PAW: A Deep Learning Model for Predicting Amplitude Windows in Seismic Signals

Ariana M. Villegas Suarez*, Delaine Reiter†, Jonathan Rolfs†, and Abdullah Mueen*
* Department of Computer Science, University of New Mexico, USA
† Applied Research Associates, USA
E-mails: arianavillegas@unm.edu, dreiter@ara.com, jrolfs@ara.com, mueen@unm.edu

2024 IEEE 11th International Conference on Data Science and Advanced Analytics (DSAA) | 979-8-3503-6494-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/DSAA61799.2024.10722784

***Abstract*—Subsurface earthquakes and explosions generate seismic wavefields that are recorded as time-domain signals on sensor networks around the world. To compute key characteristics such as the magnitude of these seismic events, analysts must detect and select the cleanest indicators of seismic phase amplitudes and periods in noisy signals.**

Existing automated systems designed to pick seismic phase amplitudes and periods require frequent adjustments by human analysts, which becomes a nuisance when the volume of data to process grows large. To address this problem, we have developed a neural network model that accurately replicates the performance of a human analyst 80% of the time and shows potential for decreasing the correction workload for analysts by over 40%. We have performed multiple tests on the model and report on its performance compared to existing deep learning techniques.

***Index Terms*—Period, Time Series, Windows, Seismic Signal, Transformer**

I. Introduction

The accurate selection of seismic phase amplitudes and periods is a crucial data-processing task required to characterize underground events such as earthquakes, mining explosions, and nuclear tests. Although automated signal-processing algorithms exist to pick amplitudes and periods, human analysts must adjust measurements frequently. Figure 1 shows examples of seismic phases, where three existing algorithms (DETPRO [1], ALLSSA [2] and EQT [3]) fail to pick the amplitude window (i.e. the largest and the most uninterrupted half-cycle). A human analyst must identify the right amplitude window in order to properly measure the amplitude and period of the seismic signal, followed by a correct estimation of the magnitude of the event. The number of manual adjustments required to generate a high-quality event bulletin highlights the ongoing need for improved automated methods for accurate amplitude and period determination. To explain the industrial impact of the proposed model, consider a monitoring network such as IMS (International Monitoring System) that uses the DETPRO method to detect the amplitude windows [1]. The CTBTO (Comprehensive Nuclear-Test-Ban Treaty Organization) employs human analysts for three shifts a day to adjust the machine detection so that a high-quality bulletin can be produced daily. Often times the analysts experience backlogs due to major earthquakes and analyst shortages. This project aims at reducing this dependency on humans in a seismic monitoring network.

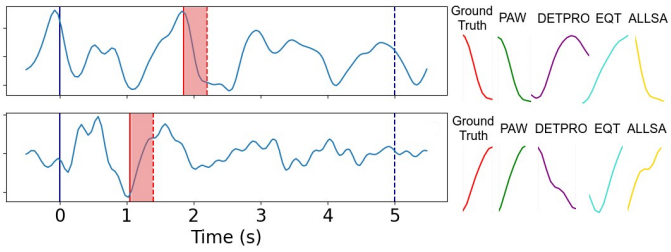

Fig. 1: Amplitude window contains the largest uninterrupted half-cycle (shaded as a red rectangle in each 5-second signal window). Although it is easy for a human to pick the correct half-cycle, existing algorithms frequently fail to find the ground truth by picking interrupted and/or partial cycles.

The challenge in learning from the analyst-labeled data is that the analysts make many heuristic choices depending on events that have happened before, geography of the earth, noise content of the signal, etc. In addition, there are subjective biases among the human analysts. Therefore, despite a straightforward definition of an amplitude window (as the largest and most uninterrupted half-cycle), the goal is to learn the heuristics analysts use in their decision-making process as opposed to devising an algorithm to find the half-cycle matching a mathematical definition.

In this paper, we design a deep architecture, PAW, to predict the amplitude window of a signal that contains what analysts identify as the cleanest half-cycle. This architecture is specifically designed to reduce the number of human interventions over existing automated predictions in a monitoring network. The architecture takes an encoder-decoder approach with large filter banks and a CNN-LSTM-Transformer combo in between the encoder and decoder. In addition, we exploit a dataset of over eighty thousand analyst-adjusted windows to calculate a loss function involving the amplitudes and periods derived from these windows. We demonstrate the proposed model can reproduce the windows picked by the analysts more than 80% of the time, and more importantly, can reduce the analysts' correction workload by 43.69%.

The uniqueness of our approach is that the proposed method is adaptive to the nuances analysts consider, while existing methods are deterministic to the signal. This is made possible by our special dataset of over one hundred thousand analyst-annotated signals and a very specific encoder-decoder architec-

ture, wider than existing models, that can capture the heuristics used by the analysts.

Our contributions are highlighted below and discussed in the following sections [1]:

- A high-quality proprietary dataset containing thousands of labeled half-cycles in equal-sized seismic time series.
- An encoder-decoder neural network model (PAW) that automatically detects an amplitude half-cycle in a seismic waveform.
- A custom loss function named *Amper Loss* to augment the learning of the model against multiple target metrics.
- Detailed sensitivity analysis regarding seismic sensor locations and signal properties.
- Comparison of our model against several state-of-the-art methods in terms of prediction performance.

II. Background

A. Seismic Signal Time Windows

For the data in our study, we extracted five-second windows that start at the onset time of the first-arriving compressional wave (i.e., a P or Pn arrival). Our labeled metadata also contains the *start time* of the phase amplitude window, which occurs either at or later than the onset time and should contain a clean amplitude half-cycle (i.e. $\cos x$ to $\cos(\pi + x)$, where $x \rightarrow 0$ or $x \rightarrow \pi$). This start time is initially generated by an existing algorithm named DETPRO. A human analyst examines the automatically picked start time and decides whether to adjust the amplitude window in favor of another window. This review process produces the label for each five-second signal window that consists of the final amplitude window start time, period, and amplitude (peak-to-trough or vice versa). The amplitude measurement window starts at *amptime* and ends at *amptime* + $\frac{1}{2}$ *per*. We note that the start time of the period (*amptime* - $\frac{1}{2}$ *per*) may not align precisely with the peak or trough of the signal. Figure 2 shows a visual representation of the amplitude window and period measurement.

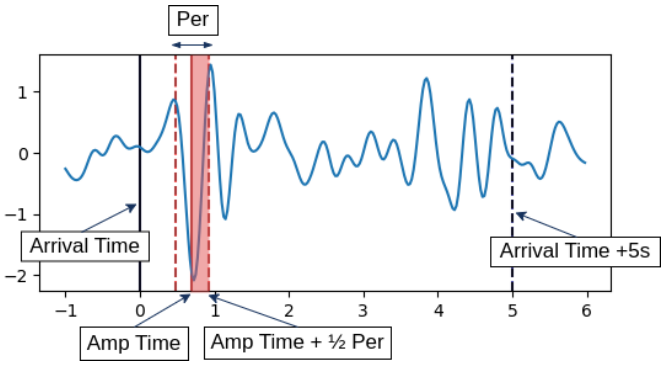

Fig. 2: Example showing a sample time window with the phase arrival onset time (black vertical line), start and end time of the selected amplitude window (red rectangle), and the end time of window (black dashed vertical line).

