# **Earthquake AI**
*Handover Report*

Predicting the maximum magnitude of earthquakes in the next 30 days in California

**Maximilian Knuth**
`mknuth@mit.edu`

**Caio Iglesias**
`caiopigl@mit.edu`

Date
Aug 13, 2025

## 1 Introduction

This document is a partial report containing a subset of experiments and analyses conducted to date. It is not a final report, and its primary goal is to facilitate the handover of the project by documenting the current methodology, preliminary findings, and implementation details.

The objective of this project is to forecast the **maximum earthquake magnitude in the Los Angeles region over the next 30 days** using a hybrid modeling framework. The methodology integrates two complementary approaches: (i) a *seismic waveform–based model* that leverages continuous waveform recordings and deep representation learning, and (ii) an *event-based probabilistic model* grounded in the Gutenberg–Richter relation and historical seismicity rates. The final prediction system fuses features from both approaches to enhance predictive robustness and accuracy.

Figure 1 summarizes the overall workflow. The **top row** illustrates the seismic waveform pipeline: 10 days of continuous three-component waveforms (BHZ, BHN, BHE) are divided into short overlapping windows, processed into embeddings using the SeisLM foundation model, and temporally aggregated via an LSTM network to produce a compact sequence representation. The **bottom row** depicts the tabular feature pipeline: earthquake catalog data is filtered by station, enriched with rolling seismological features (e.g., $b$-values, ETAS intensities, energy- and time-based measures), and transformed into daily station-level features. In both cases, the resulting representations are processed with class-imbalance handling and fed into a classifier that predicts the discrete magnitude class (0–2 low, 2–4 medium, 4+ high) expected within the next 30 days. The hybrid architecture allows joint or separate use of these data sources, enabling the fusion of raw waveform patterns with interpretable seismological indicators.

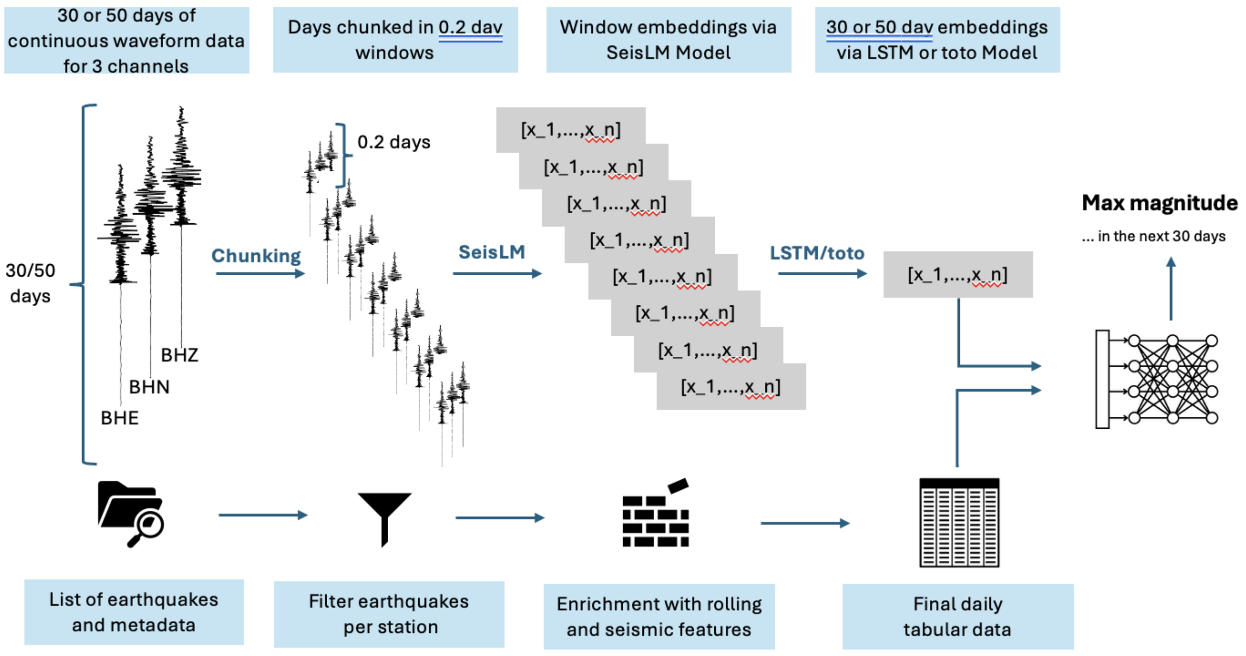

30 or 50 days of continuous waveform data for 3 channels
Days chunked in 0.2 dav windows
Window embeddings via SeisLM Model
30 or 50 dav embeddings via LSTM or toto Model
30/50 days
BHZ
BHN
BHE
0.2 days
Chunking
SeisLM
[x_1,...,x_n]
LSTM/toto
[x_1,...,x_n]
Max magnitude
... in the next 30 days
List of earthquakes and metadata
Filter earthquakes per station
Enrichment with rolling and seismic features
Final daily tabular data

**Figure 1:** Overall methodology for earthquake magnitude forecasting. **Top:** Seismic waveform pipeline using SeisLM embeddings and LSTM or toto temporal aggregation. **Bottom:** Engineered seismological feature pipeline from earthquake catalogs. Both pipelines feed into a classification stage that predicts the magnitude class within the next 30 days.

### 1.1 Background on Seismic Waveform–Driven Prediction

Seismic waveform–driven prediction leverages the continuous ground motion recordings captured by seismic networks to infer the likelihood and potential magnitude of future earthquakes. Unlike catalog-based methods, which rely solely on discrete event parameters (e.g., time, location, magnitude), waveform-based approaches retain the full temporal and spectral complexity of the seismic signal, enabling the extraction of subtle precursory patterns that may precede large events.

