AFRL-RV-PS-
TR-2023-0059

AFRL-RV-PS-
TR-2023-0059

DEEP LEARNING PROBABILISTIC REGRESSION FOR ONSET TIME DETERMINATION

Jesse Williams, et al.

Global Technology Connection, Inc.
2839 Pace Ferry Rd, SE
Atlanta, GA 30312

29 March 2023

Interim Report

AIR FORCE RESEARCH LABORATORY
Space Vehicles Directorate
3550 Aberdeen Ave SE
AIR FORCE MATERIEL COMMAND
KIRTLAND AIR FORCE BASE, NM 87117-5776

DTIC COPY

NOTICE AND SIGNATURE PAGE

AFRL-RV-PS-TR-2022-0059 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//SIGNED//	//SIGNED//
_____	_____
Dr. Megan P. Flanagan	For: Mark E. Roverse, Chief
Program Manager/AFRL/RVB	AFRL Geospace Technologies Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
29-03-2023	Final Report	30 Mar 2022 – 29 Mar 2023

4. TITLE AND SUBTITLE

Deep Learning Probabilistic Regression for Onset Time Determination

5a. CONTRACT NUMBER
FA9453-21-9-0054
PA-04-0003

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
C6601F

6. AUTHOR(S)
Jesse Williams, Greg Beroza[1], John Pace, and Artemii Novoselov[1]

5d. PROJECT NUMBER
1010

5e. TASK NUMBER

5f. WORK UNIT NUMBER
V1WU

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Global Technology Connection, Inc.
2839 Pace Ferry Rd, SE
Atlanta, GA 30312

[1]Stanford University
397 Panama Mall
Stanford, CA 94305

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Space Vehicles Directorate
3550 Aberdeen Avenue SE
Kirtland AFB, NM 87117-5776

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RVBN

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RV-PS-TR-2023-0059

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited (AFRL-2023-2101 dtd 02 Jun 2023).

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A software tool was developed to determine a seismic phase's arrival time, which is essential for subsequent processes, and can unburden analysts from manual onset time picking. The tool is a deep learning (DL) algorithm using a convolutional neural network to analyze the raw signal. The algorithm is a regression model that makes a probabilistic estimation. The onset times are a continuous interpolation between samples, so the algorithm could be more accurate than the discrete sample rate. The probabilistic output is achieved using an ensemble mask process. The variation in the output provides a measure of the uncertainty. The models were trained on International Data Center data. The final model had a mean absolute error of 0.574s over the test dataset, and by using the estimated STD of the probability as a measure of prediction confidence, the mean absolute error is reduced to 0.439s. Initially, the algorithm was trained and tested on P-waves but later extended to simultaneously identify the presence of P- and/or S-waves and their onset time.

15. SUBJECT TERMS
seismic arrival time, seismic phase picking, seismic onset time

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Megan P. Flanagan
Unclassified	Unclassified	Unclassified	Unlimited	42	**19b. TELEPHONE NUMBER** *(include area code),*

This page is intentionally left blank.

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES (continued)

Figure **Page**

This page is intentionally left blank.

1.0 SUMMARY

In this project, a software tool was developed to determine a seismic phase's starting position (onset time) within a given signal window. The onset time is essential for subsequent processes, such as phase association and locating events, and can unburden the AFTAC analysts from manual onset time picking. The onset time determination software tool is a deep learning (DL) algorithm using a convolutional neural network (CNN) to analyze the raw signal. Some key aspects of the DL algorithm are 1) it is a regression model and 2) it makes a probabilistic estimation. Since the algorithm is a regression, the signal onset times are not limited to discrete measurement samples but rather are a continuous interpolation between samples. In theory, this means that the algorithm can be more accurate than a human analyst picking samples and more accurate than the discrete sample rate. The probabilistic output is achieved using an ensemble mask process that produces multiple estimations at once, each derived from different portions of the DL model. The ensemble mask is similar to creating many independent models. The variation in the output indicates the model error for a given seismic waveform being analyzed and, thus, is a measure of the uncertainty. The models were trained on the IDC dataset from 2015 to 2018 to assess the viability and performance of this approach. The final model, a DL probabilistic onset regression algorithm based on Maskensembles was denoted as PhaseHunter, had a mean absolute error of 0.574s over the test dataset, and by using the estimated STD of the probability as a measure of prediction confidence, the mean absolute error is reduced to 0.439s. Initially, the algorithm was trained and tested on P-waves but later extended to simultaneously identify the presence of P- and/or S-waves and their onset time. This report also shows the viability of sub-sample accuracy using DL regression algorithms.

2.0 INTRODUCTION

2.1 Identification of the Problem

Proper determination of a phase arrival is essential for subsequent seismic signal processing. Seismologists traditionally consider wave propagation at local, regional, and teleseismic distances. Roughly speaking, this corresponds to wave propagation that is confined to the crust at local distances, that interacts with the structural complexities of the upper mantle including the transition zone at regional distances, and that propagates through the lower mantle and core at teleseismic distances. Earthquake monitoring relies heavily on body waves expressed as Pg/Sg at local distances, Pn/Sn at regional distances, and P/S at teleseismic distances. Most of the work on phase identification with machine learning has focused on local distances. A companion proposal is focused on regional distances, and this proposal focuses on teleseismic distances.

Current methods deployed at AFTAC that automatically identify the phase onset are less reliable than they should be. Approximately 63% of onset times derived by the current automated method are re-timed by analysts, with a mean re-timing of approximately 0.5 seconds (AFTAC, personnel communication, 2021). Whether the difference in the re-timed

arrival time is large or small, the time invested is similar. Reducing the need for that extra work, as well as quantifying the uncertainty of onset-time measurement so that analysts can focus their attention where it is most needed, are the major goals of this project.[1] One of our imperatives is to unburden the AFTAC analysts. Thus, methods to effectively and confidently determine phase onset times will unburden AFTAC analysts and so address this objective.

2.2 Current Approach to Onset Time Determination

Current methods of automatic onset determination use the Akaike Information Criterion (AIC), which is a quantification of generalized entropy. In this process, a rough onset search window is selected around the onset time, ideally containing only noise in the first portion and noise + signal in the remaining window. Segments of noise and signal are selected to model, and care is taken to select these segments away from the onset time to ensure they lie entirely in their given regime, shown Fig. 1a. [2] The noise and signal segments are modeled, commonly with auto-regression (AR) algorithms.

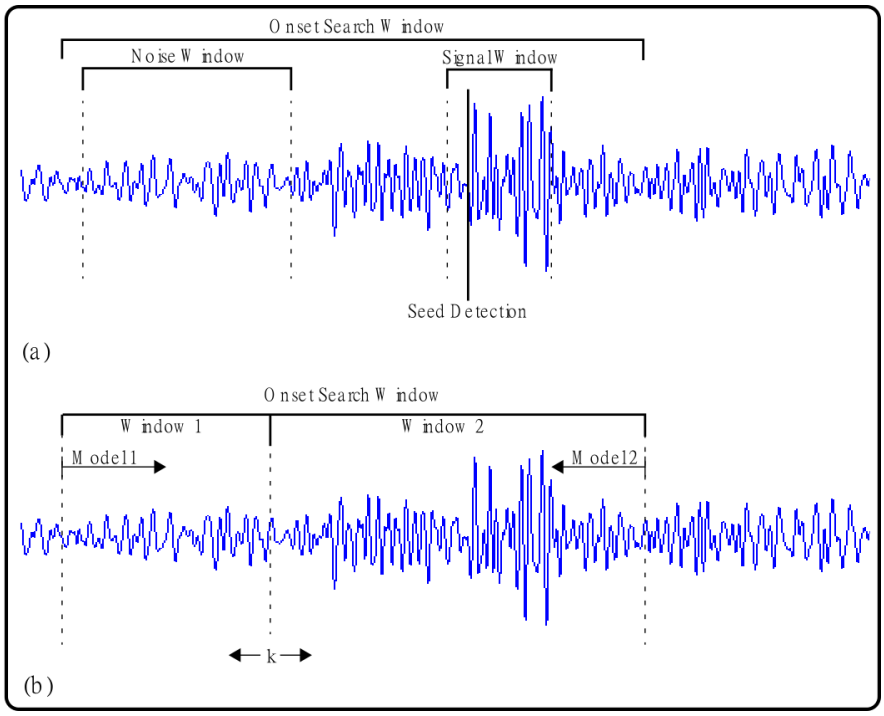

Fig. 1: Onset estimation search windows using the AIC and AR model estimation [2]

The onset search window is then bisected at position k, and window one and window two are respectively analyzed with the noise and signal models, shown in Fig. 1b. The AIC is then evaluated:

$$\Phi(k) = (k-1)log\ \sigma_1^2 + (N-k+1)log\ \sigma_2^2 + C \qquad (1)$$

where σ_1^2 and σ_2^2 are the variances of the estimated residuals (e_j) [2].

$$\sigma_1^2 = \sum_{j=1}^{k-1} \frac{e_j^2}{k-1} \text{ and, } \sigma_2^2 = \sum_{j=k}^{n} \frac{e_j^2}{N-k+1} \quad (2)$$

The AIC is iteratively calculated as K is swept through the onset search window, and the onset time is designated at the minimum AIC. Error in the AIC estimation can result from factors that include convergence on noise statistics for smaller events and emergent arrivals for small larger events. It is at this point that the analysts review the estimated onset time and adjust them as needed.

2.3 State of the Art and Limitations

Neural networks excel at non-linear pattern recognition. Convolutional neural networks (CNN) and recurrent neural networks (RNN) extend these capabilities to find translational patterns spatially and temporally, respectively. In particular, the CNN is a staple for processing 1D and 2D signals, including numerous audio and video analyses.

Over the last 3 years, the seismology community has leveraged deep learning techniques created in parallel fields with phase detection and identification as one of the leading areas of application. Onset time determination is a by-product of phase detection. I.e., identify the leading edge of the detected phase. There are numerous examples of such phase-detection/onset determination algorithms. Leading examples can be found in the following published works: [3,4,5,6,7]. The most recent of these examples are mature in their developmental process and have complex architectures.

Supervised learning can be delineated as either classification or regression. Classification is a categorical decision, but probabilities of that categorical decision can be extracted to yield extra quantitative measures, whereas regression is a quantitative estimation. Historically, CNNs have been developed for classification purposes. Primarily this has been the detection and classification of signal and spoken words in 1D data and objects in photos and videos for 2D data. Supervised deep learning analysis within the seismological community has also been predominantly classification. This may be because it suits the need it was designed for or because these classification techniques are readily available and well-developed.

