USING LONG SHORT-TERM MEMORY MODEL VARIANTS TO SIMULATE SEISMIC WAVES

Students : 110612066 張誌剛 , 110612008 沈昱翔 , 110612158 李明翰 , 110612035 劉沛宸

Supervisor: 洪士林 教授

1. MOTIVATION

Earthquakes represent one of the most perilous threats to human life. Annually, seismic activity results in considerable loss of life and economic damage. Additionally, earthquakes affect our infrastructure and culture, it has been observed that cities and villages situated in earthquake zones are often required to have some compromise to ensure the safety of their residents. Residents must take special measures, such as earthquake-resistant structures or seismic design. Furthermore, they must conduct earthquake drills more frequently than the general population. By developing AI models that are capable of more accurately earthquakes, we can provide warnings prior to the occurrence of an earthquake. This will enable residents to take the necessary and ensure the optimal allocation of resources including aid and rescue teams. Simultaneously, this advancement of optimistic models for earthquake prediction can contribute to more profound comprehension of the Earth's seismic activities and enhance our ability to address other natural disasters.

2. ASSUMPTION

Since earthquakes are represented by time-series waveforms, we can use Fast Fourier Transform (FFT) to understand the frequency distribution that makes up the waveform. Therefore, we can assume that if the frequencies of each earthquake follow a certain pattern, learning the characteristics of earthquakes could enable us to simulate them.

2.1. Fourier Transform

There is a function $f(x)$ with period 2π. There is an assumption if the function can be formed by $\sin nx$, $\cos nx$:

$$f(x) = a_0 + \sum_{n=1}^{\infty} a_n \, cosnx + b_n sinnx \qquad (1)$$

After derivation, replacing 2π by $2p$:

$$f(x) = a_0 + \sum_{n=1}^{\infty} a_n \, cos\frac{nx\pi}{p} + b_n sin\frac{nx\pi}{p} \qquad (2)$$

By the result, the hypothesis is proved.

Value: converting a data using time as variable to data using frequency as variable, which represents energy density under every frequency.

Application: earthquake-resistant building

First, we assume that there is regularity in earthquakes. To prove the hypothesis (the earthquakes have the same regularity), we have 15 earthquakes data from Taiwan Central Weather Administration in the last decade. We capture 30 seconds before and after the maximum amplitude of every seismic wave respectively and convert the time domain to the frequency domain by using Fourier Transform. Due to the large number of data point, and the equally spaced manner in which the points captured by instrument, we choose to convert the data by Fast Fourier Transform to save computing time.

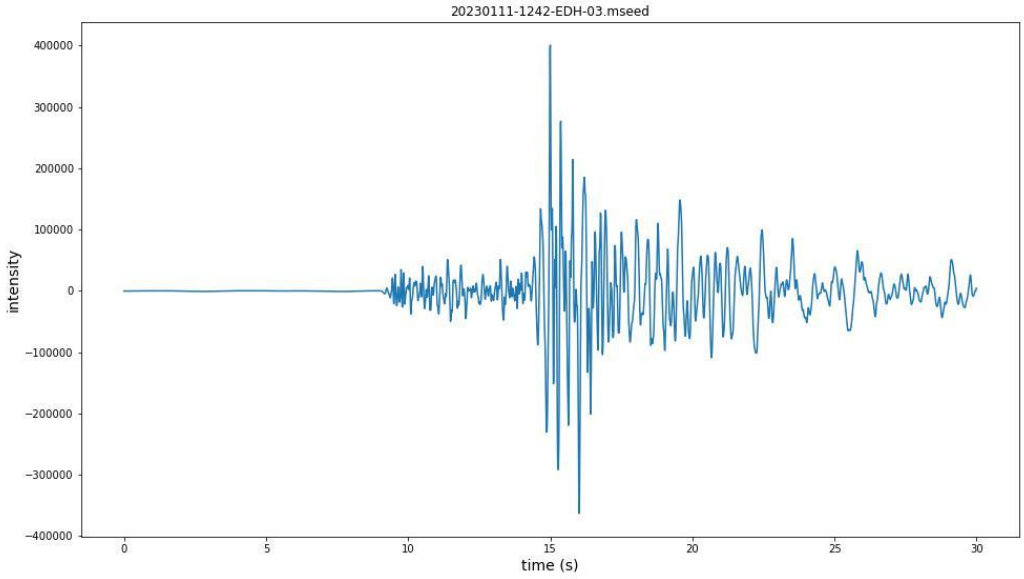

Figure 1-1. Seismic Waveform (Data 1)

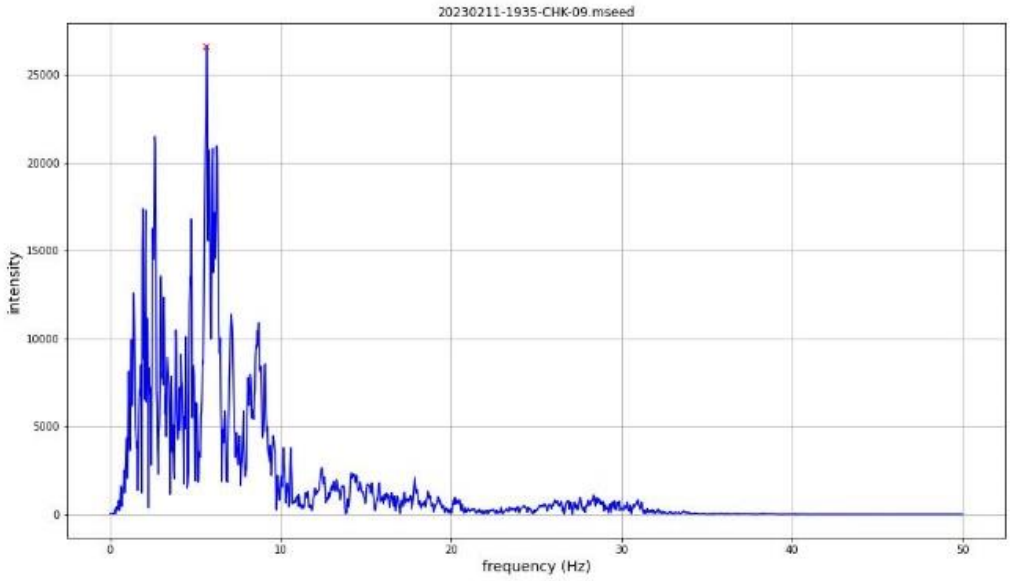

Figure 1-2. Seismic Waveform Frequency Spectrum (Data 1)

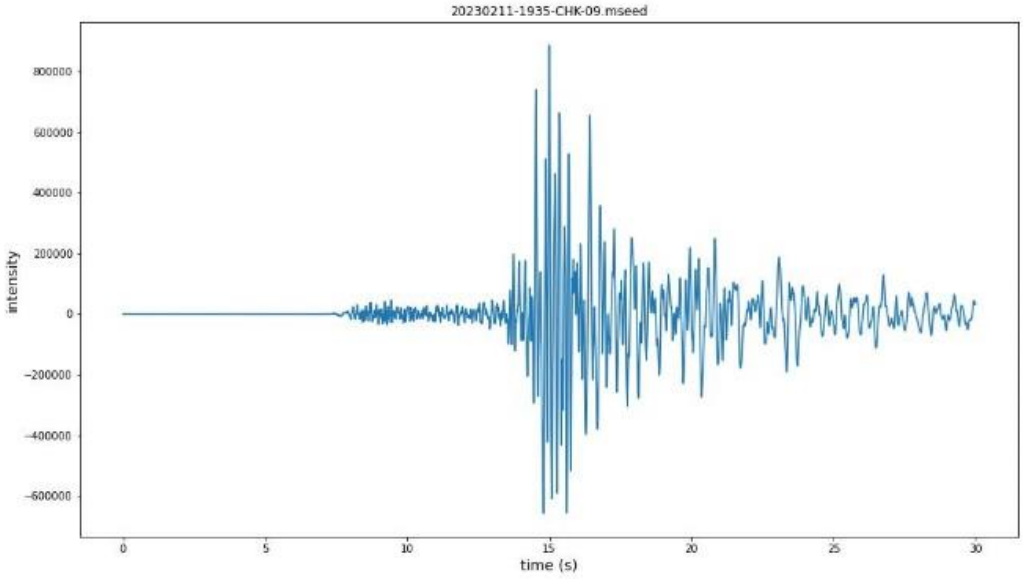

Figure 1-3. Seismic Waveform (Data 2)