Recent advances in deep learning and foundation model architectures have substantially expanded the potential of waveform-based analysis. The Seismic Language Model (SeisLM) Liu et al. (2024) exemplifies this shift, introducing a self-supervised, transformer-based foundation model trained on vast volumes of unlabeled seismic waveforms. By learning general-purpose seismic representations, SeisLM achieves strong performance across diverse downstream tasks—such as event detection, phase picking, and foreshock–aftershock classification—while requiring minimal labeled data. Its embeddings capture discriminative features separating noise from earthquake signals, highlighting the feasibility of reusing such pretrained representations for predictive modeling.

Other research has explored the direct forecasting of seismic activity using deep neural networks applied to spatial and temporal features derived from seismic signals. For example, Zhang et al. Zhang et al. (2025) employed fully convolutional networks (FCNs) to predict future earthquake occurrence probabilities in California based on maps of past released seismic energy. Their approach achieved performance comparable to the established Epidemic-Type Aftershock Sequence (ETAS) model while offering substantial gains in computational efficiency. Similarly, Mao et al. (2024) combined seismic parameters extracted from catalog data with signal decomposition techniques such as Variational Mode Decomposition (VMD), enabling deep learning models to predict monthly maximum magnitudes in high-seismicity regions.

Waveform-based learning has also been extended to multi-station settings. Kim et al. (2022) introduced a hybrid convolutional and graph convolutional network (GCN) framework to aggregate features from multiple seismic stations, improving event classification accuracy and reducing false alarms when working with long-duration continuous data. Large-scale datasets such as the Stanford Earthquake Dataset (STEAD) Mousavi et al. (2019) have been instrumental in enabling these models, providing millions of labeled waveform segments for both earthquake and non-earthquake classes.

In the Los Angeles context, Yavas et al. Yavas et al. (2024) demonstrated that advanced machine learning architectures, including ensemble models and feature-rich neural networks, can achieve up to 97.97% accuracy in predicting the maximum earthquake category within the next 30 days (we believe this is highly suspicious, and an examination of their dataset suggests the presence of data leakage).

Collectively, these studies highlight three converging trends: (i) the move towards *foundation models* capable of generalizing across regions and tasks, (ii) the exploitation of large, high-quality waveform datasets for training data-hungry models, and (iii) the integration of spatio-temporal and spectral signal attributes to enhance predictive skill. These advances form the scientific basis for the waveform-based component of this project, which employs SeisLM embeddings as a compact yet information-rich representation of continuous seismic activity in the Los Angeles area, subsequently used for maximum magnitude forecasting.

## 2 Data Sources and Preprocessing

### 2.1 Seismic Waveform Data

The dataset used in this work consists of continuous broadband seismic recordings obtained from the **Southern California Earthquake Data Center (SCEDC)** public data repository, hosted on the Amazon Web Services (AWS) Public Dataset Program.[1] Data were accessed via the `scedc-pds` Amazon S3 bucket using unsigned public access, eliminating the need for AWS credentials.

#### Acquisition Procedure

Daily waveform files were retrieved for over 25 broadband seismic stations in Southern California, each equipped with three orthogonal components:

[1]`https://scedc.caltech.edu/data/aws.html`

- `BHZ` – Vertical component
- `BHE` – East–West horizontal component
- `BHN` – North–South horizontal component

Data acquisition was performed by iterating over each day of the target year for all selected stations and channels. The retrieval script downloaded daily MiniSEED (`.mseed`) waveform files directly from the AWS S3 object paths:

```
continuous_waveforms/{year}/{year}_{jday}/CI{station}__{channel}___{year}{jday}.ms
```

where `jday` denotes the Julian day of the year.

**Data Completeness and Selection**

Data quality varied considerably across stations and years, with some stations exhibiting frequent outages, instrument downtime, or archival gaps. To ensure temporal consistency and avoid introducing bias from missing days, only station–year combinations with *complete daily coverage* across all three channels were retained. This filtering step reduced the number of stations used in the final dataset but ensured that all retained data represented uninterrupted continuous recordings for the full year. A table of the collected data can be found in Appendix 5.

### 2.1.1 Data Preparation (Waveform Preprocessing)

The daily MiniSEED files downloaded from SCEDC often contained multiple traces for the same day and channel, and in some cases included temporal gaps. Multiple traces for a single day can occur when the recording is split into segments due to routine station maintenance, network transmission limits, or temporary outages. Similarly, gaps within a single trace may appear if the station temporarily stopped recording, experienced telemetry issues, or if certain intervals failed to be archived.

To create a single, continuous 24-hour record for each station, channel, and day, every file was processed as follows. First, the file was read using ObsPy, and the number of traces and their start and end times were inspected. If multiple traces were present, they were merged into a single continuous trace using ObsPy's `merge` function, filling any gaps with zeros. The duration of the merged trace was then checked and logged in hours and minutes, with the target duration being exactly 24 hours.

After merging, the processed trace was saved as a new MiniSEED file in the same directory, with "_processed" appended to the filename to preserve both the raw and processed versions. A log file was maintained for each station and year, recording all processed files, their time coverage, and any errors encountered. Files already containing "_processed" in their names were skipped on subsequent runs to avoid redundant processing.