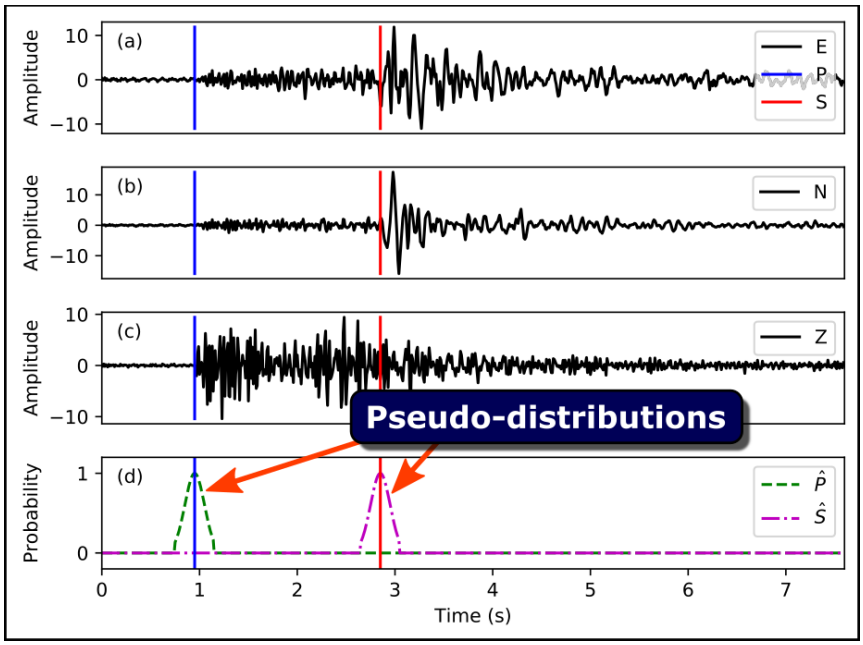

Fig. 2: PhaseNet's [5] classification approach utilized artificially generated pseudo-distributions. *Thus, the resultant onset time confidence is not properly related to the priors or current observations*

While phase detection and identification are classification problems, onset time determination is a regression problem. There are a few reasons that performing onset time determination via classification is inferior to regression:

1. Classification-based onset determination necessitates that all points in a window of interest be analyzed and given a classification. Thus, if a waveform contains 1000 points, the model must make 1000 estimations. This is not an efficient architecture for the onset problem. Once a phase is identified, it is more straightforward to have the model explicitly quantify the onset time. Here the entire model is working in concert for one specific estimation.

2. Extracting meaningful confidence estimations using classification techniques is limited at best. Extracted classification probabilities are estimations of the model's performance in general and not an estimation of the confidence of a given observation (phase). Even more sophisticated approaches, such as PhaseNet's probability distributions, are not related to statistical information in the data. Rather they are artificially imposed and not connected to meaningful parameters other than the architect's guess at a general uncertainty. Fig. 2 shows these pseudo-distributions used in PhaseNet

A potential advantage of treating onset-time as a regression problem is that the estimate of the onset time is not rounded to the nearest sample, as it is when treated as a classification problem. There is a long history of sub-sample measurement precision in seismology for the measurement of similar seismic sources using either the phase of the cross-spectrum [8] or by interpolating across samples for the peak of the cross-correlation function [9]. In these cases,

relative arrival times were measured to a precision of approximately 1 millisecond, or a tenth of a sample. These measurements are of relative arrival time, meaning that what is attained is precision rather than accuracy. For the relative-event location, that is often enough, and with a sufficiently large network of stations and a sufficiently broad distribution of earthquake sources, it is possible to develop sensitivity to absolute location, and hence accuracy, from precise measurements [10]. An intriguing question for the project is whether it is possible to obtain sub-sample accuracy for arrival time measurement. We explore this possibility as described below under tasks.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Dataset Curation and Cleaning

For our study, we obtained a partition of the IDC seismic dataset that was shared with us by AFTAC. Our subset covered the time period from August 12, 2015, to December 31, 2018, and contained approximately 20.71 terabytes of data. Before using these data in our study, we had to apply multiple quality control and cleaning steps to ensure that the dataset was reliable and accurate.

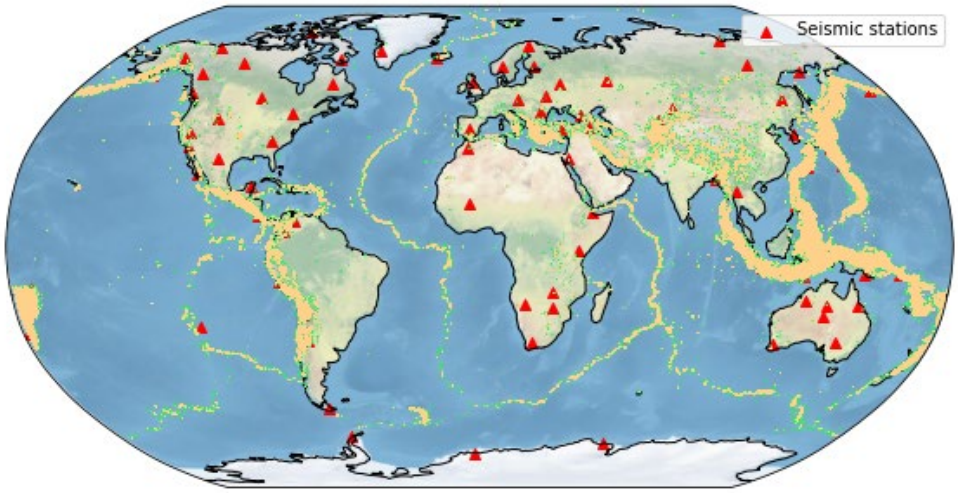

Fig. 3: Map of earthquakes and seismic stations in the dataset

Our first step was to select only the waveforms that were recorded using seismic instruments from the dataset. The seismic instruments we selected were identified as:

['CMC-3TB + Nanometr', 'CMG-3T + DM24S3EAM', 'CMG-3T Broadband 1', 'CMG-3T Broadband 3', 'CMG-3T+GD2 digitis', 'CMG-3TB + SHI AIM', 'CMG-3TB + Guralp C', 'CMG-3TB + Quanterr', 'CMG-3TB + SHI AIM2', 'CMG3T hybrid + DM2', 'CMG3TB + SHI AIM24', 'CMG3V Hybrid + DM2', 'DASE HM500', 'DASE HM500 seismom', 'DASE HM500 sensor', 'DASE LPHA-12S', 'DASE LPHA-12S sens', 'DASE LPHA-12s seis', 'DASE LPZA-12S', 'DASE LPZA-12S sens', 'DASE LPZA-12s

seis', 'DASE ZM500', 'DASE ZM500 seismom', 'DASE ZM500 sensor', 'GS-13+Europa-T dig', 'GS-13V', 'GS13+Europa-T digi', 'GS21 and AIM24S', 'GS_13H', 'GS_13H + AIM24S', 'GS_13V', 'GS_13V + AIM24S', 'Geotech', 'Geotech 20171A', 'Geotech 23900', 'Geotech 23900 + S', 'Geotech 23900 + AI', 'Geotech 23900 + SH', 'Geotech 23900 + Sc', 'Geotech 23900 sens', 'Geotech 7505A', 'Geotech 8700C', 'Geotech BB-13V', 'Geotech GS-13', 'Geotech GS-13 + ID', 'Geotech GS-13 + Na', 'Geotech GS-13 + SH', 'Geotech GS-13 + Sm', 'Geotech GS-13+Nano', 'Geotech GS-13+Scie', 'Geotech GS-13, Gur', 'Geotech GS-13H', 'Geotech GS-13V', 'Geotech GS-21', 'Geotech GS-21 + Na', 'Geotech GS-21 + SH', 'Geotech GS-21 + Sc', 'Geotech GS-21+Gura', 'Geotech GS-21+Nano', 'Geotech GS-21+Scie', 'Geotech GS13 + IDA', 'Geotech GS13 + Nan', 'Geotech GS13 + Q33', 'Geotech GS13 + Qua', 'Geotech GS13-1 + I', 'Geotech GS13-1 + Q', 'Geotech GS21+ SHI', 'Geotech KS-36000', 'Geotech KS-5400 Bo', 'Geotech KS-54000', 'Geotech KS-54000 +', 'Geotech KS-54000 B', 'Geotech KS-54000+S', 'Geotech KS-54000-I', 'Geotech KS5400 + S', 'Geotech KS54000 +', 'Geotech KS54000 se', 'Geotech KS54000+Sc', 'Geotech S-13', 'Geotech S-13 + DM2', 'Geotech S-13 + Nan', 'Geotech S-13 + Qua', 'Geotech S-13 + Sci', 'Geotech S-13+digit', 'Geotech S-500', 'Geotech short peri', 'Guralp', 'Guralp CMG', 'Guralp CMG-3', 'Guralp CMG-3ESP +', 'Guralp CMG-3ESPV +', 'Guralp CMG-3NSN +', 'Guralp CMG-3T', 'Guralp CMG-3T + Ca', 'Guralp CMG-3T + DM', 'Guralp CMG-3T + Eu', 'Guralp CMG-3T + Ge', 'Guralp CMG-3T + Gu', 'Guralp CMG-3T + ID', 'Guralp CMG-3T + Na', 'Guralp CMG-3T + Q3', 'Guralp CMG-3T + Qu', 'Guralp CMG-3T + SH', 'Guralp CMG-3T + Sm', 'Guralp CMG-3T Broa', 'Guralp CMG-3T Hybr', 'Guralp CMG-3T Seis', 'Guralp CMG-3T hybr', 'Guralp CMG-3T sens', 'Guralp CMG-3T+ CMG', 'Guralp CMG-3T+ Eur', 'Guralp CMG-3T+Euro', 'Guralp CMG-3T+Gura', 'Guralp CMG-3T+Nano', 'Guralp CMG-3T+SAIC', 'Guralp CMG-3T+Scie', 'Guralp CMG-3TB + A', 'Guralp CMG-3TB + G', 'Guralp CMG-3TB + K', 'Guralp CMG-3TB + N', 'Guralp CMG-3TB + Q', 'Guralp CMG-3TB + S', 'Guralp CMG-3TB+Kin', 'Guralp CMG-3TB+Nan', 'Guralp CMG-3V + Gu', 'Guralp CMG-3V + Na', 'Guralp CMG-3V Broa', 'Guralp CMG3-ESP', 'Guralp CMG3-ESPV', 'Guralp CMG3-ESPV +', 'Guralp CMG3-ESPV+G', 'Guralp CMG3-ESPV+N', 'Guralp CMG3-ESPV+S', 'Guralp CMG3-TB + S', 'Guralp CMG3ESP + N', 'Guralp CMG3T + Gur', 'Guralp CMG3T + Q33', 'Guralp CMG3TB + ID', 'Guralp CMG3TB + Na', 'Guralp CMG3TB + Q3', 'Guralp CMG3TB+Gura', 'Guralp CMG3TB+Nano', 'Guralp CMG3V + Nan', 'Guralp CMT-3T + Qu', 'Guralp Systems CMG', 'KS5400 + Quanterra', 'KS54000 + Quanterr', 'KS54000-01M1', 'Nanometric Europa-', 'Nanometrics T120 +', 'Nanometrics T240 +', 'Nanometrics Trilli', 'STS 2.5 + Quanterr', 'STS-2 + Nanometric', 'STS-2 + Quanterra', 'STS-2 HG + Quanter', 'STS-2.5 + Quanterr', 'STS-2.5 Seismomete', 'STS-2HG + Quanterr', 'STS2 HG + Quanterr', 'STS2 Streckeisen +', 'STS2.5 + Quanterra', 'STS2.5 Seismometer', 'Seismowave absolut', 'Strecheisen STS-2', 'Streckeisen STS-1', 'Streckeisen STS-1H', 'Streckeisen STS-1V', 'Streckeisen STS-2', 'Streckeisen STS-2+', 'Streckeisen STS-2.', 'Streckeisen STS-2/', 'Streckeisen STS-2H', 'Streckeisen STS-2V', 'Streckeisen STS2 +', 'Streckeisen high-g', 'T-240 + Quanterra', 'Trillium 120 + Q33', 'Trillium 120+ Q330', 'Trillium T-120PH +', 'Trillium T-120PH+', 'Trillium T120PH +']

Once we had selected the appropriate seismic instruments, we then applied multiple QC and cleaning steps to converge to a dataset that was suitable for our study. This included removing any data that were corrupted or incomplete, and ensuring that the remaining data were properly calibrated and aligned.

The AFTAC partition of the IDC seismic dataset contains three subdatasets: "IDCIDCX_RO", "IDCLEB_RO", and "IDCSEL3_RO". For our study, we selected "IDCLEB_RO" as our working partition and only used data from this subdataset.

