Karlsruher Institut für Technologie

Advances in Machine Learning for Seismic Event Detection

Zur Erlangung des akademischen Grades eines
DOKTORS DER NATURWISSENSCHAFTEN (Dr. rer. nat.)

von der KIT-Fakultät für Physik des
Karlsruher Institutes für Technologie (KIT)
angenommene

DISSERTATION

von

M.Sc. Jack Woollam

Tag der mündlichen Prüfung
11.11.2022

Referent: Prof. Dr. Andreas Rietbrock
Korreferent: Prof. Dr. Frederik Tilmann
Korreferent: Prof. Dr. Joachim Ritter

Abstract

Advances in computational power and storage are facilitating a new era of modelling. As the amount of information captured within datasets has evolved, tools have emerged to better exploit the statistical properties of such datasets. Machine Learning (ML) methods are one such family of techniques, leading the revolution of 'data-driven' solutions which achieve state-of-the-art performance in solving tasks across both business and the sciences. ML is concerned with the automated discovery of the governing relationships within data distributions. The algorithms can be thought of as a suite of generalized algorithms for extracting information. Seismology is a field naturally suited to the application of ML, containing high-quality catalogs of seismic recordings - collected over decades - which are crucial inputs into many seismological studies.

With seismology only starting to widely integrate the latest state-of-the-art ML research over the last few years, the lack of uptake means that huge performance increases may be possible for traditional tasks. The detection of arriving seismicity is one such area. Having more complete seismic catalogs means imaging smaller magnitude events, 'closing the gap' between seismicity observed in nature and what can be simulated in laboratory environments. Better-resolution seismic catalogs are, therefore, crucial for enhancing any physical understanding of seismogenic rupture processes.

This thesis investigates the extent to which ML can improve the task of detecting seismic events. We first explore and propose new methods for seismic phase arrival identification, applying a Convolutional Neural Network (CNN) trained for supervised classification of labelled seismic arrivals. We also propose a novel algorithm for the subsequent stage of the event detection pipeline, the task of seismic phase association. Our proposed association algorithm is designed to operate efficiently in a scalable and robust manner. We adapt a parametric model fitting framework to extract a physical model of the seismic wavefield moveout to associate picks to events. We then turn to the question of how to best evaluate the state-of-the-art ML in seismology. Here, we can also leverage practices from ML-focused fields such as computer vision, and natural language processing. Access to both open-access benchmark datasets and models is crucial for accelerating the research process. This thesis introduces software specifically designed for this task - SeisBench. It aims to significantly reduce the amount of work required to conduct ML research in seismology, accelerating development and iteration. We finally explore how the performance of the latest ML models varies when moving from well-curated benchmark training datasets to practical pipelines in less well-explored seismic environments.

We hope our exploratory work and subsequent results, including productionized software, will facilitate further in-depth research in applying ML to one of the most fundamental tasks in seismology.

Acknowledgements

I would like to express my complete thanks to my supervisor Prof. Andreas Rietbrock, who throughout both my PhD studies, all the way through to my undergraduate years at the University of Liverpool in the UK, has provided invaluable mentorship, guidance, and an infectious attitude towards tackling difficult problems which I will take with me throughout the rest of my life. He has been the primary driver behind my personal development over the years from my initial minor contributions within the research group, through to pushing my own specialised and independent research agenda. I am grateful for all the patience, and support offered.

I would also like to thank all the co-authors and collaborators who I worked with throughout my academic studies, their feedback was invaluable. The works produced as part of this thesis are the consequence of a meritocratic welcoming environment which allowed me to foster and develop my skills.

Finally I would like to thank both my family, for their continuous support, along with my partner Shannon. Her unwavering positivity and constant encouragement has been invaluable through the inevitable ups and downs of the PhD.

Contents

1 Introduction

We live in an era of constant data creation and consumption. Over the last two decades, there has been an explosion in the amount of data generated and processed daily. This has resulted in the creation of datasets of unprecedented scales and resolution, fundamentally changing the way in which we interact with them. To learn from such datasets in an optimal manner, scalable, generic routines are required. The rise of *data-driven* techniques which model underlying data distributions, is, therefore, becoming ever-more prominent. Arising from the realm of statistics, *machine learning* (ML) encompasses a range of such techniques used to exploit the information contained in the latest datasets to perform inference. Due to rapid advances in data storage and lower-cost computation, these techniques are achieving exceptional results in solving tasks in technology, business and across the sciences. The widespread adoption of these methods is one of the major disruptive technologies of our time, which has been compared to the adoption of general computer usage in the 1980s and 1990s [14]. Machine learning encompass a suite of techniques which operate under the following principles: How can we create models which learn through experience?; and are there fundamental theoretical laws of learning systems which we can use to model phenomena and solve unknown questions [51].

In the physical sciences, the typical way to gain a deeper understanding of the underlying fundamental laws of nature is by using models based on intuition, using combinations of known and inferred physical relationships to solve problems. In contrast to this, ML models learn the underlying rules from the data. These methods can be thought of as generic routines for learning which can be applied across domains, and tasks.

1.1 Motivation

Seismology is well-poised to benefit from the application of ML routines. Access to larger datasets containing high-quality examples is crucial when training a ML algorithm, as the model learns directly from the data. When provided with sufficient training examples, the general properties of the task can then be determined, improving performance to new, unseen cases. In seismology, labelled examples of seismic waveforms have been collected since the inception of the field. As of 2019, the number of stored waveform data in the Incorporated Research Institutions for Seismology (IRIS) data archive reached approximately 500 TB in total (Figure 1.1), providing a myriad of potential training data, perfectly suited to ML applications.

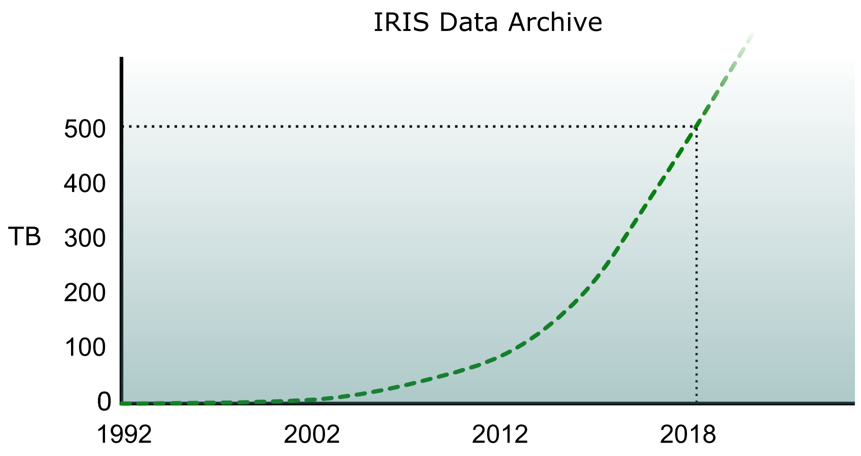


Figure 1.1: Size of total IRIS data archive through time. Modified from [60].