**Rationale for zero-filling missing intervals.** Zero-filling was chosen over interpolation for handling missing data to ensure that no artificial seismic signal was introduced into the dataset. Interpolation could produce unrealistic waveform segments that might be interpreted by the model as genuine seismic activity, potentially biasing downstream feature extraction and prediction. Zero values act as explicit markers of missing information while preserving the fixed-length input requirement for representation learning. This makes zero-filled gaps both detectable and safe for machine learning pipelines that expect consistent time series dimensions.

This procedure ensured that all data used for feature extraction were in a consistent format, with exactly one continuous trace per day and channel, making them directly usable for downstream modeling.

### 2.1.2 Combining Daily Waveforms into Multi-Day Streams

After producing single-day, continuous, zero-filled traces for each {station, channel, day}, the next step was to merge these daily waveforms into longer multi-day streams. This was necessary because the downstream feature extraction and modeling benefited from capturing temporal patterns spanning multiple days, rather than being limited to 24-hour intervals.

The merging procedure was implemented as follows. For each {station, year, channel} triplet, all "_processed" daily MiniSEED files were collected and sorted chronologically by date. A sliding window approach was then applied to determine which files to merge together. The window length (`window_len`) defined the number of consecutive days to include in each merged stream, while an optional shift parameter (`shift`) controlled the advance between successive windows:

- `shift` = `window_len` produces non-overlapping multi-day segments.
- `shift` < `window_len` yields overlapping (sliding) windows.
- `shift` > `window_len` creates gapped segments.

Before merging, the procedure standardized the *location code* across all traces in the current window. Location code inconsistencies can occur when the same physical sensor is reported under different codes due to changes in station configuration, network metadata updates, or temporary fallback telemetry. For each window, the most common location code was identified, and all traces were reassigned to this dominant code.

Once location codes were unified, all traces in the window were merged into a single continuous stream. Interpolation was chosen here instead of zero-filling because the gaps were typically short within multi-day sequences, and interpolation preserved smoothness for certain time-domain features while avoiding artificial discontinuities. The merged multi-day stream was then written to disk in MiniSEED format, with filenames indicating the station, channel, and date range (e.g., `STATION_BHZ_YYYY-MM-DD_to_YYYY-MM-DD.mseed`).

This process produced a set of consistent, location-harmonized, multi-day waveform segments for each station and channel, with flexible control over the temporal coverage via the window length and shift parameters. These files served as the direct input for the subsequent embedding extraction stage.

## 2.2 Engineered Seismological Feature Data

In addition to the continuous waveform dataset, this project also makes use of a structured, event-based dataset derived from historical earthquake catalogs and station metadata. These records are obtained from the **Southern California Earthquake Data Center (SCEDC)** as the primary source, with the *Incorporated Research Institutions for Seismology* (IRIS) database serving as a fallback to ensure completeness. The dataset covers the Caltech (`CI`) seismic network over the period `2020--01--01` to `2025--01--01`, and includes all earthquakes with magnitude $M \geq 1.0$ located within a 50 km radius of each station.

#### Station Coverage

The catalog spans **50 broadband and short-period seismic stations** across Southern California, including high-priority stations such as PAS, BSC, MAN, MAG, and ABL, as well as a wide distribution of regional stations to ensure spatial coverage of the Los Angeles region. Each station record includes precise coordinates, elevation, and network affiliation, enabling distance-based filtering of events during processing.

#### Initial Data Preparation

Raw earthquake events are retrieved via the FDSN service in **30-day chunks** to avoid network timeouts and to maintain operational reliability when downloading large datasets. For each chunk, the pipeline:

1. Downloads event origin time, hypocenter coordinates, depth, magnitude, magnitude type, and event type from the SCEDC or IRIS catalog.
2. Retrieves associated station metadata (coordinates, elevation) for all target stations.
3. Applies a 50 km proximity filter based on a circle (Haversine) distance to ensure only local, relevant events are retained.
4. Replicates each earthquake record for every station within the defined radius, producing *station-specific event lists.*

The output of this stage is stored in `Parquet` format for efficient storage and downstream processing. Each row in the raw dataset corresponds to a unique {station, event} pair and includes:

- Earthquake metadata (time, location, depth, magnitude, magnitude type, event type)
- Station metadata (station code, coordinates, elevation)
- Calculated epicentral distance between the station and event

At this point in the pipeline, the dataset remains in an event-based format, ready for later aggregation into daily station-level time series for modeling.

## 3 Modeling and Feature Engineering

### 3.1 Problem Formulation

The objective of this modeling stage is to predict the **maximum earthquake magnitude** expected in the Los Angeles region within the next 30 days, using only continuous seismic waveform data as input. This is framed as a *multi-class classification* problem, where each sample corresponds to a fixed observation period for a given seismic station.

Ground-truth labels are derived from the Southern California Seismic Network (SCSN) earthquake catalog. For each observation period end date, the subsequent 30 days are examined, and the maximum magnitude recorded within that window is identified. This magnitude is then mapped into discrete classes via the following binning function:

$$\text{class} = \begin{cases} 0, & M < 1.0 \\ 1, & 1.0 \le M < 2.0 \\ 2, & 2.0 \le M < 3.0 \\ 3, & 3.0 \le M < 4.0 \\ 4, & 4.0 \le M < 5.0 \\ 5, & 5.0 \le M < 6.0 \\ 6, & 6.0 \le M < 7.0 \\ 7, & 7.0 \le M < 8.0 \\ 8, & M \ge 8.0 \end{cases}$$

where $M$ denotes the maximum magnitude in the target horizon.

This formulation allows the model to learn discriminative patterns in the preceding continuous seismic waveforms that are predictive of the likelihood of higher-magnitude events in the upcoming month. By structuring the problem as classification rather than regression, the approach mitigates the sensitivity to small magnitude estimation errors and aligns directly with discrete magnitude thresholding commonly used in operational earthquake forecasting.

