [8]. Across diverse datasets and configurations, PhaseNet [17] and EQTransformer [11] have emerged as top performers, showcasing the highest accuracy in seismic event detection and phase picking. PhaseNet relies on convolutional neural networks, while EQTransformer adopts a transformer architecture with multi-head attention, highlighting the versatility of deep learning approaches in seismic data analysis.

In the pursuit of advancing the state of the art in seismic data analysis, our work addresses a critical aspect often overlooked in traditional methodologies: uncertainty estimation. To achieve this, we propose leveraging Bayesian methods, in particular Monte Carlo dropout [2] and SWAG [9], to approximate the uncertainty in deep-learning models by estimating the posterior distribution based on their parameters. By incorporating uncertainty quantification into the workflow, our methodology not only enhances the reliability of seismic event detection but also contributes to a more comprehensive understanding of the associated uncertainties, facilitating informed decision-making in subsequent tasks.

II. Explanation of the Data

A. Seismic data

Seismic data, stemming from sudden fault movements during earthquakes, captures the release of elastic energy in rocks. Approximately fifty daily global earthquakes, exceeding magnitude 2.5, include occasional structural threats. Even smaller, imperceptible earthquakes below magnitude 2.5 provide essential insights, recorded by modern instruments.

Seismic waves, documented by seismographs, are most often recorded by three-component seismographs, revealing observable P and S phases. P waves involve material movement along the wave's direction, while S waves entail perpendicular movement.

Magnitude, a logarithmic measure determined from seismograph wave amplitudes, signifies an earthquake's size. As magnitude increases, wave amplitude grows exponentially, with energy release rising by about 32 times per unit increase.

Seismic waves are not exclusive to earthquakes; various sources, like explosions and human activities, contribute to recorded ground motions. Continuous seismic

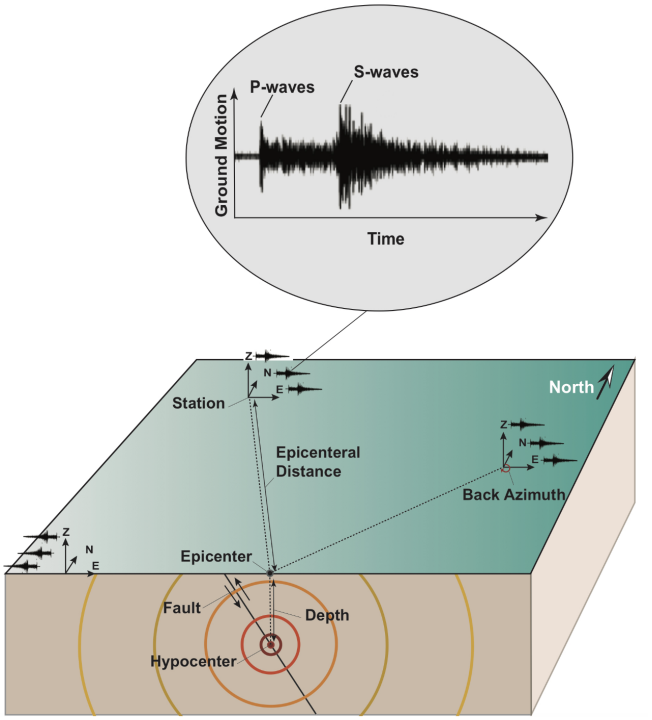

Figure 1. Earthquake-measurement schematic.

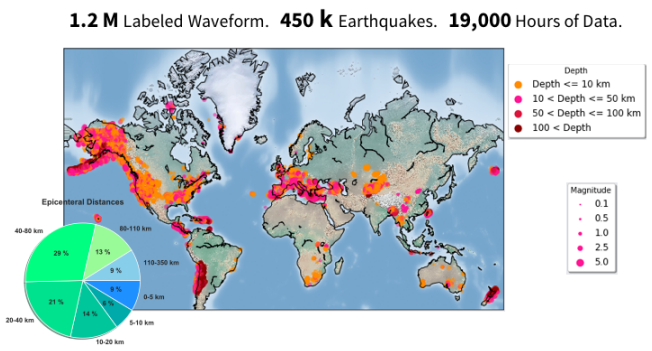

Figure 2. Location and sizes of earthquakes

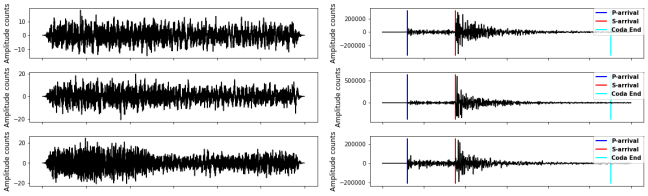

Figure 3. Example of noise and earthquake signals—STEAD

monitoring generates a vast daily dataset, including nonearthquake signals.

Seismology, a data-driven science, grapples with accelerated data acquisition. Efficient tools are crucial for processing and extracting useful insights. Though seismologists analyze only a portion of recorded data, the growing volume underscores potential insights through reanalysis with advanced tools.