As seismic waveforms form the basic starting point of almost all seismological studies, having such a useful suite of training examples already accessible allows for investigations into the application of ML for solving a range of traditional seismic tasks. These tasks can range from the relatively simple e.g. source parameter estimation, signal denoising etc.; to the complex e.g. inverse problems such a seismic tomography. The most fundamental of such tasks is the initial detection of seismic events from timeseries recordings. The ability to detect more seismic events, at lower signal-to-noise ratios, affects all subsequent methods which operate on these data. Increasing the resolution datasets in seismology, therefore, has huge implications for improving our understanding of seismic processes, from the ability to image small scale faulting fabrics and associated rupture processes, to providing better constraints on wavefield-based inversion methods.

Constant technological advancements in the field also mean that faster, scaleable routines are sorely needed. The decreasing cost of seismic sensor deployment, along with decreased cost of compute and storage resources mean that larger node station networks are becoming more common. Traditional techniques will not be able to deal with such large data streams, which further highlights the need for a new generation of tools to efficiently and accurately detect seismic events in modern seismic recording networks.

This thesis investigates how *Machine Learning* (ML) techniques can be leveraged to improve the detection of seismic events. The focus is on detecting the regional scale, intense seismicity following large earthquakes, but the methods presented in this thesis form a general suite of detection routines which can be applied to new seismic environments to help improve detection rates. The following section introduces the task of seismic event detection, providing a general overview of how it is typically performed, problems that can potentially limit performance. We also comment on the potential advantages machine learning methods may hold over them, and what this implies for seismology in general.

1.2 The seismic event detection task

Seismic event detection is the first step of any typical seismological investigation, shown in Figure 1.2. As the seismic wavefield propagates from a given source, the ground motion is recorded at seismic sensors. Combining sensors in typical geometries, otherwise known as *seismic arrays*, or *seismic networks* enables detection of the moveout of the same event across different regions in space and time. The task of seismic event detection is to take these independent pieces of arrival identification, combining them together to identify the presence of some underlying source. Whilst this may seem conceptually simple, a number of factors greatly influence what is recorded in the continuous timeseries of a seismometer. For example, simultaneously occurring seismic and non-seismic sources of energy can also be detected across the seismic network. The moveouts will overlap with the true source. 'Untangling' the true independent arrival detections from the false detections then becomes a non-trivial process (see right-hand-side of Figure 1.2).

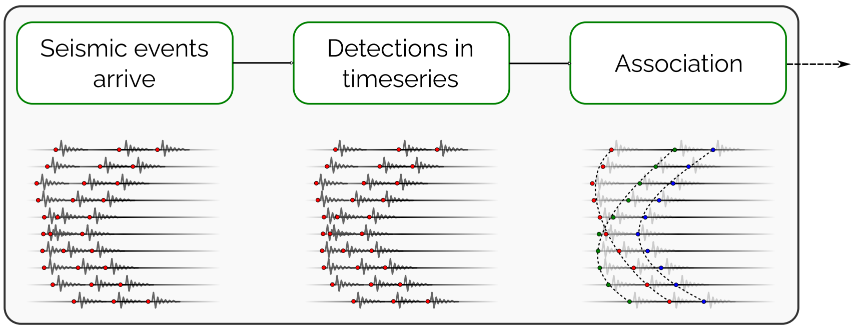


Figure 1.2: Schematic diagram showing the typical event detection workflow to identify seismic events from some continuous timeseries.

Further problems can include different noise thresholds at stations, the false assignment of non-seismic sources as seismic energy, all complicating the event detection task.

Since the 1960s, seismic arrays have been instrumental in studying earth structure and seismic source processes [114]. From the originally designed networks detect global and regional-scale seismicity, denser, higher-quality networks are now in place to monitor seismicity in real-time - present in the prominent seismic danger zones of the Western US e.g. Southern Californian Seismic Network [15], and Japan e.g. hi-Net [97]. Whilst smaller station spacing increases the resolution at which we sample the seismic wavefield, potentially resulting in more complete catalogs of recorded seismicity, this also exacerbates the factors previously described as the frequency magnitude scaling of the Gutenberg-Richter Law means that there are orders of magnitude more smaller events compared to larger events. This increases the likelihood of events simultaneously occurring in time. Traditional approaches for event detection are typically computationally expensive, so will not scale well in this new era of extensive data. Here, ML methods hold promise as the routines themselves are often 'embarrassingly parallel' in their implementation, and dependent on the accuracy of the methods, could easily be applied in real-time to analyse and detect seismicity.

1.3 Aim and scope of the thesis

With such factors in mind, this thesis investigates the extent to which ML methods can improve or outperform traditional approaches for seismic event detection. To reliably evaluate such a wide-ranging question, benchmarking investigations are vital, as they allow for the performance comparison of differing routines across some common task to see which perform best. Any newly proposed components of an event detection presented in this thesis are, therefore, benchmarked against currently used algorithms. Once benchmarked, individual components are combined together to detect seismicity in practice. The major focus is to investigate new routines for analysing the seismicity associated with major earthquakes, so any practical deployment focuses on detecting the seismicity of an aftershock sequence recorded across a dense, regional temporary seismic network.

1.4 Structure

The thesis is organised as follows; chapter 2 formulates the typical approaches to the event detection task, highlighting the motivation behind traditional approaches and both the benefits and limitations of each respective method for detecting seismic events. Here, commentary is included on how machine learning has initially been applied to improve performance.

Chapter 3 introduces a proof-of-concept case-study for improving the task of seismic phase classification with deep learning. The proposed neural network for seismic picking is composed of a convolutional architecture, trained in a fully supervised manner over a labelled training dataset of manual picks. All training pick examples were made during the aftershock sequence of the M_w 8.2 Iquique earthquake in northern Chile. The resulting picks of this initial investigation are benchmarked against a popular traditional seismic picking approach of the STA/LTA highlighting what advantages such techniques hold over traditional methods [147].

Chapter 4 focuses on the subsequent problem of seismic phase association, and whether machine learning can be leveraged to improve performance for this task. We introduce a novel adaption of a machine learning parametric model fitting algorithm, popular in the computer vision community, and apply it for correlating seismic phases to their underling source. The investigation here uses completely synthetically generated data to stress-test the performance of the algorithm on a dataset where the labels are known. This allows for quantitative evaluation of where proposed method breaks-down, and whether it offers performance improvements over traditional event association approaches [148].