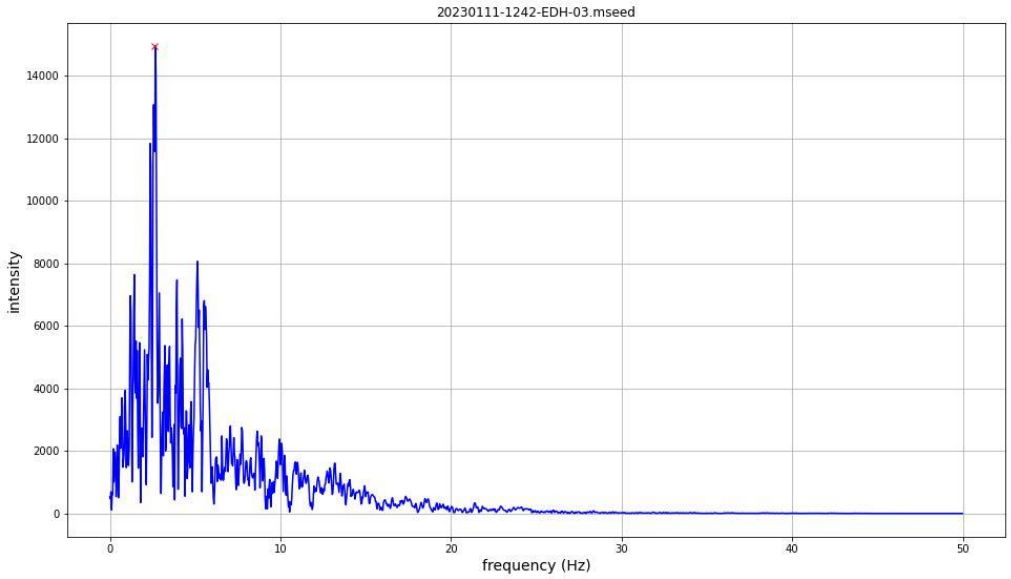

Figure 1-4. Seismic Waveform Frequency Spectrum (Data 2)

Finally, do statistics due to the frequency when peak intensity occurred of each graph and get the following diagram.

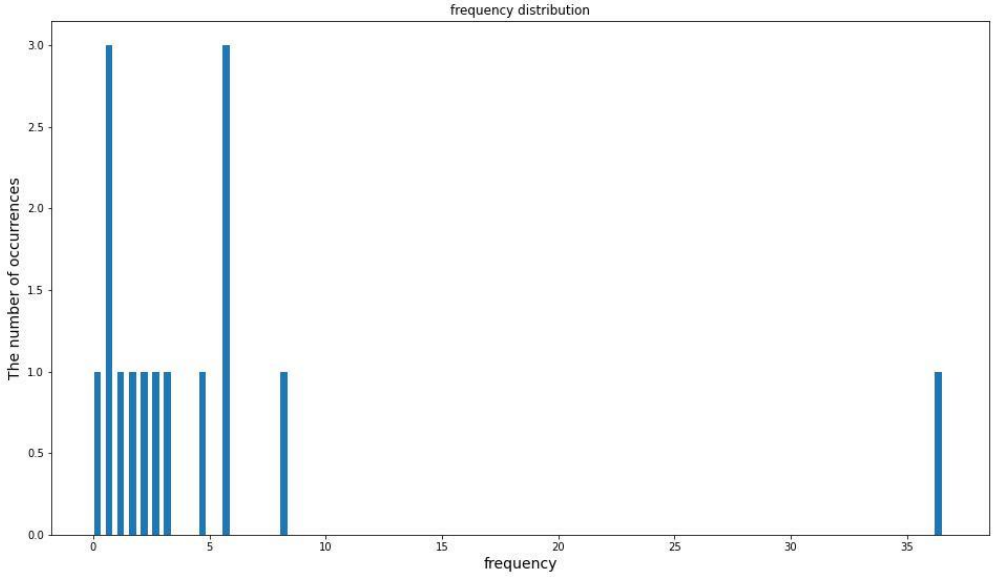

Figure 1-5. Statistical Chart of the Main Frequency Distribution for 15 Seismic Spectrograms

From the graph above, it can be observed that frequency earthquakes in Taiwan falls between approximately 1 to 6 Hertz. Due to the result, the hypothesis is proved to be true.

3. MODEL ARCHITECTURE

3.1. Recurrent Neural Networks

Recurrent Neural Network is designed for processing sequential data. It maintains a memory of previous inputs in the sequence, making them suitable for processing time series data prediction and speech recognition. The model generates output by multiplying inputs with weights and the activation function tanh. In other words, RNNs add the previous output to the inputs of current time step. These are two equations in RNN:

$$h_t = \tanh\left(W_h \times h_{t-1} + W_x \times x_t\right) \quad (3)$$

$$y_t = W_y \times h_t \quad (4)$$

h_t: current hidden state

W_h: weight of previous hidden state

h_{t-1}: previous hidden state

W_x : weight of current input state

x_t: current input

y_t: output state

W_y: weight of output state

The hidden state allows the model to maintain a memory capturing information about past inputs, which let RNNs are capable of learning the correlations between data. It is worth noting that RNNs struggle with vanishing and exploding gradients, this disadvantage can make it untrainable sometimes.

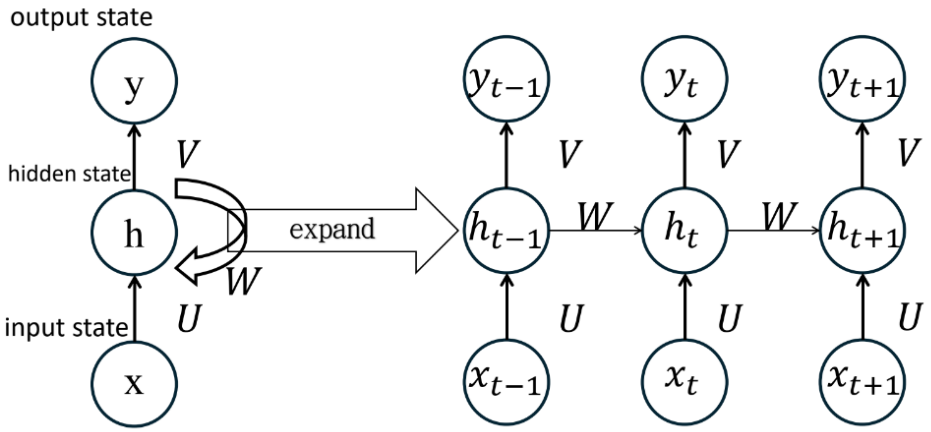

Figure 2. Recurrent Neural Network Structure

3.2. Long Short-Term Memory

Long Short-Term Memory is a type of RNN designed to solve the limitations of traditional RNNs—the problems of vanishing and exploding gradients. The LSTM cells are enhanced by three components which are called the input gate, the forget gate, and the output gate respectively. The input gate uses sigmoid activation function; The candidate cell state uses tanh activation function. Multiplying both states determines how much new information should be added. The forget gate determines how much of the previous state needs to be abandoned. The cell state is the memory of the network. It uses additional operation to avoid the vanishing problem. The output state controls which is allowed to be output. After selecting, the hidden state remembers information from previous time steps by tanh function. These are the equations in LSTM:

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) \quad (5)$$

$$\sim c_t = \tanh(W_c \times [h_{t-1}, x_t] + b_c) \quad (6)$$

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f \quad (7)$$

$$\mathrm{C}_t = f_t \times C_{t-1} + i_t \times {\sim}c_t \quad (8)$$

$$o_t = \sigma(W_o \times [h_{t-1}, x_t] + b_o) \quad (9)$$

$$h_t = o_t \times \tanh(C_t) \quad (10)$$

$W_{\square}$: weight

$b_{\square}$: bias

x_t: input

h_{t-1}: previous hidden state

LSTMs can learn long-term dependencies and sequential data effectively without worrying about the problems of vanishing and exploding gradients. Therefore, LSTM is more suitable for our project due to its ability to predict various time series.

