Life Could Be a Dream: Somnial Units for Earthquake Signal Detection

Gregory Lim, Ryan Cheong
Singapore University of Technology and Design
Singapore, SG 487372
{gregory_lim,ryan_cheong}@mymail.sutd.edu.sg

Abstract

We propose a novel architectural unit we call a *somnial unit*, and use it to achieve state-of-the-art results on an earthquake signal detection task.

1 Introduction

Every day, seismic signals are recorded by seismometers across planet Earth. Earthquake signal detection can be defined as the task of determining whether a seismic signal corresponds to an earthquake. Detecting earthquakes in real time from seismic signals can be useful for early warning and building fortification systems. Traditional algorithms which use Short-Term Average over Long-Term Average (STA/LTA) or template matching can struggle in noisy environments [5]. Machine learning algorithms are increasingly being adopted. In this paper, we propose a novel architectural unit we call a *somnial unit*, and use it to achieve state-of-the-art results on an earthquake signal detection task.

This project is available at

`http://github.com/glimeuxe/quake`

where instructions to reproduce our results can be found in the `readme.md` file.

2 Related work

Machine learning models have revolutionised seismic signal analysis by learning features directly from signal data. In 2020, Ellsworth et al. introduced EQTransformer, a multi-head transformer which employs a hierarchical attention mechanism to perform simultaneous earthquake detection and P/S phase picking, achieving human-level precision even in noisy conditions [1]. In 2023, Li et al. introduced SeisT, a model which integrates multi-scale depth-wise separable convolutions with self-attention modules to perform earthquake detection, phase picking, polarity classification, magnitude estimation, and more, outperforming EQTransformer and PhaseNet on benchmarks like DiTing and PNW [2]. In 2024, Rose developed a long short-term memory (LSTM) network trained on computed seismic signal envelopes, for tasks of earthquake detection, magnitude estimation, and phase picking [4]. By contrast, our work concerns the sole task of earthquake detection. In the case of real-time earthquake signal detection, the practical utility of a model can often depend both on how correct and how quick its predictions are. Our goal was thus to develop a model that could make correct predictions while staying computationally efficient.

3 Proposed unit

To support our goal of accurate and efficient earthquake signal detection, we propose a novel convolutional architectural unit inspired by neurobiological notions of dreaming in animals.

Let $C \in \mathbb{N}_{\geq 1}$ be an image channel count, and $H \in \mathbb{N}_{\geq 1}$ be an image height, and $W \in \mathbb{N}_{\geq 1}$ be an image width, and $L \in \mathbb{N}$ be a memory buffer size, and rand denote a uniform pseudorandom selection function.

Definition 1 (Memory Buffer)**.**

$$\mathcal{M}_t = \{\mathsf{x}_s : \max(0, t - L) \leq s \leq t\}$$

Definition 2 (Recollector)**.** Let $\mathbf{W}_\rho \in \mathbb{R}^{1 \times 1 \times C \times C}$ be a recollector weight, and $\boldsymbol{b}_\rho \in \mathbb{R}^C$ be a recollector bias.

$$\rho : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{H \times W \times C}$$
$$\rho(\mathsf{x}) = \mathsf{x} \circledast \mathbf{W}_\rho + \boldsymbol{b}_\rho$$

Definition 3 (Modulator)**.**

$$\mu : \mathbb{R}^{H \times W \times C} \times \mathbb{R}^{H \times W \times C} \to (0, 1)^{H \times W}$$
$$\mu(\mathsf{a},\, \mathsf{b}) = \sigma\left(\frac{\langle \boldsymbol{a}_{h,w},\, \boldsymbol{b}_{h,w} \rangle}{\|\boldsymbol{a}_{h,w}\|_2 \|\boldsymbol{b}_{h,w}\|_2}\right)_{(h,w) \in \{1,\ldots,H\} \times \{1,\ldots,W\}}$$

An algorithm for training is given in pseudocode as follows.

Algorithm 1 Train(x_t)

$\mathcal{M}_t \leftarrow \mathcal{M}_t \cup \{\mathsf{x}_t\}$
$\mathsf{x}_s \leftarrow \mathrm{rand}(\mathcal{M}_t)$
$\hat{\mathsf{x}}_s \leftarrow \rho(\mathsf{x}_s)$
$\mathsf{m} \leftarrow \mu(\hat{\mathsf{x}}_s,\, \mathsf{x}_t)$
return $\mathsf{m} \odot \hat{\mathsf{x}}_s + (1 - \mathsf{m}) \odot \mathsf{x}_t$

An algorithm for inference is given in pseudocode as follows.