Chapter 5 explores how benchmarking of machine learning algorithms can be performed in a standardised manner. The ability to benchmark competing algorithms is crucial for accelerating development. This is especially true for machine learning, where performance is highly dependant on the training data. We introduce the toolbox - SeisBench - which has been designed specifically for benchmarking of machine learning algorithms in seismology. The toolbox itself is designed for general comparison of models covering the entire variety of seismic tasks [150].

Referring back to the context of the event detection task, we show how the toolbox can be used to benchmark the current state-of-the-art seismic picking algorithms. All algorithms are applied across a range of benchmark datasets which cover the entire range of potential earthquake-related seismic environments. The results highlight how frameworks such as SeisBench can help development and assessment of state-of-the-art routines for seismic event detection [93].

Chapter 6 integrates the results of the previous works to set up multiple end-to-end seismic event detection pipelines for detecting regional seismicity in practice. For this final case-study, the methods are applied to analyse a subset of the M_w 6.4 2019 Durrës aftershock sequence of Albania. The results are again benchmarked against the manual analysis of 2 seismic experts.

We finish with a discussion on the viability of the proposed machine learning methods for event detection in chapter 7, concluding with what we have learned, and a general outlook for ML applied to the task of seismic event detection.

1.5 Contributions

This thesis is partially based upon the following publications, listed by order in which they are discussed in the chapters:

- **J. Woollam**, A. Rietbrock, A. Bueno, and S. De Angelis. "Convolutional neural network for seismic phase classification, performance demonstration over a local seismic network". in *Seismological Research Letters* 90:2A, 2019, pp. 491–502. DOI: `http://dx.doi.org/10.1785/0220180312`.

- **J. Woollam**, A. Rietbrock, J. Leitloff, and S. Hinz. "HEX: Hyperbolic event extractor, a seismic phase associator for highly active seismic regions". in *Seismological Society of America* 91:5, 2020, pp. 2769–2778. DOI: http://dx.doi.org/10.1785/0220200037.
- **J. Woollam**, J. Münchmeyer, F. Tilmann, A. Rietbrock, D. Lange, T. Bornstein, T. Diehl, C. Giunchi, F. Haslinger, D. Jozinović, A. Michelini, J. Saul, and H. Soto. "SeisBench—A Toolbox for Machine Learning in Seismology". in Seismological Research Letters, 2022. DOI: http://dx.doi.org/10.1785/0220210324.
- J. Münchmeyer, **J. Woollam**, A. Rietbrock, F. Tilmann, D. Lange, T. Bornstein, T. Diehl, C. Giunchi, F. Haslinger, D. Jozinović, A. Michelini, J. Saul, and H. Soto. "Which picker fits my data? A quantitative evaluation of deep learning based seismic pickers". in *Journal of Geophysical Research: Solid Earth*, 2022, http://dx.doi.org/10.1029/2021JB023499.

The author additionally contributed towards further manuscripts which lie outside the main argument and scope of this thesis - listed below:

- A. Bueno., L. Zuccarello., A. Díaz-Moreno., **J. Woollam.,**, M. Titos., C. Benítez., S. De Angelis., (2020). PICOSS: Python interface for the classification of seismic signals. in *Computers geosciences, 142*, 104531. DOI: https://doi.org/10.1016/j.cageo.2020.104531.
- S. León-Ríos., L. Bie., H. Agurto-Detzel., A. Rietbrock., A. Galve., A. Alvarado., **J. Woollam**., (2021). 3D local earthquake tomography of the Ecuadorian margin in the source area of the 2016 M_w 7.8 Pedernales earthquake. in *Journal of Geophysical Research: Solid Earth, 126*(3), e2020JB020701. DOI: https://doi.org/10.1029/2020JB020701.

2 Theory

2.1 Machine learning overview

Before introducing the specific problem of seismic event detection and describing the various ML techniques applied to this task, we first provide a general, theoretical overview of the background ML theory. Where this high-level introduction will provide the basis for understanding the techniques introduced in the following sections. As repeatedly highlighted throughout the introduction, the concept of ML is not new. In fact, many standard ML methods are an assimilation of traditional statistical techniques which aim to model underlying data distributions in order to perform inference. Such techniques have existed for decades. Figure 2.1 overview figure displays the base taxonomy for the different facets of ML. ML algorithms are a subset of the concept of *artificial intelligence*. *Artificial intelligence* techniques are concerned with building intelligent machines or algorithms to operate in an independent manner.

Within the taxonomy of Figure 2.1 there is a further division - *deep learning*. Deep learning involves the application of a single family of algorithms, neural networks. These algorithms are mathematical approximations of the bioligical decision making processes occurring in nature. They are trained in a highly parallelized manner and often involve optimizing many millions of free parameters.

> “ *Building intelligent machines or algorithms to operate in an independent manner* ”
>
> *Artificial Intelligence*

Following this general introduction of the differing ’families’ of algorithms and how they fit together, the next logical question is: How are these methods optimized for a a given task? The ways in which ML can be applied is summarised in Figure 2.2. There are 3 predominant approaches, supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning is where a given model is trained by providing it with a set of examples, and each example has a corresponding label which is known beforehand. This label is also input into the algorithm. Examples of this sort of problem includes classification and regression problems. A

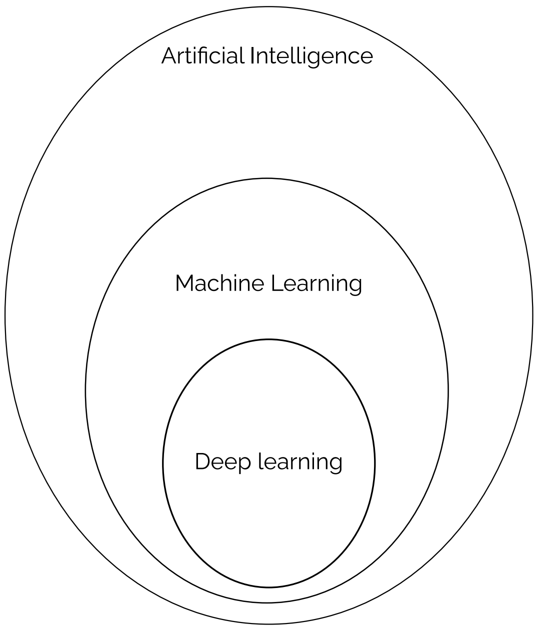


Figure 2.1: Schematic diagram showing the overall taxonomy of where machine learning fits into the field of artificial intelligence.

typical classification example is: Given a dataset of pictures of cats and dogs, with the an associated set of labels e.g. 'cat', and 'dog', train some ML model to predict the classification label when provided with an input image.

