Available Online: 16 June 2026 DOI: 10.54254/2977-3903/2026.34386

Self-supervised seismogram learning using Siamese CNN

Phyu Phyu Win

Queens College, City University of New York, New York, USA

gutdorcas@gmail.com

Abstract. Earthquakes are sudden releases of energy in the Earth's crust that generate seismic waves recorded as time-series signals called seismograms. Earthquake analysis depends on interpreting the temporal structure of seismic phases, yet many machine-learning approaches require large manually labeled datasets that are difficult to obtain at scale. To address this, we propose a self-supervised framework that learns waveform continuity and ordering cues without human annotations. Our approach consists of two steps: preprocessing and model training. In preprocessing, each one-dimensional waveform is converted into a standardized waveform image and partitioned into a fixed number of consecutive segments with small gaps to simulate missing observations and create non-contiguous inputs. In model training, these segments are randomly shuffled and used to train a Siamese convolutional neural network to predict the shuffled order through a permutation-classification objective, where pseudo-labels are generated automatically from the applied shuffle. We evaluate the framework across multiple segment settings and summarize its performance trends as task difficulty increases. Overall, this study demonstrates that permutation-based self-supervision can enable learning useful waveform structure from unlabeled data, and it suggests a practical direction for developing phase-aware representations that can support future seismic analysis when labeled data are limited.

Keywords: self-supervised, labeled, waveform segmentation, Siamese CNN model

1. Introduction

Earthquakes are natural phenomena caused by the sudden release of energy in the Earth's crust, producing seismic waves that can result in significant damage to infrastructure and loss of life. They can be classified into several types, including tectonic earthquakes, volcanic earthquakes, collapse earthquakes, and mining-induced earthquakes. Among these, tectonic earthquakes are the most prevalent and occur due to the movement and collision of tectonic plates, driven by convection currents in the Earth's semi-fluid mantle [1].

Seismic waves provide a direct measure of an earthquake's impact. Seismic waves begin with Primary waves (P-waves), followed by Secondary waves (S-waves), and finally surface waves, which are typically the most destructive. A seismogram is used to record these waves, producing a waveform that visually represents the ground motion over time [2].

Seismograms, by providing quantitative records of ground motion, establish a reliable basis for computational analysis. Building on this foundation, Machine Learning, which is a subfield of artificial intelligence focused on the development of algorithms and statistical models that enable computer systems to

learn through experience, rather than explicit programming, has become increasingly important in the interpretation of seismic data.

Machine learning has been applied across multiple dimensions of earthquake research. For instance, the Earthquake Transformer, an attentive deep-learning model that enables simultaneous earthquake detection and phase picking, significantly improves sensitivity to smaller events [3]. Another contribution is an optimized hybrid Recurrent Neural Network - Long Short-Term Memory (RNN-LSTM) architecture developed for seismicity analysis and earthquake prediction [4]. For phase detection specifically, DNN-based approaches such as PhaseNet have demonstrated significant improvements in picking P- and S-wave arrivals [5].

Prior research has explored the application of machine learning techniques, including Deep Neural Networks (DNNs) [6], Convolutional Neural Networks (CNNs) [7], and recurrent models such as RNNs [8]. These approaches have primarily relied on supervised learning and large labeled datasets. However, seismic waveform data, by nature, are abundant yet sparsely labeled [9], making supervised methods difficult to scale. To date, there has been limited investigation into self-supervised learning strategies for seismic waveforms, particularly approaches that enable models to learn the structure and sequential characteristics of P-waves, S-waves, and surface waves directly from raw time series data.

In this study, we use a self-supervised learning method to analyze seismic waveforms as time-series data. This allows the model to learn the unique features of P-waves, S-waves, and surface waves, as well as the order in which they appear, without requiring a large set of labeled examples. This matters for arrival-time estimation because phase arrivals are exactly the moments when the waveform changes in a consistent, physically meaningful way. For example, a P-wave onset often looks like a shift from quieter background noise to a more coherent pattern. A model that has learned to recognize which pieces of a waveform can plausibly follow one another has implicitly learned features that are sensitive to these transitions.