[1]We provide the dataset and the source code of PAW at https://github.com/ArianaVillegas/PAW.

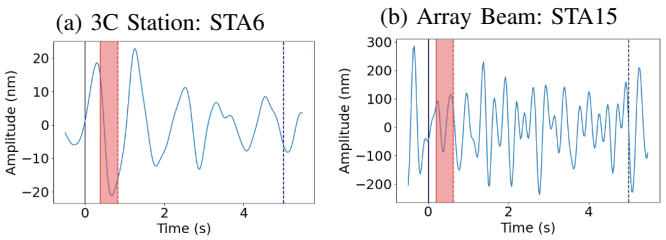

Fig. 3: Examples of unintuitive amplitude cycles in time-windowed seismic signals for 3C and array stations.

Although the amplitude measurement process is fairly straightforward, complications can arise (e.g. complex wave propagation velocities, inconsistencies across sensors in an array, etc.) that cause expert analysts to pick not-so-clean amplitude cycle windows. For example, in Figure 3 we show two time-window examples with unusual amplitude labels. We note that we do not have complete knowledge of the processing pipeline that produced these not-so-clean cases. However, the occurrences of these cases offer valuable insights into the variability of the amplitude selection process and challenge our learning system.

B. Our Dataset

Our dataset comprises multiple seismic event bulletins for years 2017 and 2018, each of which lists quantitative features extracted from seismic waveforms that were used to confirm seismic events at a particular location and time. The features in the test dataset are extracted from waveforms recorded on a global network of individual three-component (3C) stations and single-component, multi-station arrays. 3C stations record seismic data at one location on three channels oriented along vertical, north-south, and east-west directions, while arrays record data on a network of sensors arranged in a regular geometric pattern. Figure 4 shows the global distribution of seven 3C stations and eight arrays with usable information in our dataset.

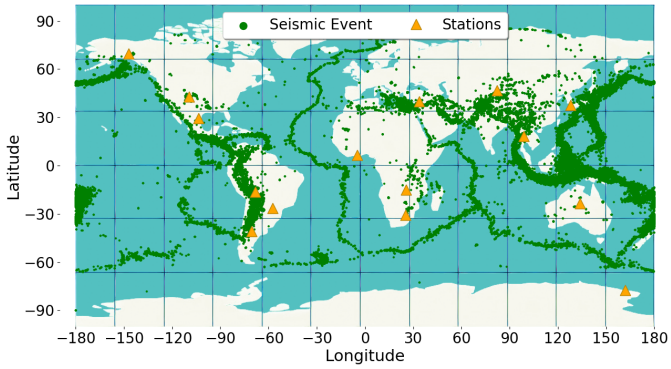

Fig. 4: Map showing the global distribution of arrays and 3-C stations in our curated dataset. The geographic locations of global seismic events from 2017 and 2018 are plotted as green dots.

To identify the waveform time windows needed to train our model, we filtered the fully automated and final analyst-

reviewed bulletins to find amplitude measurements that met several criteria, such as a certain type of amplitude measurement, seismic phase label, and other characteristics in the bulletin information. In our dataset, we have 80,648 seismic signal windows coming from 15 stations worldwide. To be specific, 10.67% of the data comes from 3C stations, while 89.33% comes from array beams. Additional details about the dataset is presented in Table I.

	Min	25th	50th	75th	Max
Start Time (s)	0.000	0.500	0.900	1.775	4.775
End Time (s)	0.175	0.850	1.250	2.150	5.000
Period (s)	0.300	0.550	0.700	0.800	1.450
Amplitude (nm)	0.000	0.900	1.990	5.470	3.263×10^4
Magnitude	3.500	3.780	4.030	4.380	6.710

TABLE I: Summary of dataset characteristics: start time, end time, period, amplitude, and magnitude features of seismic signal windows.

III. Deep Learning Model Description

A. Model Input

Our model input consists of a signal (i.e. one-dimensional time series), which we select following established guidelines for a specific station type. For 3C stations, we select the vertical-component channel. For arrays, we compute a beam signal that combines all array element (sub-station) signals according to an accepted shift-and-sum recipe. Each signal window has a five-second duration (i.e. 0 sec to 5 sec with respect to the phase onset time) to which we add a half additional second before and after the 5-second window, producing an input time series with 240 samples based on the waveform sample rate of 40 Hz.

B. Model Output

The output from the deep learning model is a step function with the same 240-sample length as the input, which is non-zero only where the model predicts a high probability of a clean amplitude half-cycle. This approach was decided after considering different output strategies, such as preserving the original input waveform values within the amplitude window or predicting a Gaussian distribution with its mean value in the middle of the amplitude window. We evaluated each output model empirically and concluded that a step-function produces the most accurate results (details are in the Experimental Results section).

To generate the final prediction for the amplitude window, we identify the maximum value in the predicted output to start the construction of the window. Then, we start building the window by extending both the left and right sides until we reach a value that is less than 0.5. Instead of choosing the maximum, one may consider learning a mapping function from the continuous prediction to the discrete windows, which we leave for future work to keep the proposed method simple.

C. Baselines

a) CNN Autoencoder.: The architecture, shown in Figure 5, consists of an encoder and a decoder. It processes a single-channel sequence of 240 inputs. The encoder includes three CNN blocks with output channels of 32, 64, and 32, using a kernel size of 13 and padding of 6. The decoder mirrors the encoder's structure, employing the same kernel sizes and number of output channels (32, 64, 32).

b) LSTM Autoencoder.: The architecture, depicted in Figure 6, comprises an encoder and a decoder. It handles a single-channel sequence of 240 inputs. The encoder consists of three one-directional LSTM blocks with two recurrent layers, having 64, 128, and 32 features in the state of each LSTM cell, respectively. The decoder mirrors the encoder's structure, utilizing LSTMs with hidden sizes of 32, 128, and 64.

D. Model Architecture

a) PAW (Predicting Amplitude Window).: Figure 7 shows the architecture of the PAW (Predicting Amplitude Window) model. It consists of an encoder and a decoder, with three block transformations in the latent space. The encoder processes a single-channel sequence with a length of 240. It includes two CNN blocks, with 32 and 64 output channels and a kernel size of 13. The output of the encoder is a feature vector of 60×64, which then undergoes additional transformations in the latent space. These transformations include a CNN block with 64 output channels and a kernel size of 13, followed by an LSTM block with 64 features in the hidden state, and end with a transformer block with 2 layers, each containing 8 heads. The decoder mirrors the structure of the encoder, maintaining the same number of layers and filter properties, but in reverse order. The main advantage of this model over previous ones lies in its latent space transformations, which prioritize width over depth. This design enriches the model's representations to boost performance while also enhancing parallelization by prioritizing width, leading to reduced forward time.