**Class Imbalance Considerations** A notable challenge in our modeling setup is the pronounced class imbalance in the target variable: the vast majority of samples correspond to low-magnitude events, while medium- and especially high-magnitude classes are rare. This skew limits the model's ability to learn discriminative patterns for the less frequent classes and often biases predictions toward the majority category. We have experimented with a range of mitigation strategies, including sample matching across magnitude levels, class-weighted loss functions, and traditional resampling techniques (over- and under-sampling). While these efforts yielded some localized improvements, we have not yet converged on a consistently effective solution, making class imbalance an open issue for future work.

## 3.2 Seismic waveform modeling and feature engineering

### 3.2.1 Inputs and Windowing Pipeline

The model operates on processed, continuous MiniSEED files representing multi-day waveform segments for a specific seismic station and channel. These files are produced through the preprocessing and merging pipeline described in Section 2.1, ensuring that all inputs are gap-free (or zero-filled) and consistently formatted.

**Input sources.** Each sample originates from a single {station, start date} pair, with three orthogonal components:

- `BHZ` – vertical component,
- `BHE` – east–west horizontal component,
- `BHN` – north–south horizontal component.

For every selected observation period, the corresponding waveform files are read into memory and processed jointly.

**Downsampling.** To reduce computational load while retaining the relevant frequency content, each waveform is *decimated* by a fixed factor (default: 16). This step lowers the effective sample rate and shortens the number of time steps per window, which significantly decreases the memory footprint during batch processing.

**Per-trace normalization.** Each channel trace is normalized independently to zero mean and unit variance:

$$x' = \frac{x - \mu}{\sigma},$$

where $\mu$ and $\sigma$ are computed per trace over the full observation period. This ensures scale invariance and mitigates the influence of absolute amplitude variations between stations or days.

**Window segmentation.** After normalization, the multi-day input is split into contiguous, non-overlapping *windows* of fixed temporal length (`window_size_days`). Let $f_s'$ denote the downsampled sample rate and $D$ the window duration in days; each window contains

$$N_{\text{samples}} = D \times 24 \times 3600 \times f_s'$$

time steps per channel. Windowing serves two purposes:

1. It constrains the input size for the downstream embedding model (SeisLM), ensuring consistent tensor dimensions across samples.
2. It enables the model to learn from multiple temporal slices within a single observation period, capturing evolving seismic patterns.

**Tensor construction.** The final input tensor for a given sample has shape

$$[T, C, N_{\text{samples}}],$$

where $T$ is the number of windows extracted from the multi-day segment, $C = 3$ is the number of channels, and $N_{\text{samples}}$ is the per-channel window length. Each tensor is returned alongside its class label, as well as metadata identifying the station and the start/end time of the observation period.

### 3.2.2 Representation Learning with SeisLM

To transform raw waveform segments into compact, informative representations, each input window is processed through the pretrained *SeisLM* model Liu et al. (2024). SeisLM is a transformer-based foundation model trained in a self-supervised manner on a large corpus of global seismic waveform data. It is designed to learn general-purpose seismic features that are transferable to a variety of downstream tasks, including event detection, phase picking, and, in this case, earthquake magnitude forecasting.

**Frozen backbone.** In the present setup, the SeisLM backbone parameters are *frozen*, meaning they are not updated during training. This decision is motivated by two factors:

1. The pretrained model has already learned rich, general seismic representations from extensive training on diverse datasets, making full fine-tuning unnecessary for the available training set size.
2. Freezing the backbone drastically reduces memory usage and computation time, allowing the workflow to scale to longer input sequences and larger batch sizes. Typically, I ran into out of memory issues when trying to fine tune the backbone.

**Embedding extraction.** Each waveform window is passed through SeisLM, which outputs a sequence of projected hidden states over time. Denoting the hidden state sequence for a given window as $\mathbf{H} \in \mathbb{R}^{L \times d}$, where $L$ is the sequence length and $d$ is the embedding dimension (256 in this configuration), a temporal mean is applied:

$$\mathbf{h}_{\text{win}} = \frac{1}{L} \sum_{t=1}^{L} \mathbf{H}_t.$$

The result is a fixed-dimensional embedding vector $\mathbf{h}_{\text{win}} \in \mathbb{R}^{256}$ for each window, summarizing the seismic content of that time span.

**Sequence of embeddings.** For an observation period segmented into $T$ windows, this process yields a sequence of embeddings:

$$\mathbf{E} = \left[\mathbf{h}_{\text{win}}^{(1)}, \mathbf{h}_{\text{win}}^{(2)}, \ldots, \mathbf{h}_{\text{win}}^{(T)}\right] \in \mathbb{R}^{T \times 256}.$$

These embeddings retain temporal ordering and serve as the input to the aggregation stage described in Section 3.2.3, where information across windows is combined into a single feature vector for classification.

### 3.2.3 Temporal Aggregation Across Windows

The sequence of fixed-dimensional embeddings produced by SeisLM for each observation period retains temporal ordering across windows. While each embedding $\mathbf{h}_{\text{win}}^{(t)} \in \mathbb{R}^{256}$ summarizes the seismic activity within a single window, the predictive signal for upcoming earthquake magnitude may depend on *patterns over time*, such as gradual changes in waveform statistics or sustained anomalous activity.