While self-supervised learning has been widely used in other domains, applying a segment-reordering objective to seismic waveform images to learn ordering and continuity cues is, to our knowledge, still not commonly explored in standard phase-picking pipelines. Beyond demonstrating that the model can solve the reordering task reliably, this approach could help in real scenarios where labels are limited, or recordings are incomplete, such as improving phase picking with fewer annotated events, assisting analysts by proposing candidate arrival windows, and supporting quality control by detecting discontinuities or corrupted segments that can interfere with accurate arrival-time estimates.

2. Materials and methods

2.1. Dataset

Our study utilizes a self-created corpus of 2,000 discrete one-dimensional waveforms with dimensions $(N, T) = (2000, 3000)$. Each waveform represents an independent signal instance defined by 3,000 sequential sample points. For downstream modeling, the waveforms are treated as high-resolution time-series traces, where the ordering of samples encodes the signal's temporal structure.

2.2. Data preprocessing

2.2.1. Converting a 1D waveform to an image

In the data preprocessing stage, we convert each one-dimensional waveform into a fixed-resolution waveform plot image before segmentation and model training. This design choice was motivated by an observed mismatch between the raw array representation and the intended learning objective: when the signals are provided directly as T-length numeric sequences, models can converge extremely quickly by exploiting

representation-specific shortcuts. For instance, strong local continuity, autocorrelation, and implicit absolute position along the index. In practice, this produces performance that resembles cheating, where high accuracy is achieved through trivial cues embedded in the sequential format rather than meaningful structural understanding of the waveform. By transforming the waveforms into images, we reduce direct access to explicit time indexing and force the model to infer patterns from waveform morphology, such as peak shape, slope changes, curvature, and relative geometry, rather than from raw sample adjacency. Overall, the image transformation serves as a regularizing preprocessing step that discourages shortcut learning and encourages learning of higher-level, more transferable waveform features.

2.2.2. Segmentation and gapping

To train the network without manual annotations, we cast the task as a self-supervised jigsaw reordering task. Each rendered waveform image is partitioned into N consecutive segments $s_0, s_1, ..., s_{N-1}$, in its true left-to-right order $(0, 1, ..., N-1)$, where $N \in \{3, 4, 5\}$, indicating that the experiment was carried out with three, four, and five segments. We then generate pseudo-labels by synthetically shuffling these segments and training the network to identify the applied shuffle. Concretely, we sample a permutation $\pi \in s_N$, where $|s_N| = N!$, and form the shuffled input tuple $X_\pi = (s_{\pi(0)}, s_{\pi(1)}, \ldots, s_{\pi(N-1)})$. The learning objective is to infer which permutation π produced the observed segment order. This forces the model to learn continuity and compatibility cues across segment boundaries, rather than relying on absolute timing metadata.

2.2.3. Pseudo-label encoding

Each permutation $\pi$is treated as a distinct class. For a given segment count $N \in \{3, 4, 5\}$, we enumerate all $N!$ permutations of $\{0, 1, \ldots, N-1\}$ and construct a deterministic mapping from permutations to class indices: Label = perm_to_idx(π)$\in \{0, 1, \ldots, N! - 1\}$. Thus, labels are pseudo-labels derived directly from the shuffling operation, meaning no human labeling is required. During training, the permutation is chosen on the fly each time a sample is loaded, so the same waveform can be presented in different shuffled orders across epochs. This increases the variety of training examples the model sees and encourages it to learn general ordering and continuity cues rather than memorizing a single arrangement for each waveform.

2.2.4. Dataset construction and input formatting

For each waveform, the data loader loads the pre-generated segment images and randomly shuffles their order. We repeat this setup using three, four, and five segments, so the number of possible orders changes with the segment count. The shuffled segments are resized and normalized, stacked into a single input, and paired with a pseudo-label that records the chosen shuffle. This on-the-fly shuffling creates many training variants from the same waveform and encourages the model to learn ordering cues from waveform structure without any manual annotations.