E. Loss Functions

a) Binary Cross-Entropy (BCE): It is a loss function that measures the difference in each point between the output label y (the step function) and the predicted output $\hat{y}$. It is expressed as equation 1, where N is the length of the sequence.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i) \quad (1)$$

b) Amper Loss.: Our loss function has two elements: the BCE loss and the Euclidean norm of two distinct loss components related to the period and amplitude. These components are the Mean Squared Error (MSE) computed between the expected and predicted values of the normalized period (denoted as per_loss) and the MSE between the expected and predicted values of normalized amplitude (denoted as amp_loss). Mathematically, the Amper loss is expressed as equation 3, where α is a hyperparameter that modulates the influence of the Euclidean norm component:

$$MSE_{tot} = [MSE_{amp}, MSE_{per}] \quad (2)$$

$$Amper\ Loss = BCE(\hat{y}, y) + \alpha ||MSE_{tot}||_2 \quad (3)$$

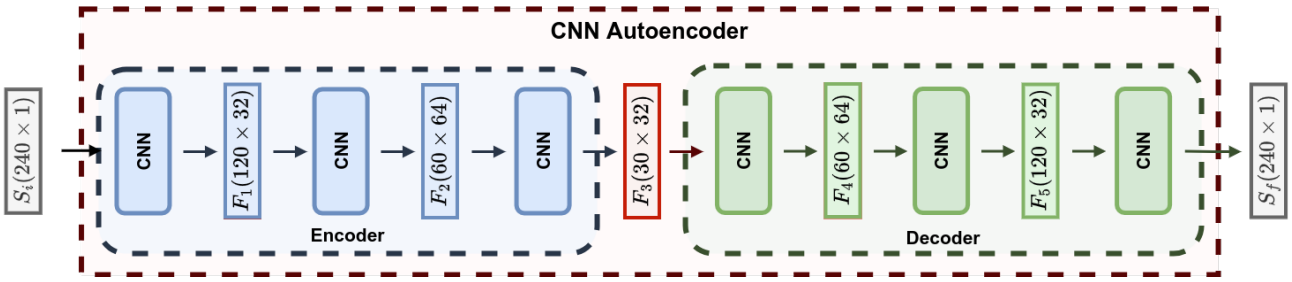

Fig. 5: Base CNN autoencoder consists of an encoder and a corresponding decoder. The encoder is constructed with 3 CNN blocks, while the decoder mirrors the structure of the encoder blocks.

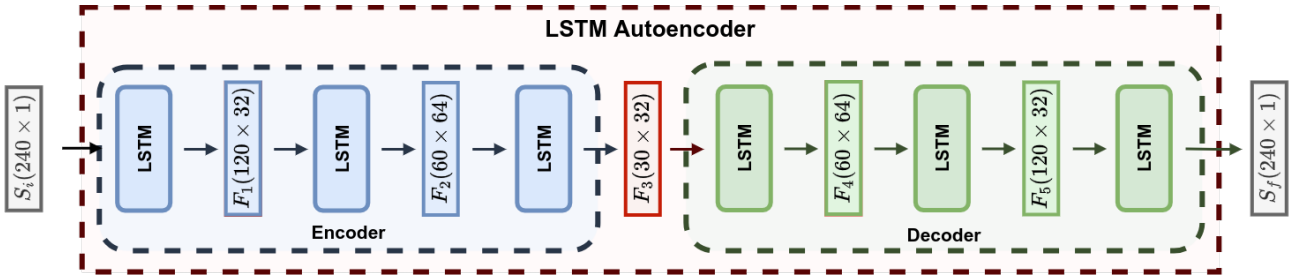

Fig. 6: Base LSTM autoencoder consists of an encoder and a corresponding decoder. The encoder is constructed with 3 LSTM blocks, while the decoder mirrors the structure of the encoder blocks.

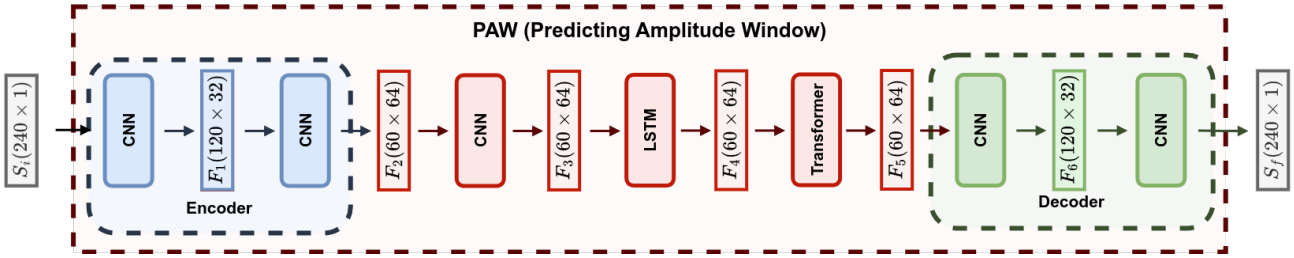

Fig. 7: PAW consists of an encoder, its corresponding decoder, and transformations in the latent space. The encoder is designed with 2 CNN blocks. In the latent space, there are CNN blocks, an LSTM block, and a transformer block with 8 heads.

The rationale behind adding the errors in predicted period and amplitude is to consider situations where we might predict the right amplitude but at the wrong time, or vice versa, predicting the wrong amplitude with the correct time window.

F. Model Performance Metrics

a) Window Root Mean Square Error (WRMSE): This metric computes the norm of errors between the predicted output ($\hat{y}_i$) and the label output (y_i). Equation 4 shows the formula we use to compute WRMSE. A lower WRMSE signifies a better fit between predicted and true values.

$$WRMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{4}$$

b) Window Accuracy (WA): The window accuracy metric measures the percentage of an output window that is predicted with an error no greater than a specified threshold (0.1 sec in our experiments). We compute WA by summing the differences between the predicted and true start times ($start\_err$) and the predicted and true end times (end_err) of the amplitude window, as shown in equation 5.

$$\text{CORRECT PREDICTION} = \begin{cases} 1 & \text{if } (start\_err + end\_err) \leq 0.1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{WA} = \frac{\text{\# CORRECT PREDICTIONS}}{\text{\# ALL PREDICTIONS}} \tag{5}$$

The value of window accuracy decreases as the predicted start or end time for the output amplitude window deviates from the true window start or end.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

To train and evaluate our deep learning models, we filtered out all accepted amplitude windows; i.e., we trained our models to learn the analyst-adjusted behaviors rather than the existing automated system predictions. We divide the remaining dataset (48,589 seismic signals) into three distinct subsets, with the following distribution: 70% for training, 20% for validation, and 10% for testing. Each experiment was repeated ten times, and we subsequently computed both