To capture these temporal dependencies, the embedding sequence

$$\mathbf{E} = \left[\mathbf{h}_{\text{win}}^{(1)}, \mathbf{h}_{\text{win}}^{(2)}, \ldots, \mathbf{h}_{\text{win}}^{(T)}\right] \in \mathbb{R}^{T\times 256}$$

is passed to a recurrent neural network (RNN) layer. In this configuration, a **Long Short-Term Memory (LSTM)** network processes the sequence in order, producing a hidden state $\mathbf{s}_t$ for each time step:

$$\mathbf{s}_t, \mathbf{c}_t = \text{LSTM}(\mathbf{h}_{\text{win}}^{(t)}, \mathbf{s}_{t-1}, \mathbf{c}_{t-1}),$$

where $\mathbf{c}_t$ denotes the LSTM cell state. The LSTM is capable of modeling both short- and long-range temporal relationships, making it well-suited for sequences where relevant patterns may span multiple windows.

**Attention pooling.** Rather than relying solely on the final LSTM hidden state, an *attention pooling* mechanism is applied to learn a weighted average of all time-step outputs:

$$\alpha_t = \frac{\exp(\mathbf{w}^\top \tanh(\mathbf{W}_a \mathbf{s}_t))}{\sum_{t'=1}^{T} \exp(\mathbf{w}^\top \tanh(\mathbf{W}_a \mathbf{s}_{t'}))},$$

$$\mathbf{z} = \sum_{t=1}^{T} \alpha_t \mathbf{s}_t,$$

where $\alpha_t$ is the learned attention weight for time step $t$, $\mathbf{W}_a$ and $\mathbf{w}$ are trainable parameters, and $\mathbf{z} \in \mathbb{R}^{256}$ is the aggregated representation. This allows the model to focus on the most informative segments of the observation period, regardless of their position in the sequence.

**Alternative: Toto model head.** As an alternative to the LSTM-based aggregation, the architecture supports the use of the **Toto** model head, built on Datadog's Time Series Foundation Model Cohen et al. (2024). Toto is a transformer-based backbone pretrained on a wide variety of multivariate time series, designed to extract semantically rich representations from sequential data. Its design allows it to model long-range temporal dependencies and cross-channel relationships in a single, unified architecture.

In this configuration, the sequence of window-level embeddings is passed to the `TotoHead` module. The head first loads a pretrained Toto backbone (e.g., `Toto-Open-Base-1.0`) and adapts the input dimension via a linear projection. The model supports multiple fine-tuning regimes:

- *Head*: the backbone is frozen, and only the classification layers are trained.
- *Finetune*: all backbone parameters are updated during training.
- *Partial*: all layers are frozen except for the last $k$ transformer blocks, which are unfrozen to allow partial adaptation while keeping most of the pretrained parameters fixed.

Before feeding the sequence into the backbone, the embeddings are reshaped into a channel-first layout $[B, C, T]$, padded to a multiple of the backbone's patch size, and accompanied by padding and identifier masks. The backbone then processes the sequence into *patch-level embeddings*, which are averaged over the temporal dimension to produce a fixed-size vector.

Unlike the LSTM+attention approach, which processes the sequence in order and learns an attention distribution over time, Toto's transformer backbone applies self-attention globally across all positions in the sequence, enabling it to directly model long-range interactions between any two windows. This global receptive field, combined with its pretrained initialization, makes the Toto head particularly well-suited for tasks where temporal dependencies are not strictly sequential or where patterns repeat over extended horizons.

In both the LSTM+attention and Toto configurations, the output of the aggregator is a single fixed-size vector $\mathbf{z}$ that is passed to the classifier head together with the tabular features to produce the final magnitude classification.

## 3.3 Tabular feature engineering

The tabular feature set is derived from the event-based earthquake catalog by transforming it into a daily, station-specific time series enriched with physically interpretable seismological variables. The objective is to encode both the short-term and long-term temporal patterns, spatial proximity, and magnitude-dependent behaviors of regional seismicity in a form suitable for machine learning models.

### Event Aggregation and Temporal Resolution

Earthquake occurrences are first grouped by *date* and *station* to produce a single record per day per station. For each daily group, the following fundamental statistics are computed:

- **Daily maximum magnitude** $M_{\max,d}$
- **Daily minimum magnitude** $M_{\min,d}$
- **Daily mean magnitude** $\overline{M}_d$
- **Total count of events** $N_d$

These features capture the baseline seismicity rate and the range of magnitudes on a given day.

### Magnitude Standardisation and Energy Scaling

To ensure comparability across stations and catalogs, all magnitudes are converted to the *Local Magnitude* (ML) scale (some of them are not trivially convertable, so we just mantain the same value). For each event, the associated seismic energy $E$ is estimated using the Gutenberg–Richter relationship:

$$E = 10^{(11.8+1.5M_{\mathrm{L}})}$$

and subsequently square-root transformed, $\sqrt{E}$, to reduce the extreme dynamic range while retaining proportionality.

### Recurrence Interval Features

For each magnitude class $c$, the *time since last event* is calculated as:

$$\mathrm{TSLE}_c(d) = d - \max\{\tau < d \mid \exists\, M \in \text{class } c\}$$

This metric captures magnitude-dependent recurrence behavior, where small events are expected to recur more frequently than large ones.

### Frequency–Magnitude Relationship (B-value)

The Gutenberg–Richter law,

$$\log_{10} N(M) = a - bM,$$

describes the statistical relationship between earthquake frequency and magnitude. The slope $b$ is estimated over a rolling time window (e.g., 30 days) for each station, providing a local indicator of stress state and seismicity regime.