2.2.5. Training and validation

Even though our pretext task generates pseudo-labels automatically, we still split the dataset into training and validation with an $80:20$ ratio because evaluating on the same data used for optimization would inflate performance and make it impossible to tell whether the model is truly learning generalizable ordering cues or simply memorizing dataset-specific patterns. Training updates the network weights to minimize loss on the training portion, while the held-out validation portion provides an unbiased check of how well the learned representation transfers to unseen waveforms. We perform this split at the waveform-folder level so that all segment images derived from a single waveform remain in the same split, preventing leakage where the model might effectively see the same underlying signal in both training and evaluation, and therefore result in cheating. The validation set is then used for model selection by saving the best checkpoint and for monitoring

overfitting across epochs; without it, there would be no principled way to decide whether more training improves real generalization or only improves fit to the training examples.

2.3. Siamese CNN model

As shown in the Figure 1: Our model takes the shuffled waveform segments as one training example and predicts which ordering produced that shuffle. We ran the same model setup with three, four, and five segments. Rather than feeding all segments into one single CNN at once, we use a Siamese-style design: the same convolutional feature extractor, which shares the same weights, is applied to each segment independently [10].

Each segment image is passed through a stack of convolution layers with batch normalization and ReLU activations. Max pooling is used between blocks to gradually reduce the spatial size while keeping the most important shape information. At the end of this encoder, adaptive average pooling converts each segment into a fixed-length feature vector, which serves as a compact summary of that segment's waveform shape.

The Figure 2 shows that we concatenate their feature vectors into one combined representation after encoding all segments. This combined feature is then fed into a small fully connected classifier with dropout for regularization. The classifier outputs one score for every possible segment ordering, and the number of classes depends on whether we use three, four, or five segments. We train the network using cross-entropy loss against the pseudo-label from the shuffle, so the model learns to recover the correct temporal order by recognizing how waveform structure should connect across segment boundaries.