B. Stanford Earthquake Dataset

The STanford EArthquake Dataset (STEAD) [10] is introduced as a high-quality, large-scale global annotated dataset comprising seismic signals recorded by seismic instruments. The dataset includes two categories of signals: local earthquake waveforms (recorded within 350 km of earthquakes) and seismic noise waveforms devoid of earthquake signals, totaling over 1.2 million time series or more than 19 000 hours of seismic signal recordings. The dataset addresses the scarcity of high-quality labeled datasets in seismology, providing a reliable benchmark for machine learning applications.

STEAD serves as a pivotal resource for seismic signal analysis, offering distinctive features:

- STEAD leverages noise waveforms for model training, enabling the classification of signals as seismic events or nonevents. All signal waveforms are uniformly represented as 60-second windows sampled at 100 Hz, simplifying integration into machine learning models. Each seismic data point is encapsulated as a three-dimensional NumPy [4] array, comprising 6000 points per waveform.
- Enriching the dataset, each seismic data point includes metadata with essential information, such as the P and S wave locations, distance to the epicenter, magnitude, and source depth. This combination of noise waveforms, standardized signal representation, and rich metadata positions STEAD as a robust tool for advancing seismic signal classification and analysis.

In fig. 3, you can find a example of what a noise and a an earthquake signal looks like with the annotation for the P and S waves of interest.

C. Bedretto Underground Laboratory for Geosciences and Geoenergies

The Bedretto Underground Laboratory for Geosciences and Geoenergies (BedrettoLab), managed by ETH Zurich, is a cutting-edge underground research facility located 1.5 kilometres below the Swiss Alps. Situated in a 5.2-kilometre tunnel, it provides an ideal setting to study the Earth's interior. Equipped with state-of-the-art technology, the lab focuses on experimental research to understand the behaviour of the deep underground, particularly in the realms of geothermal energy and earthquake physics. The facility also contributes to the development of novel techniques and sensors for these scientific purposes.

We have been provided with a dataset comprising

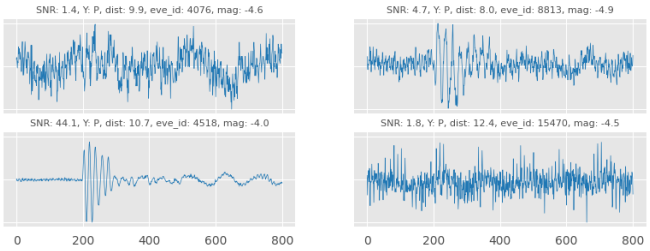

Figure 4. Noise and earthquake signals—BedrettoLab

20 000 samples. Each sample corresponds to 0.1 s and a total of 17 227 picks (P and S) that have been manually annotated. In comparison to the STEAD dataset, which has three component waveforms, this dataset only contains single component waveforms, similar to the z component, corresponding to vertical movement of the seismograph. Additionally, the frequency is much higher with 200 000 Hz compared to the 100 Hz of STEAD and the number of samples inferior by two orders of magnitude. In fig. 4, you can find an example of what a waveform of this dataset looks like. Note that this dataset does not contain any noise samples.

III. Models and Methods

A. EQTransformer—original

The EQTransformer model architecture comprises a multi-task structure with a very-deep encoder and three distinct decoders. The encoder processes seismic signals in the time domain, generating high-level representations and temporal dependencies. The decoders use this information to produce probabilities for earthquake signal existence, P-phase, and S-phase at each time point. To handle growing memory in self-attentive models, a down-sampling section with convolutional and max-pooling layers is added to the encoder. Down-sampled features undergo transformation through residual convolution and LSTM blocks. A global attention section directs the network's focus to earthquake signal-associated parts. The high-level features are mapped to probability vectors for earthquake signal existence and seismic phases using three decoder branches. Residual connections and network-in-network techniques enhance depth without compromising error rates or training speed, resulting in a very deep network with 56 layers and around 372 000 trainable parameters. More details can be found in the original paper [11].

Thanks to its three-blocks architecture and the utilisation of self-attention layers, the model is able to simultaneously perform the classification (labelling a sample as noise or not) and the regression task (localising P and S waves). Figure 6 shows an example of the output from EQTransformer once trained on STEAD.

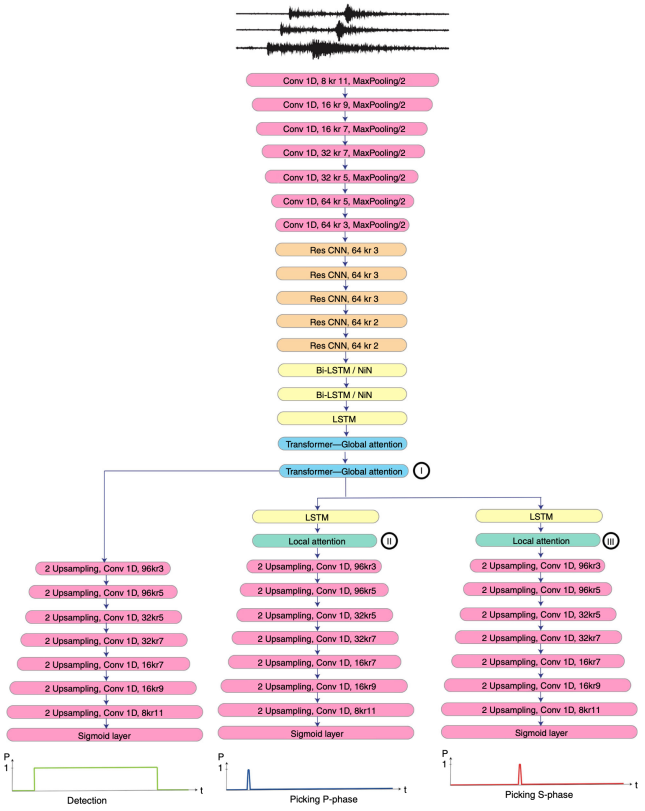

Figure 5. EQTransformer architecture.