Unsupervised learning algorithms are where there are no labels known beforehand. Examples of this type of application can include clustering, and dimensionality reduction. For this type of task, the model learns directly from the underlying data. Going back to the typical example of a training dataset of images of cats and dogs. Applying an unsupervised learning pipeline would involve feeding in the input images, with no corresponding label information for each training example. The model would then have to cluster the inputs into a predefined number of distinct groups based on the differences of the input training examples only.

Finally, there is the task of *reinforcement learning*, which involves trained algorithms to optimise a decision making policy through a delayed feedback mechanism via a reward/cost function. These models are widely applied to learn general policies in the field of robotics, with typical examples including controller optimization, autonomous driving, and agents to outperform humans in controlled environments where all rules are known (e.g. autonomous agents for playing games such as Chess, Go, and other simple games). This final type of ML paradigm is one which to date, has not been widely applied to the task of seismic event detection, and also the more generally has been limited in its applications within seismology.

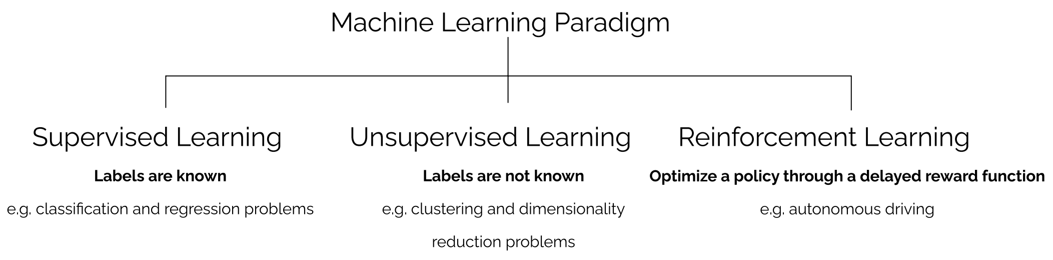

Machine Learning Paradigm
Supervised Learning
Labels are known
e.g. classification and regression problems
Unsupervised Learning
Labels are not known
e.g. clustering and dimensionality reduction problems
Reinforcement Learning
Optimize a policy through a delayed reward function
e.g. autonomous driving

Figure 2.2: Schematic diagram showing the different ways in which machine learning can be applied.

Following our general definition of the ML landscape, we can now provide a more mathematical grounding for the techniques relevant to this thesis. In this section, we will first explore a range of linear parametric methods for supervised classification and regression. Section 2.3 will then expand on this concept to include non-linear algorithms trained in a supervised manner. The techniques presented in the aforementioned sections provide the baseline understanding for the ML methods applied in this thesis. It also provides a basic grounding, more generally, when discussing how these techniques are being leveraged for the event detection task in seismology. Following the theoretical introduction, we turn to a review of the state-of-the-art methods for event detection present in the literature (section 2.4).

2.2 Supervised machine learning

We have commented on the general form of the supervised learning problem, which is; given some input features, predict a given output where the label information is both known and simultaneously input into the algorithm during training. This very high-level definition can be thought of as the ML interpretation of a concept which has existed for decades - if not centuries. The fundamental goal is to find the features of the inputs which directly affect the output. In general statistical terminology, the inputs are often termed the *predictors*, or the *independent variables*; the outputs are the *dependant variables* or *target*.

The type of output dictates type of the supervised learning approach. If the target output is a discrete set of categories, then it a classification task. For example: Given an input set of measurements of an individual, predict whether the person has diabetes. In this case there are 2 discrete groups, or categories $(does\ not\ have\ diabetes,\ \ has\ diabetes)$. Which is encoded as a *categorical variable* $(0,\ 1)$. The numeric representation of the categorical variable for K groups provides the label information for the classification task.

For the regression task, the output response is just the simpler case of outputting some quantitative value given a set of input features. As a simple example for this case: Given the rainfall for the last 10 days, predict the amount of rainfall tomorrow. In this case, the label information is the amount of rainfall for a given day.

2.2.1 Classification

We first consider the simple case of linear estimators for regression and classification. This will highlight the main underlying concepts behind some fundamental ML techniques, and also solidify understanding of typical ML nomenclature. One of the most fundamental approaches in statistics is the approach of least squares. Given an input vector $X^T = (X_1, X_2, ..., X_N)$, where X denotes a column vector and therefore, X^T is the transposed row representation; we wish to output a optimal suite of prediction values $\hat{Y}$ via some linear model

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{N} \boldsymbol{X}_j + \hat{\beta}_j. \tag{2.1}$$

Here, β_i represents the coefficients for each model parameter and β_0 is the intercept term. Eq. 2.1 can be equivalently written as an inner-vector product where the intercept term takes a value of 1.

$$\hat{Y} = \boldsymbol{X}^T \hat{\beta}. \tag{2.2}$$

Given this linear model, the aim is the then find the set of coefficients which output the optimal set of predictions. The least-squares approach does this by finding the coefficients $\hat{\beta}$ which minimize the sum of the squared residuals between the predictions and true values

$$\epsilon^T \epsilon = (y - \boldsymbol{X}\beta)^T (y - \boldsymbol{X}\beta). \tag{2.3}$$

Where the residual $\epsilon = (y - \boldsymbol{X}\beta)$. As 2.3 is quadratic, the solution can be found directly via by differentiating with respect to β, providing the *normal equations*

$$\boldsymbol{X}^T (y - \boldsymbol{X}\beta) = 0. \tag{2.4}$$

If $\boldsymbol{X}^T \boldsymbol{X}$ is invertable, then the unique solution is

$$\beta = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \quad or, \quad \beta = \boldsymbol{X}^{+} y, \tag{2.5}$$

where $\boldsymbol{X}^{+}$ is the Moore-Penrose pseudo-inverse of $\boldsymbol{X}$.