Algorithm 2 Infer(x_t)

$\mathsf{x}_s \leftarrow \mathsf{x}_t$
$\hat{\mathsf{x}}_s \leftarrow \rho(\mathsf{x}_s)$
$\mathsf{m} \leftarrow \mu(\hat{\mathsf{x}}_s,\, \mathsf{x}_t)$
return $\mathsf{m} \odot \hat{\mathsf{x}}_s + (1 - \mathsf{m}) \odot \mathsf{x}_t$

The memory buffer maintains some number of feature maps for recollection. In practice, the memory buffer can be implemented as a double-ended queue of finite maximum length, for efficiency. Sampling from the memory buffer introduces inductive bias from any previously encountered examples, which may encode useful information. The randomness of this sampling can have a regularising effect on network activations, which could promote generalisation or mitigate overfitting.

The recollector is a learnable convolution for a recalled feature map. The modulator can be seen as a non-learnable factor which modulates between a current feature map and a recollected feature map. In particular, the modulator computes a cosine similarity between a current feature map and a recollected feature map, at each spatial position. This similarity is mapped to a soft attention-like score by a sigmoidal function. In practice, we used the sigmoid function simply for its smooth properties. Finally, the somnial unit returns a modulated linear combination between a current feature map and a recollected feature map. In practice, this interpolation between present and recalled information can then be passed to other (e.g. linear) layers in a neural network.

4 Methodology

4.1 Dataset

4.1.1 Description

Our raw dataset comprised chunks 1 and 6 of the Stanford Earthquake Dataset (STEAD) [3]. STEAD contains seismic waveforms measured at various seismic stations. Each waveform is 60 seconds long, sampled at 100 Hz, and labelled as either noise (class 0) or earthquake (class 1).

4.1.2 Exploratory analysis

To inform modelling, we analysed STEAD across some of its features.

Table 1: Summary statistics for some features of STEAD.

Feature	Minimum	Maximum	Average
Magnitude	-0.36	7.90	1.52
Depth (km)	-3.46	341.74	15.42
Distance to station (km)	0.00	336.38	50.58

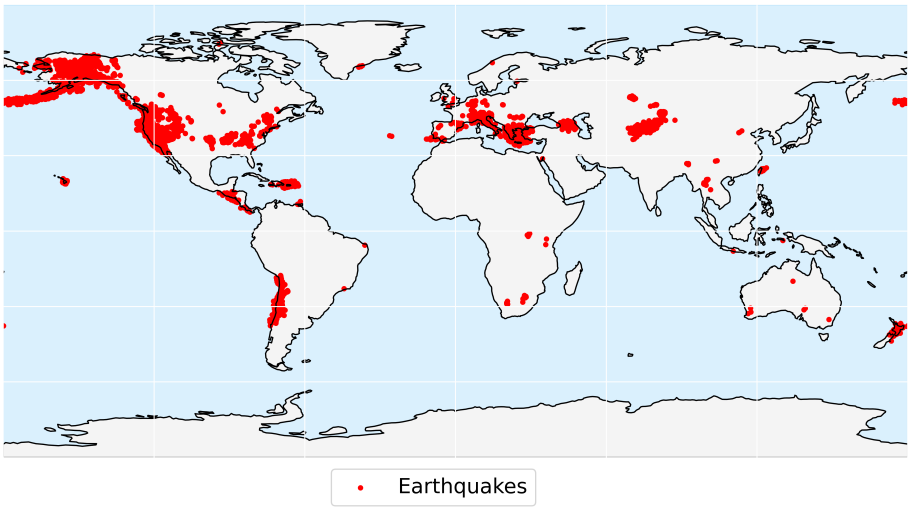


Figure 1: Location distribution of earthquakes in STEAD.

We also randomly selected a few samples, and plotted their waveform and spectrogram images to look for any visually obvious patterns.