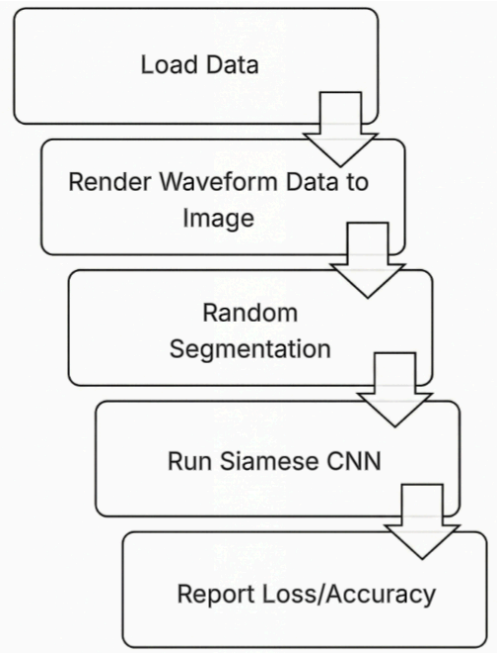

Figure 1. Workflow chart

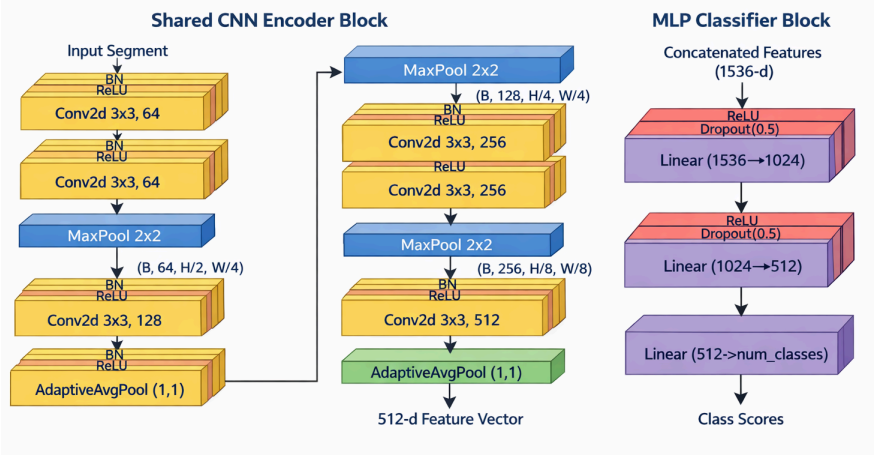

Figure 2. Siamese CNN model internal structure

3. Results

3.1. Overview

This study evaluates the efficacy of self-supervised learning for extracting high-dimensional temporal features from a two-dimensional visual domain and implementing a stochastic fragmentation protocol. We establish a permutation-based pretext task that necessitates an understanding of signal continuity. The methodology utilizes a Siamese Convolutional Neural Network (Siamese CNN) to encode discrete, non-contiguous segments into a shared latent space. To resolve the original chronological sequence, the model must identify and correlate invariant morphological signatures, such as local gradients, periodic oscillations, and structural trends. Achieving high accuracy in sequence reconstruction demonstrates that the architecture has successfully internalized the latent dynamics and global temporal dependencies inherent in the time-series data without the requirement for manually annotated labels.

3.2. Experiment

3.2.1. Waveform rendering

We start from a corpus of earthquake waveforms stored as one-dimensional time-series arrays. To make the inputs compatible with standard convolutional neural networks, each waveform is rendered into a clean line-plot image at a fixed resolution of 512×256 pixels. The waveform trace is drawn in black on a white background, and all axes, ticks, labels, and borders are removed so that the model sees only waveform geometry. To ensure visual consistency across samples, we apply global amplitude scaling: the vertical limits used for rendering are determined once from the minimum and maximum values across the entire waveform collection and then reused for all images. This prevents per-sample auto-rescaling from changing the apparent steepness, curvature, or relative amplitude patterns across waveforms.

3.2.2. Random segmentation with gaps

After rendering, each waveform image is partitioned along the horizontal, which is the time axis, into multiple consecutive segments. Segment boundaries are sampled randomly so that segment lengths vary across waveforms, which increases diversity in the segmentation patterns observed during training. The sampling is constrained to preserve usable visual structure: each segment is required to have a minimum width of 80 pixels, and a minimum gap of 10 pixels is enforced between adjacent segments. Gap regions are treated as

missing observations and are excluded from the cropped segment images. For quality control, we verify the segmentation process by visualizing the original waveform image alongside a reconstructed canvas where the saved segments are pasted back into their original horizontal positions and the gaps are highlighted.

3.3. Data organization and learning procedure

After preprocessing, we saved the generated data in a folder-based structure, creating one directory per waveform instance. Each folder contains the full rendered waveform image and its corresponding segment images, which makes the dataset easy to load and ensures that all segments from the same waveform stay grouped. To confirm that the preprocessing was consistent, we randomly selected waveform folders and visually inspected them by reconstructing a canvas from the saved segments. We do this by pasting each segment back at its original horizontal position and checking that the gaps and boundaries are aligned with the original waveform. Once the saved segments passed these checks, we proceeded to training.

During training, the data loader reads the pre-generated segment images for a waveform and constructs a training example by randomly shuffling their order. Just as we can see in Figure 3, 4, and 5, we ran this procedure using three, four, and five segments, so the number of possible shuffle orders changes depending on the segment count. Each segment image is processed in the same way before entering the model: it is resized to a fixed resolution, converted into a tensor, and normalized so that all inputs share a consistent shape and intensity scale. The processed segment tensors are then stacked together into a single input sample where multiple segment images are treated as one example, and a pseudo-label is attached to indicate which shuffle was applied. We trained the network using a batch size of 16, optimized the permutation classification objective with cross-entropy loss, and updated parameters using the Adam optimizer.