### Depth and Energy Distribution Features

Daily distributions of hypocentral depths and event energies are summarised via:

$$\begin{aligned}\overline{D}_d &= \text{mean depth on day } d,\\ \Delta D_d &= D_{\max,d} - D_{\min,d},\\ E_{\text{tot},d} &= \sum_{i=1}^{N_d} \sqrt{E_i},\\ \overline{E}_d &= \frac{E_{\text{tot},d}}{N_d}.\end{aligned}$$

Depth range $\Delta D_d$ reflects the vertical extent of seismicity, while $E_{\text{tot},d}$ and $\overline{E}_d$ capture total and average daily energy release.

### Advanced Seismological Indicators

Three advanced temporal–statistical descriptors are computed over moving windows of recent events:

- **T-value**: the elapsed time spanned by the last $k$ events, inversely related to seismicity rate.
- $dE_{1/2}$: the normalised energy release rate, sensitive to changes in stress release.
- **Magnitude deficit**: the difference between the observed $M_{\max}$ and that predicted from the Gutenberg–Richter law for the window, an indicator of potential stress accumulation.

### ETAS-based Aftershock Intensity

We also compute a daily aftershock intensity proxy based on the Epidemic-Type Aftershock Sequence (ETAS) model:

$$\lambda_i = \mu + \sum_{j<i} K\, 10^{\alpha(M_j - M_0)}\, (t_i - t_j + c)^{-p}\, \exp\left(-\frac{\text{dist}_{ij}^2}{2\sigma^2}\right),$$

where $\lambda_i$ is the conditional intensity for day $i$ given all prior events $j$. This term captures both temporal clustering and spatial decay of aftershock triggering.

Overall, this feature engineering strategy integrates *physical seismology* (energy release, recurrence intervals, frequency–magnitude statistics) with *statistical forecasting tools* (rolling windows, ETAS-based clustering) to yield a rich, interpretable tabular dataset for earthquake magnitude prediction.

## 3.4 Final model

The final stage of the framework integrates the two data modalities produced in earlier steps: waveform-based features learned from continuous seismic recordings, and tabular seismological indicators computed from earthquake catalogs. Both modalities are available in a station–day format, allowing them to be aligned in time and by station code. The modeling framework implements two alternative methods to exploit these combined data: a gradient boosting model and a graph neural network (GNN) model.

### 3.4.1 Combination of Feature Types

For both modeling approaches, each station–day observation is represented by a concatenation of its waveform-based and catalog-based feature vectors. These include lagged values to capture short-term temporal context (e.g., previous-day features). In the gradient boosting approach, this combined feature vector is treated as a flat input, whereas in the GNN approach it is assigned as the node attribute in a spatial graph of stations. This setup enables direct comparison between a spatially agnostic tabular learner and a spatially aware graph learner, both trained on the same underlying information. We systematically evaluate three feature combination strategies: (1) catalog features only (seismologically-engineered features), and (2) hybrid approach (catalog + waveform embeddings).

### 3.4.2 Gradient Boosting Model

The gradient boosting approach employs a LightGBM multi-class classifier operating directly on the flattened station–day feature vectors. This method is effective for heterogeneous tabular inputs, capturing non-linear interactions among seismological variables and learned waveform features. Hyperparameter tuning explores a search space including learning rate, number of estimators, tree depth, regularization strengths, and subsampling parameters, with temporal cross-validation used to ensure the forecasting setup is respected. Class imbalance is addressed via built-in weighting options (`is_unbalance` and `scale_pos_weight`). This approach offers a computationally efficient baseline that does not model explicit spatial dependencies between stations.

### 3.4.3 Graph Neural Network Model

We represent the seismic station network as a weighted, undirected graph

$$G = (\mathcal{V}, \mathcal{E}, \mathbf{X}),$$

where $\mathcal{V}$ is the set of stations, $\mathcal{E}$ is the set of spatial edges, and $\mathbf{X} \in \mathbb{R}^{N\times F}$ contains the $F$-dimensional features for each of the $N$ stations. An edge $(i, j) \in \mathcal{E}$ is included if the Haversine distance between stations $i$ and $j$ is less than 100 km.

Two GNN architectures are implemented:

**GraphSAGE.** Each layer updates node embeddings by combining a station's own features with the average of its neighbors' features:

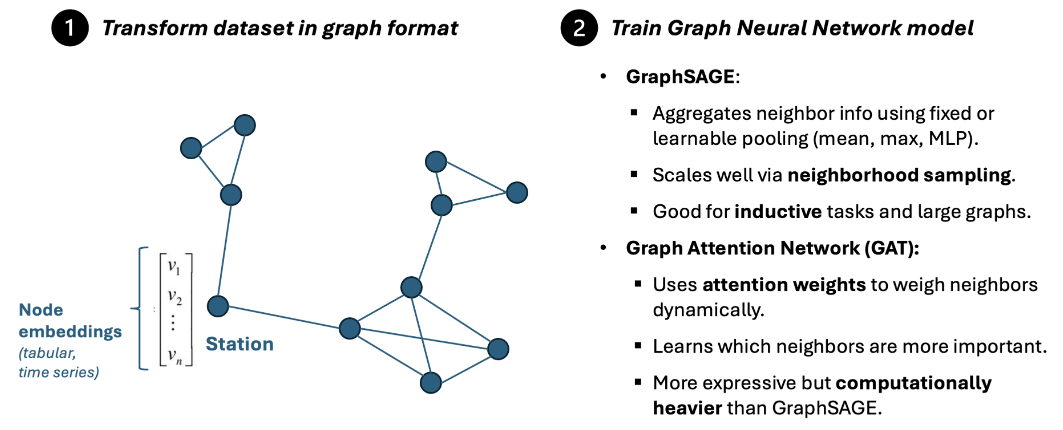
$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \frac{\mathbf{h}_i^{(l-1)} + \sum_{j\in\mathcal{N}(i)} \mathbf{h}_j^{(l-1)}}{|\mathcal{N}(i)| + 1}\right),$$

where $\mathcal{N}(i)$ are the neighbors of station $i$, $\sigma$ is a non-linear activation, and $\mathbf{W}^{(l)}$ is a trainable weight matrix.