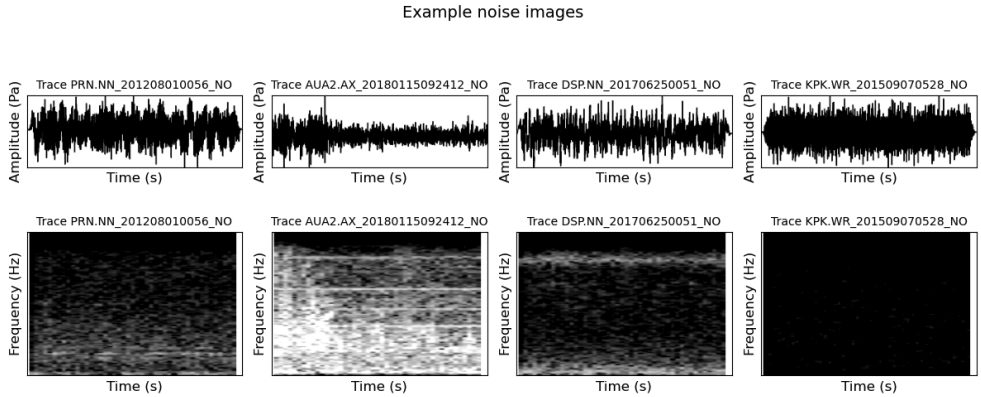


Figure 2: Examples of class 0 waveform and spectrogram images.

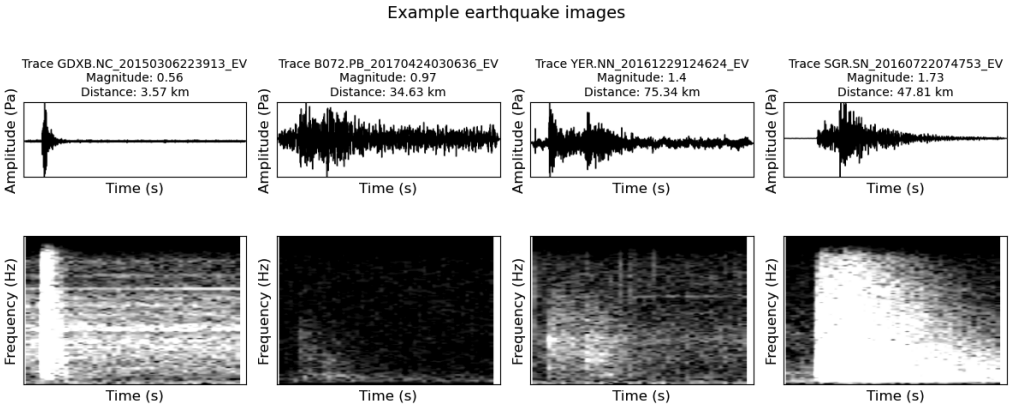


Figure 3: Examples of class 1 waveform and spectrogram images.

We noted that phase information might not be too useful for classification.

4.1.3 Preprocessing and feature engineering

We randomly selected 5000 class 0 waveforms and 5000 class 1 waveforms. This was done due to hardware limitations, and for each model to learn about both classes in equal measure. Then, we processed the raw dataset using two separate methods. The first method was used to generate spectrogram images, while the second method was used to generate resampled envelopes.

The first processing method is as follows.

Let $\varsigma(I)$ denote the standard deviation of pixel intensity for an image I.

1. **Trace to vertical component.** For each trace, we extracted the vertical component.
2. **Vertical component to spectrogram.** For each vertical component, we applied a short-time Fourier transform (STFT) with a Hamming window of length 256, and a hop size of 128. Then, we applied the base-10 logarithm function multiplied by 10. This was done to discard phase information.
3. **Spectrogram to image.** For each spectrogram, we clipped each value to a range of $[-10, 25]$. Then, we plotted the spectrogram as a grayscale image of height 200 and width 300.
4. **Class 1 image to noised class 1 image.** For each class 1 image I, we noised each pixel of I by an amplitude $\eta \sim \mathcal{N}(0, (\alpha\varsigma(x))^2)$ for some $\alpha \sim \mathcal{U}(0.001, 0.01)$. This was done to improve model robustness to noise.

The second processing method for resampled time series envelopes is as follows.

1. **Trace to vertical component.** For each trace, we extracted the vertical component.
2. **Vertical component to filtered component.** For each vertical component, we applied a fourth-order Butterworth bandpass filter from 1 to 49.9 Hz with a sampling rate of 100 Hz.
3. **Filtered component to envelope.** For each filtered component, we applied a Hilbert transform and computed its magnitude.
4. **Envelope to demeaned envelope.** For each envelope, we computed a 2-second rolling mean with a 200-sample window, discarded the initial 199 undefined values, and subtracted the global mean of the samples. This was done to reduce the noise in the features.
5. **Demeaned envelope to padded envelope.** For each demeaned envelope, we prepended the first demeaned value 199 times.
6. **Padded envelope to resampled envelope.** For each padded envelope, we downsampled from 6000 to 300 samples using FFT-based resampling.

Each processed dataset was split into training, validation, and test sets in a ratio of $8 : 1 : 1$.