To ensure the reliability and accuracy of our data, we then selected only those events for which we had a P-phase pick in the database. Our definition of a P-phase was broad and included P, Pn, Pg, and Px phases. This selection process resulted in a dataset of 1,043,358 event-station pairs (please refer to Fig. 3). It's worth noting that multiple waveforms may contain the same event if it was recorded on several stations.

To compile our dataset, we followed a specific procedure. For each event-station pair, we cut a waveform with a 90-second window centered at the P arrival time for that station. We selected only the BH[ZNE] channels from the available waveforms and applied a 0.1 Hz highpass zero-phase filter to pre-filter them. Additionally, we removed linear trends from the cut waveforms to ensure the reliability and accuracy of our dataset.

However, not all data files contained the selected channels or data, which left us with a final dataset of 288,356 event-waveform pairs. To train our neural network, we split this dataset into three parts: training (230,684 samples), validation (28,836 samples), and testing (28,836

samples). This division allowed us to assess the performance of our neural network and validate its predictions.

By following this procedure and compiling our dataset in this manner, we obtained a high-quality subset of the IDC seismic dataset that was suitable for our study. This dataset was then used to analyze seismic activity during the selected time period and provided valuable insights into the patterns and trends of seismic activity in the region.

3.2 Build/Train Deterministic Deep Learning Onset Regressor

This section discusses the development of the DL deterministic regressor; a regression model that estimates only one onset value. The subsequent section will incorporate methods for probabilistic estimation. This 2-staged approach of first building/refining the deterministic model followed by the probabilistic model was structured to isolate the developmental challenges. In the first stage, we wanted to ensure that the model and architecture were highly capable of estimating the onset time, and in the second stage, we added the complexity of probabilistic estimation.

Model architectures

Many model architectures were investigated while developing the DL onset regressor, and the strategy was, to begin with simple architectures and to increase the complexity systematically as necessary to achieve better performance. The models were quantified using mean square error (MSE) and reported using mean absolute error (MAE), which is in units of seconds and thus readily understandable.

First, we explored variations of convolutional neural networks (CNN). CNNs are an effective DL tool for extracting/ordering information from time-series data. Here greater than ten CNN layers were required for moderate performance. Then residual blocks (res-blocks) were introduced to maintain a gradient throughout the layers (Fig.4). By using the res-blocks, we could reduce the model size and improve the performance. We present a detailed discussion of the model layer types in section 3.3, and discuss the model metrics in section 4.1.

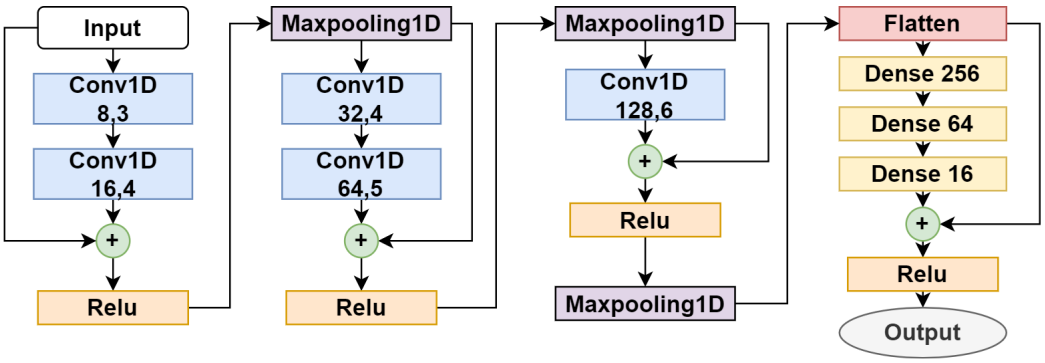

Fig. 4: Deterministic DL architecture using res-blocks

In addition to the data pre-processing and partitioning discussed in section 3.1, the three channels (3C) were globally normalized (i.e., normalized by the max absolute value found in all three components over the window duration). The channels were then stacked in a 2D matrix to form the input vector.

The model training curves are shown in Fig. 5. The two plots display the loss (MSE) and metric (MAE), respectively. The train and test curves shown in both plots are highly overlapped. Moreover, both curves are still trending down. These curves indicate that the model is not overfitting, and further training will improve the model's performance.

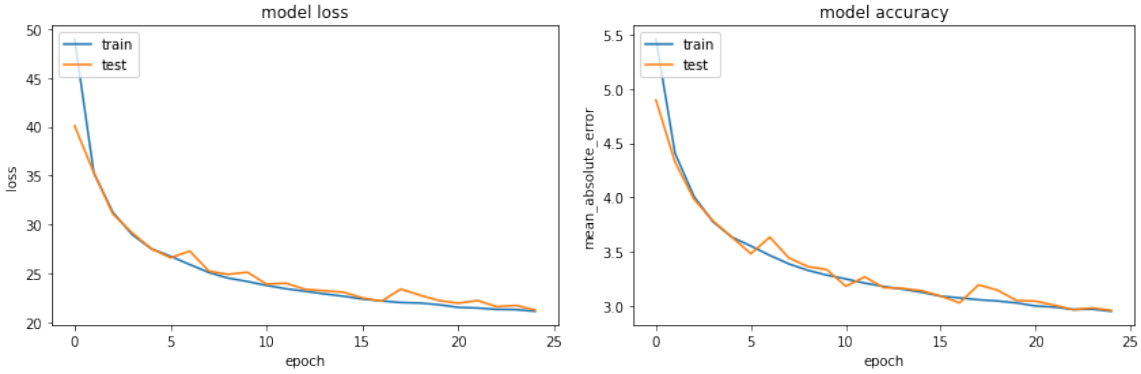

Fig. 5: Loss and metric (MAE) curves during the deterministic onset regressor training process

Sub-sample accuracy

In theory, one of the benefits of using a regression model (as opposed to a classification model) is in sub-sample accuracy. In this context, that means finding the exact onset time between two measurement points. It is difficult to quantify subsample accuracy on real data since there is no ground truth. However, we created a simple prototype problem with known sub-sample onsets. The signal was a triangular pulse with varied onset positions, pulse widths, and signal-to-noise ratios (SNR). Fig. 6 shows examples of pulses with different SNRs.

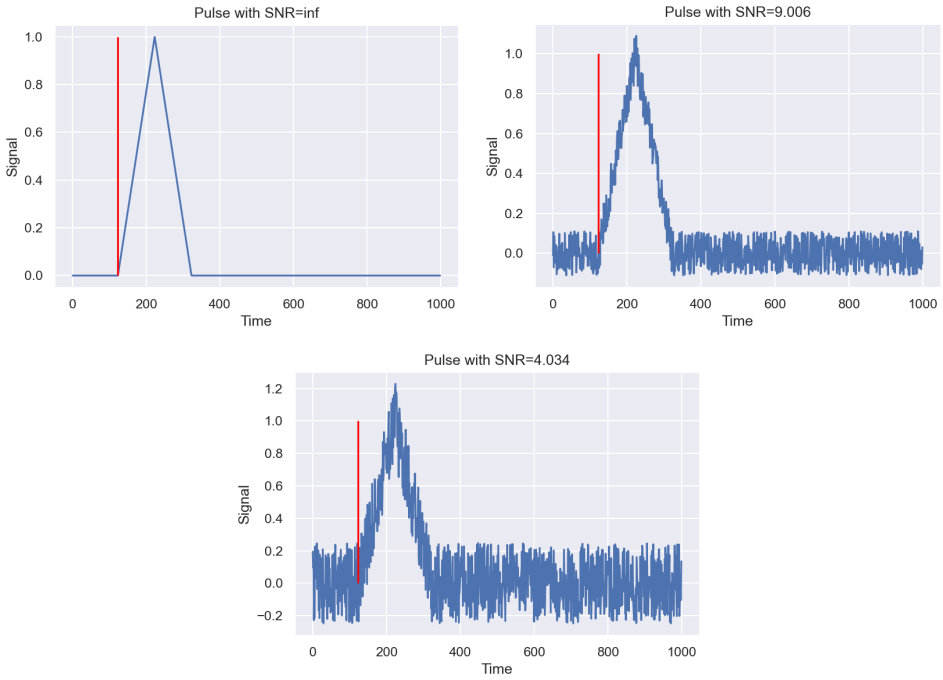

Fig. 6: Example signal pulse with varied SNR used to investigate subsample accuracy

Using the prototype pulse model, a deterministic CNN was trained to estimate the onset time. Note that the onset time and SNR are known quantities and irrational numbers. The results are presented in section 4.1.

3.3 Build/Train Probabilistic Deep Learning Onset Regressor

To build and train the Probabilistic Deep Learning Onset Regressor, which we refer to as "PhaseHunter," we designed a model that takes a seismogram segment as input and predicts the arrival time of the seismic phase within it. The model is composed of convolutional layers that extract features from the data, followed by a fully connected layer that maps the features to N outputs representing the predicted onset time of the desired phases (e.g. in the

case of N=2 - P and S phases, please refer to Fig. 7 for the detailed architecture of the model).

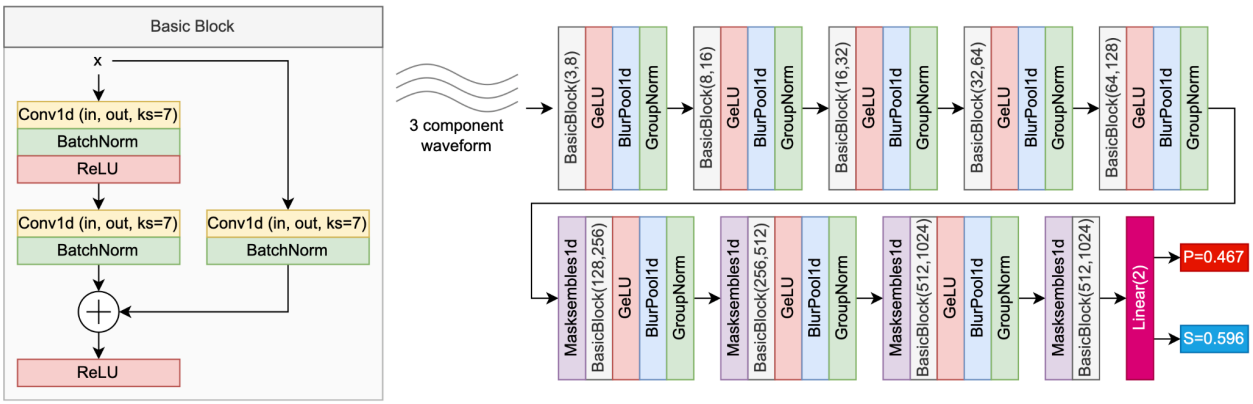

Fig. 7: Architecture of PhaseHunter. PhaseHunter takes a 3-component seismogram in the time domain as an input, and outputs N predictions of onset times as an array of floats. Please refer to the text (chapter 3.3) for the information on each layer used in this neural network

To improve the uncertainty estimates of the network, we also included Masksemble layers, which are a type of dropout layer (described later in the text). During training, we optimized the model using Mean Absolute Error (MAE) as the loss function, which measures the difference between the predicted and labeled phase arrival times. We chose MAE because of its useful properties, such as being insensitive to outliers and not sensitive to the order of the predictions and true values.

For the P phase, we compared the predictions to the ground truth labels, while for the S phase, we compared the predictions only to the subset of the data where the labels are available. This was necessary because many labels for the S phase do not exist. The network's parameters were updated to minimize the difference between the predictions and the ground truth labels.