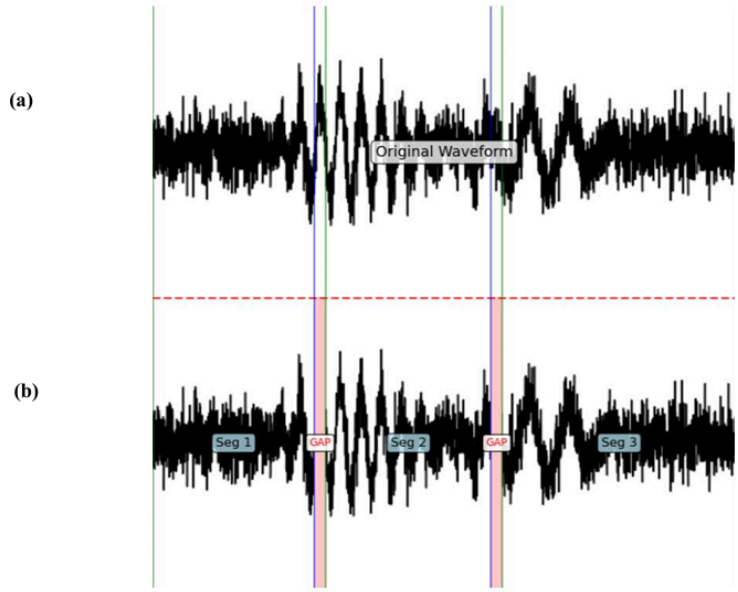

Figure 3. (a) Waveform with three randomized segments; (b) waveform with labels of three segments and random gaps

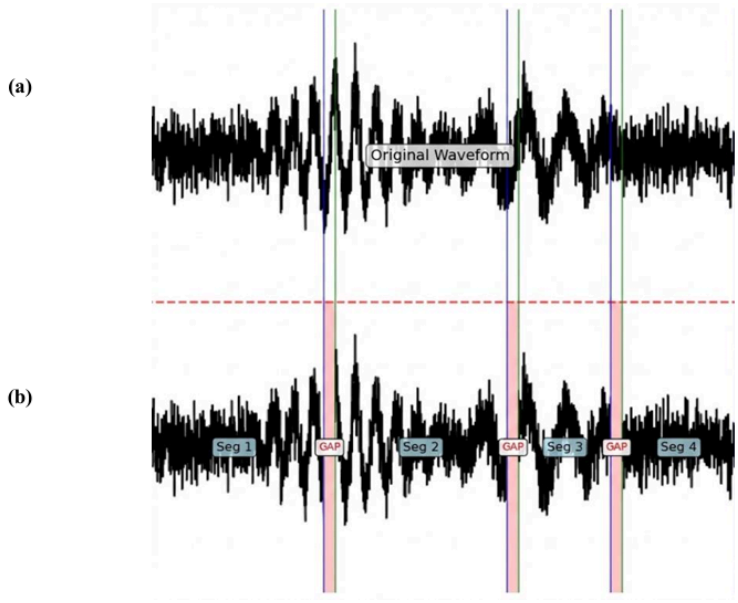

Figure 4. (a) Waveform with four randomized segments; (b) waveform with labels of four segments and random gaps

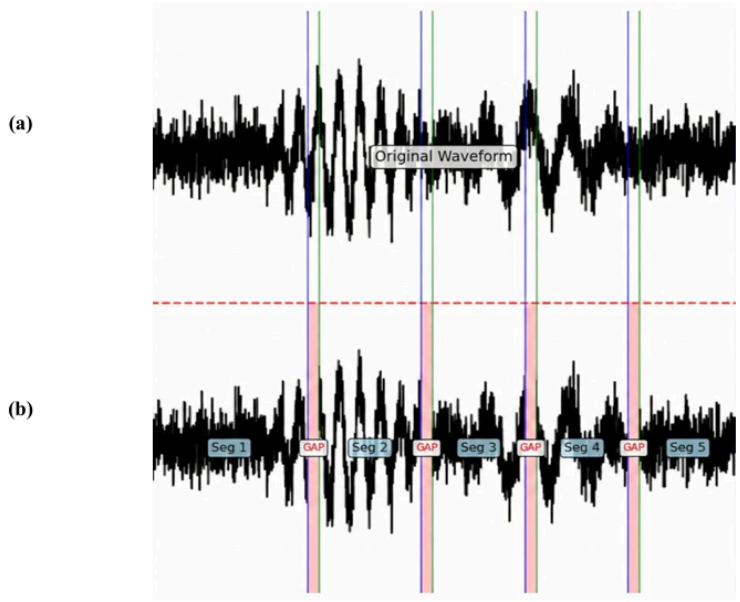

Figure 5. (a) Waveform with five randomized segments; (b) waveform with label of five segments and random gaps