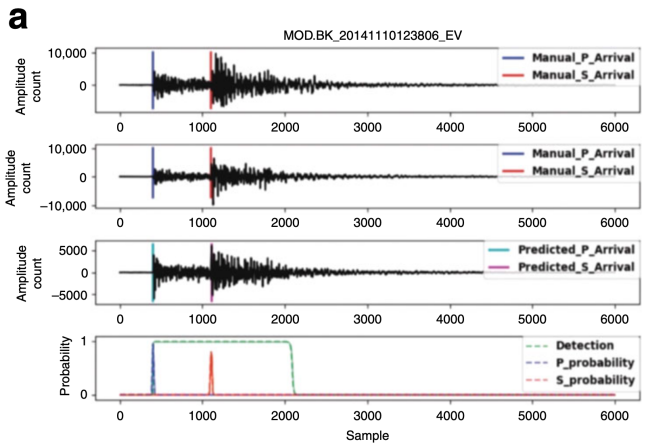

Figure 6. EQTransformer output example on STEAD.

B. EQTransformer—modified

We reused the EQTransformer implementation from SeisBench and adapt it to the specificities of the BedrettoLab data. The main difference being the longer signal windows and the single channel. The high complexity of EQTransformer compared to other deep learning solutions like PhaseNet motivated us to experiment with simplified variations of the original architecture.

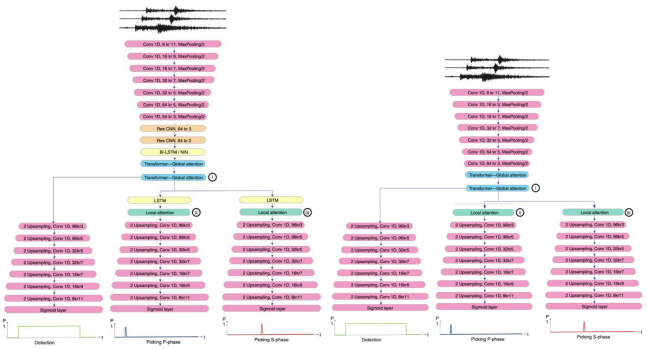

Figure 7. Schema of the modified EQTransformer architecture.

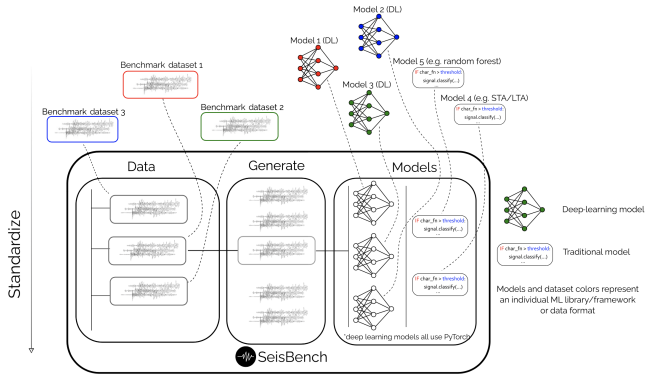

Figure 8. Schema of SeisBench Motivation

We propose EQTRedEnc with a shallower encoder and EQTNoResLSTM, where all CNN and LSTM layers are dropped.

C. SeisBench

SeisBench [16] is a Python-based framework that serves as a centralised platform for machine learning development and application in the seismological community. It integrates state-of-the-art models and datasets into a unified framework, allowing users to compare algorithms across various seismic environments.

The main advantage of SeisBench lies in its standardisation of the data format with an HDF5 waveforms file and a CSV metadata file. The STEAD dataset for pretraining purpose is accessible in this format and the data from BedrettoLab has been converted in this format too.

D. Data Augmentation

As the BedrettoLab dataset represents a limited size of labelled points (20 000) compared to the complexity of the the model, data augmentation becomes a fundamental tool. It enables an increased number of training samples and added diversity, allowing to improve model generalisation while reducing over-fitting. The data augmentation is adapted from the SeisBench implementation. It enables in 66 % of the cases to sub-sample a smaller 6000 points window around the pick from the original 12 000 and in 33 % returns the original trace to keep diversity high. For the training data more augmentations is enabled by adding a second event to the trace, Gaussian noise, array rotation and added gaps. This data augmentation alongside the normalisation represents the main preprocessing steps.

E. Model Training

Since the BedrettoLab dataset only became available during the final third of the semester and only contains a single time-series component per sample (compared to three components more typically available in seismology), we initially trained on STEAD using only one channel, hoping the model would transfer well to the BedrettoLab data later on.

This was motivated in part through domain knowledge: Seismic signals appear quite self-similar—they 'look the same' whether one considers a signal 1 min long, as in STEAD, or 0.1 s long, as in the BedrettoLab data.