Once the model is trained, it can be used for inference (prediction) on new seismograms. The same convolutional and fully connected layers are used to process the data during inference, and the model uses its learned knowledge to make a prediction of the phase arrival time.

Here we briefly explain each type of layer is used in PhaseHunter:

- **Conv1D** is a PyTorch layer used for 1D convolutional operations on 1D input tensors. It applies a specified number of filters (also known as kernels) to the input tensor to extract features. The filters slide over the input tensor in a specified stride and generate feature maps. Conv1D is commonly used in deep learning models for processing time series data, such as in speech recognition, signal processing, and natural language processing applications.

- **BatchNorm** (Batch Normalization) is a technique used in deep neural networks to normalize the inputs of each layer in a batch-wise manner. It works by normalizing the inputs to the layer using the mean and standard deviation of the inputs of the current batch. This helps to stabilize and speed up the training process by reducing the effect of internal covariate shift, which is a change in the distribution of layer inputs that occurs during training as the parameters of the previous layers change. By normalizing the inputs, BatchNorm reduces the dependence of the gradients on the scale of the parameters, making it easier to optimize the model. Additionally, BatchNorm acts as a regularizer by adding noise to the layer inputs, which can help to prevent overfitting.

- **GroupNorm** is a type of normalization layer in neural networks, similar to BatchNorm. However, instead of computing normalization statistics over the entire batch, GroupNorm splits the channels of the input into groups and computes the normalization statistics within each group. This can be useful when the batch size is small or when the inputs have a high variance in their feature statistics across channels. The groups are typically chosen to be smaller than the total number of channels, but larger than the batch size. By computing normalization statistics within each group, GroupNorm reduces the dependence on batch size, which can lead to improved training stability and convergence.

- **ReLU** stands for Rectified Linear Unit. It is an activation function used in neural networks that applies a non-linearity to the output of a neuron. It is defined as $f(x) = \max(0, x)$, where x is the input to the function. The ReLU function is commonly used in deep learning models because it is computationally efficient and allows the network to learn more complex features. It also helps to alleviate the vanishing gradient problem by preventing saturation of the neuron output.

- **GeLU** (Gaussian Error Linear Units) is an activation function used in neural networks. It is a variant of the ReLU activation function. Like ReLU, GeLU is a non-linear activation function that is applied to the output of a neuron in a neural network. It is designed to introduce non-linearity into the network, which is important for the network's ability to learn complex relationships in the data. GeLU has been shown to outperform ReLU in some cases, particularly in natural language processing tasks.

- **BlurPool1D** is a type of pooling layer in deep learning, specifically designed for 1D input data such as signals, audio, and time-series data. The main purpose of the layer is to perform pooling operations while reducing the aliasing effect that occurs in traditional pooling operations, which can lead to a loss of information.

  The BlurPool1D layer applies a blur filter to the input data before performing pooling. The blur filter helps to smooth the input data and reduce the aliasing effect, which can lead to improved accuracy in downstream tasks such as classification and regression. The pooling operation is then performed on the smoothed data, reducing the size of the input and helping to capture the most salient features of the data.

Overall, the BlurPool1d layer is useful for reducing the negative effects of pooling on 1D input data, while still allowing the network to capture important information from the data.

- **Masksembles1D** is a layer in deep learning that provides a way to estimate uncertainty in a neural network's predictions. It is a type of dropout layer that introduces randomness during training by randomly masking (i.e., setting to zero) certain activations in the previous layer. The masking is performed independently for each sample and channel, which allows for the estimation of the distribution of possible outputs. During inference, the layer scales the activations by the probability of the activation being non-zero, which results in a weighted average of the possible outputs. This approach has been shown to improve the accuracy and uncertainty estimates of deep learning models, particularly in situations with limited training data (see more below).

- **Linear** is a PyTorch layer that implements a linear transformation of the input tensor. Given an input tensor x of shape (batch_size, input_dim), Linear applies a matrix multiplication between x and a learnable weight matrix W of shape (input_dim, output_dim) and adds a learnable bias vector b of shape (output_dim). The output tensor of the layer is of shape (batch_size, output_dim).

  The layer can be used for various purposes, such as implementing a fully connected layer in a neural network, or for linear regression tasks where the goal is to predict a continuous value from a set of input features.

To make our deep learning model perform better, we use multiple augmentations during training. First, we convert the target phase arrival times from time [0,1] to the corresponding sample index by multiplying them by the length of the original sample. Then, we randomly shuffle the sample by selecting a random start index for the cropped sample. This helps introduce variability and prevent the model from overfitting to specific patterns in the data [11]. We then cut the sample to the desired length by selecting a subset of the sample starting at the chosen start index and ending at the end index, which is calculated by adding the desired length to the start index. We then normalize the waveform using global normalization (absolute maximum out of 3 channels). Finally, we apply two augmentations to the sample: rotation and channel dropout. The rotation augmentation randomly rotates the waveform by an angle, while the channel dropout augmentation randomly sets one of the channels in the sample to a small constant value. Overall, these augmentations help introduce variability and noise into the training data, which can help the model learn more robust and generalizable patterns, leading to improved performance.

For the uncertainty estimation in this paper, we leverage the method known as Masksembles [12]. This method is based on the observation that dropout can be seen as a special case of ensembling. Dropout is a regularization method used in training neural networks, in which a random subset of the connections between the layers of the network are dropped out (set to 0) during each training iteration. This has the effect of reducing overfitting, as it forces the network to learn multiple independent representations of the data, rather than relying on one

specific set of connections. Dropout methods can also be used for weight regularization, which was shown to help suppress the effects of label noise [13]. Masksembles extend this concept by using masks, rather than randomly dropping connections, to achieve a similar regularization effect.

In addition to its use as a regularization method, dropout - and by extension, Masksembles - can be used to estimate the confidence of a trained neural network in its predictions. Because dropout effectively trains multiple independent versions of the network during each iteration, the resulting model can be treated as an ensemble of smaller networks, each of which makes its own prediction on a given input. By aggregating the predictions of these sub-networks, we obtain an estimate of a distribution of a model's predictions that expresses the uncertainty inherent in the individual sub-networks (see Fig. 8).

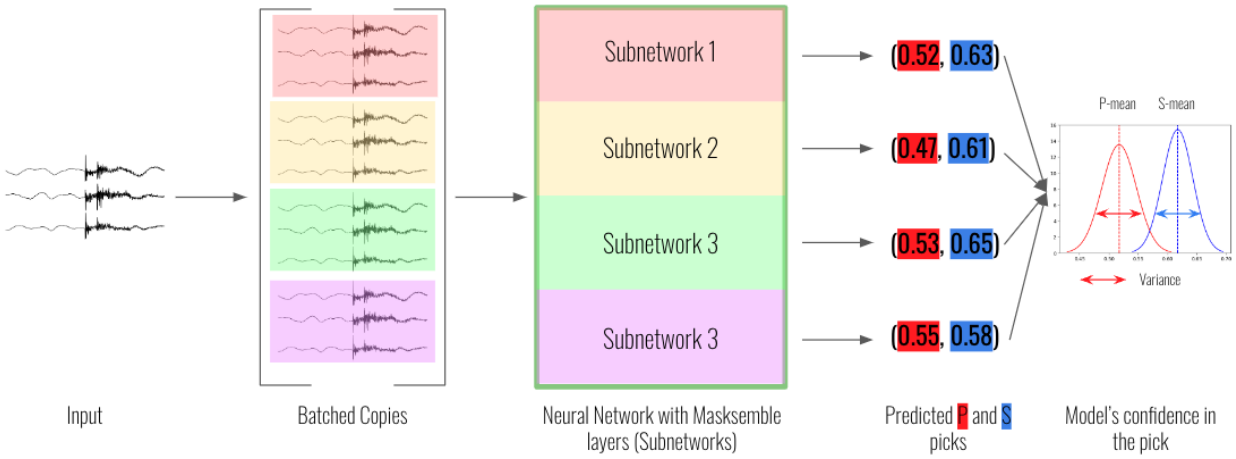

Fig. 8: Schematic explanation of the process used for uncertainty estimation. *The figure illustrates the Masksembles method, which uses masks to drop out a random subset of activations in a neural network during training. This creates an ensemble of sub-networks that can be used to estimate the model's confidence in its predictions. By combining the predictions of these sub-networks, a distribution of confidence values can be obtained that takes into account the inherent uncertainty in the individual sub-networks*

Masksembles can be used to estimate the confidence of a trained neural network in its predictions. This is because dropout, and by extension, Masksembles, effectively trains multiple independent versions of the network during each iteration. Each sub-network in the ensemble is randomly masked during training, and therefore, it learns to make predictions based on different subsets of features. This results in a diverse set of sub-networks, each with its own strengths and weaknesses.

By aggregating the predictions of these sub-networks, we obtain an estimate of the distribution of a model's confidence in its predictions. This distribution takes into account the uncertainty inherent in the individual sub-networks. Specifically, if the predictions of the

sub-networks are consistent for a given input, the model's confidence in its prediction is high. On the other hand, if the sub-networks produce divergent predictions, the model's confidence is lower, as there is higher uncertainty in the prediction.

Therefore, by using Masksembles, we can estimate the confidence of a model's predictions by examining the variance (standard deviation) of the sub-network predictions. If the variance is low, the model is confident in its prediction, while a high variance indicates that the model is less certain. By taking into account the uncertainty inherent in the sub-networks, Masksembles can provide more accurate and reliable confidence estimates for neural network predictions.

Tensorflow Probability

Before settling on the Maskensemble probabilistic method (discussed above), the team investigated using Tensorflow Probability (TP). Where Maskensembles estimates many discrete samples, TP estimates the parameters that describe a predefined distribution model. Here we began the investigation with a normal distribution. As shown in Fig. 9, a probabilistic DL model can be created with TP by adding two additional layers at the end of the deterministic model: the Dense Variational and Distribution Lambda.

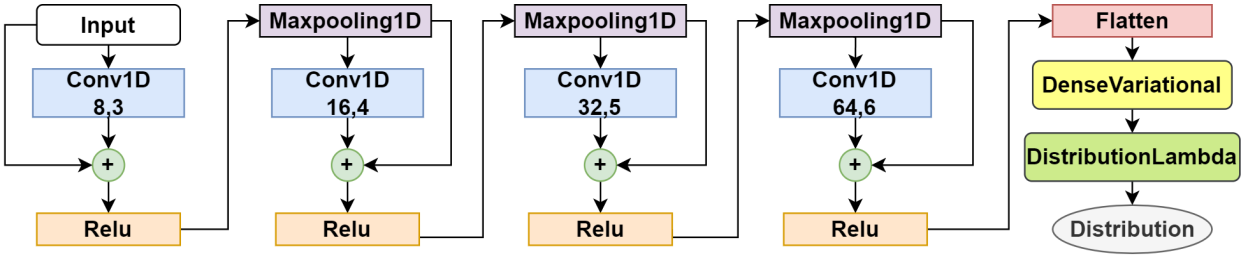

Fig 9: Example probabilistic model architecture using Tensorflow Probability. *The Dense Variation and Distribution Lambda layers enable the probabilistic estimation*

The Dense Variational layer uses the convolutional filters to perform variational inference and create a surrogate posterior. While the Distribution Lambda creates the user-defined distribution. Another important difference between the TFP model and its deterministic counterpart is in the loss function used to train the model. The deterministic models used MSE to calculate loss, but the TFP models used the negative of the log-likelihood (negloglik or NLL).

While the TFP model behaved in the intended way, it produced broader distributions when the estimates were more error-prone, it tended to train much more slowly and had more residual error in its mean prediction. That is, the onset picks were not as good as the deterministic model or the Maskensemble method. As such, the TFP results will not be discussed in detail in the results section, though Fig. 10 shows how the distribution broadens on the pulse problem when more noise is applied to the signal.