3.4. Model performance

In terms of model performance, the difficulty of the permutation task increases noticeably as the number of segments increases. With three segments, the network reaches near-perfect performance very quickly: both training and validation accuracy rise to about 100% by around 15 epochs, and both training and validation loss continue decreasing until it becomes essentially zero by roughly 48 epochs. With four segments, performance is still very strong but follows a more typical convergence pattern, stabilizing smoothly at about 97-98%

validation accuracy, with low losses roughly around 0.05-0.15, and validation loss roughly 0.03-0.10. In contrast, the five-segment setting is substantially harder and shows more variability. Even after extending training to 150 epochs, training accuracy peaks at around 95%, while validation accuracy reaches only about 86% at best, with training loss around 0.1-0.2 and validation loss remaining higher at about 0.5-0.6. Although the rapid saturation in the three-segment case may initially look like cheating, the overall trend is consistent with task complexity: fewer segments create fewer boundaries and far fewer possible orders, while five segments greatly increase the number of orderings and make generalization more challenging. Figure 6, 7, and 8 respectively illustrate the above conclusions.

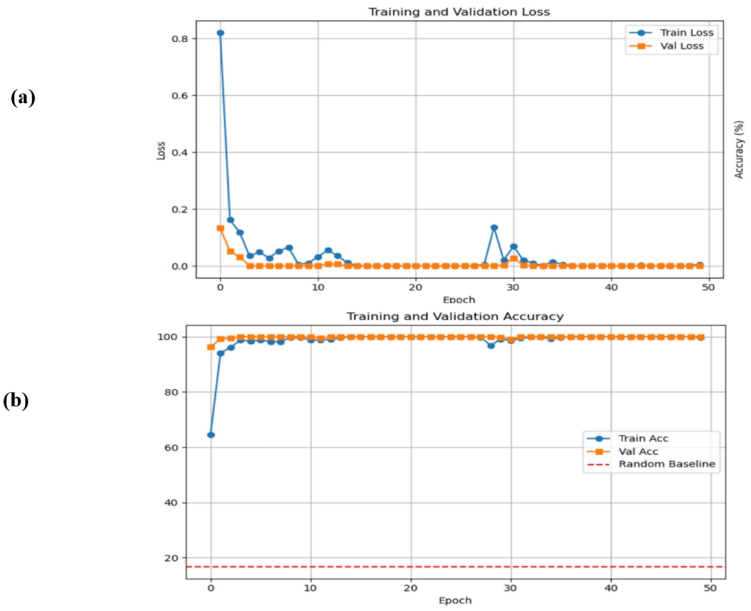

Figure 6. (a) Three segments training and validation loss, (b) three segments training and validation accuracy