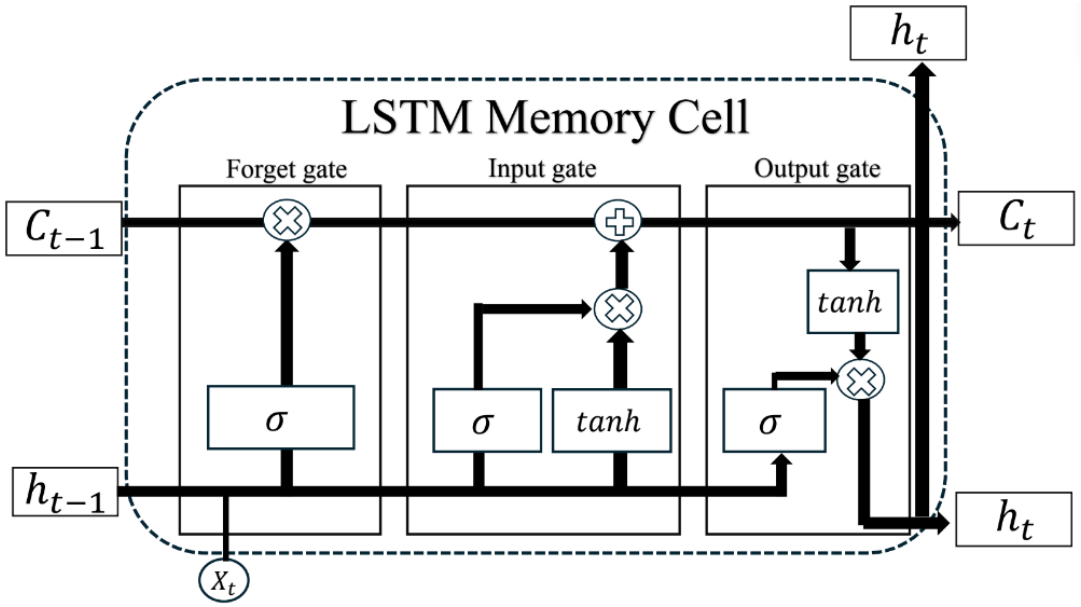

Figure 3. Long Short-Term Memory Structure

4. LSTM MODELS FOR SEISMIC WAVES SIMULATION

4.1. Dataset

We have 20 earthquake records with a magnitude of 4 or above, and the data is sourced from the Taiwan Central Weather Administration's Central Weather Administration Seismographic Network (CWASN).

4.1.1. Data preprocessing

After removing data unrelated to earthquakes, the original dataset contains three channels. We selected the channel with the highest PGA for our analysis. Since there is no STA/LTA method to accurately determine the time frame from the start to the end of the earthquake, we use data from the 60 seconds following the earthquake event.

In the 20 seismic data points, we used 15 as the training set and 5 as the test set to evaluate the trained model.

4.2. Methodology

Seismic events belong to time series data, and traditional linear methods often fail to capture their complex dynamic changes. However, LSTM models can learn long-term dependencies and short-term fluctuations, making them highly suitable for detecting patterns in seismic data, aligning with our goal of simulating seismic waveforms.

The process involves three key steps:

1. **Data Scaling:** Using Min-Max scaling (range -1 to 1) from formula 12 to transform the dataset. This standardizes the features, matching the tanh activation function of the LSTM, preventing certain features from dominating the model.
2. **Data Segmentation and Model Training:** The data is segmented into sequences, with each time step's value used as training data and the subsequent value as testing data. We use RMSE from formula 11 to calculate the error, and the Adam optimizer (selected as the best choice after various comparisons) to minimize the error.
3. **Simulation and Evaluation:** After training, we adjust the mean of the simulated data to align it with the original data, focusing on the seismic data 0.7 seconds before and after the Peak Ground Acceleration (PGA). We compare the difference between the predicted and actual values using RMSE, and apply FFT (Fast Fourier Transform) to calculate the frequency, observing whether the spectra of the original and simulated data are similar, assessing whether the model has successfully captured the characteristics of the earthquake.

Finally, we will compare the performance of LSTM, Bidirectional LSTM, CNN-LSTM, and CNN models to determine which is most suitable for earthquake simulation.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2} \qquad (11)$$

$$z = \frac{x - \min(x)}{\max(x) - \min(x)} \qquad (12)$$

4.3. Simulation

4.3.1. LSTM

This model is a simple single-layer LSTM model. Our architecture consists of one LSTM layer followed by one Dense layer as the output. The LSTM layer uses 60 units of neurons, with tanh as the activation function and hard-sigmoid as the recurrent activation function. We chose hard-sigmoid because it slightly outperformed sigmoid in practical testing. In the Dense layer, we use a linear activation function to prevent extreme values from being improperly handled due to function constraints.

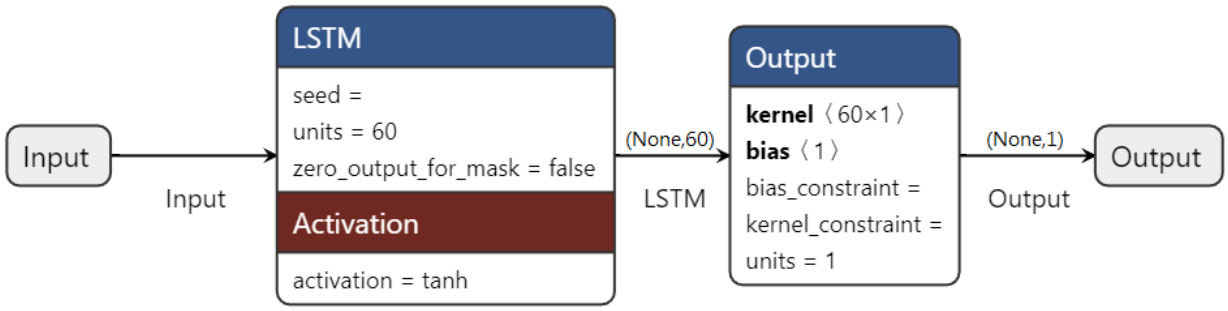

Figure 4-1-1. LSTM model Structure

Layer (type)	Output Shape	Param #
LSTM (LSTM)	(None, 60)	14,880
Output (Dense)	(None, 1)	61

Total params: 14,941 (58.36 KB)
Trainable params: 14,941 (58.36 KB)
Non-trainable params: 0 (0.00 B)

Figure 4-1-2. LSTM model parameters and output shape

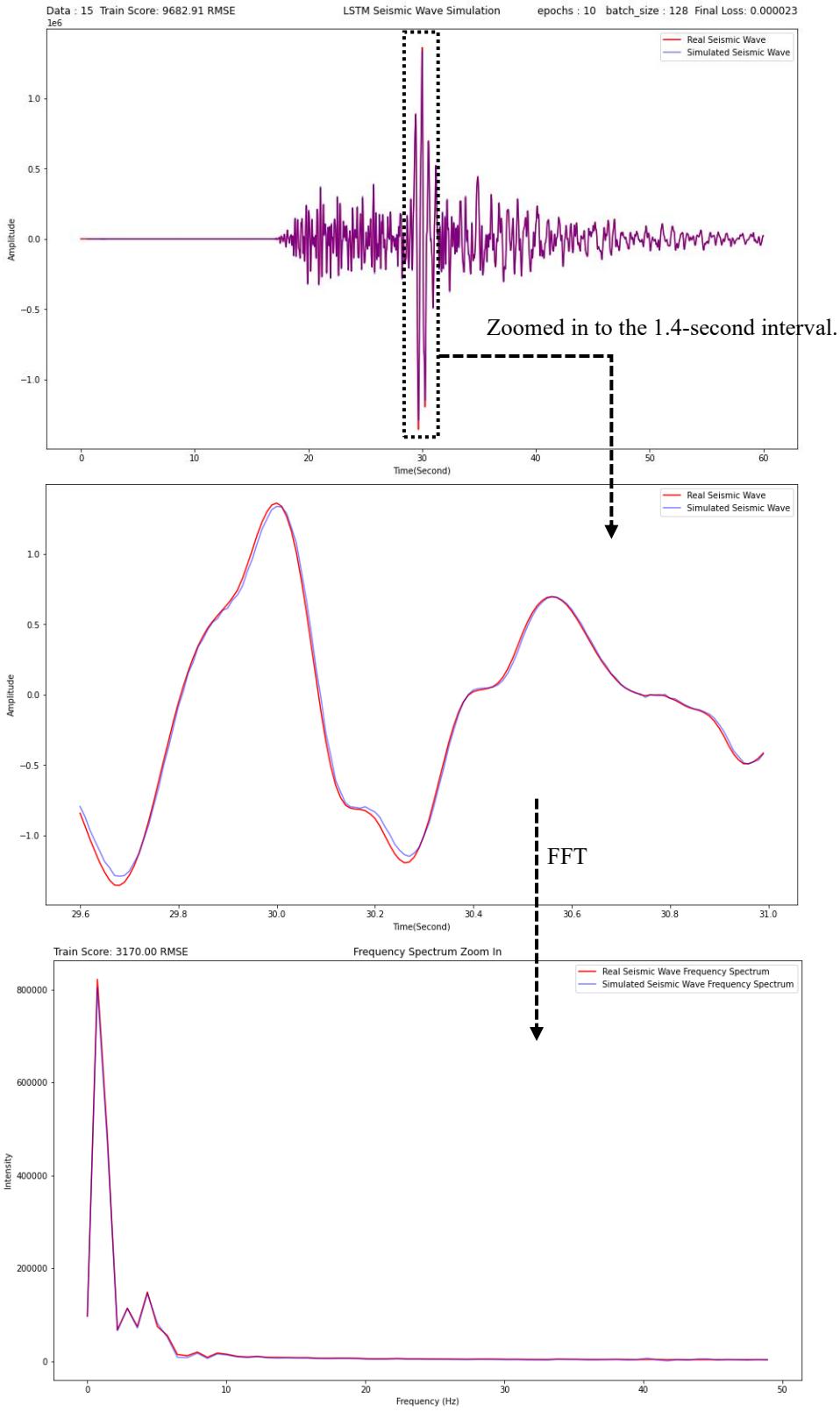

Figure 4-1-3. Waveform simulation and actual overlap chart and the spectrum chart (Data 1)