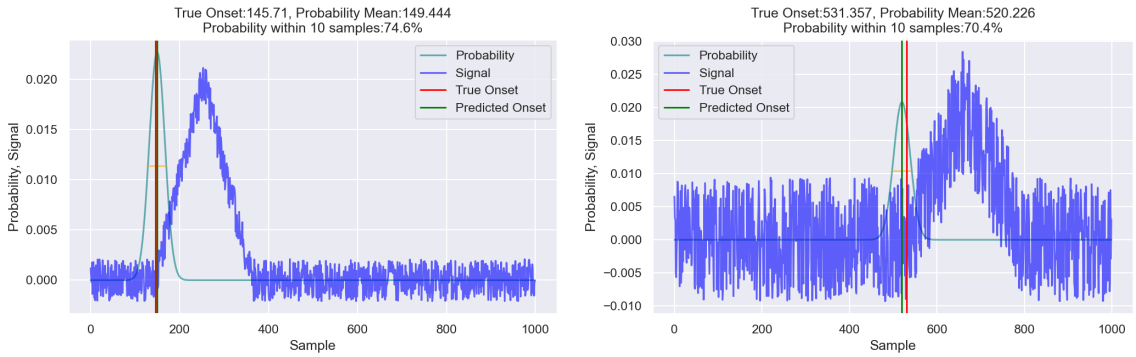

Fig 10: Example outputs using Tensorflow Probability to estimate the onset times in the pulse problem

4.0 RESULTS AND DISCUSSION

In the sections below, we present the results of the deterministic and probabilistic onset regression estimators, followed by the software package developed for AFTAC's testing purposes. The analysis of the probabilistic estimator is more detailed since it is the target product.

4.1 Deterministic Onset Estimation

Many different variations of DL architecture were investigated while developing the deterministic onset regressor, but the model described in Fig. 9 proved to be the best performing. An example pick is shown in Fig. 11, where the difference between the analyst and model pick is 0.05 seconds or two samples, given that the data are sampled at 40 Hz. The discussion below regarding the deterministic model is on results using this particular model.

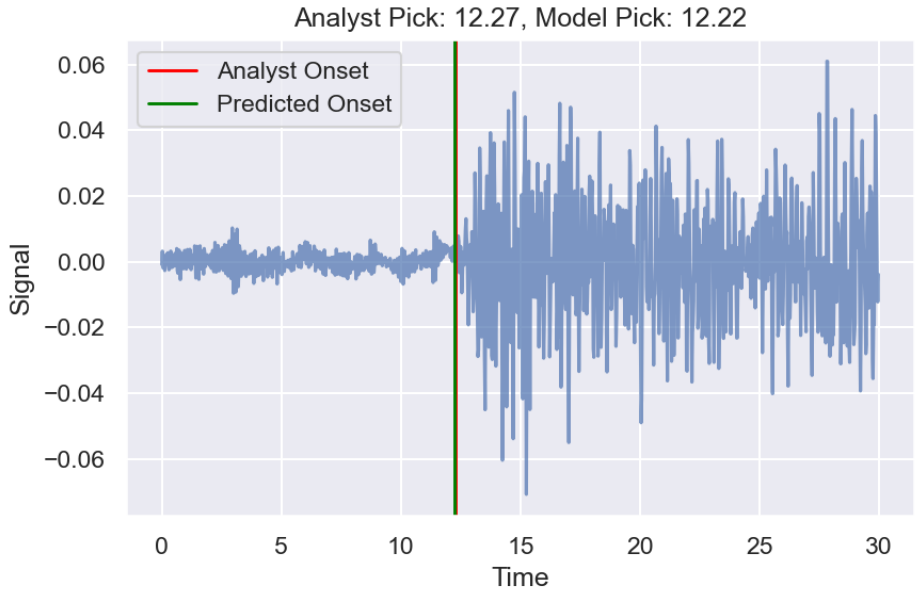

Fig 11: Example onset pick using the leading deterministic model using res-blocks

The MAE for the deterministic model is 0.72s, and the statistical distribution of estimation error is presented in Fig 12. Across the whole data set, the mean and median errors are 0.035 and 0.015s, respectively. This indicates that the model is making estimations that are, on average, about one sample later than the analyst. The residual distribution also shows that the error does not fit a normal distribution. Rather there is a tight central mode between ±0.2s and a long tail outside of that. We investigate the source of this tail in Sec. 4.2

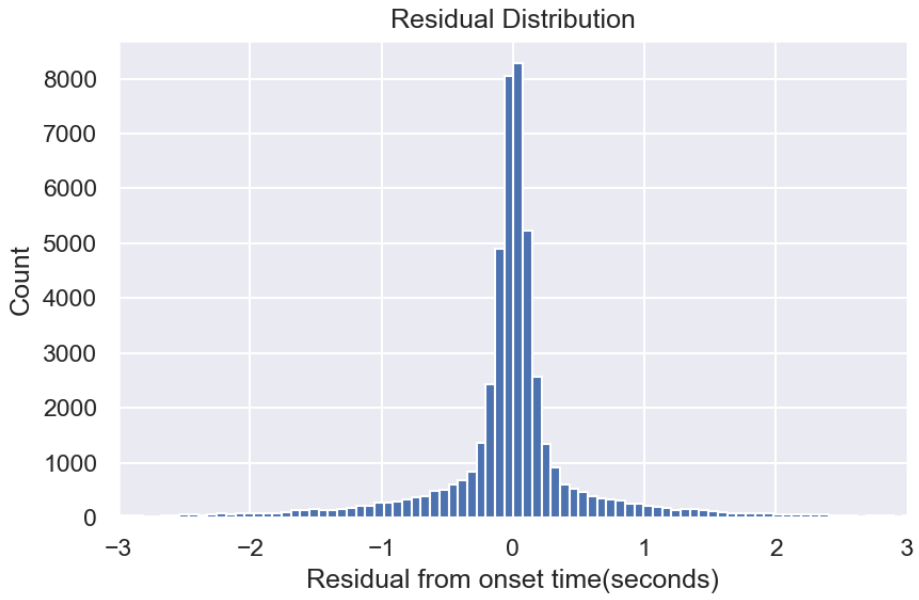

Fig 12: Residual error for the deterministic onset model

Fig. 13 shows a scatter plot of actual and estimated (predicted) onsite times. A perfect result is unity, actual = predicted, or a line of y=x. Most estimates follow this trend. But this plot also gives insight into the spread of the outliers. Note that the upper and lower bounds constrain the range of the biggest possible outliers, and they are asymmetric; an actual value of 11.75 has an estimation of 20, and an actual value of 18.5 has an estimation of 12. The skew in error is due to the training conditions since the model learns that there is never a prediction outside of the training limits. This is a form of bias by design. Later when developing the probabilistic onset estimator, this effect is minimized by expanding the allowable onset time in the training window.

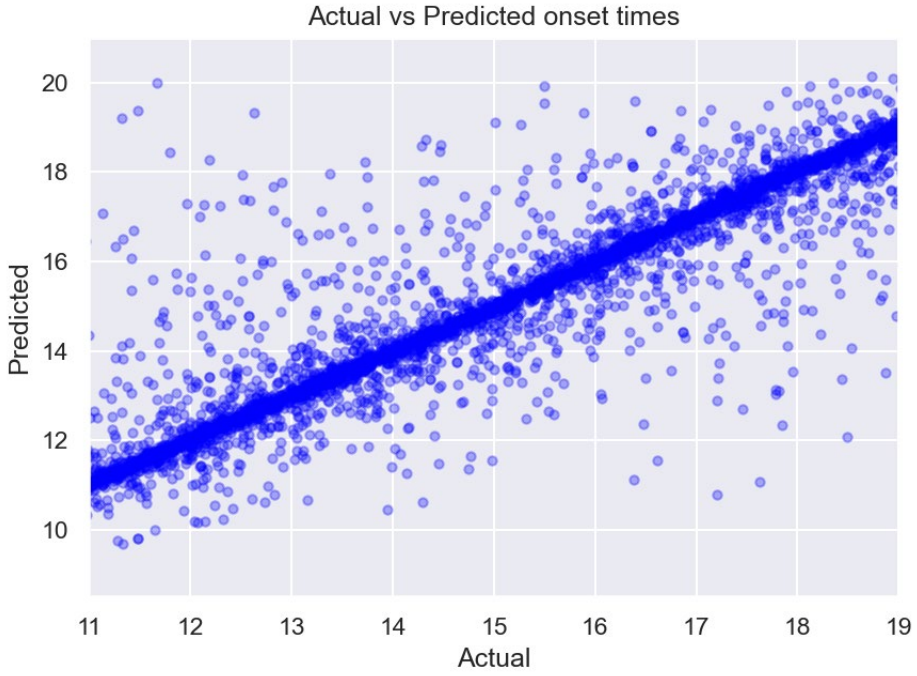

Fig 13: Actual (analyst) vs predicted (model estimated) picks. Axes are in seconds. X=Y is the ideal result

The SNR correlates to the estimation error as expected. Fig. 14 shows the estimation error vs. SNR. This is a subset of the test data capped at an SNR of 35. The error progressively decreases as the SNR increases beyond what is shown in Fig. 14. In this figure, the orange line indicates the median and the yellow lines bound the 95% confidence interval. At the upper limit of SNR shown, the 95% confidence interval is just under ±1s error. The error starts to increase more rapidly at SNR < 10. Also, at low SNRs, there starts to be a bias toward a late estimation. At the lower SNR limit of the figure, there is a positive shift in mean error and 95% confidence interval. The shift in mean error is small, < 0.25s, but the confidence interval shift is more pronounced with an approximate range of -2s to 4s.

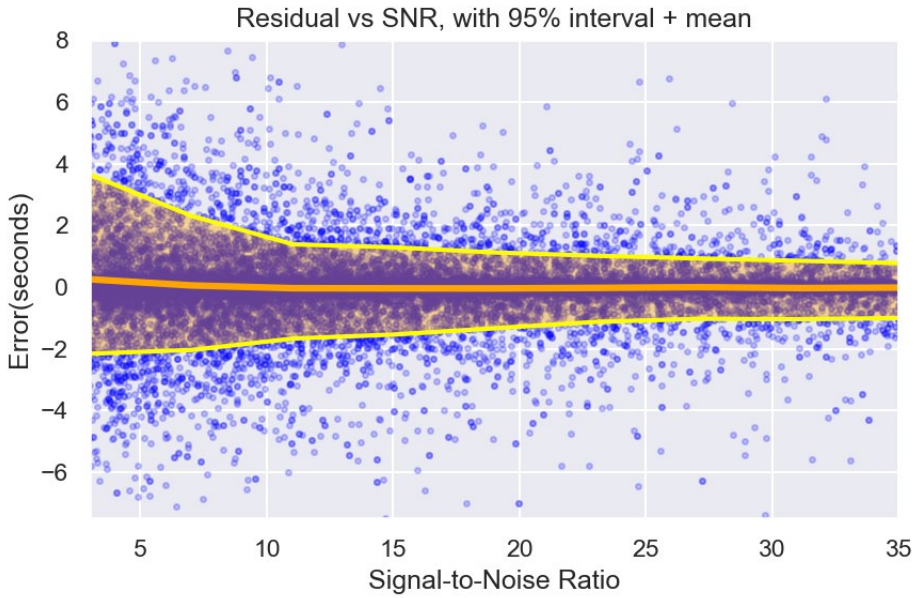

Fig. 14: Estimation error vs. SNR with the mean (orange line) and 95% confidence interval (yellow band). *Note the bias towards late estimations grows for SNR < 10*

Sub-sample accuracy on pulse prototype

In the following section, we examine the results from the pulse problem designed to test the capacity for sub-sample accuracy in the DL regressor. Fig. 15 shows the estimation error histogram of signal without noise. The MAE is 0.0145 samples, an initial result that indicates that sub-sample accuracy is achievable. The histogram also displays a shift toward late estimations.

