Regional Earthquake Seismic Phase Picking Using Convolutional and Recurrent Neural Networks

Albert Leonardo Aguilar Suarez (aguilars@stanford.edu) and Paige Given (prgiven@stanford.edu)

June 3, 2021

1 Introduction

Latest developments in machine learning and data science are transforming the way observational seismology is done [Bergen et al., 2019], [Kong et al., 2019]. Deep learning powered earthquake detectors and phase associators are fueling a new generation of high resolution earthquake catalogs [Zhu and Beroza, 2019], [Mousavi et al., 2020]. Previously unseen patterns in space and time are becoming visible now due to the generalization and scalability of these deep learning algorithms.

Currently, most of the work regarding seismic phase and earthquake detection deep learning algorithms has focused on areas with a dense availability of seismic instruments, such as California and the Central US, where most earthquakes are recorded by sensors located tens of kilometers away from the epicenter. However, the majority of the world has sparser seismic networks. As a consequence, earthquakes are typically recorded at distances of hundreds of kilometers (km). The complexity of the waveforms, as well as their length, increase as a result of these longer distances. The aforementioned studies have centered on picking P and S phase arrivals in the seismic waveforms without making the distinction between Pn and Pg, or Sn and Sg. In this project, we aim to expand the capabilities of deep learning seismic phase pickers to optimize identification of earthquake events in these sparser seismic networks. In addition, we aim to pick and classify more regional seismic phases, such as Pn, Pg, Sn, and Sg. This extended capability will facilitate earthquake monitoring in regions where most significant earthquakes occur outside of instrumented regions, like offshore areas. We intend to use 5 minute long waveforms, which should contain the full ground shaking for an earthquake at a distance of approximately 1000 km.

2 Related Work

[Zhu and Beroza, 2019] trained a CNN to pick P and S phases (without making the distinction between Pg and Pn) on seismic data from the dense Northern California Seismic Network using 30 second long waveforms sampled at 100 Hz. [Ross et al., 2018] did a similar approach, using data from the Southern California Seismic Network. [Mousavi et al., 2020] trained an earthquake detector using data where the earthquake sensor distance is less than 300 km. Their approach uses attention mechanisms to first detect the presence of earthquakes in continuously recorded seismograms, then the attention mechanism focuses in part of the waveform to pick P and S arrivals (again, without making the distinction between Pg and Pn or Sn and Sg). Their model uses 1 minute long waveforms sampled at 100 Hz. These approaches have been focused on lowering detection thresholds, i.e. finding small earthquakes that previous detection methods missed. Our proposed work is different from these studies because we will focus on less dense seismic networks and on longer time series, aiming to pick a larger set of earthquake phases.

Leonardo Aguilar, developed a classifier for the first arriving P wave as the class project for CS229 in the Fall of 2020.The work presented here is related and builds upon that project. You can find the report here: https://stanford.box.com/s/jb0djmg9fose0stblxzrwsm84fsukh7j.

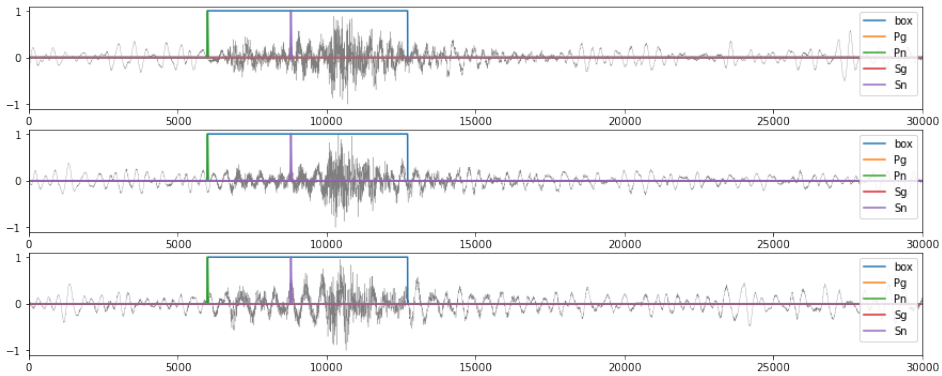

Figure 1: Example of earthquake time-series data with seismic phases labeled accordingly.