Model details					RMSE				Window Accuracy		
Name	**Size (MB)**	**Forward (ms)**	**Loss**	**Train Time**	**Window**	**Amp**	**Per (s)**	**Mag**	**Test**	**Adj**	**Acc**
Classical Methods											
DETPRO [1]	-	-	-	-	0.1596	0.1143	0.3818	0.4572	64.20%	18.30%	100.00%
ALLSSA [2]	-	1.7232	-	-	0.2173	0.2415	0.6930	0.8975	28.37%	27.28%	30.29%
Regressors											
GDP [4]	6.958	1.2328	MSE	0:28:41	0.2529	0.2608	0.5715	1.0049	20.30%	17.50%	23.76%
DPP [5]	0.492	0.9678	MSE	0:31:17	0.2844	0.3597	0.4147	1.0847	14.08%	12.04%	16.76%
Autoencoders											
Basic AE [6]	0.134	0.8181	BCE	0:57:18	0.1582	0.1117	0.4442	0.5056	62.45%	56.67%	70.21%
			Amper	0:34:37	0.1575	0.1097	0.4332	0.4549	64.41%	58.79%	72.10%
GPD AE [4]	7.278	1.3313	BCE	0:22:27	0.1591	0.1079	0.3826	0.4287	68.87%	64.48%	76.95%
			Amper	0:20:31	0.1581	0.1076	0.3794	0.4523	69.04%	64.72%	77.05%
DPP AE [5]	0.540	0.9034	BCE	0:31:04	0.1649	0.1125	0.4226	0.4684	62.44%	56.02%	69.70%
			Amper	0:28:21	0.1605	0.1119	0.4828	0.4714	62.64%	55.95%	70.55%
CRED [7]	1.108	0.9526	BCE	0:32:41	0.1513	0.1008	0.2513	0.3844	74.62%	66.23%	82.66%
			Amper	0:25:56	0.1522	0.1000	0.2494	0.3814	75.44%	67.43%	82.89%
EQT [3]	1.508	1.0031	BCE	0:45:10	0.1605	0.1162	0.4004	0.5074	60.84%	55.46%	68.19%
			Amper	0:39:53	0.1586	0.1116	0.3897	0.4994	64.31%	58.44%	72.58%
Dyna [8]	16.2	6.1167	BCE	6:27:49	0.1594	0.1563	0.1802	0.3832	73.99%	68.17%	78.57%
			Amper	5:38:59	0.1586	0.1542	0.1799	0.3795	74.51%	68.12%	80.97%
SeisT [9]	1.513	2.1663	BCE	1:30:31	0.1527	0.1040	0.2014	0.3746	74.78%	68.33%	81.17%
			Amper	1:13:43	0.1502	0.1021	0.1946	0.3708	76.06%	69.52%	82.74%
Our Autoencoders											
CNN AE	0.431	1.0183	BCE	0:31:10	0.1515	0.1056	0.1813	0.3931	73.37%	65.32%	82.30%
			Amper	0:21:39	0.1510	0.1024	0.1739	0.3914	73.93%	66.27%	83.47%
LSTM AE	0.189	3.2514	BCE	5:09:39	0.1681	0.2249	0.2770	0.5277	45.53%	48.56%	48.47%
			Amper	1:39:48	0.1635	0.2235	0.2568	0.5218	55.65%	48.75%	61.55%
PAW	2.596	1.3640	BCE	0:59:02	0.1463	0.1002	0.1704	0.3760	79.22%	72.60%	86.16%
			Amper	0:37:03	**0.1445**	**0.0983**	**0.1686**	**0.3573**	**80.04%**	**73.28%**	**87.28%**

TABLE II: Method comparison: classical methods, state-of-the-art methods (regressors and autoencoders), and our autoencoders (including PAW). Results cover various loss functions (MSE, BCE, Amper) and metrics like model size, training time, window RMSE, amplitude RMSE, period RMSE, magnitude RMSE, and window accuracy (within 0.1 seconds error). The PAW model, using the Amper loss function, achieves superior performance across the evaluated metrics.

the average and standard deviation of our results. To ensure consistency, the same seed was used for each repetition, resulting in identical subset distribution and network initialization. We utilized several metrics for evaluation, including window RMSE, amplitude RMSE, period RMSE, and Window Accuracy on the test set, which we divided into adjusted and accepted subsets. We performed all computations on a desktop computer equipped with an Intel Core i7 processor running at 3.40 GHz.

We used Python 3 and the PyTorch library to develop and test the models. The hyperparameters of the model were fine-tuned through empirical selection via random search on the validation set. We trained for a maximum of 200 epochs, utilizing a batch size of 32 and we configure the Adam optimizer with an initial learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-7}$. Finally, we saved the model with the lowest validation WRMSE for further evaluation.

B. Model Comparison

To evaluate the performance of our PAW model, we compared it against seven state-of-the-art methods (GDP [4], DPP [5], Basic AE [6], CRED [7], EQTransformer [3], DynaPicker [8], SeisT [9]), along with basic CNN and LSTM autoencoders for reference. GDP and DPP were utilized as regressors for predicting the start and end times of the amplitude window, both trained with the MSE loss function. Additionally, we adapted GDP, DPP, CRED and DynaPicker to serve as autoencoders, while using the existing autoencoder architecture of Basic AE, EQTransformer and Seist. These autoencoders were trained using both the BCE loss function (1) and our Amper loss function (3).

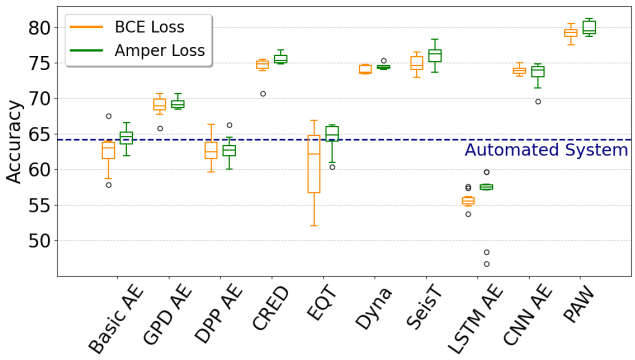

Fig. 8: Comparison of test window accuracy, including state-of-the-art methods (Basic AE, GPD AE, DPP AE, CRED, EQT, DynaPicker, SeisT) and our autoencoder models (CNN AE, LSTM AE, PAW) under BCE and Amper loss functions.

The preliminary findings of window accuracy for the autoencoder models are shown in Figure 8, trained using both BCE loss and our custom Amper loss. GDP AE, CRED, DynaPicker, SeisT, CNN autoencoder, and the PAW model achieved higher test window accuracy compared to DETPRO. However, Basic AE, DPP AE, and EQT models intersected

with DETPRO's window accuracy, while the LSTM AE model performed worse than the automated system, suggesting these methods didn't surpass DETPPRO and displayed greater variability. Overall, our Amper loss function consistently outperformed the BCE loss function.