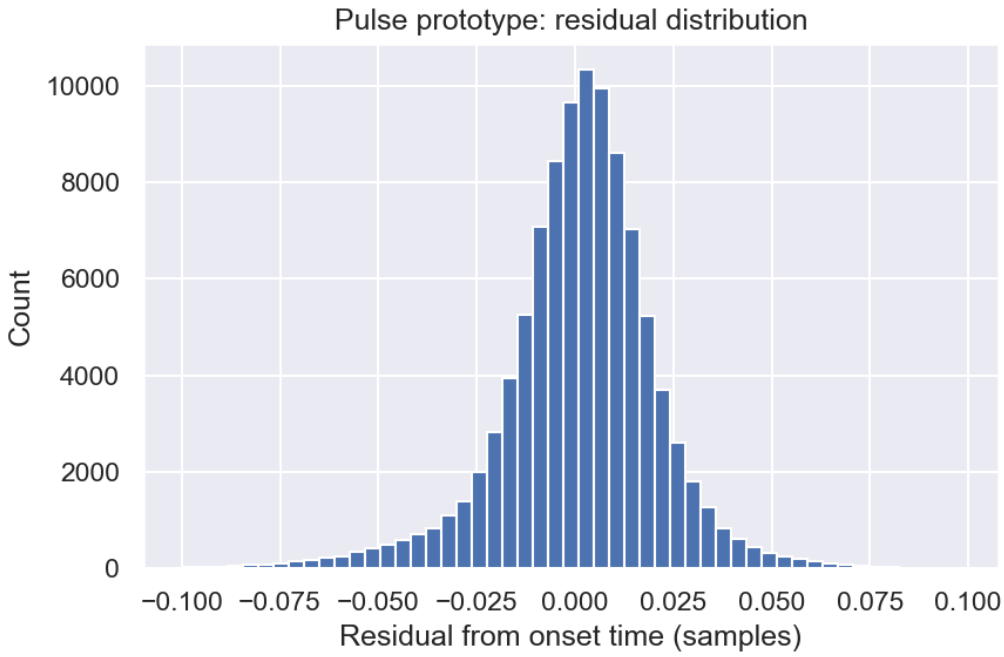

Fig 15: Residuals on the pulse prototype problem with no noise showing that DL regression can produce sub-sample accuracy

The effect of SNR on estimation error is shown in Fig. 16. The trend is similar to that seen with real data in Fig. 16. Increased SNR reduces estimation error. Note that this figure does not display any results where SNR > 1, where the error trends toward zero. This plot also displays a positive error bias at low SNRs, and can be seen on the 95% confidence interval as it deviates from linearity at SNR < 0.4. A further indication that this methodology is prone to late estimations for low SNRs.

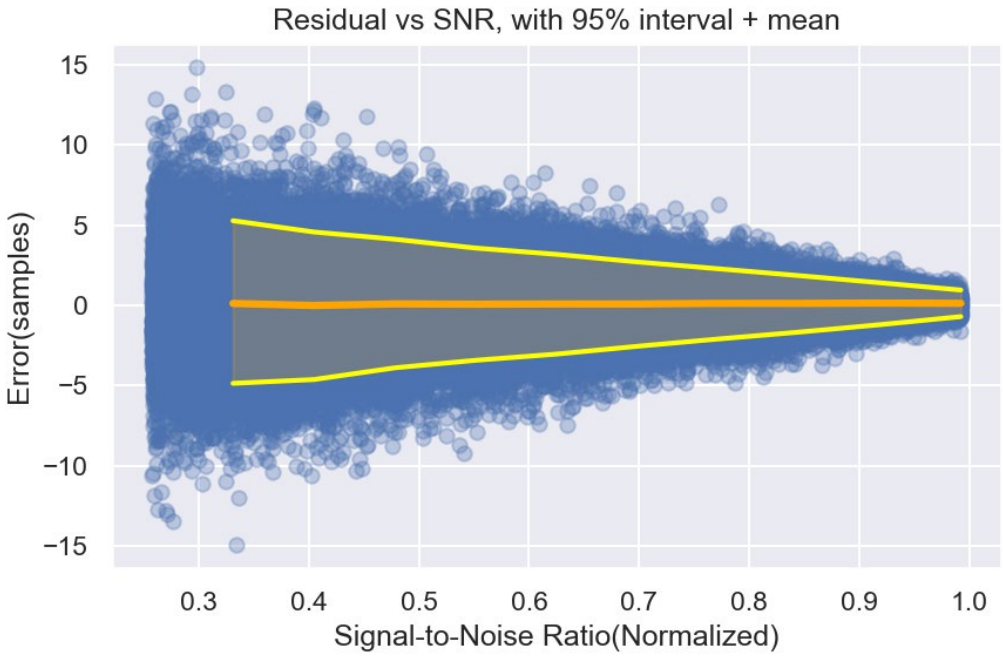

Fig 16: Residuals on the pulse prototype problems when noise is added. *The orange and yellow lines are the mean and 95% confidence intervals, respectively*

4.2 Probabilistic Onset Estimation

Using PhaseHunter we were able to achieve MAE of 0.35 seconds on the test set. This improvement was possible due to uncertainty estimation. For example, if we don't utilize the available information of model uncertainty, the results for the same split would be 0.5786s. Because we estimate the uncertainty, we gain the ability to filter out picks with the highest uncertainty (in this case, we removed 25% of the test split samples). The distribution of residuals is shown in Fig. 17. The residual shown here is estimated onset - analyst onset, with a mean of 0.12s and a median of 0.017s, indicating that the prediction time is slightly later than the analyst prediction. An example of prediction is demonstrated in Fig. 18.

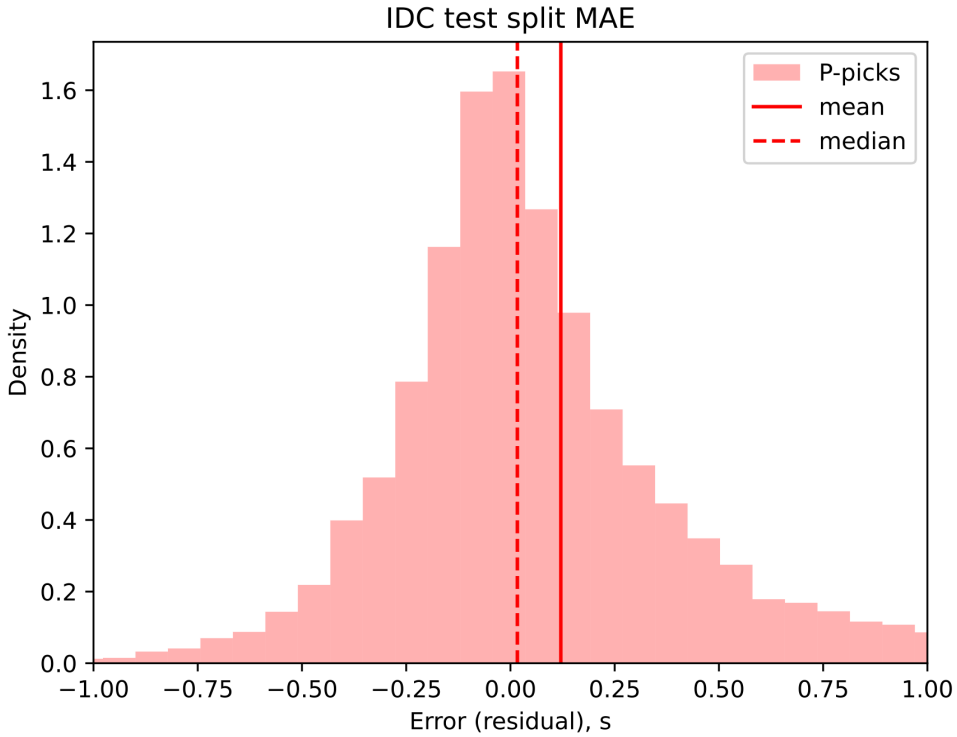

Fig. 17: Distribution of residuals. *Mean is shown as a bold red vertical line, median is shown as a dashed red line. Residuals are defined such that an estimated time later than the analysts prediction plots as a positive residual*

The Masksembles approach is an alternative to a more conventional dropout. While the effectiveness of the method will depend on the specific characteristics of the data, the task at hand and the implementation of the approach, the authors of the Masksembles paper argue that their approach has some advantages over dropout in certain situations

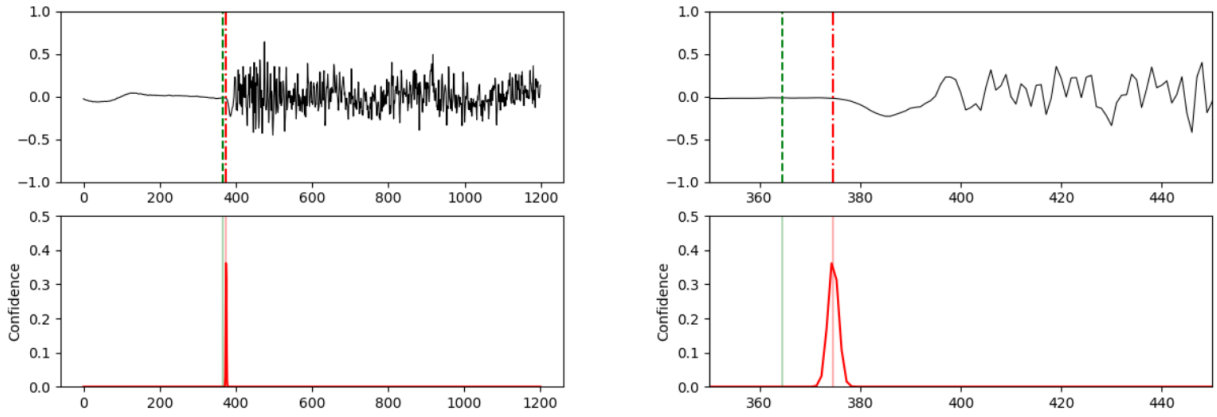

Fig. 18: a) A full waveform, with predicted P onset time shown in red and ground truth shown in green. b) Zoom-in, demonstrating how the prediction of the Neural Network is different from the ground truth

One potential advantage is that it can provide more stable and consistent regularization. Because the masks used in Masksembles are fixed and do not change during training, the regularization effect is more predictable and consistent compared to a dropout, where the dropped connections are randomly selected at each iteration. This can make it easier to tune the regularization strength and may result in more stable training. Another potential advantage is that Masksembles can provide stronger regularization when applied to deeper networks.

These potential advantages come with trade-offs. For example, the Masksembles approach requires more memory to store the masks, compared to traditional dropout, which only requires storage for the dropped connections. Additionally, the fixed nature of the masks used in Masksembles may make it less effective in certain situations, such as when the data distribution changes during training.

Overall, the Masksembles approach may provide some advantages over traditional dropout in certain situations, but it is not necessarily superior in all cases. As with any regularization method, it is important to evaluate its effectiveness on a case-by-case basis and choose the approach that works best for a given task.

Sources of Bias

In the following section, we investigate sources of bias found in PhaseHunter, the DL probabilistic onset regression algorithm utilizing Maskensembles. In section 4.1, the deterministic model showed an aggregate shift toward a late estimation. Here we parse the data by feature to see if some are more susceptible to bias. Each plot shows the results of the inference on the test dataset of 13,786 P-wave observations.