3 Dataset and Pre-processing

Earthquake arrival times were retrieved from the National Earthquake Information Center (NEIC) earthquake catalogs. The information collected includes the network and station codes, the instrument type, the timestamps of the Pn/Pg or Sn/Sg arrivals, along with the evaluation mode of each pick, which is mainly manual or automatic. The information described here is commonly referred as earthquake metadata.

After the metadata collection, we collected the waveforms, which are the recordings of ground shaking produced by the passage of seismic waves that originated in an earthquake. Using the metadata, we downloaded 6 minutes long waveforms, this extra length allows us to take a random slice of 5 minutes, ensuring that not all of the arrivals are located in the same position. All the datacenters available through Obspy [Beyreuther et al., 2010] were queried for the waveforms. The available waveforms range in sampling rates from 20 Hz to 200 Hz. We kept the time series with the sampling rate that is closest to 100 Hz. If the data is not 100 Hz sampled, it was resampled to 100 Hz. Then, we ensured a 5 minutes length, which amounts to 30000 sample points. Taking into account that there are three channels, East, North and Vertical channels, we filled any missing channel with zeros. By concatenating the three time series, we got matrices of shape (3,30000). These waveforms were then detrended and bandpass filtered between 0.1 and 45 Hz. This filter is intended to remove low frequency noise, which is fairly common in seismometers. We chose a set of examples that contain both P and S arrivals, and that have the required number of samples per channel (30000) as shown in Figure 1. This turned out to be a fairly scarce resource. We ended working with a set of 9663 examples, 2626 corresponding to n phases and 7037 corresponding to g phases. Then, we created the labels, which correspond to a box, which starts with the first P arrival and ends 1.4 times the S-P time difference after the S arrival. Then, for each other phase we created a truncated gaussian around the timestamp of the arrival. In order to combat the challenges of data sparsity in the original labeled time-series data, we turned our focus towards exploiting the spectral information contained in the waveforms. We transformed the waveforms into spectrograms, using the parameters displayed in the appendix. These spectrograms are of shape (996,65) for the vertical component input and (996, 195) for all components (North-South, East-West, and Vertical) and were the inputs into our two CNN Frameworks detailed in the next section.

Finally, for the scope of this quarter long process, our results focus on the arrival of a single seismic phase at a time. This provides proof of concept for our CNN and opens up opportunities for us to expand and develop our model to chart all phase arrivals together in the future.

4 CNN Framework

Currently, the most reliable form of earthquake phase picking is done manually, or automatically for regions with dense seismic networks. Our work aims to implement Convolutional Neural Networks (CNN) to estimate

earthquake arrivals and label the phase type. In order to do so, we use a training dataset comprised of approximately 90% of the 9,000 manually labeled 5-minute long seismic events and their respective spectrograms. To configure the CNN framework, 10% of the original, unseen 9,000 seismic events are designated as a testing dataset in order to get an updated accuracy scoring on unseen data.

4.1 Baseline Model

Using inspiration from Coursera CNN labs, Tensorflow Quickguide, and [Huot, 2021], our initial baseline framework was a Keras Sequential model with an input shape of (3,30000) - this represents the seismic data in 3 dimensions (north, east, and vertical), with 30000 time-series data points. The model consisted of a 1-Dimensional Convolution layer with 'same' padding and ReLU activation, followed by batch normalization to normalize the output according to its mean and standard deviation. It then had another 1-Dimensional Convolution layer with 'same' padding and ReLU activation, followed by a flattening layer and dense layer to output the data to 4 seismic phase location predictions.

The results of the baseline test were a bit sporadic due to the low volume of input data for the baseline test (only 100 events), but overall the results still fared relatively well for a baseline run: with a training accuracy of between 30-60% and an evaluation accuracy of between 15-65%.

In order to improve our results, we converted the inputs into spectrogram data and altered the CNN to follow a Recurrent Neural Network design, as detailed in the next section.

4.2 Updated Framework: Recurrent Neural Network

Drawing inspiration from the way human analysts detect earthquakes and the Trigger Word Detection lab on Coursera, we wanted to have an architecture that takes into account information from the past and the future when dealing with time series. Thus, we turned to recurrent neural networks (RNN).