Training on only one channel is a departure from the original EQTransformer paper ([11]) which uses all three components of every available seismogram. As part of this study, we investigate whether this leads to significant performance degradation.

We train each model using the same random 85 % subset of STEAD, using the Adam optimiser ([6]) together with binary cross entropy loss and a constant learning rate of 0.001. Training for 20 epochs already yields decent results; see section V.

We tried three different approaches to transfer the existing pipeline to the BedrettoLab dataset. In the first we trained the one-channel EQTransformer model from scratch with an input size of 20 000 samples for 50 epochs. In the other two models, we used a one-channel EQTransformer model that was trained as stated below as a base for retraining the model for 30 epochs with the BedrettoLab dataset. The two strategies applied to reduce the input size from 20 000 samples down to 6000 were as follows: First we used the `RandomWindow` function from SeisBench and in a second attempt we wrote our own function, which randomly picks 6000 from the original 20 000 data points. In total all three models were trained for 50 epochs using the same dataset split, optimizer, loss and learning rate as for the training with STEAD mentioned above. Unfortunately (see section V), all attempts ended up being only moderately successful.

IV. Uncertainty Estimation

Traditional deep learning models lack a representation of uncertainty, leading to often overconfident and miscalibrated predictions. Bayesian methods offer a probabil-

istic representation of uncertainty but are challenging to scale and sensitive to hyperparameter choices.

In order to obtain probabilistic estimates for P- and S-picks, we use Bayesian Model Averaging (e.g., [9], [13]). Nominally, this requires sampling from the predictive distribution—a probability distribution p, which integrates out (marginalises) the model weights, θ, for the given output y, conditioned on the current input x and the training data D:

$$p(y \mid x, D) = \int p(y \mid x, \theta) p(\theta \mid D) d\theta. \tag{1}$$

As closed forms of predictive distributions are generally unobtainable in deep learning, approximate approaches are used. Two such approaches are dropout ([2]) and Stochastic weight averaging with Gaussian (SWAG-Gaussian, SWAG) proposed in [9].

To perform Bayesian inference in practice, the predictive distribution $p(y_* \mid x_*, D) = \int p(y_* | \theta, x_*) p(\theta | D) d\theta$ of a (new) sample (x_*, y_*) is approximated through Monte Carlo sampling:

$$p(y_* \mid x_*, D) \approx \frac{1}{N} \sum_{n=1}^{N} p(y_* \mid \theta_n, x_*), \quad \theta_n \sim p(\theta \mid D). \tag{2}$$

A. Monte Carlo Dropout

Traditionally, Monte Carlo dropout randomly zeros neural-network weights during training as a form of regularisation. This idea can be adapted to Bayesian Model Averaging in the following way.

Suppose we independently zero weights during inference with probability p. Then the posterior distribution over the network weights is given by

$$q(\theta \mid \lambda) = \prod_{j=1}^{d} q_j(\theta^{(j)} \mid \lambda^{(j)}), \tag{3}$$

where d is the total number of network weights and

$$q_j(\theta^{(j)} \mid \lambda^{(j)}) = p\delta_0(\theta^{(j)}) + (1-p)\delta_{\lambda^{(j)}}(\theta^{(j)}). \tag{4}$$

Here, δ_α denotes the Dirac delta function with point mass at α and $\lambda = \lambda(D)$ are the 'normal' weights of the network. In other words, the j-th weight $\theta^{(j)}$ has a value of 0 with probability p and a value of $\lambda^{(j)}$ with probability $1-p$.

Bayesian inference is done as described above, by Monte Carlo sampling from the distribution $q(\theta \mid \lambda)$ and then approximating the predictive distribution as in (2).

In our experiments, we used a value of $p = 0.1$.

B. Stochastic Weight Averaging with Gaussian

SWAG is a scalable approximate Bayesian inference technique for deep learning.

SWAG proposes a novel approach by utilising information from the Stochastic Gradient Descent (SGD) trajectory to efficiently approximate the posterior distribution

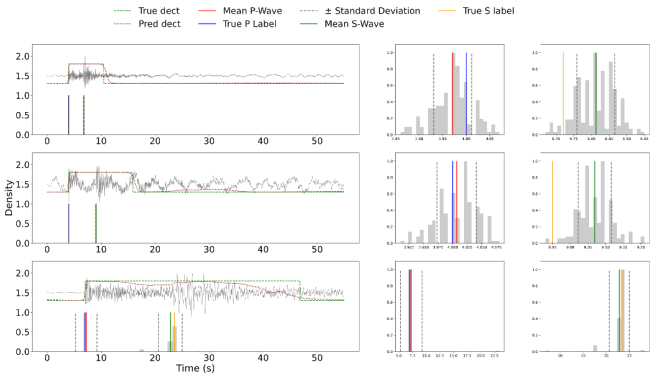

Figure 9. Example output of EQTransformer together with SWAG on three sample waveforms. Mean and standard deviation are from approximating the posterior pick distribution using 20 predictions.

over neural network weights. It employs Stochastic Weight Averaging (SWAG) to compute an average of SGD iterates with a constant learning rate, enhancing generalisation and interpreting SGD as approximate Bayesian inference. SWAG also computes a low-rank plus diagonal approximation to the covariance of the iterates, defining a Gaussian posterior approximation over neural network weights.