Fig. 19 shows estimation error vs SNR on a log-normal plot. There is a clear correlation between SNR and error: signals with low SNR tend to have greater error in the estimation, which is an expected trend. In this plot, a bias toward late (or early) prediction is represented by asymmetric error. While it seems that there is a positive bias (late estimation) across all SNRs, the magnitude of the differential is greater at lower SNRs, which can be seen with the positive shift in the 95% confidence interval and the median. Note the noise in the 95% confidence interval at low SNRs is due to the fact that there are few samples per bin in these regions, an artifact of the log-normal plot.

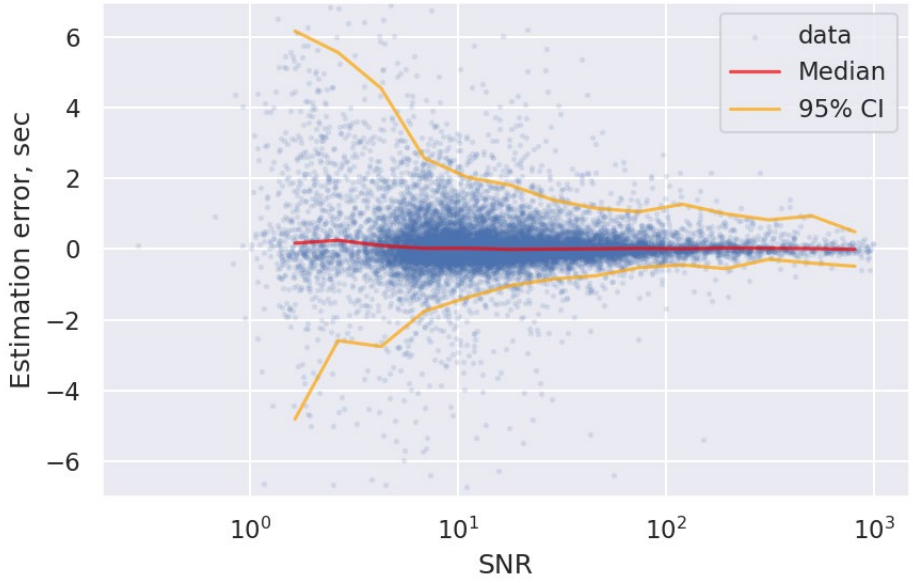

Fig 19: PhaseHunter estimation error vs. SNR, a positive bias increase with decreasing SNR

Fig. 20 and 21 show PhaseHunter's estimation error as a function of magnitude and distance. In both cases, there is an aggregate positive bias, but the bias does seem to increase with decreasing magnitude and distance. However, at very small values, the error decreases. In these two cases, more data could clarify a potential trend.

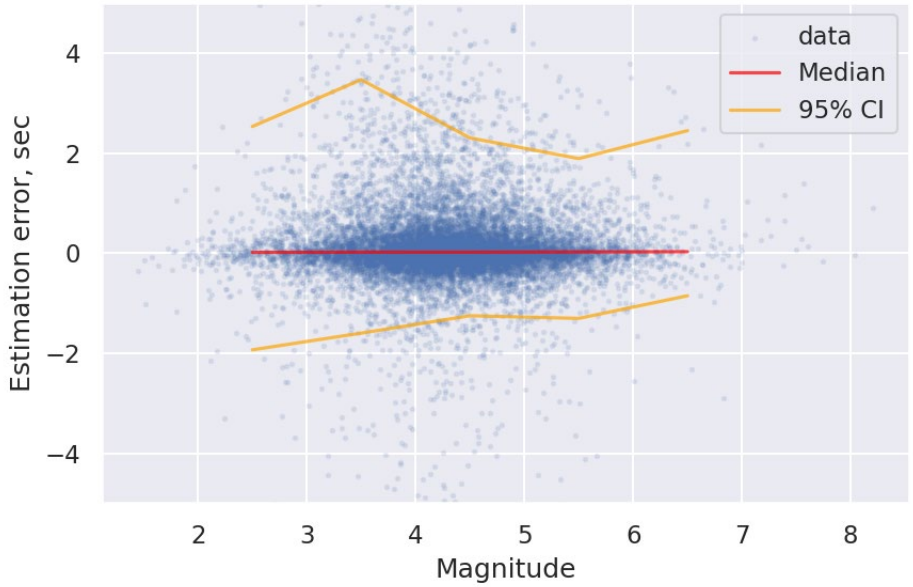

Fig. 20: PhaseHunter estimation error vs. magnitude

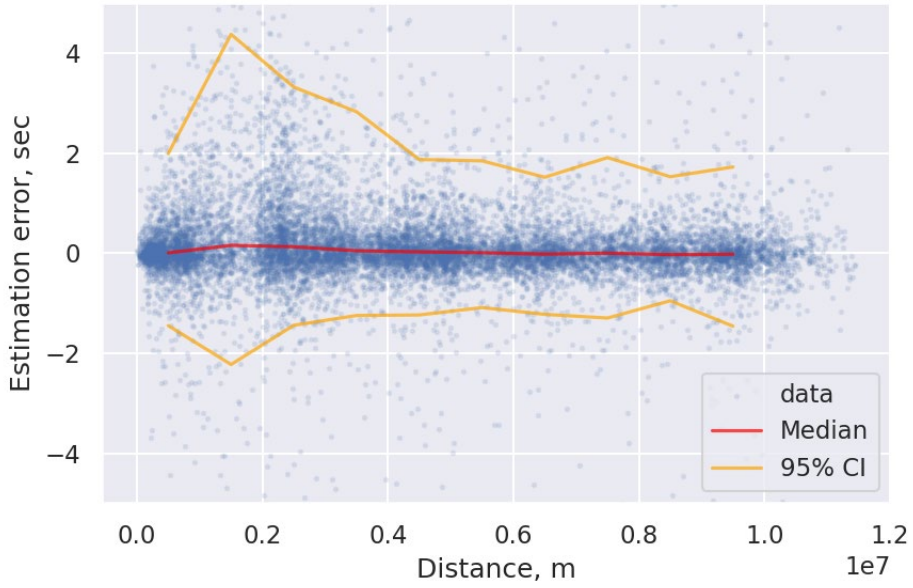

Fig. 21: PhaseHunter estimation error vs. distance

Fig. 22 and 23 show PhaseHunter's estimation error as a function of depth and energy. Energy is the mean of the squared signal starting from the onset time. With these parameters, there is an aggregate positive bias. Also, there is increased error and positive bias with decreased parameter value.

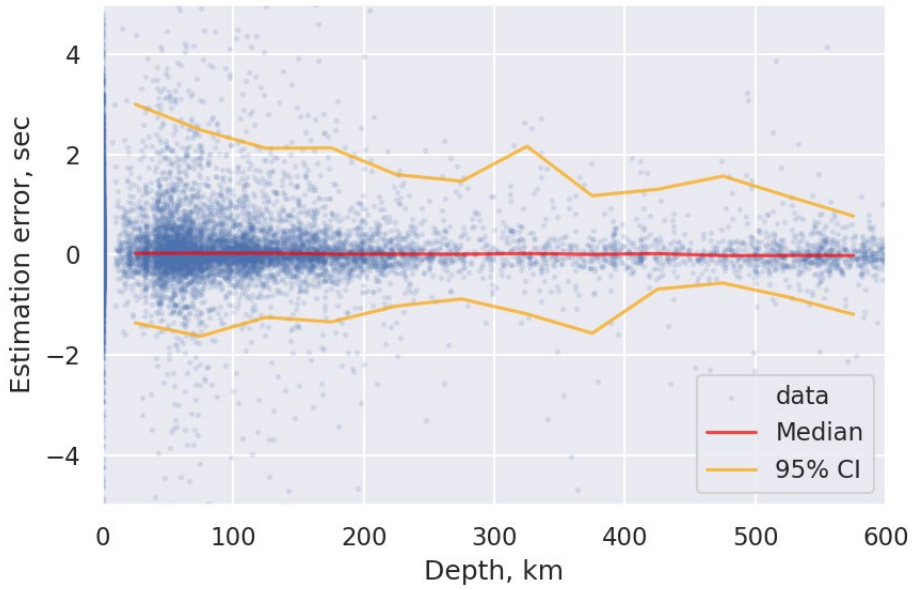

Fig. 22: PhaseHunter estimation error vs. depth

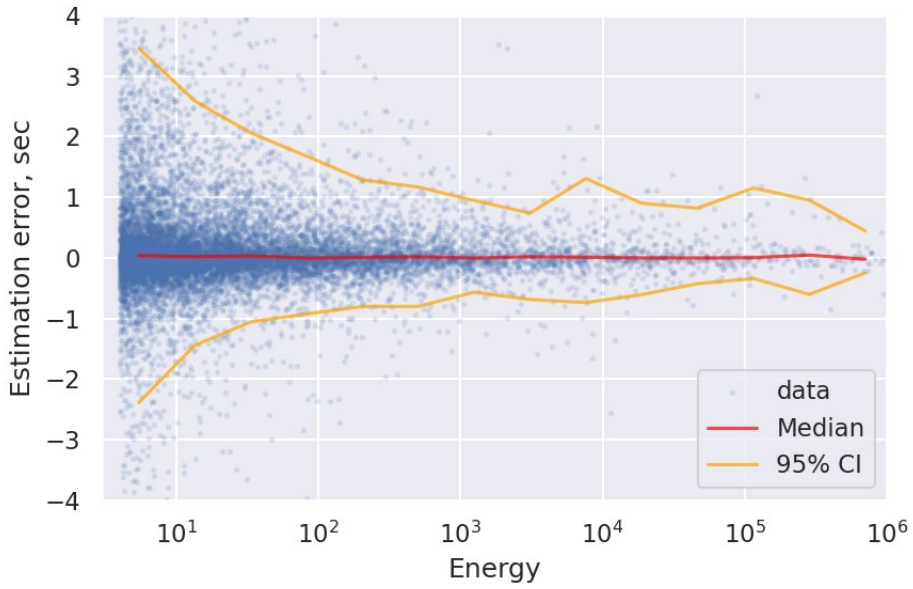

Fig 23: PhaseHunter estimation error vs. energy

Use of Probabilistic Confidence (STD) with Estimation

In this section, we analyze the probabilistic nature of PhaseHunter using the P-wave test dataset discussed above. For each observation, we investigate the error derived from the mean value of the multi-estimation regression and the confidence as expressed by the STD. The aim is two-fold, 1) to investigate if PhaseHunter's confidence is useful, and 2) to understand how it can be used in a deployment scenario.

To assess PhaseHunter's probabilistic utility, its estimated STD is plotted as a function of the error, Fig. 24. In deployment, the estimation error will not be known unless an analyst reviews the pick, but the STD is produced along with the pick. Here we see that there is a correlation between estimation error and STD, indicating that PhaseHunter's STD can be used as a measure of pick confidence. Moreover, the error rate greatly increases around an STD of 0.1s.