The final model architecture (detailed in Appendix 10.3) consisted of three layers. The first was a 1-Dimensional Convolution layer followed by batch normalization, ReLU activation, and a 2% dropout. The second and third layers were identical and consisted of a Bi-directional Long short-term memory (LSTM) layer, followed by batch normalization, ReLU activation, and a 2% dropout. Finally, we applied a Time Distributed Dense layer to format the output data into an array of size (996,1). We implemented this model in two phases: one that takes the input as the spectrogram of the vertical component, and one that takes the input as the three individual spectrogram components. We implemented the second phase after noticing that some of the prediction examples did not contain any discernible earthquake signal, and thus were not contributing to the model's learning ability. This might occur because one of the channels of the seismometer is malfunctioning, but the earthquake signal is still observed in the other channels. Both phases are discussed in the next section.

Our model was compiled using an ADAM optimizer with a learning rate of 0.0005 and a decay of 0.00001. Since we formatted our labeled data to follow box-function, composed of zeros and ones, we can think of the task as a binary classification one, per timestep. As a consequence, we used a Binary Cross-Entropy loss function and examined this output, as well as a general accuracy metric to evaluate our model. We recognize that due to the nature of the labels, accuracy might not be the best metric. Take for instance the label in the right plot of Figure 5, even if the model predicted all zeros, the accuracy would be above 90 %.

5 Experiments and Results

5.1 Hyperparameter tuning

In the initial stages of implementation, we faced the problem of the cost function becoming Not a Number, but this problem was solved by tuning the learning rate. Learning rate values of 10^{-5} to 10^{-1} were tested, with the smaller values producing the better results. The size of the mini batch was also tuned, with a batch size of 32 examples producing the faster training times. Our results are produced using 10 epochs, due to the time requirements to train the model, but for a longer term project we would likely during more epochs.

5.2 Phase 1 Results: Vertical Component Input Data

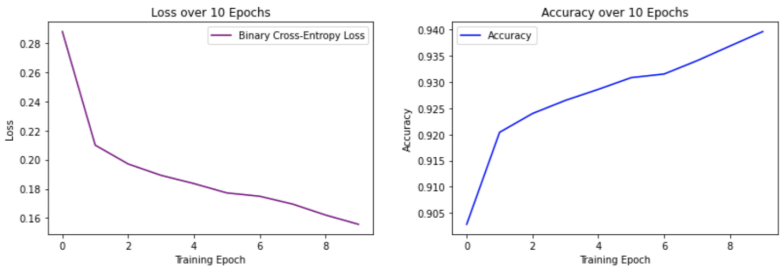

Figure 2: On the left: Binary Cross-Entropy loss value plotted over 10 epochs for Phase 1 - Vertical Component model. On the right: Accuracy value plotted over 10 epochs for Phase 1 - Vertical Component model.

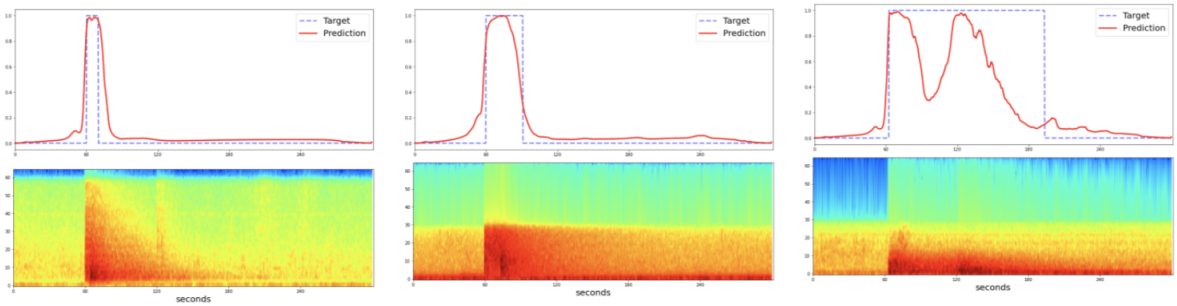

Figure 3: Random prediction plots of Phase 1 model. Top plots show true label versus predicted, while bottom plots show the input data.