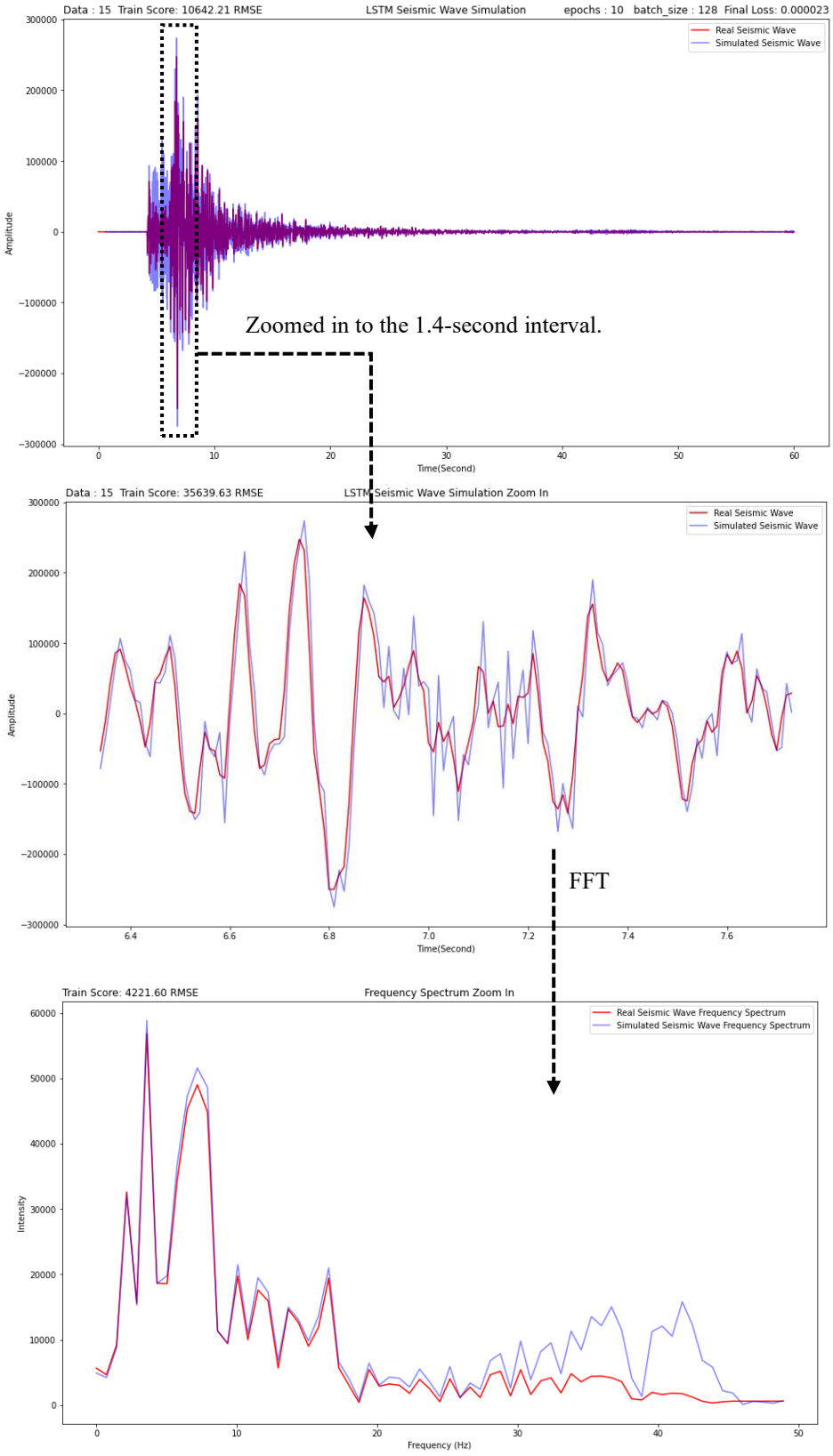

Figure 4-1-4. Waveform simulation and actual overlap chart and the spectrum chart (Data 2)

4.3.2. CNN

Before using the CNN-LSTM, we first tested whether CNN alone could achieve our expected goals, which is to capture features in the data through convolutional layers rather than relying solely on LSTM for feature extraction. In this CNN architecture, we used a 1D convolutional layer with ReLU activation, 1D global max pooling, and a fully connected layer. For the 1D convolutional layer, we set the number of filters to 128, with the window size the same as the step size. This way, when the CNN receives data with 60 values as one time step, it can read the entire time step at once and output 128 features, which are then passed to the global max pooling layer to find the maximum value in the feature map as the output.

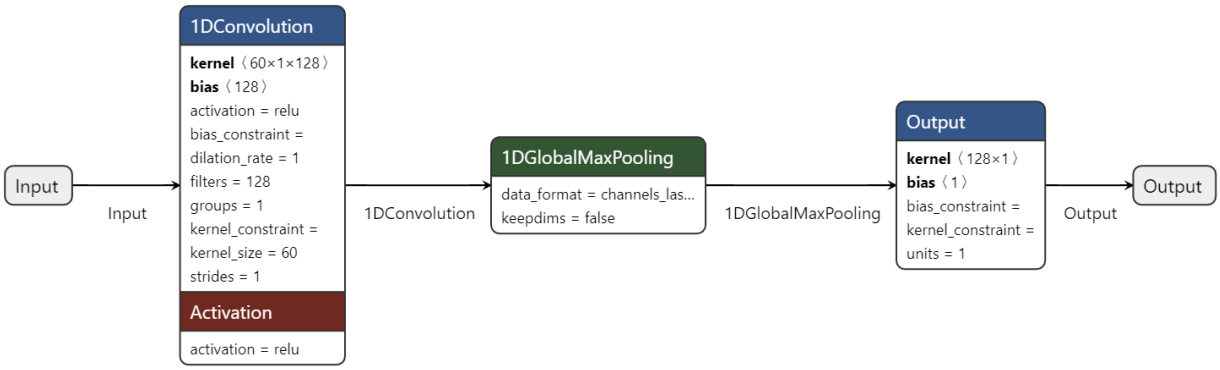

Figure 4-2-1. CNN model Structure

Layer (type)	Output Shape	Param #
1DConvolution (Conv1D)	(None, 1, 128)	7,808
1DGlobalMaxPooling (GlobalMaxPooling1D)	(None, 128)	0
Output (Dense)	(None, 1)	129

Total params: 7,937 (31.00 KB)
Trainable params: 7,937 (31.00 KB)
Non-trainable params: 0 (0.00 B)