Motivated by the theoretical analysis of the stationary distribution of SGD iterates, SWAG leverages the geometry of the posterior distribution within the low-dimensional subspace spanned by SGD iterates. Despite non-convexity and overparameterisation, SWAG remarkably captures the geometry of the posterior. The original code used for this project can been found here [15].

After regular training, we train for an additional five epochs using the Adam optimiser and a constant learning rate of 0.01 and store the resulting set of parameters, $\{\, \theta_i : i = 1, \ldots, 5 \,\}$. These are then used to calculate a Gaussian distribution over the parameter space with mean θ_{SWA} and covariance matrix Σ given by (see also [9])

$$\begin{aligned} \theta_{\text{SWA}} &= \frac{1}{5} \sum_{i=1}^{5} \theta_i \\ \Sigma &= \frac{1}{2} \left(\operatorname{diag} \left(\frac{1}{5} \sum_{i=1}^{5} (\theta_i^2 - \theta_{\text{SWA}}) \right) \right. \\ &\quad \left. + \frac{1}{5-1} \sum_{i=1}^{5} (\theta_i - \overline{\theta}_i)(\theta_i - \overline{\theta}_i)^{\text{tr}} \right), \end{aligned}$$

where $\overline{\theta}_i$ is the running mean of the first i parameters. This corresponds to a diagonal plus low-rank[1] approximation of the covariance matrix of $\{\, \theta_i : i = 1, \ldots, 5 \,\}$.

[1] Rank five in our case

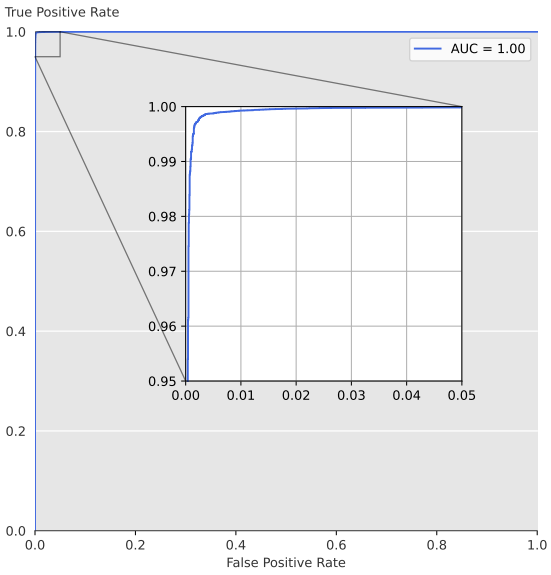

Figure 10. ROC curve of EQTransformer on STEAD.

Table I: Detection performance.

Model	Pr	Re	F1
EQTransformer (3 channels)	1.0	0.92	0.96
EQTransformer (1 channel)	1.0	1.0	1.0
EQTRedEnc	1.0	1.0	1.0
EQTNoResLSTM	1.0	1.0	1.0

Pr, Re, and F1 are precision, recall, and F1-score respectively. All models were trained on STEAD.

Table II: P-phase picking.

Model	μ	σ	MAE	MAPE	RMSE
EQTransformer (3 channels)	−0.85	6.81	0.92	0.17	6.86
EQTransformer (1 channel)	−0.02	0.75	0.08	0.01	0.75
EQTRedEnc	−0.01	0.59	0.08	0.01	0.60
EQTNoResLSTM	−0.04	1.23	0.11	0.02	1.23

μ and σ are mean and standard deviation of error (ground truth – prediction) in seconds respectively. MAE, MAPE, and RMSE are mean absolute error, mean absolute percentage error, and root mean square error respectively.

V. Results

We evaluate each architecture on an independent (from the training set) test set. As another point of comparison, we additionally evaluated the three-channel variant of EQTransformer (denoted by 'EQTransformer (3 channel)' in tables I to III) together with its original weights on the same test set. Curiously, the resulting performance is worse than either of the retrained, one-channel models considered in this study and also worse than the performance stated in the original paper ([11]). As our aim was never to perfectly reproduce said paper, we did not investigate further and merely state these findings as is.

A. Classification Performance

The classification task corresponds to the capability of the model to distinguish between a noise signal and a quake waveform. For this evaluation, we label a sample as an earthquake whenever the maximal value of the detection-output layer across the 6000 time points exceeds a threshold of 0.5. As seen in fig. 10, the detection performance of the one-channel model is good enough so that optimising the threshold further yields at best marginal improvements.

In table I the evaluation of the three different EQTransformer models can be found. It can be seen that all three models have the same performance on the classification task and that our one-channel models perform better for this specific tasks than the pretrained three channel model.

For now the classification accuracy of the model could not be assessed on the BedrettoLab data as we have been given only earthquakes signals. Noise signal would be needed to train the model for the classification task and measuring its accuracy.

B. Phase-Picking Performance

The regression task corresponds to the prediction of the correct time arrival for the P and S wave in the signal. For an earthquake signal detected as such we follow [16] in setting the P (resp. S) pick as the location of the maximal value in the P-phase (resp. S-phase) output layer. A worthwhile further point of enquiry might be to experiment with more sophisticated approaches; however, performance appears quite good already using this naïve approach.