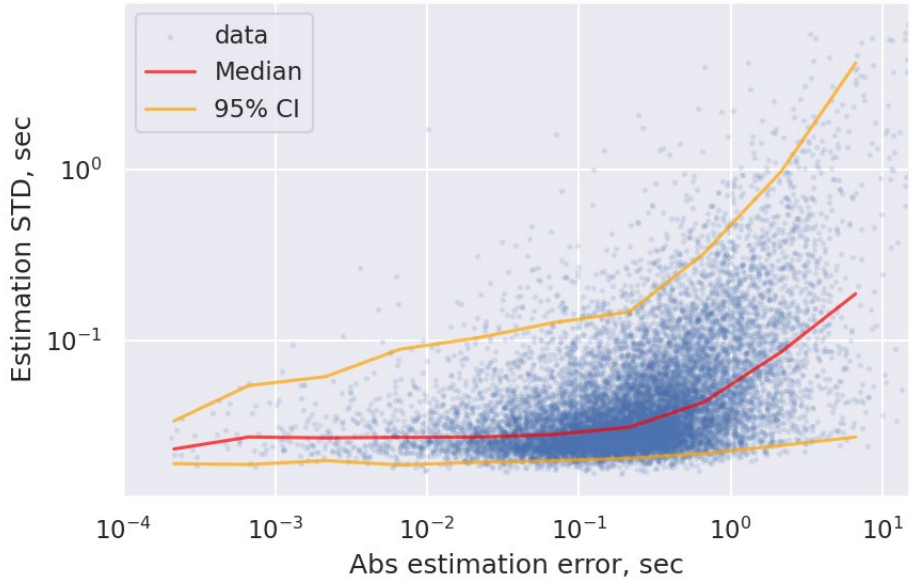

Fig. 24: PhaseHunter's estimation error vs STD on the test dataset. *A clear trend is seen, indicating that STD can be used as a measure of prediction confidence*

PhaseHunter's STD can be used to assess the confidence of the pick estimation. One deployment strategy is to identify a threshold value STD above which the pick should be reviewed by an analyst. Fig. 25 plots the MAE as a function of the STD threshold. To generate this plot, observations above the threshold (x-axis) were removed, and the MAE was calculated for the remaining observations. The plot demonstrates that the aggregate model performance improves when samples with a high STD are removed.

The bottom plot of Fig. 25 shows the remaining test dataset size once the observations above the STD threshold value have been removed. Both plots show a clear knee in the curves between 0.1s and 0.2s. The orange vertical line is an example threshold at STD = 0.15s; approximately the middle of the knee. Using this threshold value, the dataset is 91.5% of its original size, and the MAE is 0.439s.

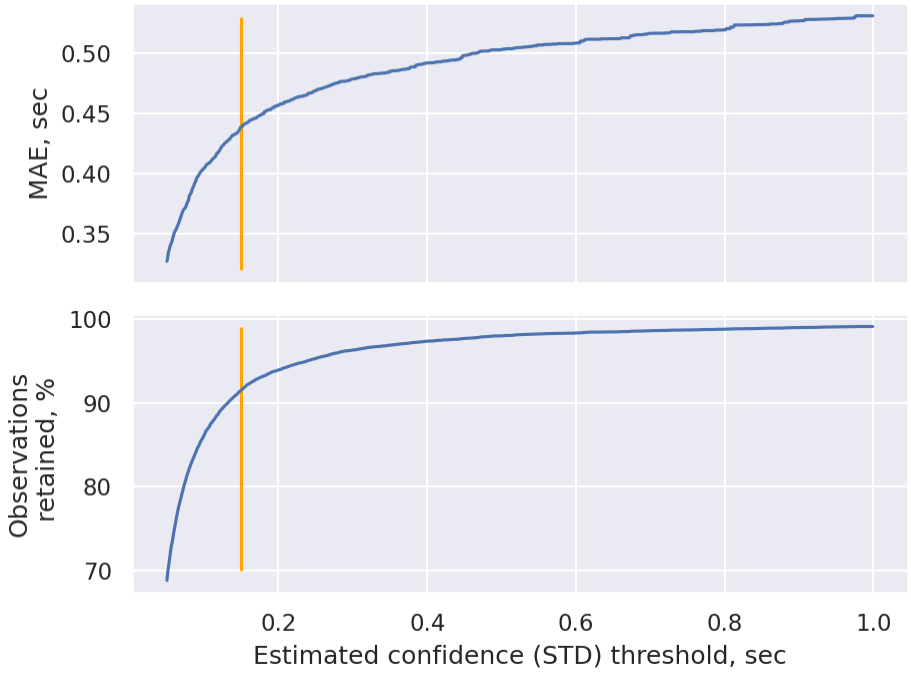

Fig 25: PhaseHunter performance MAE (above) when using the STD as a threshold. i.e., events above the threshold are removed. Remaining dataset size (below) when the threshold is applied. *The orange line is an example threshold at STD = 0.15s*

Finally, Fig. 26 shows where the 0.15s STD threshold lies within the estimated error scatter plot. Using this visual approach, one may tend to push the threshold lower, perhaps to ~0.1s. It should be noted that the model performance will be improved when trained on more data, and the estimated error will be reduced. Thus, this threshold trade-off analysis should be re-conducted whenever the model is retrained.

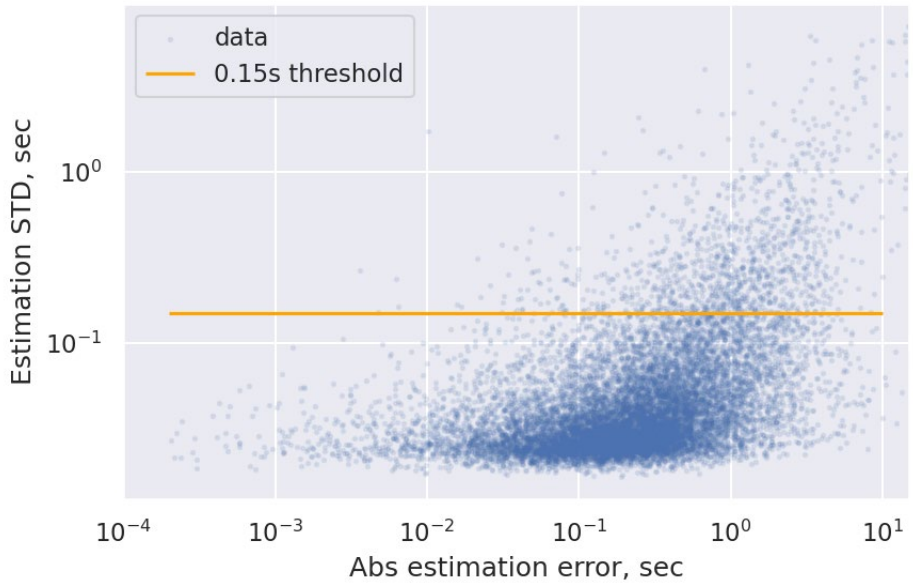

Fig. 26: The STD threshold visualized on the error-STD scatter plot

4.3 Software package

During this project, a software package was created for AFTAC to test the as-trained PhaseHunter model. Included in the software package were the following items:

Item	Description
README.md	Read me file describing the files included and required python packages
onset_model.py	The PhaseHunter DL model
onset_model_weights.py	Pretrained weights for the PhaseHunter DL model
onset_model_training.py	Jupyter notebook that shows how to (re)train PhaseHunter
web_service.py	Python web service providing RestFUL web API designed as a drop in replacement for AFTAC's current onset tool
web_functions.py	Python functions supporting web_service.py
curl_query.sh	Example shell commands showing how to form a query for the web service
onset_web_service_guide.docx	Further documentation on how to use the web service.

5.0 CONCLUSIONS

In conclusion, our deep learning model for predicting phase arrival times in seismic data appears to be performing well. The model achieved high accuracy and low mean absolute error, as well as a low root mean squared error. Additionally, we propose using the STD as a useful metric for estimating uncertainty in the model's predictions. For the supplied pretrained model, we propose using an STD of 0.15s as a confidence threshold. Sources of bias were found to be related to the SNR, depth, and energy of the measured signal. In general, when any of these factors are low, there tends to be a bias toward later predictions.

It is important to note that the model's performance may improve even further if it is trained on data specific to the user's location and station. Furthermore, based on the learning curves, additional training could result in even better performance. However, caution should be exercised when introducing the model to new station data, as it should be tested and evaluated before being used in practice.

Overall, the results are promising and suggest that deep learning models can be effective for phase arrival time prediction in seismic data, with potential applications in earthquake early warning systems and seismic hazard assessment.

REFERENCES

[1] Reiter, Delaine and Vanessa Napoli, “A Collaborative Research and Development Program to Advance the Use of Machine Intelligence in Nuclear Explosion Monitoring,” *Abstracts, Seismological Society of America Annual Mtg.,* 2022.

[2] I. D. 5.2.1, H. IDC Processing of Seismic, and I. Data, 1999.

[3] Ross, Zachary E., Men-Andrin Meier, Egill Hauksson, and Thomas H. Heaton, "Generalized seismic phase detection with deep learning," *Bulletin of the Seismological Society of America* 108, no. 5A, 2018, pp. 2894-2901.

[4] Zhu, Lijun, Zhigang Peng, James McClellan, Chenyu Li, Dongdong Yao, Zefeng Li, and Lihua Fang, "Deep learning for seismic phase detection and picking in the aftershock zone of 2008 Mw7. 9 Wenchuan Earthquake," *Physics of the Earth and Planetary Interiors* 293, 2019, p. 106261.

[5] Zhu, Weiqiang and Gregory C. Beroza, "PhaseNet: a deep-neural-network-based seismic arrival-time picking method," I 216, no. 1, 2019, pp. 261-273.

[6] Dickey, Joshua, Brett Borghetti, and William Junek, "Improving regional and teleseismic detection for single-trace waveforms using a deep temporal convolutional neural network trained with an array-beam catalog," *Sensors* 19, no. 3, 2019, p. 597.

[7] Mousavi, S. Mostafa, William L. Ellsworth, Weiqiang Zhu, Lindsay Y. Chuang, and Gregory C. Beroza, "Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking," *Nature communications* 11, no. 1, 2020, p. 3952.

[8] Poupinet, G., W. L. Ellsworth, and J. Frechet, "Monitoring velocity variations in the crust using earthquake doublets: An application to the Calaveras Fault, California," *Journal of Geophysical Research: Solid Earth* 89, no. B7, 1984, pp. 5719-5731.

[9] Schaff, David P. and Paul G. Richards. "Lg-wave cross correlation and double-difference location: Application to the 1999 Xiuyan, China, sequence," *Bulletin of the Seismological Society of America* 94, no. 3, 2004, pp. 867-879.

[10] Waldhauser, F. and Ellsworth, W. L., A double-difference earthquake location algorithm: Method and application to the northern Hayward fault, California, Bulletin of the seismological society of America, 90(6), 2000, pp. 1353-1368.

[11] Zhu, Weiqiang, S. Mostafa Mousavi, and Gregory C. Beroza, "Seismic signal augmentation to improve generalization of deep neural networks," In *Advances in geophysics*, vol. 61, Elsevier, 2020, pp. 151-177.

[12] Durasov, Nikita, Timur Bagautdinov, Pierre Baque, and Pascal Fua, "Masksembles for uncertainty estimation," In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13539-13548.

[13] Jindal, Ishan, Matthew Nokleby, and Xuewen Chen, "Learning deep networks from noisy labels with dropout regularization," In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, IEEE, 2016, pp. 967-972.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AIC	Akaike Information Criterion
AFTAC	Air Force Technical Applications Center
AR	Auto regression
CNN	Convolutional neural network
DL	Deep learning
GeLU	Gaussian error linear units
MAE	Mean absolute error
MSE	Mean square error
NLL	Negative log likelihood
ReLU	Rectified linear unit
SNR	Signal-to-noise ratio
TF	Tensorflow
TFP	Tensorflow Probability

DISTRIBUTION LIST

DTIC/OCP
8725 John J. Kingman Rd, Suite 0944
Ft Belvoir, VA 22060-6218 1 cy

AFRL/RVIL
Kirtland AFB, NM 87117-5776 1 cy

Official Record Copy
AFRL/RVB/Dr. Megan P. Flanagan 1 cy

This page is intentionally left blank.