Figure 4-2-1. CNN model output shape and parameters

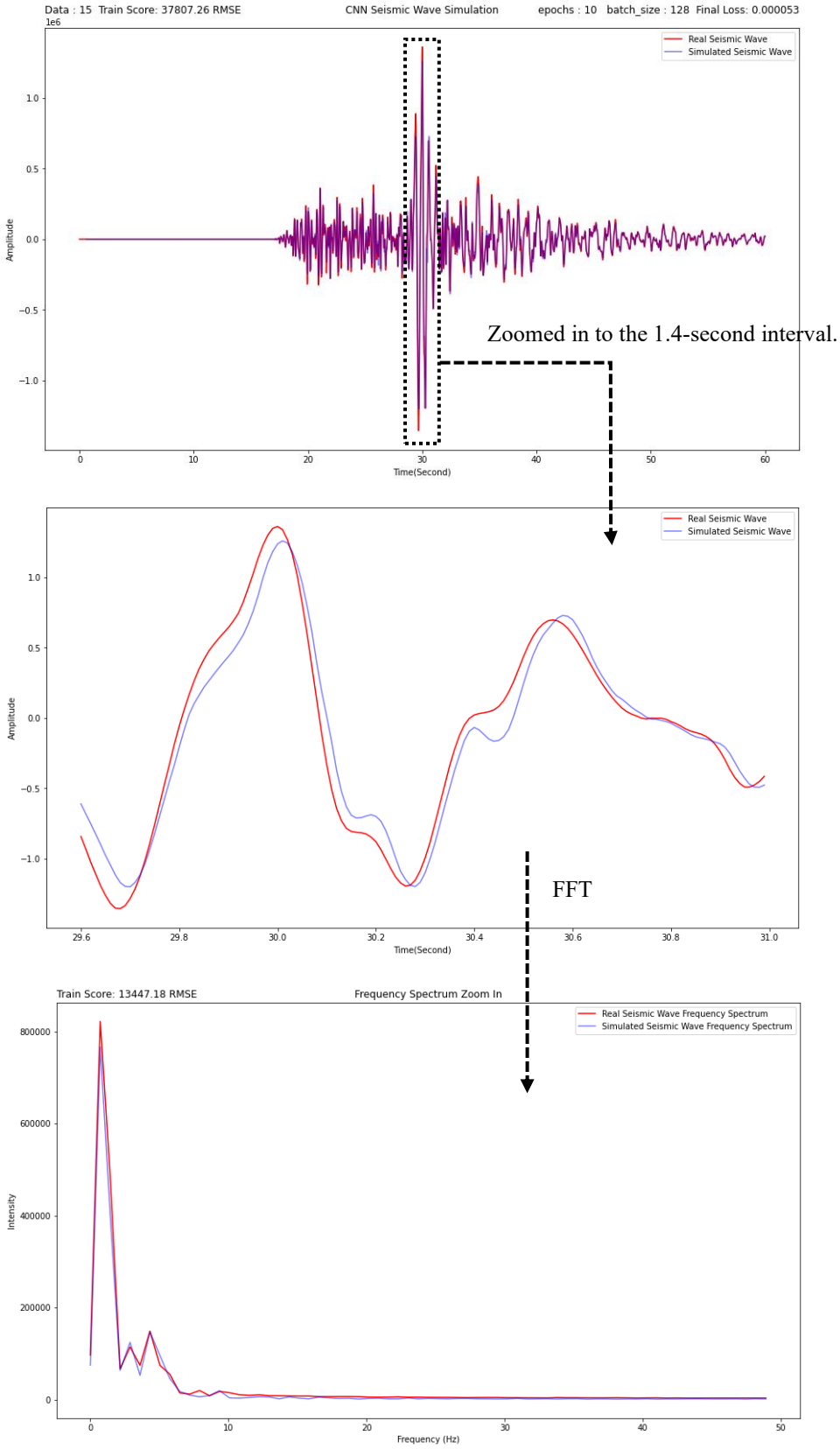

Figure 4-2-3. Waveform simulation and actual overlap chart and the spectrum chart (Data 1)

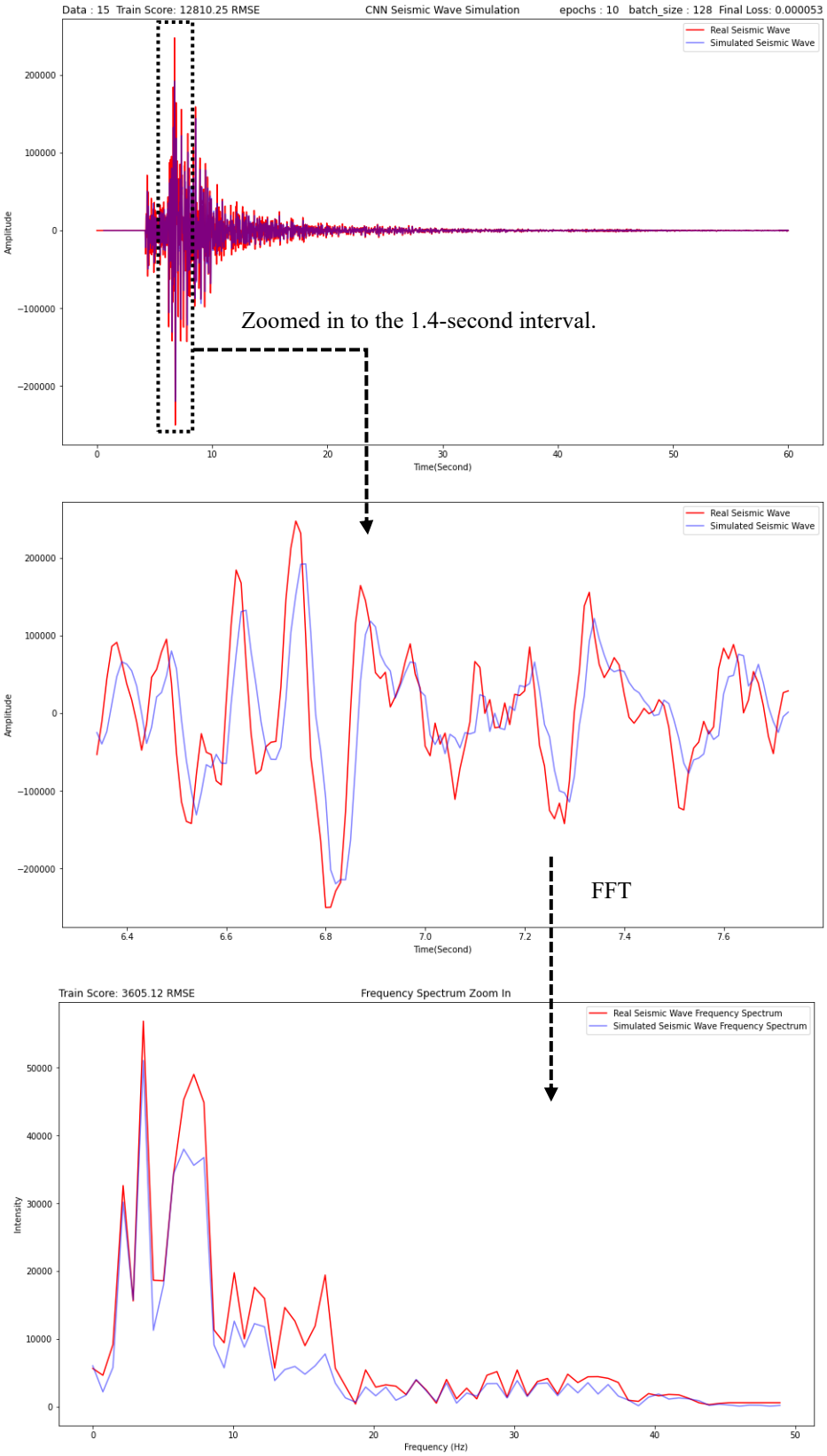

Figure 4-2-4. Waveform simulation and actual overlap chart and the spectrum chart (Data 2)

4.3.3. CNN LSTM

Since the CNN achieved the goal of feature extraction during testing, we propose adding a CNN before the LSTM to capture features from the data first, and then pass these processed features to the LSTM for learning the temporal patterns in the time series. This could improve the efficiency of the simulation or enhance the generalizability of our model.

In our CNN-LSTM architecture, we use a 1D convolutional layer with the ReLU activation function, followed by 1D global max pooling, an LSTM layer, and a fully connected layer. The CNN part is configured the same as in the CNN model simulation, while the LSTM is configured the same as in the LSTM model simulation.