For the classification task, converting the output prediction $\hat{Y}$ to some discrete representation $\delta(X)$ can be achieved through some *discriminant rule* which maps continuous values to discrete classes. For the linear estimator, this classifier typically suboptimal when compared against other

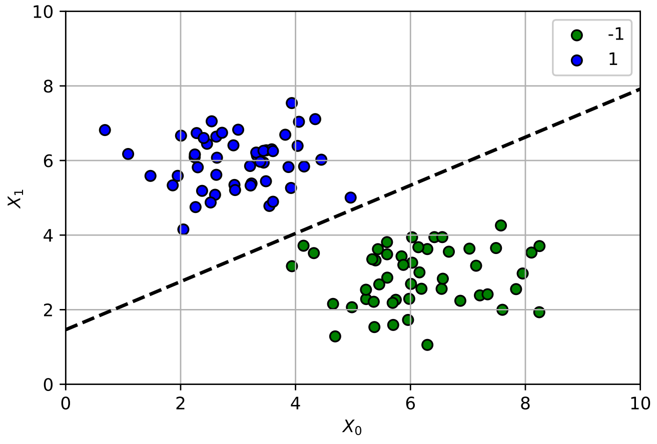


Figure 2.3: Linear decision boundary for classifying data generated from two independant Gaussian distributions with varying mean.

more advanced methods, but it helps to solidify conceptually how we can move between cases of classification and regression. A simple example would be

$$\delta(X) = \begin{cases} 1 & \boldsymbol{X}^T\hat{\beta} \geq 0 \\ -1 & \boldsymbol{X}^T\hat{\beta} < 0. \end{cases} \tag{2.6}$$

From here, the *decision boundary* to place any new input X_i into a given group is given by $\delta(X) = 0$. Eq. 2.6 denotes a binary classification task, predicting a total of two potential groups, but it should be easy to see how the discriminant rule can be extended further, introducing extra cases to incorporate multi-class classification problems. Figure 2.3 shows a schematic example to classify input data into 2 discrete groups using the linear least-squares approach, with the decision boundary of Eq. 2.6 also plotted as a dashed line. Any points lying on either side of the decision boundary are put into distinct groups.

2.2.2 Regression

Linear approximators

For the regression case, the aim is map some function $f(\boldsymbol{X}, \hat{\beta})$ to output a suite of predictions $\hat{Y}$, where the form of $\hat{Y}$ is now a column vector of continuous valued predictions. Figure 2.4 highlights how Eq. 2.5 can be used to fit a linear function to some continuous data.

For understanding, again it makes sense to first just considering the basic case of linear estimators only. Non-linear functional approximators are often extensions of the linear case, where

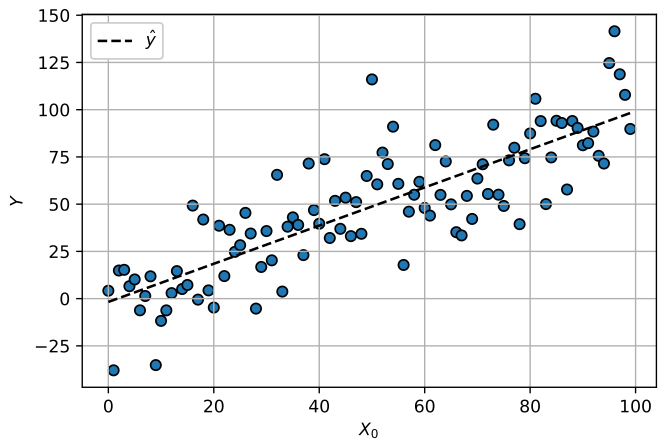


Figure 2.4: Linear regression using Moore-Penrose pseudo inverse of some linear model $y = mx+b$ which is contaminated with noise sampled from a Gaussian distribution with fixed variance.

direct solutions to the approximating function do not exist. For such models, methods are minimised via iterative optimization procedures, but the general methodology for solving often remains constant between both the linear and non-linear variants.

The Gauss-Markov Theorem states that the least-squares regression estimator (Eq. 2.5) provides the best linear unbiased estimator of the parameters β. 'Best' means that this estimator will have the smallest variance amongst all linear unbiased estimators. Provided, linearity in parameters, no multi-co-linearity between input features, errors having constant variance (homoscedascity), and independence of the error terms.

Non-linear approximators

We have formulated how supervised learning can be performed models which assume linearity of the input features. The methods are often simple and powerful, but the assumption of linearity in the features imposes a strong bias on the simplicity of the model and, therefore, limits types of relationships it can fit. In the physical world, non-linearities exist, and many processes are in fact best approximated using highly non-linear functions such as partial differential equations.

We will now introduce how linear regression can be applied to fit non-linear functions. Under these conditions, the convexity of the solution space cannot be guaranteed, so there is the potential for the first-order steepest descent method to get stuck in local minima. One way to improve convergence towards an optimal solution is by utilizing higher-order gradients. We will start with the basic approach of Newton's method. This provides faster convergence when compared against standard gradient descent, which utilises the first-order gradients only (see Figure 2.5).

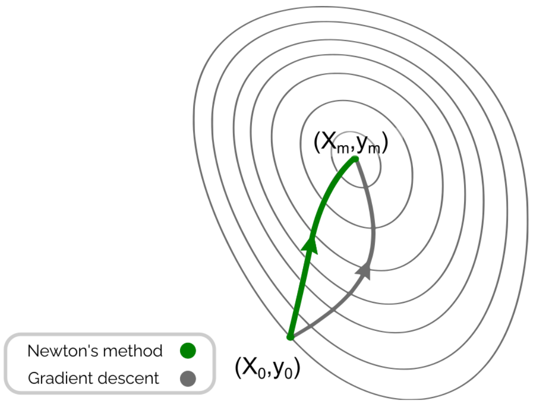


Figure 2.5: Schematic of convergence of iterative optimisation algorithms. The convergence path of gradient descent is in gray, Newton's method is in green. Newton's method uses second-order curvature properties to improve the convergence for convex functions.

Starting with the 1-D case, given some input variable x, providing an initial starting value $x_o \in R$, Newton's method solves the recurrence relation

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}. \tag{2.7}$$

This can be interpreted as finding the minima of a quadratic approximation of $f(x)$, as opposed to a linear approximation associated with steepest descent. There are a number of caveats with this approach, which we will see if we expand this out to to higher dimensions. Here, the gradient calculations generalise to the well-known Jacobian

$$\mathbf{J}_{i,j} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial u_1} & \cdots & \frac{\partial f_m}{\partial u_n}, \end{bmatrix} \tag{2.8}$$

and Hessian

$$\mathbf{H}_{i,j} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2}, \end{bmatrix} \tag{2.9}$$

matrices respectively. Equation 2.7 now generalises to

$$x_{n+1} = x_n - [\mathbf{H}f(x_n)]^{-1} \nabla f(x_n) \quad for \quad x \geq 0. \tag{2.10}$$

Calculating the second-order derivatives in Equation 2.10 makes the solution unstable when $\mathbf{H}$ is non-invertable, or if the $\mathbf{H}$ is not positive-definite. This ill-conditioning can lead to Newton's method not converging to some solution. In addition to this, the storage requirements of the algorithm are now $\mathcal{O}(n^2)$ due to the $(n\,x\,n)$ $\mathbf{H}$; requiring $\mathcal{O}(n^3)$ flops to solve the system of equations at each iteration. For most cases this is not practical. To overcome this, the Gauss-Newton iterative descent method takes advantages of the properties of the sum of squares error function to improve convergence.