In specific, our PAW model trained with Amper loss achieves the highest window accuracy (80.04%) among all methods. Although domain experts must verify all predictions, the increased accuracy of our model reduces the number of incorrect predictions they encounter. As a result, the time and effort required to correct these errors decreases significantly, which reduces the proportion of measures needing adjustment from 36% with DETRO to 20%, and thus reduces the correction workload by 43.69%.

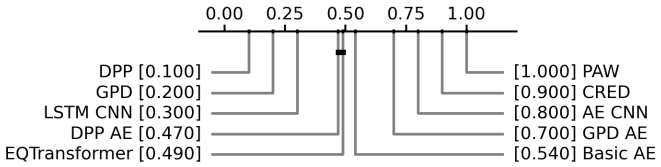

Fig. 9: Critical difference diagram comparing test window accuracy. The results are from the model using the loss function (Amper loss, BCE loss, MSE loss) that achieved the highest mean test accuracy across 10 repetitions.

Figure 9 illustrates the critical difference (CD) diagram comparing test accuracy based on average ranks, shown in brackets. The PAW model stands out with the highest rank of 1.0, showing its superior performance. Horizontal bars indicate statistically significant differences. Models connected by a line do not differ significantly, while those without a connecting line do. The absence of connecting lines for the PAW model highlights the statistical significance of its performance.

Table II expands the comparison between the PAW model and other existing methods, including the automated system and a frequency domain method called ALLSSA [2], as well as state-of-the-art models (regressors and autoencoders), and our autoencoders. The model details used for comparison include the model size, forward time, training time, and loss functions. Additionally, performance metrics such as window RMSE, amplitude RMSE, period RMSE, magnitude RMSE, and test Window Accuracy are included. The division into adjusted and accepted subsets for Window Accuracy allows for a more comprehensive assessment of the results.

Furthermore, our evaluation covers RMSE analysis across various aspects of the results. Firstly, we focus on WRMSE, which reflects the error in the methods' output. Secondly, we analyze amplitude, period, and magnitude RMSE, which hold greater significance in the magnitude detection task. Across all these RMSE metrics, our PAW model with the Amper Loss achieves the best results. The PAW model with BCE loss follows closely, outperforming the others in RMSE metrics, except for the case of amplitude RMSE.

Our PAW model, trained with the Amper loss function, emerges as the top performer across all metrics, surpassing previous transformer based models like EQT and SeisT in the state-of-the-art. Despite its larger size, our model achieves similar training times than previous models and a lower forward time than SeisT. This efficiency is due to our focus on width over depth. Prioritizing width simplifies parallelization during training, resulting in faster training. When we prioritize width over depth, it means that while EQT has 2 blocks of 4 layers with 4 heads each after fine-tuning, and SeisT consists of 4 blocks of 2 layers with 4 heads, our model consists of 1 transformer block with 2 layers, and 8 heads. In addition, when utilizing the Amper loss function, the training is accelerated, leading to quicker convergence and improved window accuracy.

Moreover, it is important to highlight the value of accurate predictions for both amplitude and period. Any inaccuracies in either of these aspects can significantly impact the overall magnitude estimation, leading to incorrect global results. Additionally, our comparison emphasizes the consistent challenge of predicting adjusted windows compared to accepted ones. This difficulty likely stems from human bias introduced by analysts when adjusting amplitude windows.

C. Window Accuracy Error

To further highlight the window accuracy of the new PAW model in predicting correct amplitude windows versus the existing automated system, we computed the window accuracy error between labeled (i.e., 'true') analyst-adjusted windows versus windows produced by DETPRO and those predicted by the PAW model. Figure 10 shows the results of this comparison, which demonstrate that the PAW model consistently outperforms the existing automated system DETPRO in predicting the amplitude windows, regardless of specific window error tolerance levels. Further, the PAW model starts to achieve high performance at a low window-error level. At higher window error levels, the PAW model converges towards a window accuracy plateau of approximately 80%. We also note that the PAW model's standard deviation on window accuracy remains notably low across most values.

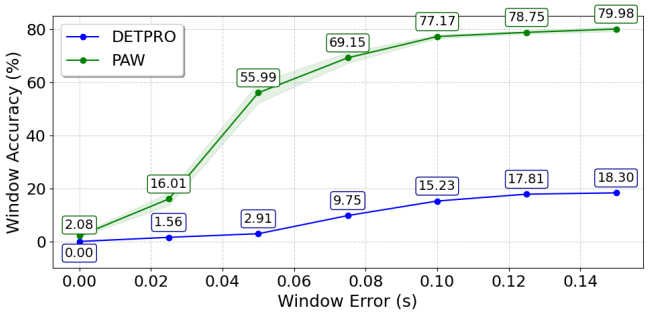

Fig. 10: Predicted Window Accuracy (WA) as a function of amplitude measurement window (in seconds) for DETPRO versus the PAW model. These values are computed between the analyst-adjusted amplitude windows and the respective models for different window error thresholds (in seconds).

D. Station Holdout Tests

To assess the generalization capabilities of the PAW model to new stations, we performed a *station holdout* evaluation. This involved excluding all window samples from a particular station and using the remaining data from all other stations for training. We then evaluated the performance of the PAW model on the station left out of training.

STA	Adjusted	Accepted	Proportion	WA
3C Stations				
STA1	33.91%	66.09%	1.74%	78.84%
STA2	36.36%	63.64%	0.99%	71.77%
STA3	26.61%	73.39%	1.24%	82.92%
STA4	25.44%	74.56%	1.14%	81.55%
STA5	33.75%	66.25%	1.60%	76.12%
STA6	25.00%	75.00%	1.68%	80.67%
STA7	28.95%	71.05%	2.28%	79.05%
Array Stations				
STA8	34.81%	65.19%	22.84%	**84.49%**
STA9	32.97%	67.03%	4.64%	76.05%
STA10	51.92%	48.08%	10.17%	**66.32%**
STA11	37.11%	62.89%	15.17%	77.04%
STA12	36.61%	63.39%	4.89%	80.35%
STA13	34.42%	65.58%	16.50%	81.17%
STA14	33.89%	66.11%	7.82%	77.58%
STA15	33.01%	66.99%	7.30%	79.93%

TABLE III: Station labels, their respective percent distribution of analyst-adjusted versus accepted amplitude windows, proportion of the dataset, and their results from station holdout tests. PAW was trained using analyst-adjusted windows while systematically holding out each station. The window accuracy at each held-out station shows the capability of the model to generalize effectively.

Table III displays the percent distribution of amplitude windows for each of the 15 stations in our data archive as a function of 'accepted' (analyst agreed with the automated selection) and 'adjusted' (analyst adjusted the automated selection). A higher proportion in the fourth row indicates a greater number of window amplitude samples. Approximately 89.33% of our dataset comes from the arrays, with STA8, STA11, and STA13 stations dominating the overall distribution. In addition, 3C stations have a higher proportion of amplitude measurements produced by the automated system.