4.2 Models

We explored various architectures, including convolutional neural network (CNN), vision transformer (ViT), recurrent neural network (RNN), and hybrid architectures. These architectures are defined more precisely in the GitHub repository. For training and testing, the RNNs used the second processed dataset, while the others used the first processed dataset.

4.2.1 Training

On each training set, we trained various models using binary cross-entropy (BCE) loss on the following settings, where settings not mentioned are assumed to use library defaults. The libraries we used are listed in the GitHub repository. For reproducibility, we seeded each function we used to the greatest extent practicable to our knowledge.

Call the maximum number of epochs without an improvement in validation loss (before early stopping), patience.

Table 2: Training hyperparameters for models.

Seed	37
Batch size	32
Optimiser	AdamW
Learning rate	0.001
Number of epochs	$\{50, 100\}$
Patience	$\{7, 10\}$
Non-linearity	relu

Table 3: Training hyperparameters for models equipped with a somnial unit.

Memory buffer size	10

Table 4: Training hyperparameters for CNN models.

Padding	1
Kernel size	3×3
Pooling type	Max
Pooling size	2×2
Pooling stride	2
Dropout	$\{0.25, 0.37, 0.5\}$

Table 5: Training hyperparameters for RNN models.

Memory vector size	$\{64, 128, 256\}$
Number of recurrent layers	$\{1, 2, 3, 4\}$
Number of linear layers	$\{2, 3\}$
Dropout	$\{0.2, 0.5\}$

We used a patience of 10 for some of the RNNs, and a patience of 7 for the others. We also ran 100 epochs for some RNNs, and 50 epochs for the others. This was done as a few of the RNNs generally seemed to take longer to converge. We also considered using additional algorithms, such as genetic and simulated annealing algorithms, to look for more optimal hyperparameters. However, we did not do so owing to hardware limitations and time constraints.

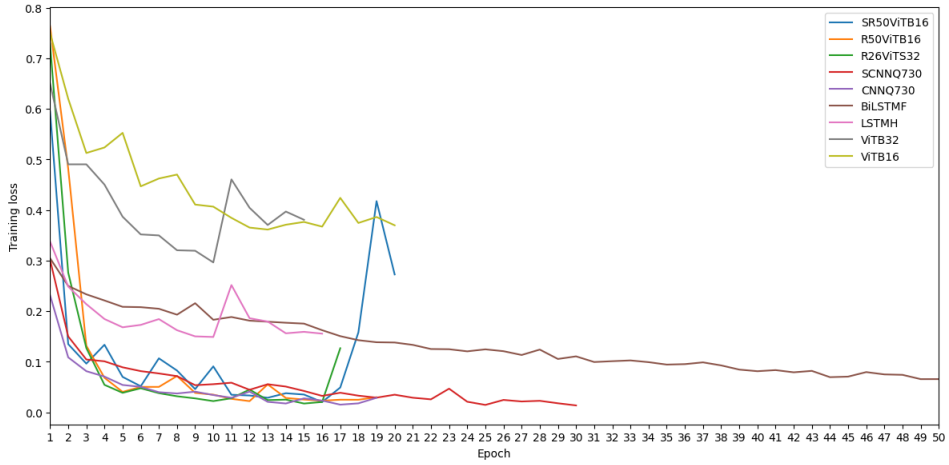


Figure 4: Training losses for models.

4.2.2 Evaluation

On each test set, we evaluated the trained models using macro F1 and recall metrics. Due to the severity of misclassifying and not being prepared for earthquake events, recall was deemed more important than precision. Our top-performing model was evaluated with and without a somnial unit, to investigate the effects of somnial unit addition. For brevity, the identifiers (IDs) of models equipped with a somnial unit are prefixed with "S", and only the top-performing models are shown.