**Graph Attention Network (GAT).** Instead of averaging, GAT learns attention weights $\alpha_{ij}$ for each neighbor:

$$\mathbf{h}_i^{(l)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}\, \mathbf{W}\mathbf{h}_j^{(l-1)}\right),$$

where $\alpha_{ij}$ reflects the relative importance of neighbor $j$ for predicting $i$'s target.


1 Transform dataset in graph format
Node embeddings
(tabular, time series)
$v_1$
$v_2$
$v_n$
Station
2 Train Graph Neural Network model

- **GraphSAGE**:
  - Aggregates neighbor info using fixed or learnable pooling (mean, max, MLP).
  - Scales well via **neighborhood sampling**.
  - Good for **inductive** tasks and large graphs.
- **Graph Attention Network (GAT):**
  - Uses **attention weights** to weigh neighbors dynamically.
  - Learns which neighbors are more important.
  - More expressive but **computationally heavier** than GraphSAGE.

**Figure 2:** Schematic illustration of the graph structure and trade-offs between GNN architectures.

**Output and Training.** After $L$ message-passing layers, the final node embeddings $\mathbf{h}_i^{(L)}$ are passed through a multilayer perceptron with a softmax output over $C$ magnitude classes. Training minimizes the cross-entropy loss over all stations and dates in the training set:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log \hat{y}_{ic}.$$

## 3.5 Evaluation and Outputs

We evaluate predictive performance, interpretability, and computational efficiency of both the GNN and boosting models under identical experimental conditions.

**Evaluation Setup.** Data from 2020–2023 is used for training and 2024–2025 for testing, ensuring temporal separation to avoid leakage. Hyperparameters for each method are tuned via 3-fold temporal cross-validation:

- **GNN**: tuned over number of layers, hidden units, learning rate, dropout, and neighborhood size.
- **Boosting**: tuned over learning rate, maximum tree depth, number of leaves, and feature subsampling.

The best configuration is retrained on the full training set and evaluated once on the held-out test set.

**Metrics.** We report overall and per-class accuracy, macro-averaged ROC–AUC, and normalized confusion matrices. Computational cost is measured as total training time to convergence.

**Outputs.** The pipeline stores configuration files, metrics tables, ROC curves, confusion matrices, and serialized model artifacts, enabling full reproducibility.

# 4 Preliminary Results

At this stage, results are exploratory and based on the best-performing configuration out of 25 tested for the GNN model, evaluated under different feature sets, data availability scenarios, and the inclusion or exclusion of learned embeddings. While embeddings do not always improve performance across all experimental configurations, in this particular setup they did yield a clear benefit. Here, the term *limited data* refers to restricting the evaluation to only those days and stations for which embedding vectors are available, ensuring a fair, like-for-like comparison between runs with and without embeddings.

For the *limited data, top features* scenario, the baseline without embeddings achieved a ROC–AUC of 0.5825 and accuracy of 0.6436. Adding learned embeddings in this setting improved ROC–AUC to 0.6583 (a gain of +0.0758) while maintaining a comparable accuracy of 0.6277. This suggests that embeddings can meaningfully enhance discrimination power even when only a small number of high-importance features are available.

When using *limited data, all features*, the inclusion of embeddings yielded no improvement in ROC–AUC (0.6409 vs. 0.6450) and accuracy remained similar (0.6395 vs. 0.6399). This may indicate that, for this configuration, embeddings provide the most benefit in feature-constrained regimes.

**Table 1:** Test ROC–AUC and accuracy for limited-data scenarios, with and without embeddings.

| Scenario | ROC-AUC (test) | Accuracy (test) |
|---|---|---|
| no_embeddings_limited_data_top_features | 0.5825 | 0.6436 |
| embeddings_limited_data_top_features | 0.6583 | 0.6277 |
| no_embeddings_limited_data_all_features | 0.6409 | 0.6395 |
| embeddings_limited_data_all_features | 0.6450 | 0.6399 |

Figure 3 shows the one-vs-rest ROC curves for the *no_embeddings_limited_data_all_features* setup for one of the configurations tested. Performance varies across magnitude classes, with the highest discrimination for Classes 3 and 4 (AUC of 0.77 and 0.74, respectively), while lower-magnitude classes remain more challenging to separate from the rest.