In this experiment, PAW achieved an average window accuracy of 78.26%, surpassing the automated system DETPRO. We note an interesting effect on model accuracy derived from the ratio of adjusted to accepted amplitude windows. Station STA10 shows lower window accuracy, mainly because it has a higher percentage of analyst-adjusted windows (52%) compared to accepted ones (48%), which introduces human biases. PAW trained without STA10 windows correctly predicts 66.3 % of the STA10 amplitude windows. Despite this challenge, PAW still significantly improved upon the 48% window accuracy of the automated system at STA10.

In contrast, station STA8 outperformed other stations in station-holdout tests, because it has a low percentage of adjusted windows. This result suggests that analyst window adjustments for this station were aligned closely with the other stations used in training, although further investigation into this case is needed to uncover underlying factors.

E. Output Design Strategy

To shape the decoder's output, we aim for a strategy that highlights the amplitude window. We experiment with three different options:

- Waveform: $f(x) = x$, if x is in the window, and 0 otherwise.
- Gaussian: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$, if x is in the window, and 0 otherwise. Here $\mu = \frac{start+end}{2}$ and $\sigma = 3 \cdot \frac{period}{2}$
- Step: $f(x) = 1$, if x is in the window, and 0 otherwise.

The results are shown in the Table V. The Step function outperforms Waveform and Gaussian outputs in window RMSE, magnitude RMSE, and test window accuracy (split into adjusted and accepted subsets). Additionally, Table V confirms consistency with previous findings, highlighting that predicting adjusted windows is more challenging than predicting accepted windows that the existing method had already predicted correctly.

F. Ablation Study.

To further demonstrate the capabilities of the PAW model, we conducted an ablation study to evaluate individual components of the model and assess their impact on overall performance. To do this, we subdivided the PAW model into its individual CNN-autoencoder, CNN, LSTM, and transformer components. We also assessed the need for a symmetric structure and skip connections within the PAW model. Our evaluation compared window RMSE, amplitude RMSE, period RMSE, window accuracy in the test subset, window accuracy in analyst-adjusted amplitude windows, and accepted amplitude window accuracy. Table IV summarizes the results of this ablation study.

Our findings indicate that each model learning component contributes to the PAW model's overall performance and ability to generalize effectively in predicting accepted amplitude windows in new seismic signals. In addition, the PAW model has the lowest values for all RMSE metrics and the highest window accuracy across the three testing scenarios, which demonstrates its robustness. We also observe the strongly detrimental effect on model performance incurred by symmetry, which highlights the benefits of the asymmetric structure incorporated into the PAW model. While skip connections appear to have a negative impact, further investigation with a larger dataset is necessary to fully understand their effects.

G. PAW Model Limitations.

Figure 11 presents another view of the results from our study, in which we compare histograms of key outputs from the existing automated system DETPRO, the labeled (analyst-adjusted) data, and the PAW model. The outputs shown in

PAW Model Components					RMSE				Window Accuracy		
CNN AE	LSTM	Transformer	Symmetry	Skip Connections	Window	Amp	Per (s)	Mag	Test	Adj	Acc
✓					0.1537	0.1015	0.2158	0.3804	75.54%	71.05%	80.48%
✓	✓				0.1526	0.0999	0.2024	0.3783	76.83%	72.25%	80.14%
✓		✓			0.1514	0.0994	0.2107	0.3792	76.04%	69.92%	82.19%
✓	✓		✓		0.2712	0.2864	0.5386	1.0083	18.31%	14.43%	22.67%
✓	✓	✓		✓	0.1493	0.1091	0.2015	0.3632	77.64%	71.19%	84.11%
✓	✓	✓			**0.1445**	**0.0983**	**0.1686**	**0.3573**	**79.84%**	**73.28%**	**87.28%**

TABLE IV: Results from the PAW model ablation study, including evaluation of window RMSE, amplitude RMSE, period RMSE, magnitude RMSE, test window accuracy, adjusted window accuracy, and accepted window accuracy. The presence of a specific PAW model component is indicated by an '✓'.

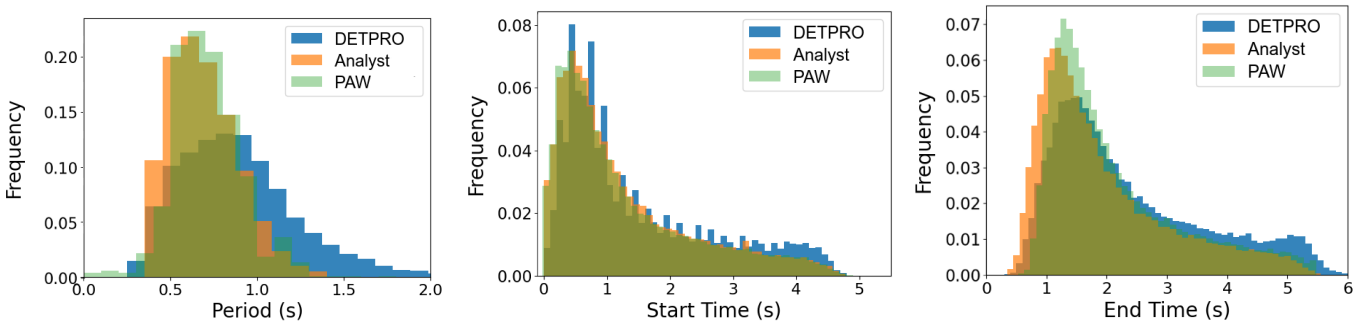

Fig. 11: Comparative histograms of period, start time, and end time for the existing automated system method DETPRO, analyst-adjusted (ground truth), and the PAW model.

	RMSE		Window Accuracy		
Function	Window	Mag	Test	Adj	Acc
Waveform	0.1621	0.4857	61.08%	60.01%	61.29%
Gaussian	0.1517	0.3798	76.52%	74.03%	77.49%
Step	**0.1445**	**0.3573**	**79.84%**	**73.28%**	**87.28%**

TABLE V: Comparison of the output design strategies (waveform, gaussian, step) for the PAW model using the Amper loss function. Evaluation includes window RMSE, magnitude RMSE, and test window accuracy (divided into adjusted and accepted subsets).

the figure include the period, start time, and end time for the amplitude windows. The results exhibit a close match between the PAW model and adjusted selections, particularly for amplitude-window start times. However, we also notice slight differences in the period distributions, where our model shows smaller gaps between peaks and troughs. This discrepancy might be caused by several factors, including learning model errors or inaccuracies in the original labels, and we will continue to investigate its cause.

Figure 12 shows that correct and incorrect predictions have similar distribution proportions across different values of magnitude, depth, or distance. While small differences exist, the overall trend suggests that amplitude window predictions are independent of these factors (magnitude, depth, or distance). In addition, we observe that magnitude values are primarily clustered between 3.5 to 4.5, depth values are concentrated in the range of 0 to 50, and distance values are centered around 5000 and 10000.