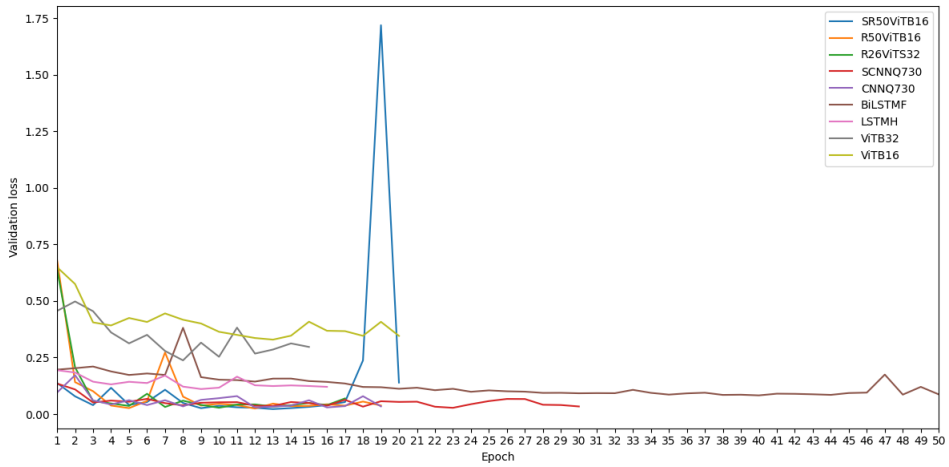
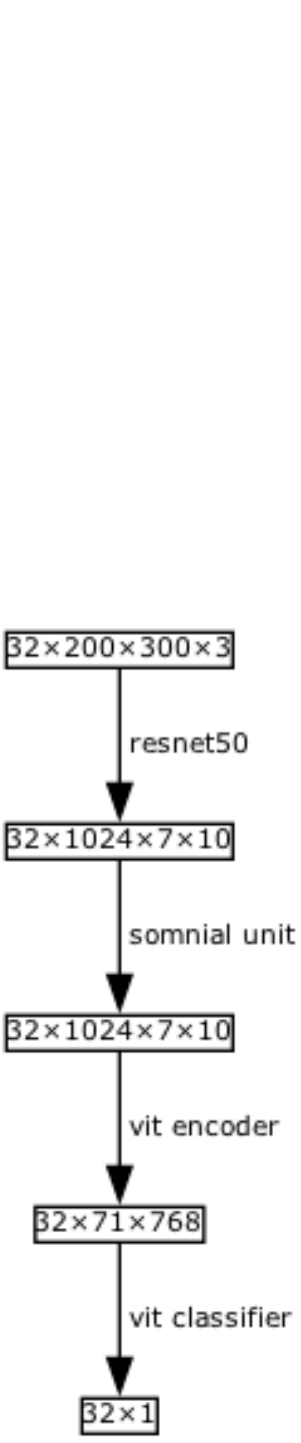


Figure 5: Validation losses for models.

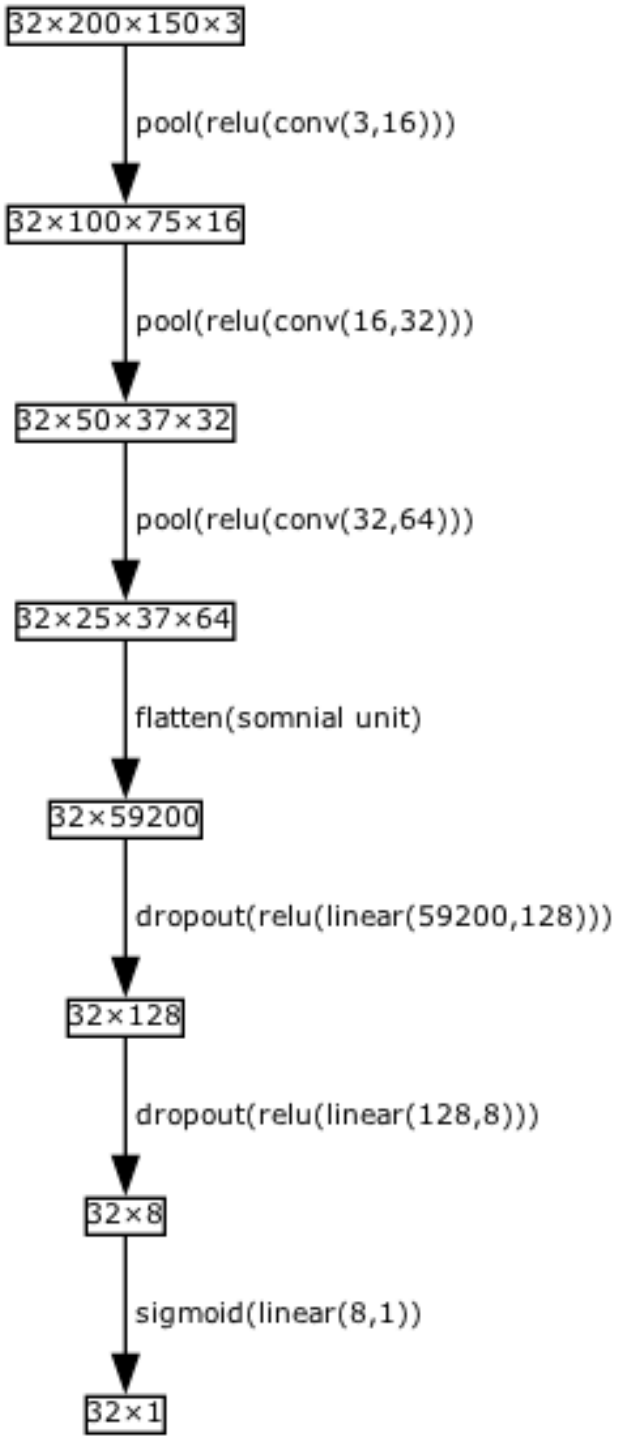
Table 6: Evaluation results for models.