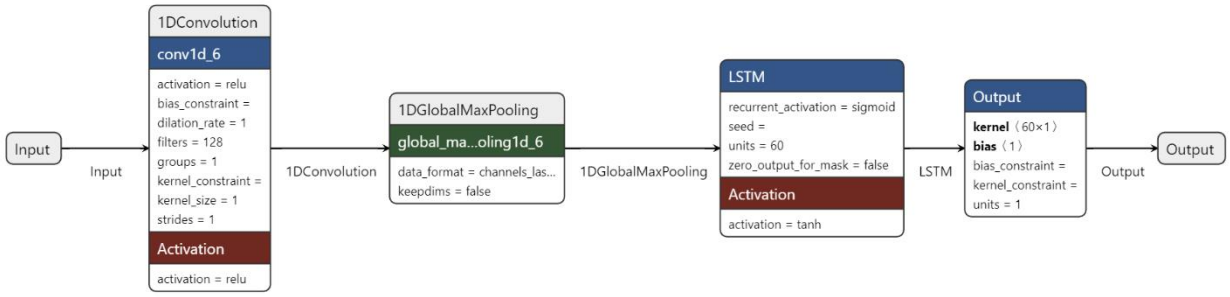

Figure 4-3-1. CNN-LSTM model Structure

Layer (type)	Output Shape	Param #
1DConvolution (TimeDistributed)	(None, None, 60, 128)	256
1DGlobalMaxPooling (TimeDistributed)	(None, None, 128)	0
LSTM (LSTM)	(None, 60)	45,360
Output (Dense)	(None, 1)	61

Total params: 45,677 (178.43 KB)
Trainable params: 45,677 (178.43 KB)
Non-trainable params: 0 (0.00 B)

Figure 4-3-2. CNN-LSTM model Shape and Parameter

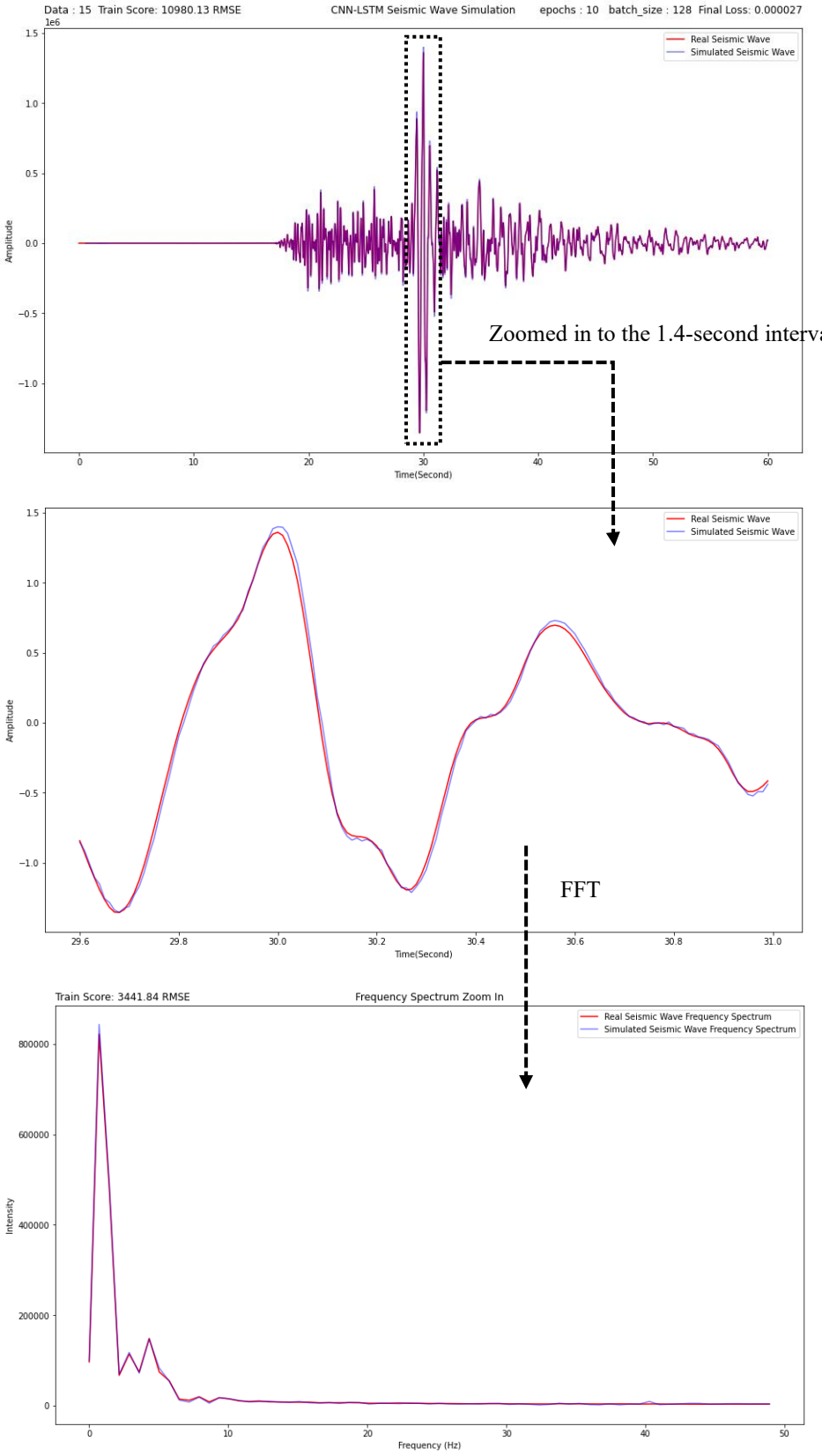

Figure 4-3-1. Waveform simulation and actual overlap chart and the spectrum chart (Data 1)

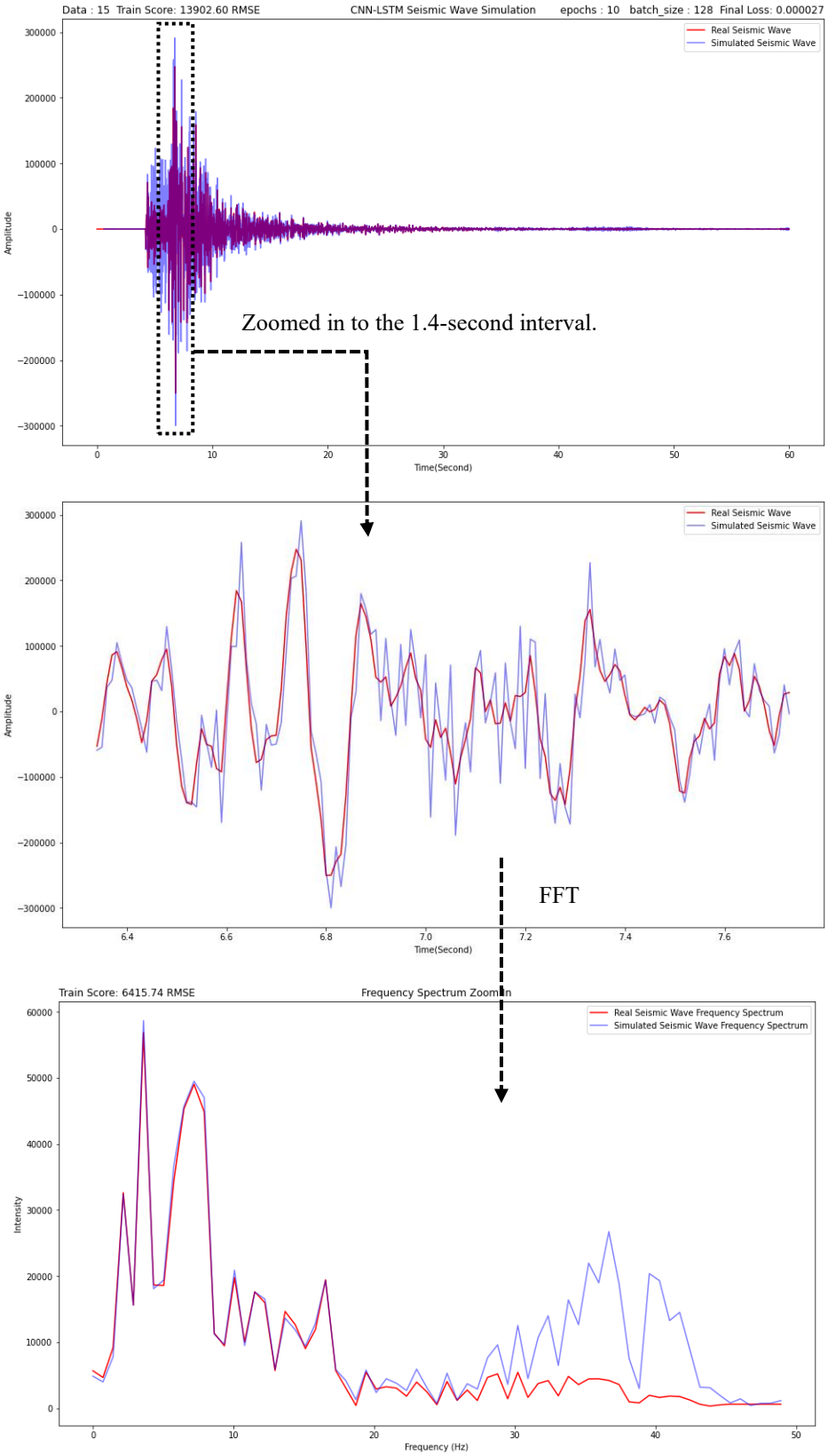

Figure 4-3-2. Waveform simulation and actual overlap chart and the spectrum chart (Data 2)

4.3.4. Bidirectional LSTM

A unidirectional LSTM processes the sequence from the start to the end, meaning it can only use past information to predict the future. In some applications, such as natural language processing or time series forecasting, this may not fully utilize the context information from later in the sequence. However, a bidirectional LSTM has both a forward LSTM and a backward LSTM processing the sequence in opposite directions, allowing us to capture features from both past and future data, leading to more accurate handling of time series data.