The PAW model sometimes predicts the amplitude window incorrectly. Figure 13 shows four cases of inaccurate model

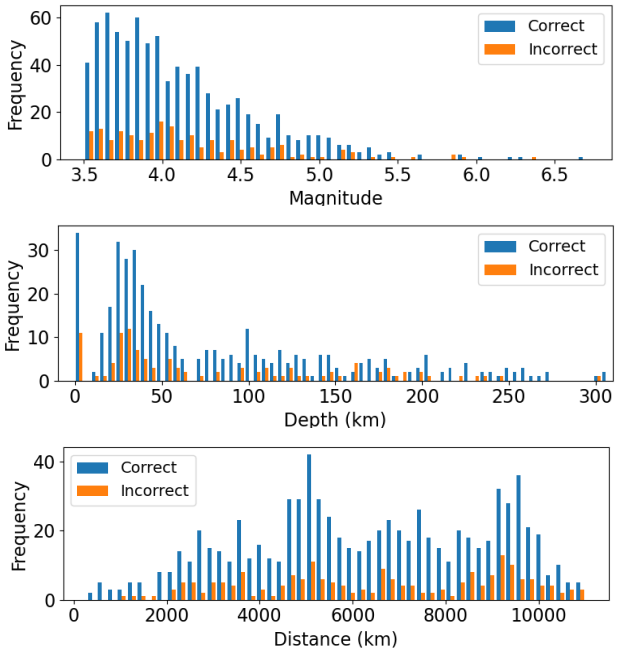

Fig. 12: Histograms comparing the frequency of correct and incorrect predictions based on varying conditions such as magnitude, depth, and distance.

behavior, chosen to highlight different types of PAW model failures. For example, in Figure 13a, our model confidently predicts a longer period and amplitude, while Figure 13c illustrates a case where the model chooses a window with a shorter amplitude but higher confidence, while also displaying some confidence in the labeled window as a secondary option. Figure 13b highlights a case where the model shows uncer-

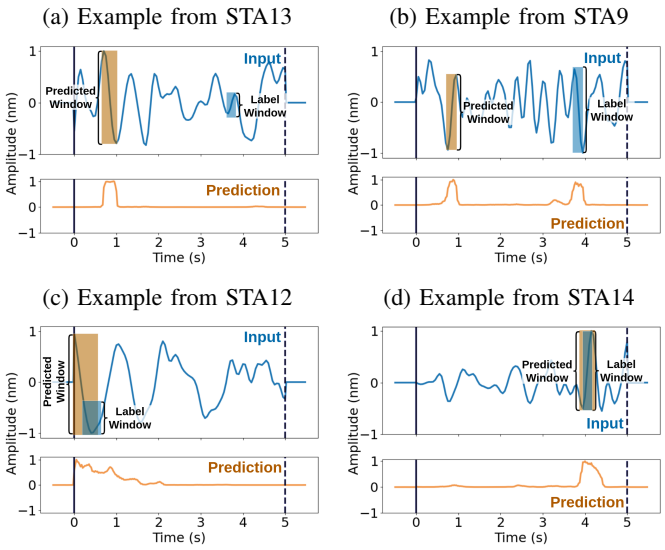

Fig. 13: Examples of inaccurate amplitude results predicted by the PAW model; (a) PAW predicted a longer period and amplitude than the labeled value; (b) PAW predicted a shorter amplitude than the labeled value; (c) PAW picked an amplitude window with a larger period and higher amplitude; (d) PAW predicted a larger amplitude window containing the labeled window.

tainty in its decision. Lastly, Figure 13d illustrates a scenario where the model predicts a larger amplitude window, primarily driven by the lack of confidence to select an exact half-cycle window. Many of these cases may be caused by differences in the data-processing techniques we used to generate our 240-sample inputs for the PAW model compared to those used to prepare the event bulletins.

V. Related Work

Deep learning plays an important role in seismology, particularly in earthquake pattern recognition [10], [11] and prediction of seismic indicators [12]–[16]. Feed-forward models combined with decision trees [17], [18] achieved high accuracy in detecting seismic features with Chilean data. Likewise, in regional contexts like Indonesia, earthquake magnitude prediction with deep neural networks and hierarchical K-means clustering [19] achieved varying success.

Time Series Regression Methods. A variety of methods, comprising CNNs [20]–[22], RNNs [23]–[25], LSTMs [26]–[28], and transformers [29]–[31], can be applied to time series regression tasks such as traffic speed prediction, renewable energy generation modeling, financial market analysis, and earthquake depth estimation [4], [5]. DeepPhaseNet [5] (DPP) combines a CNN with two RNNs, it was designed for detecting and picking local events on a regional seismic network in Chile. Generalized Phase Detection [4] (GPD) is a CNN model that can detect seismic events and classify seismic phases on the Southern California Seismic Network.

Time Series Autoencoders. Autoencoders applied to time series have diverse applications such as anomaly detection [32], [33], classification [34], [35], denoising [36], [37], dimensionality reduction [38], [39], feature extraction [40], [41] and earthquake detection [3], [7]. CRED [7] combines CNN and RNN layers to detect events in signal and noise examples from northern California. EQTransformer [3] (EQT) is an attention-based transformer network with CNN and LSTM layers to both detect and pick events.

VI. Conclusions

We have developed a novel, autoencoder-based model (PAW) that accurately replicates analyst selections of seismic phase amplitudes and periods in a waveform. Assessments of the window accuracy of the new PAW model indicate that it achieves an overall window accuracy of nearly 80% in predicting key amplitude characteristics, including the start and end times of the correct amplitude windows, and amplitude, period, and magnitude values. This represents a strong improvement over the performance of the existing automated system. In future work, we will continue to pursue performance improvements with the PAW model and evaluate its performance on other labeled seismic phase-amplitude datasets.

References

[1] "IDC Documentation Products and Services: Formats and Protocols for Continuos Data CD-1.0 /," p. 37 p. :, 2 2002.

[2] E. Ghaderpour, W. Liao, and M. P. Lamoureux, "Antileakage least-squares spectral analysis for seismic data regularization and random noise attenuation," *Geophysics*, vol. 83, no. 3, 2018.

[3] S. M. Mousavi, W. L. Ellsworth, W. Zhu, L. Y. Chuang, and G. C. Beroza, "Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking," *Nature communications*, vol. 11, no. 1, p. 3952, 2020.

[4] Z. E. Ross, M. Meier, E. Hauksson, and T. H. Heaton, "Generalized Seismic Phase Detection with Deep Learning," *Bulletin of the Seismological Society of America*, vol. 108, no. 5A, pp. 2894–2901, 2018.

[5] H. Soto and B. Schurr, "DeepPhasePick: a method for detecting and picking seismic phases from local earthquakes based on highly optimized convolutional and recurrent deep neural networks," *Geophysical Journal International*, vol. 227, no. 2, pp. 1268–1294, 2021.

[6] J. Woollam, A. Rietbrock, A. Bueno, and S. De Angelis, "Convolutional neural network for seismic phase classification, performance demonstration over a local seismic network," *Seismological Research Letters*, vol. 90, no. 2A, pp. 491–502, 2019.