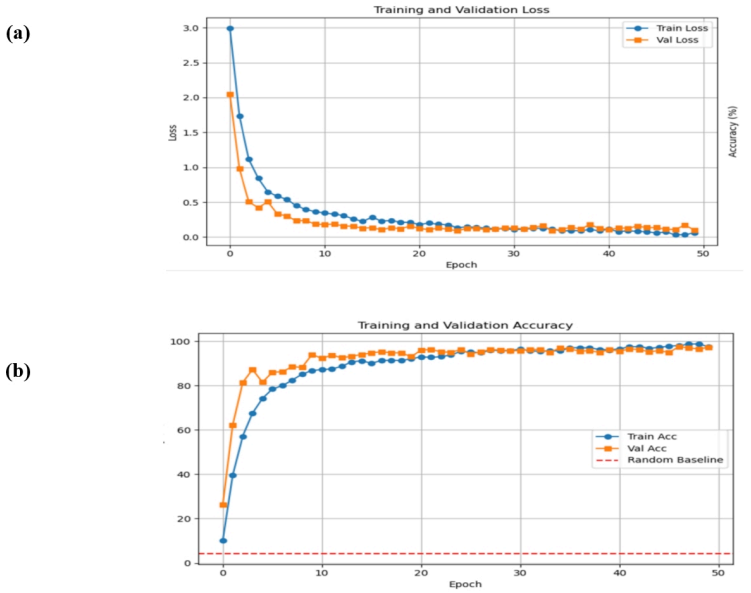

Figure 7. (a) Four segments training and validation loss, (b) four segments training and validation accuracy

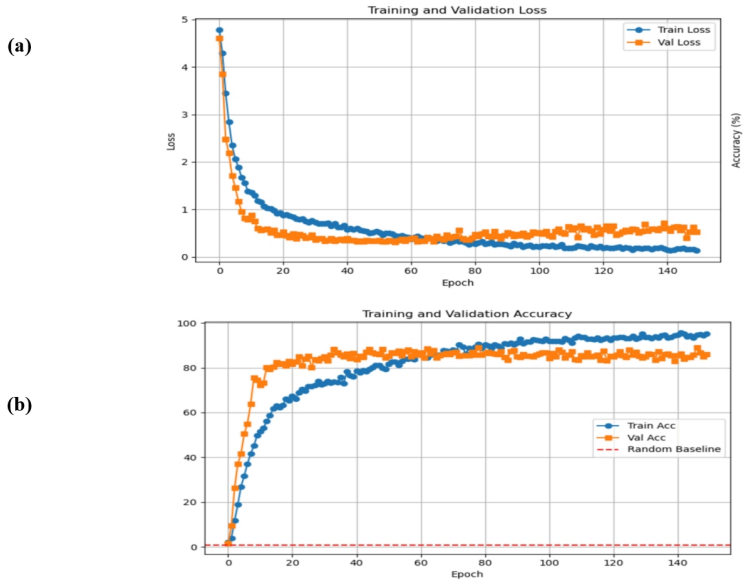

Figure 8. (a) Five segments training and validation loss, (b) five segments training and validation accuracy

4. Discussion

Across the segment-reordering experiments, the Siamese CNN consistently learns meaningful ordering cues from waveform morphology, but the final performance depends strongly on task difficulty. In the four-segment setting, the model reaches very strong and stable results: validation accuracy rises far above the random baseline and converges smoothly in the high 90% range, while both training and validation loss decrease to low values and remain closely aligned. This behavior suggests that the shared encoder and concatenation-based classifier can reliably capture continuity information across segment boundaries when the rendering and segmentation pipeline is controlled and consistent.

When we vary the number of segments, the same trend becomes clearer. With three segments, the task is substantially easier because there are fewer boundaries to reconcile and far fewer possible orders. The network reaches near-perfect accuracy quickly, which is about 100% for both training and validation by roughly 15 epochs, and the loss approaches zero by the end of the run. This rapid saturation is expected in a simplified setting where continuity cues are strong, and the output space is small. At the other extreme, the five-segment experiment is noticeably harder and exposes the limits of generalization under the same data and model capacity. Even with longer optimization, training accuracy peaks at around 95%, while validation accuracy improves more slowly and plateaus at roughly 86%, with a persistent gap between training and validation loss. This pattern is consistent with increased combinatorial complexity and greater ambiguity at segment boundaries, rather than data leakage.

Overall, these results support two main conclusions. First, the proposed image-based preprocessing and Siamese-style architecture are effective for learning segment compatibility and reordering signals without manual labels. Second, performance degrades as the problem becomes more complex, indicating that success in the easiest setting should be interpreted as a proof-of-concept rather than a guarantee of transfer to harder conditions. In real-world terms, this type of learned "continuity awareness" could be useful in scenarios where waveform records are incomplete or fragmented. For instance, it could help fill in or reason about missing parts of a recording, match and connect waveform pieces that got separated or shifted in time, flag broken or

corrupted sections, and support seismic analysis work by helping researchers organize and check waveform clips when the full continuous signal isn't perfectly recorded.