In our model architecture, there is a bidirectional LSTM layer and a fully connected layer as the output layer. The activation function for the bidirectional LSTM is tanh, and the recurrent function is hard sigmoid.

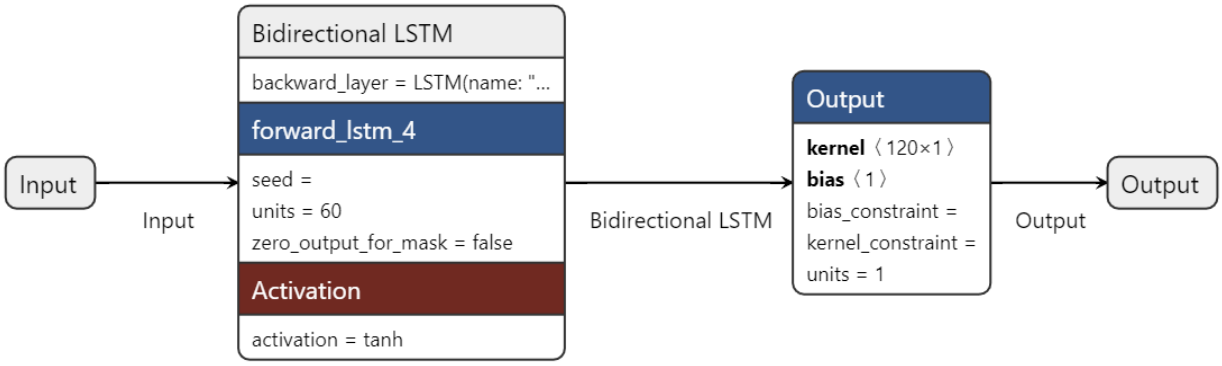

Figure 4-4-1. CNN-LSTM model Structure

Layer (type)	Output Shape	Param #
bidirectional_1 (Bidirectional)	(None, 120)	29,760
dense_1 (Dense)	(None, 1)	121

Total params: 29,881 (116.72 KB)
Trainable params: 29,881 (116.72 KB)
Non-trainable params: 0 (0.00 B)

Figure 4-4-2. CNN-LSTM model Shape and Parameters

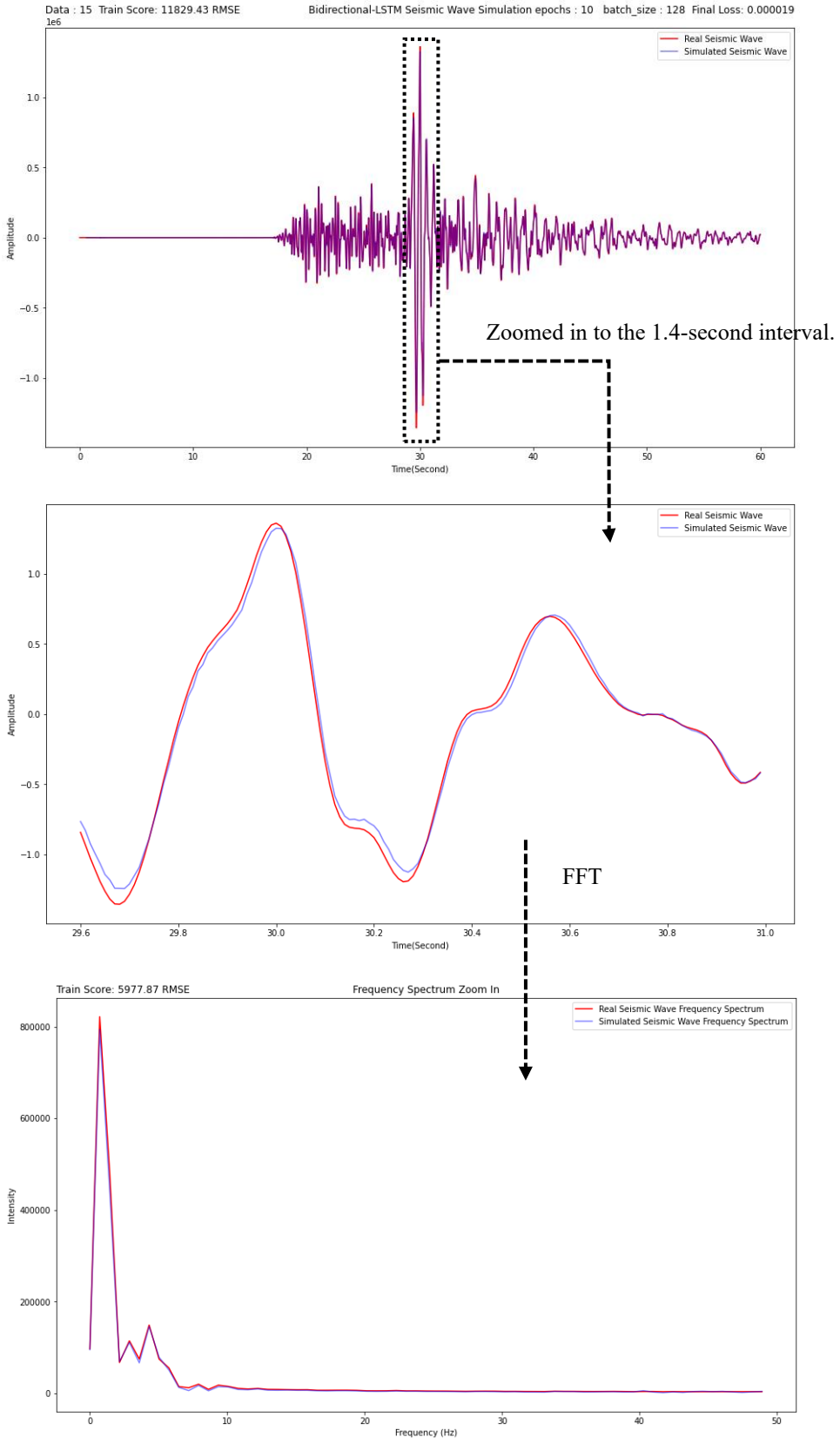

Figure 4-4-1. Waveform simulation and actual overlap chart and the spectrum chart (Data 1)