Here we are interested to get the smallest time residual which is the difference between the true annotated time arrival and the one predicted by the model. Figure 11 shows the distribution of the residuals of the one-channel model. Note that we exclude samples with a residual of more than 1 s. On the right, we include the residuals plotted against signal-to-noise ratio. S-pick residuals show notably higher variance, which matches prior domain knowledge that locating S picks is more difficult than P picks. One might expect higher signal-to-noise ratio (so a clearer signal) to improve performance, but a bit surprisingly, we see no clear difference across varying levels of SNR.

As seen in fig. 12, the accuracies across architectures are similar for the classification task. Nevertheless, the reduced model outperforms the two other ones for the regression task. This could be explained by the complex

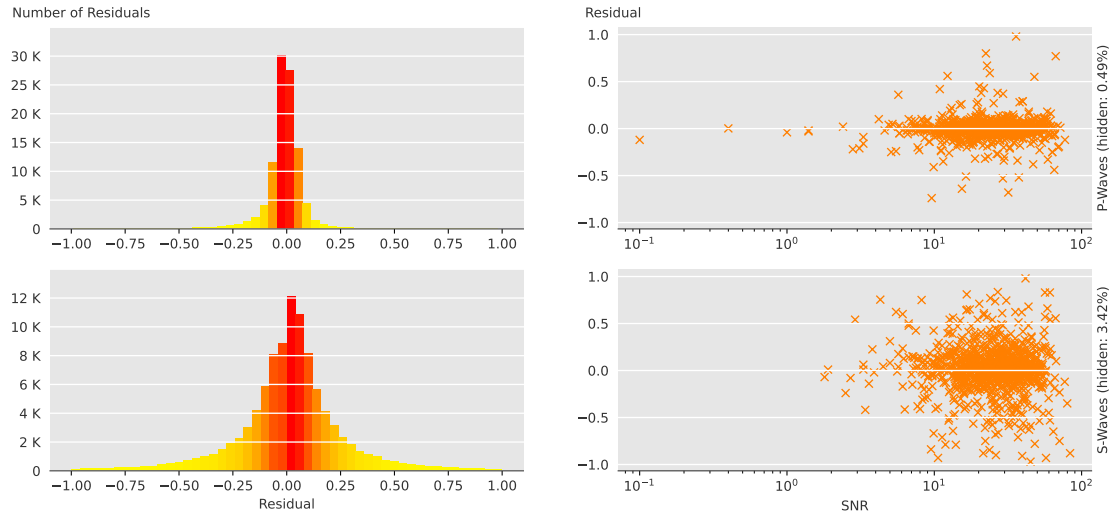

Figure 11. Residuals of P and S picks of EQTransformer

Table III: S-phase picking.

Model	μ	σ	MAE	MAPE	RMSE
EQTransformer (3 channels)	0.04	1.37	0.26	0.02	1.38
EQTransformer (1 channel)	−0.02	1.28	0.03	0.02	1.28
EQTRedEnc	0.00	1.33	0.03	0.02	1.34
EQTNoResLSTM	−0.04	1.93	0.39	0.03	1.93

μ and σ are mean and standard deviation of error (ground truth – prediction) in seconds respectively. MAE, MAPE and, RMSE are mean absolute error, mean absolute percentage error, and root mean square error respectively.

architecture of the encoder, to result in overfitting for the relatively simple task the model has to perform.

The one-channel model arguably performs quite well. This is rather remarkable, considering it only uses a third of the theoretically available seismic data (one of three components) during training and evaluation and the fact that we did not perform any sophisticated hyperparameter selection.

Reducing the size of the encoder generally improves performance, or keeps it constant, except for an increase in the standard deviation of the S-picking task. This suggests that the original EQTransformer architecture might be unnecessarily involved and complicated for the task at hand.

Finally, simplifying the architecture further by removing all CNN and lstm layers degrades performance to a level below the original one. This indicates that such a drastic reduction might be 'too much of a good thing'.

VI. Explanation of the code

To ease further development of our code, we briefly outline the structure of the repository (found at [5]) as seen in fig. 13.

The primary training script is `code/swag.py`, which allows for training each of the mentioned models on a specified dataset. Optionally, the script builds a 'SWAG-variant' of the model as well. All models are by default saved to disk on a regular interval, making them available for further training or evaluation later on. Model training is configurable via a plethora of command-line arguments. The file `code/swag.sh` contains an example usage.

The second script, `code/evaluate_model.py`, is used to calculate performance metrics of a trained model.

Both scripts expect the data used to be in SeisBench-compatible format. In particular, *train*, *dev*, and *test* splits should be defined.

The `code/utils/augmentations.py` file defines classes to perform data augmentations which are compatible with the SeisBench-preprocessing pipeline.

The code in `code/utils/evaluation.py` calculates performance metrices and is used in the `evaluate_model.py` script. Notably, predictions across the entire evaluation dataset are temporarily stored in memory. However, we never ran into issues using the *test* split of STEAD; so, this seems only a theoretical limitation of the code.

In `code/utils/visualisations.py` we collect functions used to produce the plots in this document.

Finally, `code/utils/utils.py` contains utility functions used all across the code base, for example, for training a single epoch or for calculating predictions of a model on a dataset.