ID	Family	Macro F1	Recall
SR50ViTB16	Hybrid	0.99600000	1.00000000
R50ViTB16	Hybrid	0.99499999	0.99600000
R26ViTS32	Hybrid	0.99299994	0.99600000
SCNNQ730	CNN	0.99199997	0.99000000
CNNQ730	CNN	0.98899972	0.98400000
BiLSTMF	RNN	0.96399770	0.97200000
LSTMH	RNN	0.95399540	0.94400000
ViTB32	ViT	0.90175439	0.95200000
ViTB16	ViT	0.85554700	0.91200000

5 Discussion



(a) Architecture of SR50ViTB16.



(b) Architecture of SCNNQ730.

Figure 6: Architectures of SR50ViTB16 and SCNNQ730.

We consider SR50ViTB16 and SCNNQ730 our best models as they achieved the highest macro F1 and recall scores. Compared to the latter, the former scored higher, but took more time for inference.

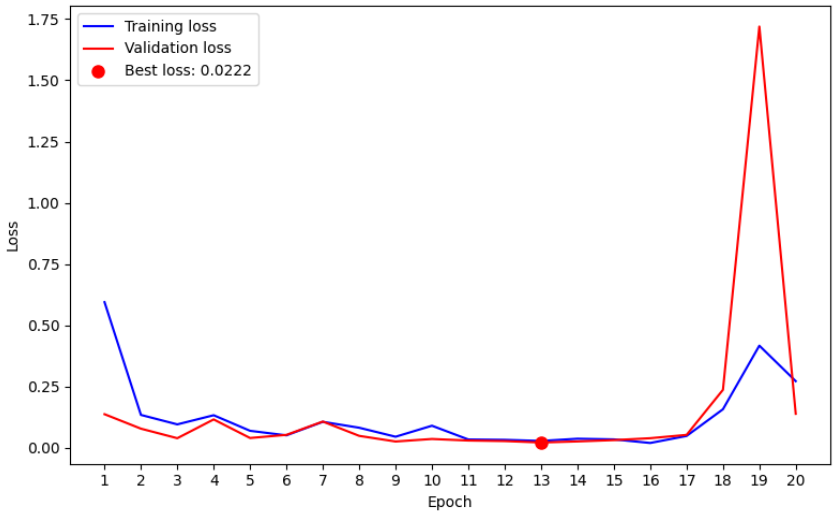


Figure 7: Training and validation losses for SR50ViTB16.

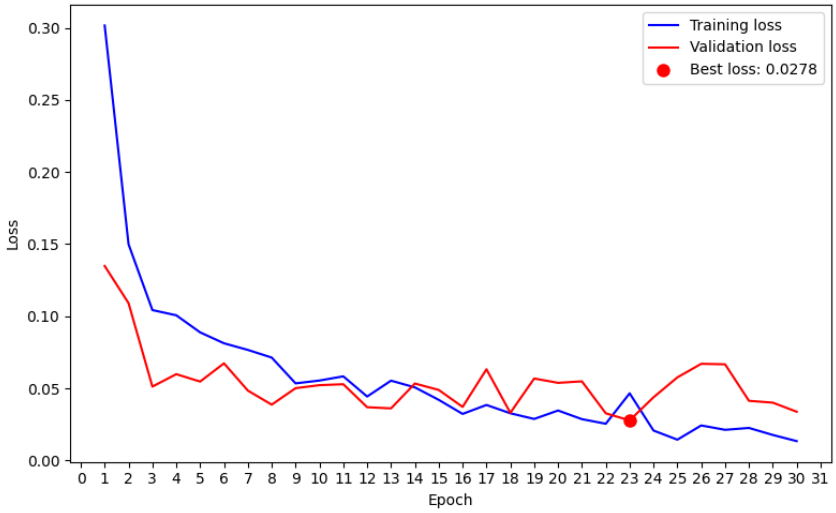


Figure 8: Training and validation losses for SCNNQ730.