5.3 Phase 2 Results: 3 Component Input Data

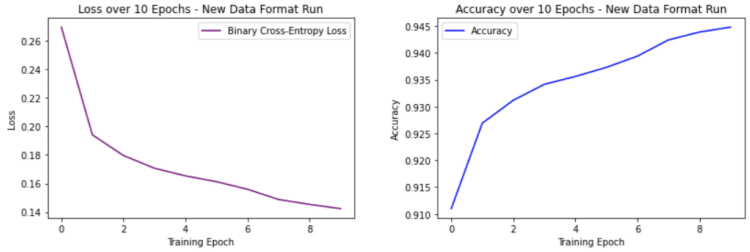

Figure 4: On the left: Binary Cross-Entropy loss value plotted over 10 epochs for Phase 2 - 3 Component model. On the right: Accuracy value plotted over 10 epochs for Phase 2 - 3 Component model.

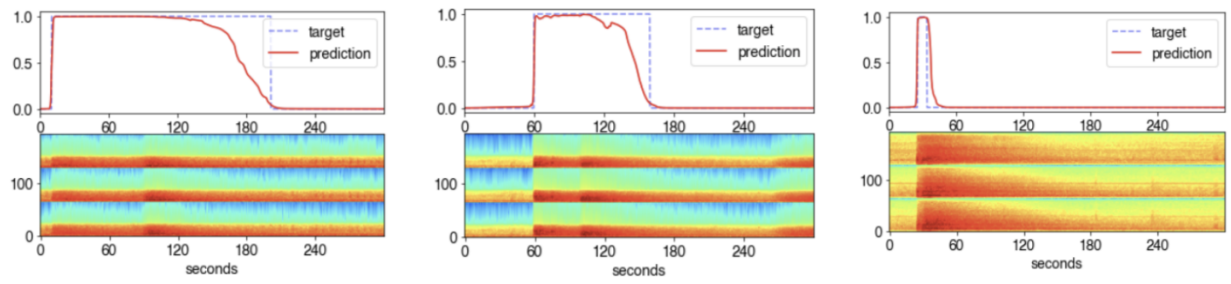

Figure 5: Random prediction plots of Phase 2 model. Top plots show true label versus predicted, while bottom plots show the 3-component input data.

6 Discussion

Phase 1 of the model, with the 1-dimensional vertical seismometer spectrogram input performed relatively well. Figure 2 displays the loss and accuracy of this model over 10 epochs. It achieved an overall training loss of 0.206 and training accuracy of 92.35%, with a testing loss of 0.207 and testing accuracy of 92.23%. As seen in Figure 3, in most cases, the algorithm is able to fit the seismic arrival relatively well, however, due to the nature of seismic phase arrivals, it is possible that some events have no motion in the vertical direction and therefore it may impact the models ability to generalize. We therefore chose to further develop our algorithm to learn from all 3-directional components.

Phase 2 of the model, with the 3-dimensional seismic spectrogram inputs showed great improvement. Figure 4 shows the loss and accuracy of our phase 2 model over 10 epochs, with an overall training loss of 0.151 and accuracy of 94.12%, and testing loss of 0.155 and accuracy of 94.12%. As displayed in Figure 5, the model fits the Pn seismic phase arrival overall very well, with errors primarily occurring in regions with low signal-to-noise ratios (i.e. low quality data). Overall, this model is able to take all 3-dimensional seismic inputs and accurately identify the seismic phase arrivals, and will be further developed to predict the onset all phases simultaneously.

7 Conclusion and Future Work

This project provides a proof of concept for the use of CNN/RNN frameworks to identify seismic phase arrivals for earthquakes. We were able to predict single phase arrivals with over 90% accuracy. This technique could be built upon to include the prediction of the other phases simultaneously. The ability to accurately and quickly identify seismic phase arrivals aids in the performance of Earthquake Early Warning and location detection. Future work for this project includes further developing the algorithm to simultaneously predict all Pn, Pg, Sn, and Sg phases in a single model. We also hope to collect and further process more seismic data as it becomes available in order to continuously update our model.

8 Contributions

Both teammates made great contributions to this work. Albert Leonardo Aguilar Suarez collected and pre-processed the data. Paige Given created and tested the baseline model. Both teammates updated the model, produced results, contributed to the presentation, and contributed to this paper.

9 Acknowledgements

We would like to extend a special thanks to our project TA, Shubhang Desai, for his guidance throughout this work. We would also like to thank all of the instructors and TAs for the CS 230 course for a very interesting and insightful quarter.