Given some function $f(\boldsymbol{X}, \hat{\beta})$ the Gauss-Newton method of minimisation iteratively finds the minimization of the residuals

$$x^{n+1} = x^n - (\mathbf{J}_r^T \mathbf{J}_r)^{-1} \mathbf{J}_r^T r(x^n). \tag{2.11}$$

Where the Gauss-Newton method is solving a linearized version of the norm

$$\min \|\mathbf{r}(x^{(n)}) + \mathbf{J}_r(x_n)\Delta\|_2^2 \quad where \quad \Delta = x - x^{(n)} \tag{2.12}$$

With no second order gradient calculations needed, this is an improvement over Equation 2.10 in terms of convergence, but the Δ term means that Equation 2.11 can still result in divergence, as the approximation has made assumptions about the sum of squared residuals having small changes close to the minima, strong convexity, and moderate non-linearity. When this is violated, Gauss-Newton descent will potentially diverge from the global solution. Further variants of this optimization process attempt to overcome this caveat when tackling the problem of non-linear modelling. Such methods implement a damping parameter α to the gradient step.

$$x^{n+1} = x^n + \alpha\Delta. \tag{2.13}$$

The most popular variation of this approach to non-linear regression is the Levenberg-Marquardt algorithm [69], provided as the default method in many curve fitting packages (e.g. MATLAB, SciPy; Figure 2.6).

2.3 Supervised Deep Learning: Neural Networks

Whilst the non-linear optimization procedures introduced until now allow for the modelling of a large range of phenomena, a-priori knowledge is still required to define the form of the model parameters to fit. Now, we introduce a new type of ML algorithm which is agnostic to the type of function to be fit - *neural networks*. In recent years, the family of neural network algorithms have demonstrated widespread success across many fields. These methods are often interpreted as 'black-box' methods which automatically map some input to some output. They can, however,

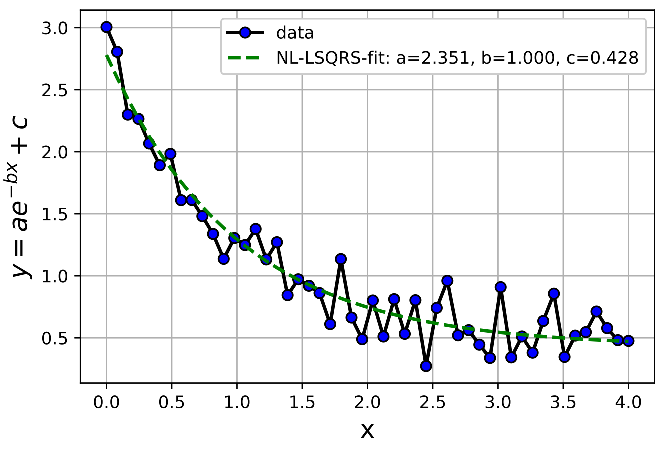


Figure 2.6: Linear regression of some non-linear function $y = ae^{-bx} + c$. The residual sum of squares is minimized using the Levenberg-Marquardt iterative optimization procedure, which comes as the default non-linear optimization for regression in the SciPy scientific computing package in Python.

be thought of as simple non-linear function approximators. The key idea behind neural networks is to find linear combinations of the input features, and apply a non-linear function, which are then optimized for globally.

2.3.1 The Perceptron

We will start with a basic or 'vanilla' neural network of standard form, the *feedforward neural network*. The fundamental concept underpinning neural networks is to use a mathematical approximation of decision making processes found throughout nature - *neurons*. In nature, biological neurons are responsible for making decisions throughout the natural world. Neurons are the mechanism in the brain which sends messages throughout the body via chemical signals and electrical impulses. Neurons allow for the body to send messages in response to external stimuli. The mathematical approximation of this phenomena is termed *The Perceptron* [110]. First presented in 1958, it's conceptual introduction predates many of the success of neural networks by decades. Simply, a perceptron takes, a suite of inputs $X^T = (x_1, x_2, ..., x_N)$, applies a corresponding weighting to each input $\theta = (\theta_1, \theta_2, ..., \theta_N)$, computes the weighted sum of the inputs, applies a non-linearity - commonly known as an *activation function* $\phi(\cdot)$, and adds a bias term b to the output $(\hat{Y})$. A schematic of the mathematical model of the perceptron is shown in Figure 2.7 [1]

[1]The term 'activation function' can also be thought of in terms of a 'basis function' and, in fact, wider generalisations exist for the simple perceptron.

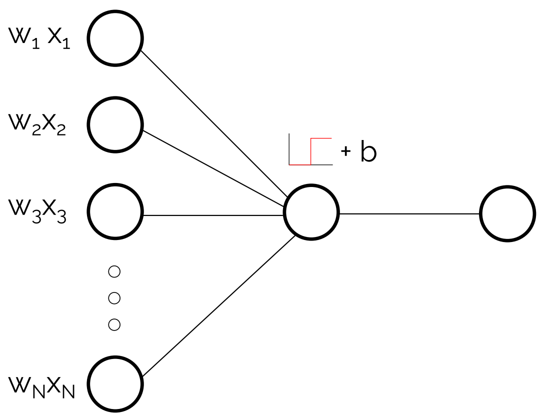


Figure 2.7: The mathematical model of the Perceptron which takes a set of inputs, applies a weighting to each one, computes the dot product of the weighted inputs, applies some activation function and adds a bias term.

$$\hat{Y} = \phi(\theta^T \cdot X) + b \tag{2.14}$$

Here, the output can encompass both regression and classification cases. For the regression case, the output is simply of length $N - 1$, for classification, the output would be of length K, where K is the number of different categories. The general idea is to find a functional approximation $\boldsymbol{f^\star}$ through a mapping, $y = f(X;\theta)$, learning the values of the parameters (θ) that result in the best approximating function.