Notably, the addition of a somnial unit to R50ViTB16 and CNNQ730 slightly improved their performances. In particular, the perfect recall score achieved by SR50ViTB16 constitutes a state-of-the-art result for this task. Comparatively, we note that the EQTransformer developed by Ellsworth et. al had a recall of 1 on STEAD, and the best CNN developed by Rose had a recall of approximately 0.9883 on a subset of STEAD different from ours.

Across the board, model families can also be ranked by their macro F1 scores. The hybrid models slightly outperformed the CNN and RNN models, while the CNN and RNN models significantly outperformed the ViT models. We conclude that the inductive bias introduced by convolution was useful for the task. Overall, we are fairly certain that more investigation could yield a better understanding of the results we obtained. Due to hardware limitations and time constraints, we were only able to run ablative studies for the somnial unit on our top-performing model and some of our simplest CNN models (e.g. CNNQ730).

Domain knowledge suggests that earthquake signals typically show P-wave and S-wave arrivals as distinctive energy bursts, and that noise signals often show consistent levels of energy in specific bands. With this in mind, we used a gradient-weighted class activation mapping (Grad-CAM) to try to generate a visual explanation for the predictions made by SCNNQ730 on its test set.

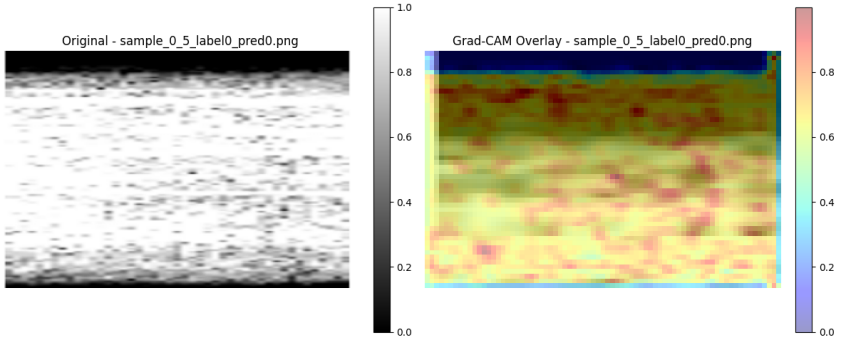


Figure 9: Grad-CAM visualisation of a true negative predicted by SCNNQ730.

SCNNQ730 correctly classified the noise sample shown in 9. Consistent horizontal bands can be observed in the spectrogram, which seems to be recognised by the model. The model also seems to focus on the black spots within the noise for its classification.

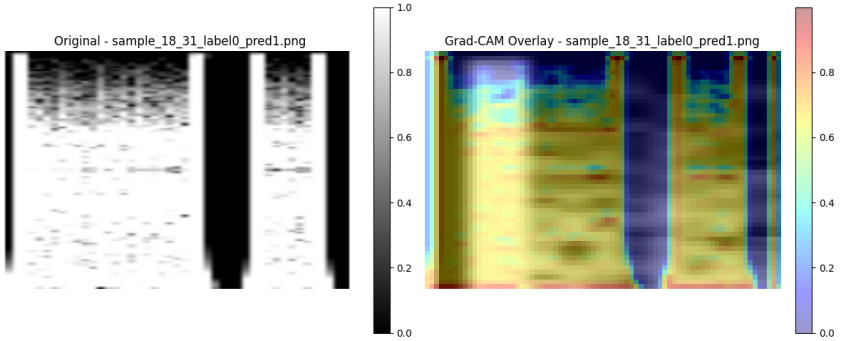


Figure 10: Grad-CAM visualisation of a false positive predicted by SCNNQ730.

SCNNQ730 failed to recognise the noise sample shown in 10. The model seemed to focus heavily on certain vertical structures with high intensity, which could resemble the phase arrivals characteristic of earthquake signals.

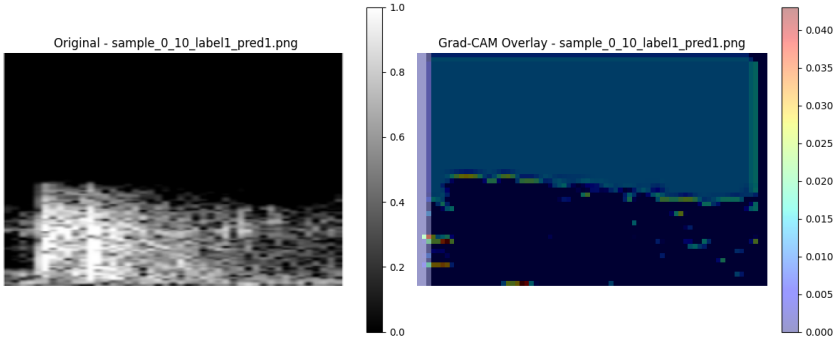


Figure 11: Grad-CAM visualisation of a true positive predicted by SCNNQ730.