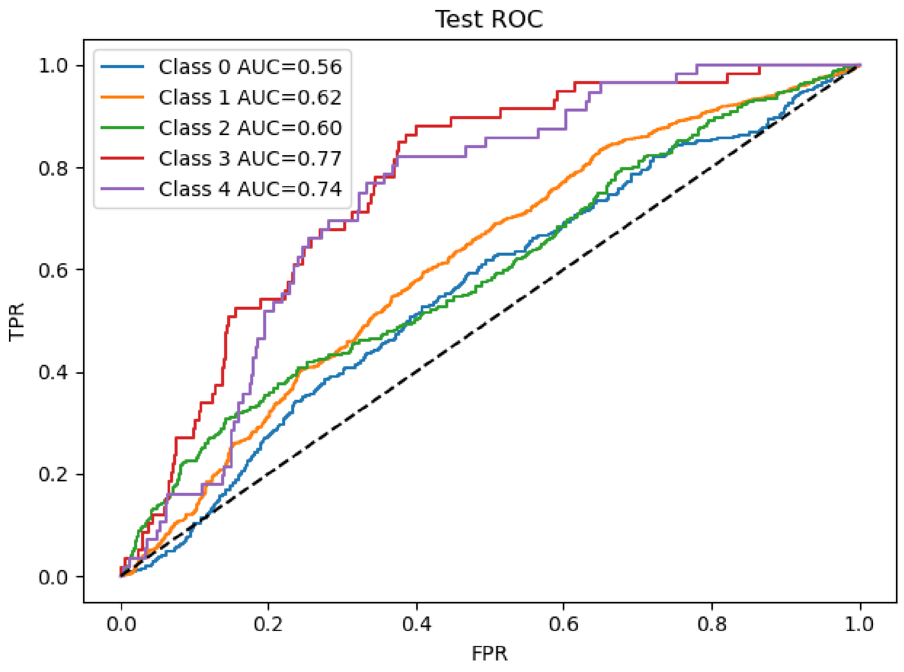


**Figure 3:** One-vs-rest ROC curves for the one of the model configurations as an example.

## 5 Data Availability by Station and Year

**Table 2:** Data Availability by Station and Year

| **Station** | **2020** | **2021** | **2022** | **2023** | **2024** |
|---|---|---|---|---|---|
| Station | 2020 | 2021 | 2022 | 2023 | 2024 |
| ABL | Missing | Missing | Missing | Missing | Complete |
| AGM | Missing | Missing | Missing | Complete | Complete |
| BAC | Missing | Missing | Missing | Missing | Missing |
| BAI | Missing | Missing | Missing | Complete | Complete |
| BHP | Missing | Complete | Complete | Complete | Missing |
| BRE | Missing | Missing | Missing | Complete | Complete |
| BSC | Missing | Missing | Missing | Missing | Missing |
| BTL2 | Missing | Missing | Missing | Missing | Missing days |
| BTP | Missing | Missing | Complete | Missing | Missing |
| CFS | Missing | Missing | Missing | Missing | Missing |
| CLI2 | Missing | Missing | Missing | Missing | Missing |
| CPO | Missing | Missing | Missing | Complete | Complete |
| CRR | Missing | Missing | Complete | Missing | Complete |
| CWC | Missing | Missing | Missing | Missing | Missing |
| CYP | Complete | Complete | Missing | Missing | Complete |
| DGR | Complete | Complete | Missing | Missing | Complete |
| DLA | Missing | Missing | Missing | Missing | Complete |
| EML | Missing | Missing | Complete | Missing | Missing |
| FMO | Missing | Missing | Missing | Complete | Complete |
| FRM | Missing | Missing | Missing | Missing | Missing |
| GOR | Complete | Missing | Missing | Complete | Complete |
| HYS | Complete | Missing | Missing | Complete | Complete |
| IPT | Missing | Missing | Missing | Missing | Missing |
| LVY | Missing | Missing | Missing | Complete | Missing |
| MAC | Missing | Missing | Missing | Missing | Missing |
| MAG | Missing | Missing | Missing | Missing | Missing |
| MAN | Missing | Missing | Missing | Complete | Complete |
| MRS | Missing | Missing | Missing | Missing | Missing |
| NWH | Missing | Missing | Missing | Missing | Missing |
| PASC | Complete | Complete | Complete | Complete |  |
| PDW | Missing | Missing | Missing | Missing | Missing |
| PLR | Missing | Missing | Missing | Missing | Missing |
| PTD | Missing | Missing | Missing | Complete | Missing days |
| RRX | Missing | Complete | Complete | Complete | Complete |
| SAN | Missing | Missing | Missing | Missing | Missing |
| SDD | Complete | Missing | Complete | Complete | Complete |
| SDG | Complete | Complete | Complete | Complete | Complete |
| SES | Complete | Complete | Complete | Complete | Missing |
| STG | Missing | Complete | Complete | Complete | Complete |
| SYN | Complete | Complete | Missing | Complete | Complete |
| WRC2 | Complete | Complete | Missing | Missing | Missing |
| YUH2 | Complete | Missing | Missing | Missing | Complete |

## References

Cohen, B., E. Khwaja, K. Wang, C. Masson, E. Ramé, Y. Doubli, and O. Abou-Amal (2024): "Toto: Time Series Optimized Transformer for Observability," .

Kim, G., B. Ku, J.-K. Ahn, and H. Ko (2022): "Graph Convolution Networks for Seismic Events Classification Using Raw Waveform Data From Multiple Stations," *IEEE Geoscience and Remote Sensing Letters*, 19, 1–5.

Liu, T., J. Münchmeyer, L. Laurenti, C. Marone, M. V. de Hoop, and I. Dokmanić (2024): "SeisLM: A Foundation Model for Seismic Waveforms," *arXiv preprint arXiv:2410.15765*.

Mao, N., K. Sun, and J. Zhang (2024): "Monthly Maximum Magnitude Prediction in the North–South Seismic Belt of China Based on Deep Learning," *Applied Sciences*, 14, 9001.

Mousavi, S. M., Y. Sheng, W. Zhu, and G. C. Beroza (2019): "STanford EArthquake Dataset (STEAD): A Global Data Set of Seismic Signals for AI," *IEEE Access*, 7, 179464–179476.

Yavas, C. E., L. Chen, C. Kadlec, and Y. Ji (2024): "Improving earthquake prediction accuracy in Los Angeles with machine learning," *Scientific Reports*, 14, 24440.

Zhang, Y., C. Zhan, Q. Huang, and D. Sornette (2025): "Forecasting future earthquakes with deep neural networks: application to California," *Geophysical Journal International*, 240, 81–95.