2.3.2 Neural Network Design

By joining together layers of perceptrons in a *network*, different non-linear functional forms can be combined together, to increase the approximating power of $f(X;\theta)$. Connections between individual functional approximators, can be viewed as an acyclic graph composed of M nested functions $f^L(X) = (f^{(1)}(f^{(2)} \ldots (f^{(L)}(X))))$. Each $f^{(l)}$ is then termed a *layer* of the neural network, which takes a suite of inputs and some non-linearity to the weighted inputs, adding a bias term and 'feeding' the outputs to the next layer of the neural network. Typical terminology is to call $f^{(1)}$ the *input layer* to the neural network, $f^{(2)}$ to $f^{(L-1)}$ are then the *hidden layers* of the network, where L is the total number of layers in the network, and $f^{(L)}$ is the *output layer* of the network

$$\begin{aligned} \hat{Y} = f^{(L)}(X) =& (f^{(1)}(f^{(2)} \dots (f^{(L)}(X)))) \\ f^{(1)} =& \phi(\theta^T \cdot X) + b^{(1)} \\ f^{(2)} =& \phi(\theta^T \cdot f^{(1)}) + b^{(2)} \\ & \vdots \\ f^{(L)} =& \phi(\theta^T \cdot f^{(L-1)}) + b^{(L)}. \end{aligned} \tag{2.15}$$

2.3.3 How to train a neural network

As no closed-form solution will exist for such highly non-linear function, neural network parameters must also be optimized through iterative, gradient-based optimization procedures, evaluated using some *cost function* J. Gradient computation is done via recursive application of the chain rule, to derive the 'direction' in which to change each respective parameter of the neural network to minimise $J(\theta)$. This process is termed *backpropagation* in deep learning nomenclature. The fact that the backpropagation step can be done efficiently [67] on massively parallelized architectures has been a key factor behind the exponential increases in adoption of deep learning across business and the sciences. The neural network is some parameterized distribution $p(y|X;\theta)$, and so, the frequentist approach of *Maximum Likelihood Estimation* (MLE) can be to optimize the network parameters, where the cost function is the cross-entropy or negative log-likelihood of the true distribution and the networks predicted distribution

$$J(\theta) = -E_{x,y\sim\hat{p}_{data}} log\ p_{model}(\hat{Y}|X). \tag{2.16}$$

Here, Eq. 2.16 has been kept in the general form, as the parameterized distributions p_{model} and $\hat{p}_{data}$ vary with task/network dimensions.

For regression cases, the cost function is typically the sum-of-squared errors

$$J(\theta) = -\frac{1}{2}E_{x,y\sim\hat{p}_{data}}||y - f(x;\theta)||^2 + c. \tag{2.17}$$

Other variations of the cost function can exist for both cases of problem, but as an overview, these are two of the most widely used. As a consequence of optimizing using MLE, Eq. 2.16 is agnostic to the parameterization of the neural network, meaning that we do not have to define a new J for each new network architecture.

Building and training a neural network, therefore, consists of the following fundamental steps: define the task (e.g. supervised classification/regression, unsupervised latent space mapping), define the cost function, optimize or 'train' the algorithm, monitor the training/optimization stage until it reaches an 'acceptable' level. At the final point the network parameters are no longer updated, and you are left with an algorithm to perform your given task.

The backpropagation algorithm

Before explaining how neural networks are optimized let us first consider the case 'forward propagation', analogous to the forward problem of inverse theory. A set of input features X are input to the network, undergoing repeated transformations, to map to some output prediction $\hat{Y} = f(X;\theta)$. This is known as the *forward-pass*. This is also where the *feedforward* term in 'feedforward neural networks' comes about as information 'flows forward' through the network only. To update the parameters, the backpropagation reverses the order of operations, and goes computing the partial derivatives of $J(\hat{Y}, Y; \theta)$ with respect to the network parameters θ. This is known as the *backward pass*. Training can then be performed to minimise the cost function through gradient descent, iterating between forward and backward passes; each forward pass outputs a new set of predictions, the backward pass computes the gradient of the cost function with respect to the network parameters to minimise the $J(\hat{Y}, Y; \theta)$.

This is the overall concept; to solidify understanding, we go through each step in detail. To avoid introducing extra notational complexity, we introduce the following key parameters. The neural network parameterization θ consists of a set of weights $w_{ij}^{(l)}$, where $w_{ij}^{(l)}$ represents the weight for the connection from the j^{th} neuron in layer $(l-1)$ to the i^{th} neuron in the l^{th} layer; and also biases, $b_i^{(l)}$, denotes the bias of the i^{th} neuron in the l^{th} layer. Following this notation format, we can write $g_i^{(l)}$ as the activation function for the i^{th} neuron in the l^{th} layer. This representation is another way of writing the individual functional mappings $f^{(i)}$ of Eq. 2.15, and allows us to succinctly define the activation of any given layer $g^{(l)}$ in vectorized form as

$$g^{(l)} = \phi(w^{(l)} g^{(l-1)} + b^{(l)}). \tag{2.18}$$

Eq. 2.18 shows how any layer of the network can now be expressed as a function of all the weights and biases of previous layers directly mapping to the current layer. If we define $g^{(l)} = \phi(z^{(l)})$, where $z^{(l)}$ is the weighted input of all the input neurons to the l^{th} layer, the forward pass, where the neural network maps inputs to a set of predictions is computed as follows,

$$\hat{Y} = g^{(L)} \begin{cases} g^{(l)} = X, & \text{if } l = 0 \\ z^{(l)} = w^{(l)} g^{(l-1)} + b^{(l)}, & \text{for } l = 1, ..., L \\ g^{(l)} = \phi(z^{(l)}), & \text{for } l = 1, ..., L. \end{cases} \tag{2.19}$$

The error for the output layer can subsequently be written as,

$$\delta^{(L)} = \nabla_g J \odot \phi'(z^{(L)}). \tag{2.20}$$

Here, $\odot$ is the element-wise product. Eq. 2.20 shows the gradient of the cost function, as a function of the input parameters in the 'activation' layers z^L. We can go further, and express the error term for each layer as a function of the error in the next layer

$$\delta^{(l)} = ((w^{l+1})^T \delta^{l+1}) \odot \phi'(z^{(l)}), \tag{2.21}$$

This allows for errors to be computed for any layer in the network, determining the change in cost function with respect to all of the parameterized weights and biases. Using Eq. 2.20 and Eq. 2.21, we can now apply the backpropogation algorithm. Starting at the output layer, the gradients of the activations g^l are computed for each of the l layers, back to the input layer.

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = g_j^{l-j} \delta_i^{(l)} \; and \; \frac{\partial J}{\partial b_i^{(l)}} = \delta_i^{(l)}. \tag{2.22}$$

Applying Eq.2.22 from layer L to layer 1, allows for the determination of the direction in which to change each individual weight and bias in the network to minimise J.

The entire process has a nice compact, vectorized representation. In addition to being easy to represent mathematically, the nature of the vectorized operations make it highly efficient. In fact, modern software packages in the deep learning community provide versions of the backpropogation, through generalised computational graph frameworks. Known as *auto-differentiation*, it allows for gradients to be automatically computed between nodes of the graph, and is the approach used by many popular deep learning libraries such as Tensorflow [1], PyTorch [99], Theano [10], amongst others. This interesting approach naturally lends itself to the scientific community, as many scientific problems involve calculus. Many software packages for deep learning can, therefore, also be used for rapid prototyping and development of ideas for solving complex functions without the use of meshes, meaning there is no reliance on finite-difference schemas. The scientific community is just starting to see the power of such toolboxes for imposing known physical constraints on powerful non-linear functional fitting ability of neural networks. This has driven a wave of interesting work combining powerful nonlinear modelling capacity of neural networks, adding known physical constraints into the loss function [104, 53].