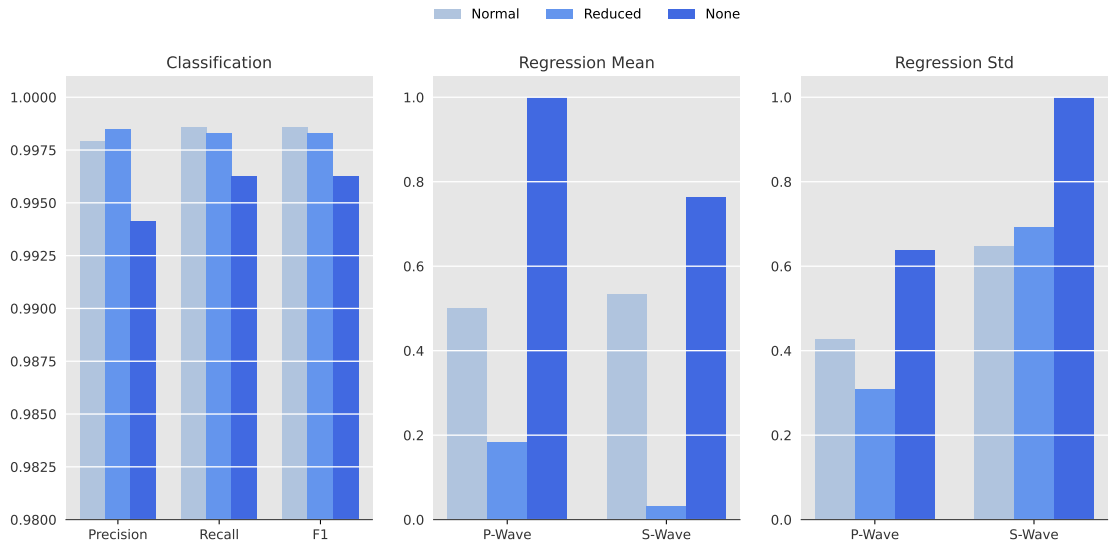

Figure 12. Comparison plot of the three models. Regression mean and regression standard deviation are rescaled such that the worst performance is 1.0.

Table IV: P-phase picking for the BedrettoLab dataset.

Model	μ	σ	MAE	MAPE	RMSE
Train from scratch	2.99	1030.71	160.46	0.02	1030.72
Retrain with self-implemented subsampling	−2.60	268.34	44.64	0.01	268.35
Retrain with SeisBench subsampling	7111.99	2214.38	7111.99	0.71	7448.75

μ and σ are mean and standard deviation of error (ground truth – prediction) in 2×10^{-5} s respectively. MAE, MAPE, and RMSE are mean absolute error, mean absolute percentage error, and root mean square error respectively. The unusual unit is due to the sampling rate of 200 000 Hz of the Bedretto data.

Table V: S-phase picking for the BedrettoLab dataset.

Model	μ	σ	MAE	MAPE	RMSE
Train from scratch	−4456.87	4628.21	4698.97	0.47	6424.56
Retrain with self-implemented subsampling	−330.72	390.34	379.00	0.13	511.61
Retrain with SeisBench subsampling	7766.97	2080.32	7766.97	0.78	8040.74

μ and σ are mean and standard deviation of error (ground truth – prediction) in 2×10^{-5} s respectively. MAE, MAPE, and RMSE are mean absolute error, mean absolute percentage error, and root mean square error respectively. The unusual unit is due to the sampling rate of 200 000 Hz of the Bedretto data.

VII. Discussion

Our seismic analysis solution draws strength from the widely used SeisBench library, providing a robust foundation for comparison with existing methodologies. Incorporating SWAG for model uncertainty estimation enhances the reliability of predictions, offering straightforward standard deviation values. The emphasis on code reusability enables easy adaptation to future available data, facilitating following work.

However, the complex architecture with numerous parameters, requires substantial computational resources and expensive training. Additionally, the lack of a real evaluation regarding the advantages of pretraining on a larger data corpus limits the comprehensive understanding the model's potential.

In conclusion, while our approach exhibits strengths in leveraging SeisBench, uncertainty estimation, and code reusability, addressing the complex architecture and exploring the benefits of pretraining are crucial for refining our model's efficiency and applicability in seismic analysis.

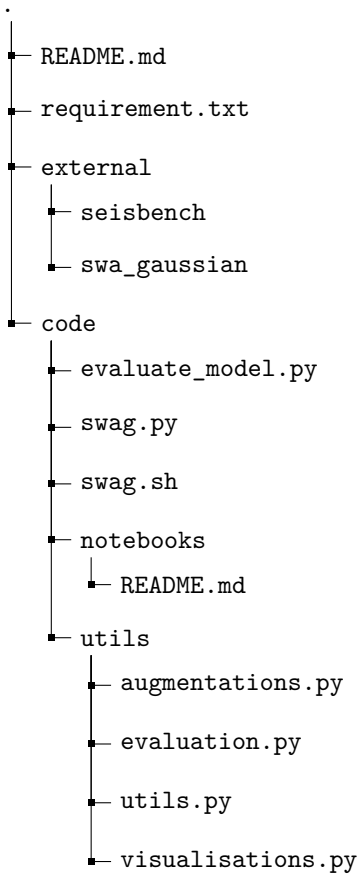

Figure 13. Directory structure.