References

[Bergen et al., 2019] Bergen, K. J., P. A. Johnson, M. V. De Hoop, and G. C. Beroza, 2019, Machine learning for data-driven discovery in solid Earth geoscience: Science, **363**.

[Beyreuther et al., 2010] Beyreuther, M., R. Barsch, L. Krischer, T. Megies, Y. Behr, and J. Wassermann, 2010, Obspy: A python toolbox for seismology: Seismological Research Letters, **81**, 530–533.

[Huot, 2021] Huot, F., 2021, Ml-framework: `https://gitlab.com/fantine/ml-framework/-/tree/master`.

[Kong et al., 2019] Kong, Q., D. T. Trugman, Z. E. Ross, M. J. Bianco, B. J. Meade, and P. Gerstoft, 2019, Machine learning in seismology: Turning data into insights: Seismological Research Letters, **90**, 3–14.

[Mousavi et al., 2020] Mousavi, S. M., W. L. Ellsworth, W. Zhu, L. Y. Chuang, and G. C. Beroza, 2020, Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking: Nature Communications, **11**, 1–12.

[Ross et al., 2018] Ross, Z. E., M. A. Meier, E. Hauksson, and T. H. Heaton, 2018, Generalized seismic phase detection with deep learning: Bulletin of the Seismological Society of America, **108**, 2894–2901.

[Zhu and Beroza, 2019] Zhu, W., and G. C. Beroza, 2019, PhaseNet: A deep-neural-network-based seismic arrival-time picking method: Geophysical Journal International, **216**, 261–273.

10 Appendix

10.1 GitHub Repository Links

Final CNN/RNN Code:
https://github.com/albertleonardo/Pn/blob/master/CS230_FinalProject.ipynb
Data Collection Example Code:
https://github.com/albertleonardo/Pn/blob/master/Example%20data.ipynb
Baseline Code:
https://github.com/albertleonardo/Pn/blob/master/CS230_Project_PhasePicking.ipynb
Other development phase code:
https://github.com/albertleonardo/Pn/blob/master/numbertwo.ipynb

10.2 Spectrogram parameters

length	30000
nfft	128
overlap	98
sampling frequency	100

Table 1: Parameters used to generate spectrograms

10.3 Network architecture

10.3.1 Phase 1 - Vertical Component Input

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 996, 128)          133248
_________________________________________________________________
batch_normalization_3 (Batch (None, 996, 128)          512
_________________________________________________________________
activation_3 (Activation)    (None, 996, 128)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 996, 128)          0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 996, 256)          263168
_________________________________________________________________
batch_normalization_4 (Batch (None, 996, 256)          1024
_________________________________________________________________
activation_4 (Activation)    (None, 996, 256)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 996, 256)          0
_________________________________________________________________
bidirectional_3 (Bidirection (None, 996, 256)          394240
_________________________________________________________________
batch_normalization_5 (Batch (None, 996, 256)          1024
_________________________________________________________________
activation_5 (Activation)    (None, 996, 256)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 996, 256)          0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 996, 1)            257
=================================================================
Total params: 793,473
Trainable params: 792,193
Non-trainable params: 1,280
```

Figure 6: Model summary for the Neural Network that takes as input the spectrograms from the vertical component.

10.3.2 Phase 2 - 3 Component Input

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 996, 195)          0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 996, 128)          399488
_________________________________________________________________
batch_normalization_1 (Batch (None, 996, 128)          512
_________________________________________________________________
activation_1 (Activation)    (None, 996, 128)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 996, 128)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 996, 256)          263168
_________________________________________________________________
batch_normalization_2 (Batch (None, 996, 256)          1024
_________________________________________________________________
activation_2 (Activation)    (None, 996, 256)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 996, 256)          0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 996, 256)          394240
_________________________________________________________________
batch_normalization_3 (Batch (None, 996, 256)          1024
_________________________________________________________________
activation_3 (Activation)    (None, 996, 256)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 996, 256)          0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 996, 1)            257
=================================================================
Total params: 1,059,713
Trainable params: 1,058,433
Non-trainable params: 1,280
_________________________________________________________________
```

Figure 7: Model summary for the Neural Network that takes as input the spectrograms from the three components.