SCNNQ730 correctly classified the earthquake sample shown in 11. A distinct increase in low frequency amplitudes can be observed. The model seems to focus on the frequency border of the new signal created by the earthquake.

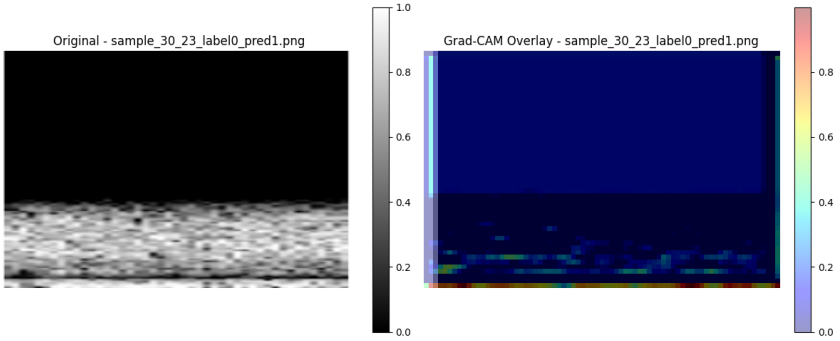


Figure 12: Grad-CAM visualisation of a false negative predicted by SCNNQ730.

SCNNQ730 also failed to recognise the earthquake sample shown in 12. In this case, the earthquake manifests as a continuous horizontal energy band rather than displaying distinct phase arrivals. The model could have erred simply because the continuous morphology deviated too much from the more pronounced P-wave and S-wave patterns represented in its training data.

6 Future Work

The somnial unit was developed iteratively in both a breadth-wise and depth-wise fashion. On one iteration of the unit, we replaced the convolutional layer of the recollector with a Fourier neural operator (FNO). However, this increased memory usage, which eventually made training infeasible on our hardware. On another iteration of the unit, we defined the modulator as a convolutional layer which learns a concatenation of a current feature map with a recollected feature map. However, we eventually scrapped this idea to reduce the computational complexity of the unit. Briefly, we also explored using Kolmogorov–Arnold layers, cyclic topologies, and Fourier feature mapping.

The somnial unit admits several modifications and extensions as follows.

- For a classification task, one could define a memory buffer for each class, so that for each class c, a memory buffer for c stores samples which belong only to class c.
- One could use a sampling function other than a uniform pseudorandom selection function.
- One could noise the generator by an amplitude to introduce regularisation.
- One could use a moving average (e.g. exponential moving average) to modulate between current and recollected feature maps.

7 Conclusion

In this paper, we proposed a novel architectural unit we call a *somnial unit*, and use it to achieve state-of-the-art results on an earthquake signal detection task.

References

[1] William L. Ellsworth et al. "Earthquake Transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking". In: *Nature Communications* (2020). DOI: 10.1038/s41467-020-18247-5.

[2] Sen Li et al. "Seismogram Transformer: A foundational deep learning model for earthquake monitoring tasks". In: *IEEE Transactions on Geoscience and Remote Sensing* (2024). DOI: 10.1109/TGRS.2024.3371503.

[3] S. Mostafa Mousavi et al. "STanford EArthquake Dataset (STEAD): A Global Data Set of Seismic Signals for AI". In: *IEEE Access* (2019). DOI: 10.1109/ACCESS.2019.2947848.

[4] Kaelynn Rose. *Earthquake Detection & Characterization with Deep Learning*. https://github.com/kaelynn-rose/Earthquake_Detection. Accessed: 2025-04-19. 2024.

[5] Weiqiang Zhu et al. "Seismic arrival-time picking on distributed acoustic sensing data using semi-supervised learning". In: *Nature Communications* 14 (2023). DOI: 10.1038/s41467-023-43355-3. URL: https://www.nature.com/articles/s41467-023-43355-3.

A Appendix

A.1 Summary of contributions

- Gregory Lim: Extraction of dataset files from STEAD, research and invention of somnial unit, training and fine-tuning of other models, first data processing method, writing of main code framework, drawing of architectural diagrams, plotting of loss curves, compilation of model results.
- Ryan Cheong: Other parts of report, training and fine-tuning of RNNs, Grad-CAM implementation, second data processing method.