VIII. Summary

Navigating large dataset, comprehension of the data and its metadata, understanding of previous research and training of large deep learning models presented significant challenges. Nevertheless, tangible results have been achieved. State-of-the-art EQTransformer model have been adapted and trained achieving strong performance on STEAD and promising initial results on Bedretto Lab. Multiple variations of the model architecture have been tested in the goal to make the model simpler. Additionally, SWAG could be implemented on top of the model to estimate the uncertainty of the predictions and make it available for down stream tasks. Finally, a crucial focus of the project has been to ensure code reusability and full compatibility with the widely used SeisBench library. This initiative aims to facilitate seamless integration and maximise the utility of the codebase within the established framework of SeisBench, promoting ease of use and collaboration across related projects.

IX. Outlook

Due to time constraints, some parts of the original project goals could not be accomplished. Below, we suggest a short list of potential further steps.

As shown above in section III-B, the complex encoder of the EQTransformer model seems to be too complex for the task it tries to solve and that the accuracy can be improved by removing certain layers in the encoder. But removing too much of the encoder decreases performance thus with careful consideration which layers are needed, we think that the performance of the model could be further improved. We think that the model could even profit from a slight reduction of encoder / decoder structure by simplifying the convolutional layer stack.

As mentioned before, the BedrettoLab dataset that was provided to us consisted of a limited number of samples. This was a limiting factor in our approaches to train a model that can meaningfully predict where the P and S picks occur. As this dataset is actively worked on, there should be a larger dataset available in the near future to which our pipeline can be applied. To increase the accuracy of the classification task, we think that some noise should be added to the BedrettoLab dataset, to also have some negative samples to help the model recognise seismic windows that neither contain a P- nor an S-pick. Additionally, due to the limiting time we had with this dataset, we had some difficulties adjusting our pipeline, such that the our model, which was trained on windows of length 6000 samples, works properly on windows with 20 000 samples.

In [1], the authors successfully apply MultiSWAG ([14]) to approximate the posterior distribution of their phase pickers. They found that it slightly increases performance compared to traditional SWAG and enables a more robust posterior approximation.

As deep-learning models tend to be overconfident in their predictions (see e.g., [3]), one generally calibrates the uncertainties estimations before using them for downstream tasks. Calibrated in this context means that a 90 %-credible interval for the location of, say, the P pick, outputted by the Bayesian model actually contains the true location 90 % of the time. One simple method for uncertainty calibration using isotonic regression was developed in [7].

References

[1] Alysha D Armstrong et al. 'A Deep-Learning Phase Picker with Calibrated Bayesian-Derived Uncertainties for Earthquakes in the Yellowstone Volcanic Region'. In: *Bulletin of the Seismological Society of America* 113.6 (2023), pp. 2323–2344.

[2] Yarin Gal and Zoubin Ghahramani. 'Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning'. In: (2015). URL: https://doi.org/10.48550/arXiv.1506.02142.

[3] Chuan Guo et al. 'On calibration of modern neural networks'. In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.

[4] Charles R. Harris et al. 'Array programming with NumPy'. In: *Nature* 585 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[5] Herwig Höhenberger, Tobias Rahn and Yanis Tournier. *Deep Learning Earthquake Monitoring*. Version 1.0.0. Dec. 2023. URL: https://github.com/ib31iat/Earthquake_Monitoring.

[6] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[7] Volodymyr Kuleshov, Nathan Fenner and Stefano Ermon. 'Accurate uncertainties for deep learning using calibrated regression'. In: *International conference on machine learning*. PMLR. 2018, pp. 2796–2804.

[8] Baer M. and Kradolfer U. 'An automatic phase picker for local and teleseismic events'. In: (1987). URL: https://doi.org/10.1785/bssa0770041437.

[9] Wesley Maddox et al. 'A Simple Baseline for Bayesian Uncertainty in Deep Learning'. In: (2019). URL: https://doi.org/10.48550/arXiv.1902.02476.

[10] S Mostafa Mousavi et al. 'STanford EArthquake Dataset (STEAD): A Global Data Set of Seismic Signals for AI'. In: *IEEE Access* (2019).

[11] S. Mostafa Mousavi and William L. Ellsworth. 'Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking'. In: (2020). URL: https://www.nature.com/articles/s41467-020-17591-w.

[12] Jannes Münchmeyer, Jack Woollam and Andreas Rietbrock. 'Which Picker Fits My Data? A Quantitative Evaluation of Deep Learning Based Seismic Pickers'. In: (2021). URL: https://doi.%20org/10.1029/2021JB023499.

[13] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[14] Andrew G Wilson and Pavel Izmailov. 'Bayesian deep learning and a probabilistic perspective of generalization'. In: *Advances in neural information processing systems* 33 (2020), pp. 4697–4708.

[15] wjmaddox. *github repository for SWAG original code*. 2019. URL: https://github.com/wjmaddox/swa_gaussian.

[16] Jack Woollam and Jannes Münchmeyer. 'SeisBench—A Toolbox for Machine Learning in Seismology'. In: (2022). URL: https://doi.org/10.1785/0220210324.

[17] Weiqiang Zhu and Gregory C Beroza. 'PhaseNet: a deep-neural-network-based seismic arrival-time picking method'. In: (2019). URL: https://doi.org/10.1093/gji/ggy423.