[7] S. M. Mousavi, W. Zhu, Y. Sheng, and G. C. Beroza, "Cred: A deep residual network of convolutional and recurrent units for earthquake signal detection," *Scientific reports*, vol. 9, no. 1, p. 10267, 2019.

[8] W. Li, J. Koehler, M. Chakraborty, C. Quinteros-Cartaya, G. Ruempker, and N. Srivastava, "Real-time earthquake monitoring using deep learning: a case study on turkey earthquake aftershock sequence," 2023.

[9] S. Li, X. Yang, A. Cao, C. Wang, Y. Liu, Y. Liu, and Q. Niu, "Seist: A foundational deep-learning model for earthquake monitoring tasks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–15, 2024.

[10] S. Kanarachos, S.-R. G. Christopoulos, A. Chroneos, and M. E. Fitzpatrick, "Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and hilbert transform," *Expert Systems with Applications*, vol. 85, pp. 292–304, 2017.

[11] M. Mirrashid, "Earthquake magnitude prediction by adaptive neuro-fuzzy inference system (anfis) based on fuzzy c-means algorithm," *Natural Hazards*, vol. 74, pp. 1577–1593, 2014.

[12] F. A. Chowdhury, M. A. Siddiquee, G. E. Baker, and A. Mueen, "FASER: Seismic Phase Identifier for Automated Monitoring," in *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2021* (F. Zhu, B. C. Ooi, and C. Miao, eds.), pp. 2714–2721, ACM, 2021.

[13] S. M. Mousavi, W. Zhu, Y. Sheng, and G. C. Beroza, "CRED: A Deep Residual Network of Convolutional and Recurrent Units for Earthquake Signal Detection," *Scientific Reports*, vol. 9, p. 10267, 12 2019.

[14] Y. Wang, Y. Chen, and J. Zhang, "The application of rbf neural network in earthquake prediction," in *2009 Third International Conference on Genetic and Evolutionary Computing*, pp. 465–468, 2009.

[15] W. S. Hu, H. Wang, and H. L. Nie, "Regional short-term earthquake prediction model based on bp neural network," in *Progress in Structures*, vol. 166 of *Applied Mechanics and Materials*, pp. 2309–2314, 2012.

[16] E. Amar, T. Khattab, and F. Zada, "Intelligent Earthquake Prediction System Based On Neural Network," *International Journal of Earth, Energy and Environmental Sciences*, vol. 8.0, 2015.

[17] E. Florido, G. Asencio–Cortés, J. Aznarte, C. Rubio-Escudero, and F. Martínez–Álvarez, "A novel tree-based algorithm to discover seismic patterns in earthquake catalogs," *Computers Geosciences*, vol. 115, pp. 96–104, 2018.

[18] J. Reyes, A. Morales-Esteban, and F. Martínez-Álvarez, "Neural networks to predict earthquakes in chile," *Applied Soft Computing*, vol. 13, no. 2, pp. 1314–1328, 2013.

[19] M. N. Shodiq, D. H. Kusuma, M. G. Rifqi, A. R. Barakbah, and T. Harsono, "Neural network for earthquake prediction based on automatic clustering in indonesia," *JOIV: International Journal on Informatics Visualization*, vol. 2, no. 1, pp. 37–43, 2018.

[20] Y. Wang, X. Li, Z. Wang, and J. Liu, "Deep learning for magnitude prediction in earthquake early warning," *Gondwana Research*, 2022.

[21] A. Gasparin, S. Lukovic, and C. Alippi, "Deep learning for time series forecasting: The electric load case," *CAAI Transactions on Intelligence Technology*, vol. 7, no. 1, pp. 1–25, 2022.

[22] P. Lara-Benítez, M. Carranza-García, J. M. Luna-Romera, and J. C. Riquelme, "Temporal convolutional networks applied to energy-related time series forecasting," *Applied Sciences*, vol. 10, no. 7, 2020.

[23] R. Chandra and S. Chand, "Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance," *Applied Soft Computing*, vol. 49, pp. 462–473, 2016.

[24] H. M, G. E.A., V. K. Menon, and S. K.P., "Nse stock market prediction using deep-learning models," *Procedia Computer Science*, vol. 132, pp. 1351–1362, 2018. International Conference on Computational Intelligence and Data Science.

[25] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021.

[26] H. Chung and K.-s. Shin, "Genetic algorithm-optimized long short-term memory network for stock market prediction," *Sustainability*, vol. 10, no. 10, 2018.

[27] L. Peng, S. Liu, R. Liu, and L. Wang, "Effective long short-term memory with differential evolution algorithm for electricity price prediction," *Energy*, vol. 162, pp. 1301–1314, 2018.

[28] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.

[29] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," vol. abs/2001.08317, 2020.

[30] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, *Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting*. 2019.

[31] N. Phandoidaen and S. Richter, "Forecasting time series with encoder-decoder neural networks," 2020.

[32] H. Gao, B. Qiu, R. J. Duran Barroso, W. Hussain, Y. Xu, and X. Wang, "Tsmae: A novel anomaly detection approach for internet of things time series data using memory-augmented autoencoder," *IEEE Transactions on Network Science and Engineering*, 2022.

[33] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, "Anomaly detection based on convolutional recurrent autoencoder for iot time series," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 1, pp. 112–122, 2022.

[34] P. Liu, X. Sun, Y. Han, Z. He, W. Zhang, and C. Wu, "Arrhythmia classification of lstm autoencoder based on time series anomaly detection," *Biomedical Signal Processing and Control*, vol. 71, p. 103228, 2022.

[35] D.-H. Kang and D.-H. Kim, "1d convolutional autoencoder-based ppg and gsr signals for real-time emotion classification," *IEEE Access*, vol. 10, pp. 91332–91345, 2022.

[36] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems," *Scientific reports*, vol. 9, no. 1, p. 19038, 2019.

[37] A. Novoselov, P. Balazs, and G. Bokelmann, "Sedenoss: Separating and denoising seismic signals with dual-path recurrent neural network architecture," *Journal of Geophysical Research: Solid Earth*, vol. 127, no. 3, p. e2021JB023183, 2022.

[38] G. Stein, U. Seljak, V. Böhm, G. Aldering, P. Antilogus, C. Aragon, S. Bailey, C. Baltay, S. Bongard, K. Boone, *et al.*, "A probabilistic autoencoder for type ia supernova spectral time series," *The Astrophysical Journal*, vol. 935, no. 1, p. 5, 2022.

[39] S. N. Shukla and B. M. Marlin, "Heteroscedastic temporal variational autoencoder for irregularly sampled time series," vol. abs/2107.11350, 2021.

[40] Z. Li, Z. Rao, L. Pan, P. Wang, and Z. Xu, "Ti-mae: Self-supervised masked time series autoencoders," 2023.

[41] L. Yang, Y. Song, S. Gao, A. Hu, and B. Xiao, "Griffin: Real-time network intrusion detection system via ensemble of autoencoder in sdn," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2269–2281, 2022.