5. Conclusions

This paper proposed a self-supervised framework for learning waveform continuity by reformulating segment ordering as a jigsaw-style permutation prediction task. The method converts one-dimensional waveform sequences into standardized waveform images, applies randomized segmentation with enforced gaps, and trains a Siamese CNN to predict the shuffled order of segments using automatically generated pseudo-labels, eliminating the need for manual annotation. The approach was tested under three settings using three, four, and five segments. The model achieved near-perfect performance in the three-segment case, reached strong and stable results in the four-segment case with validation accuracy in the high 90% range, and maintained reasonable performance in the five-segment case with increased variability and a lower validation ceiling, reflecting the higher task difficulty. Overall, these results show that the proposed pipeline can reliably learn ordering cues from waveform morphology and provides a practical foundation for handling fragmented or partially observed waveform data. In real-world applications, this capability could support tasks such as organizing waveform clips, aligning separated waveform windows, and improving quality control when continuous recordings contain gaps or corrupted sections.

References

[1] Legrand, D., Bani, P., & Vergniolle, S. (2024). Investigating the potential influence of tectonic earthquakes on active volcanoes of Vanuatu. *Journal of Volcanology and Geothermal Research, 452*, 108139. https: //doi.org/10.1016/j.jvolgeores.2024.108139

[2] British Geological Survey. (n.d.). *How are earthquakes detected?* Retrieved February 9, 2026, from https://www.bgs.ac.uk/discovering-geology/earth-hazards/earthquakes/how-are-earthquakes-detected/

[3] Mousavi, S. M., Ellsworth, W. L., Zhu, W., Chuang, L. Y., & Beroza, G. C. (2020). Earthquake transformer—An attentive deep-learning model for simultaneous earthquake detection and phase picking. *Nature Communications*, *11*(1), 3952. https: //doi.org/10.1038/s41467-020-17591-w

[4] Kaushal, A., Gupta, A. K., & Sehgal, V. K. (2025). Earthquake prediction optimization using a deep learning hybrid RNN-LSTM model for seismicity analysis. *Soil Dynamics and Earthquake Engineering, 195*, 109432. https: //doi.org/10.1016/j.soildyn.2025.109432

[5] Zhu, W., & Beroza, G. C. (2018, March 8). *PhaseNet: A Deep-Neural-Network-Based Seismic Arrival Time Picking Method*. arXiv.Org. https: //arxiv.org/abs/1803.03211v1

[6] Apriani, M., Wijaya, S. K., & Daryono. (2021). Earthquake Magnitude Estimation Based on Machine Learning: Application to Earthquake Early Warning System. *Journal of Physics: Conference Series*, *1951*(1), 012057. https: //doi.org/10.1088/1742-6596/1951/1/012057

[7] Jain, M. (2022). *Machine Learning with Convolutional Neural Networks (CNNs) in Seismology for Earthquake Prediction* (SSRN Scholarly Paper No. 5261165). Social Science Research Network. https: //doi.org/10.2139/ssrn.5261165

[8] Berhich, A., Belouadha, F.-Z., & Kabbaj, M. I. (2022). A location-dependent earthquake prediction using recurrent neural network algorithms. *Soil Dynamics and Earthquake Engineering, 161*, 107389. https: //doi.org/10.1016/j.soildyn.2022.107389

[9] Shakeel, M., Nishida, K., Itoyama, K., & Nakadai, K. (2022). 3D Convolution Recurrent Neural Networks for Multi-Label Earthquake Magnitude Classification. *Applied Sciences*, *12*(4), 2195. https: //doi.org/10.3390/app12042195

[10] Chicco, D. (2021). Siamese Neural Networks: An Overview. In H. Cartwright (Ed.), *Artificial Neural Networks* (pp. 73–94). Springer US. https: //doi.org/10.1007/978-1-0716-0826-5_3