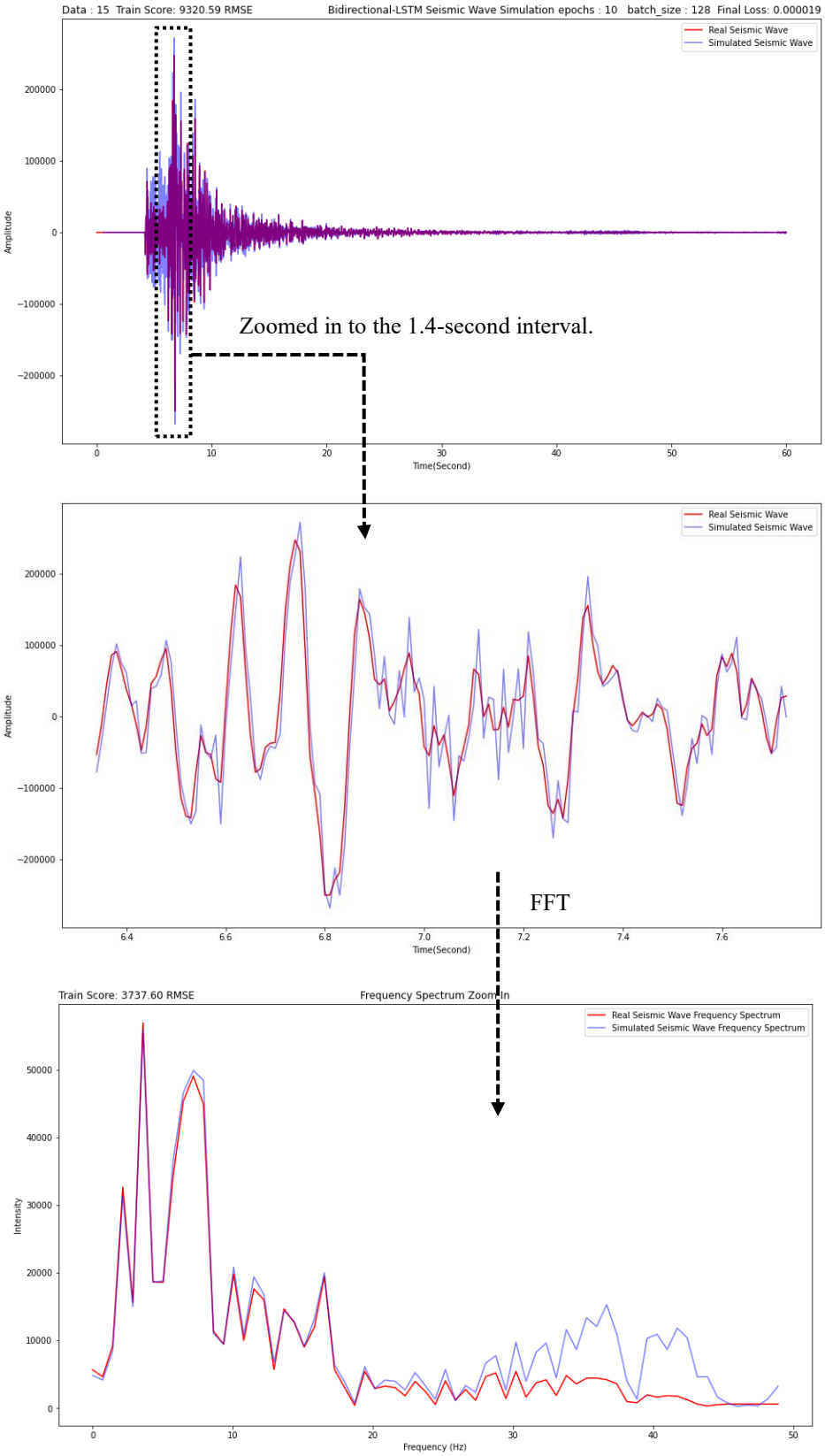

Figure 4-4-2. Waveform simulation and actual overlap chart and the spectrum chart (Data 2)

5. PERFORMANCE EVALUATION AND DISCUSSION

After training on 15 seismic data points and testing on 5, we calculated the average RMSE and loss for each test and organized the results into Table 1. The results show that the LSTM model outperforms the other three models in terms of RMSE, loss, and computation time. Although the bidirectional LSTM takes the longest computation time, its RMSE is only 2151 higher than that of the CNN-LSTM. Meanwhile, the CNN model performs the worst across all three metrics, clearly lagging behind the other three models.

Models	LSTM	Bi-LSTM	CNN-LSTM	CNN
Average RMSE	36790.86	40006.24	42151.17	110496.92
Average Loss	0.00063	0.00109	0.00090	0.00488
Time (second)	313.66	626.72	585.84	47.18

Table 1. The average RMSE, average loss, and computation time after testing different models on 5 samples.

5.1.1. Comparison of spectrum differences between different models

The various models all achieved excellent results on Data 1. In terms of waveform, they perfectly overlapped in the spectrum, capturing the frequency characteristics accurately. We believe this outcome was due to the training set consisting mostly of high-PGA data, which allowed the model to perform well on Data 1, which also had high PGA.

In contrast, Data 2, with a PGA 10 times smaller than Data 1, contains more high-frequency segments, unlike Data 1, which is concentrated in the low-frequency range. This highlights the model's generalization ability when faced with data that exhibits different characteristics from those seen during training.

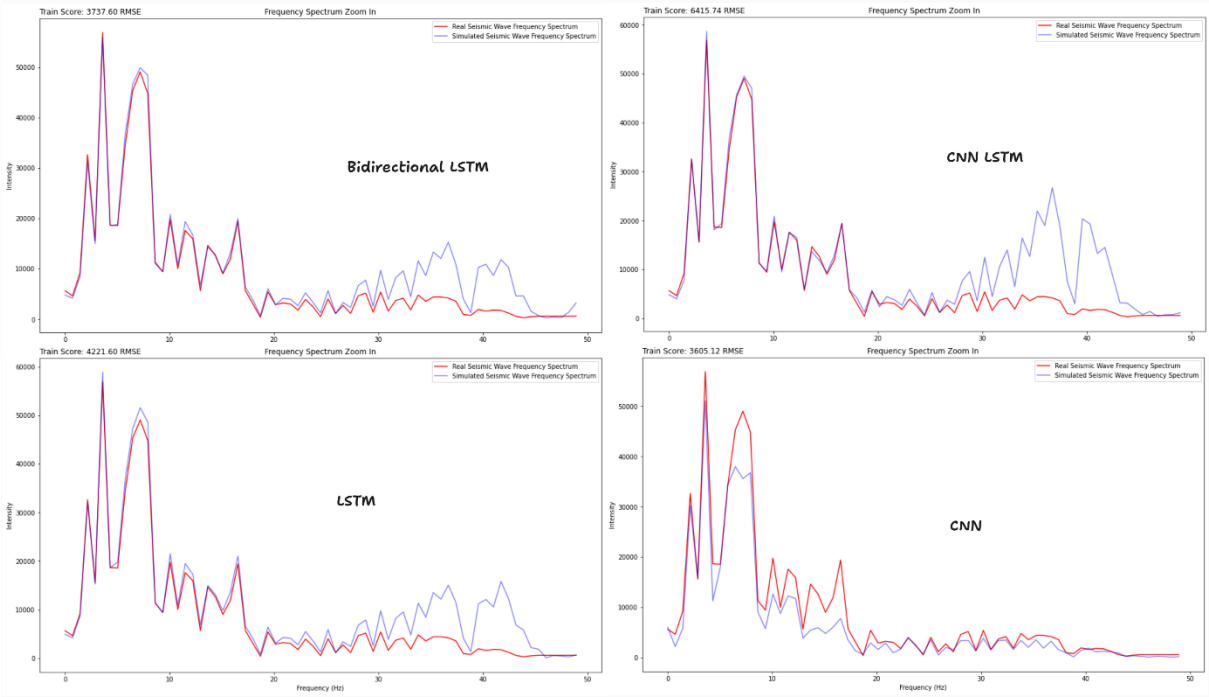

Figure 5-1. Spectrum plots of simulations from different models on Data 2.

From this figure, we can see that the accuracy of the bidirectional LSTM is slightly higher than that of the unidirectional LSTM; however, the CNN-LSTM has more severe misclassifications in the high-frequency range. Compared to the other three models, we believe that the CNN aspect is the most worthy of further exploration, as CNN does not exhibit significant errors in the high-frequency range, while the other three models do. Although there are considerable errors in the overall frequency intensity, the general patterns of the frequencies have still been captured. Additionally, the CNN requires the fewest parameters and computation time at the same time step, indicating that CNN has superior generalizability compared to the other three models.

6. CONCLUSION

First, as shown in Table 1, the LSTM model performs the best, while the CNN model significantly underperforms compared to the other models. Second, as observed in Figure 5-1, the LSTM model struggles with high-frequency earthquake simulations, especially when dealing with unfamiliar earthquake data. In contrast, although the CNN model is not perfect in simulations, it demonstrates good performance in capturing frequency patterns and high-frequency characteristics, highlighting its adaptability to different data features.

Based on these two points, we infer that the CNN-LSTM model should ideally combine the excellent RMSE performance of the LSTM model with the generalization ability of the CNN model. According to other research, CNN-LSTM should outperform LSTM in earthquake prediction. However, in our experimental results, this was not the case. We believe this is due to suboptimal hyperparameter tuning and model design in our CNN-LSTM implementation.

Additionally, from the first point, we can infer that bidirectional LSTM should outperform unidirectional LSTM in terms of prediction accuracy by better capturing future data and having a stronger grasp of time series. Other research also suggests that bidirectional LSTM should surpass LSTM in earthquake prediction. However, our experimental results did not show this advantage, likely due to inadequate hyperparameter tuning and structural design.

In conclusion, if the models can be tuned to their optimal states, we believe CNN-LSTM would be the best model structure for earthquake simulation. Since even a single-layer CNN can achieve good results, the feature-capturing capability of CNN combined with the time-series accuracy of LSTM, when properly designed and tuned, should lead to the best simulation model.

7. REFERENCES

[1] Obspy Chinese Tutorial
URL: https://docs.obspy.org/archive/ObsPy_Tutorial_2020-04_chinese.pdf

[2] Jason Brownlee. (2020). How to Develop Convolutional Neural Network Models for Time Series Forecasting.
URL: https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/

[3] Jason Brownlee. (2020). How to Develop LSTM Models for Time Series Forecasting.
URL: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

[4] Central Weather Administration, Taiwan Geophysical Database Management System
URL: https://gdms.cwa.gov.tw/