2.3.4 Choice of activation function

Until now, we have considered the activation function $\phi(\cdot)$ to be some arbitrary non-linear function. As we have shown the optimization process to involve the calculation of the activation gra-

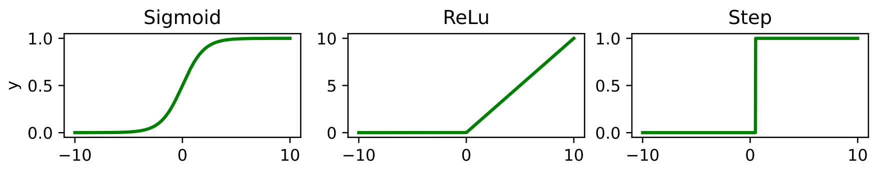


Figure 2.8: The different non-linear activation functions $\phi(\cdot)$ commonly applied in neural networks.

dient through recursive application of the chain rule, the choice of activation function is a key component of neural network architecture design. Initially, the step function was used as this was more inline with the biological understanding of neurons, where they 'fire' once the inputs pass a certain value. Naturally, repeatedly computing the gradient of the step-function is not well-suited for optimization, as small changes in the weighted inputs can produce a large change in the output during the backprogation step. As neural networks were further developed, they were more widely incorporated into modelling statistical processes, the step function was then replaced by other non-linear functions with better properties related to differentiation. Typically, the Rectified Linear Unit (ReLU) [94], or Sigmoid function are now used for the input and hidden layers. Figure 2.8 displays each of the aforementioned functions.

For the final output layer, the type of activation function applied depends upon the task. For the binary classification case ($K = 2$), the sigmoid function is used, in the case of multiple classes ($K \geq 3$), the Softmax function is used

$$f_k(\hat{Y}) = \frac{e^{Y_j}}{\sum_{j=1}^{K} e^{Y_j}}, \tag{2.23}$$

which normalises the output probability distribution amongst the K classes. For the regression case, the output activation layer g^L can use a linear activation, or a ReLU activation for positive only outputs.

2.3.5 Neural Network Variants

We have established how the conceptual underpinnings of the feedforward neural network introduced in Section 2.3.2 - the perceptron. We have also now introduced how a 'vanilla' feedforward neural network is constructed and optimised. The idea of using 'perceptons' as a mathematical model of decision making has however been around for decades, with it first discussed as early as the start of the 1960s. Throughout the 1970s to early 1980s, however, neural networks received limited interest. This was mainly due to the limited recognised work on implementing the crucial component of the backpropogation algorithm. Backpropogation had been implemented as a computational method for optimization from as early as 1970 [72], it was also shown to be a

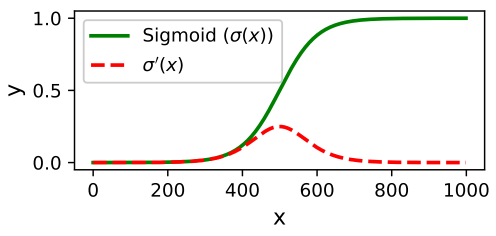


Figure 2.9: Example of how the exploding/vanishing gradient problem manifests during neural network training due to the gradient calculation of the activation function tending towards zero at extrema.

method to train feedforward neural networks from as early as 1974 [146]. The lack of interest came about as the academic community as a whole were pessimistic regarding the ability of neural networks to solve problems, leading to limited recognised publications. It was not until a concise presentation of the modelling power of neural networks, in 1986 [117] that interest in such approaches started being recognising within the scientific community.

Following this seminal publication, interest in the use of neural networks was firmly established, with the resulting interest obtaining some key findings. A key proof was that neural networks are universal function approximators [48]. This finding was crucial, as it stated that given an infinitely sized hidden layer, a feedforward neural network could be trained to approximate any function, regardless of its complexity. Practical case studies followed (e.g. handwritten digit recognition using neural networks) [66]; but, whilst theoretically feedforward neural networks could be trained to perform any given task or approximate any given function, this was never the case in practice. Limitations in approximating performance were fundamentally due to the available computational power, and the *exploding/vanishing gradient problem*.

The exploding or vanishing gradient problem is simply a consequence of the recursive application of the chain rule. Repeated gradient updates will push the update step to the extreme values of the activation function. Here, the gradient changes get asymptotically smaller towards 0, which causes the gradient to explode (tend to maxima), or vanish (tend to minima). An example of how this manifests is shown in Figure 2.9, with the gradient of the sigmoid activation function displayed as the dashed red line.

This is particularly a problem for fully-connected neural networks due to the nature of the connection between neurons, and makes the training of large networks problematic.

Convolutional Neural Networks

To overcome this problem, it was recognised that neural networks must extract features using sparser representations. One way in which to do this is via *Convolutional Neural Networks* (CNNs)

- initially termed weight sharing [66], the idea pass a convolutional filter over the input which has a common weight matrix. This feature vector will be automatically optimized to extract features for a given task. By chaining together layers of convolutional operations, more abstract features can be identified, again increasing the approximating power of the network. CNNs were the first deep learning architecture to be widely deployed in commercial applications.

Starting with the typical convolution operator, taking some real valued input $x(t)$ and apply a weighting function $w(a)$, where a is the delay term. We take a weighted average across each step of t, to give the response

$$\begin{aligned} s(t) &= \int x(a)w(t-a)da \\ s(t) &= (x \star w)(t). \end{aligned} \tag{2.24}$$

The $\star$ operator is typically used to represent a convolution. In deep learning nomenclature the weighting vector applied $w(\cdot)$ is a termed the *kernel*. In practice, the the convolution is computed over a discretized input $x(t)$.

$$s(t) = (x \star w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{2.25}$$

This operation can be generalizing to multi-dimensional inputs. If we define an input 2D input vector I, and pass though a 2D kernel K,

$$S(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-m); \tag{2.26}$$

and using the commutative property, the 2D convolutional operator is commonly implemented in ML toolboxes as

$$S(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{2.27}$$

The early successes of CNNs can be attributed to the following properties. Firstly, the concept of *weight sharing* greatly decreases the number of free parameters required to represent features when compared against a feedforward neural network. For examples, consider the case of inputting a 2D input consisting of a 28 x 28 pixel image for a classification task, for the feedforward case, each input pixel will be represented by a single neuron in the input layer which maps to the first hidden layer. If we were to then decide to represent the first input layer with 100 neurons, the total number of weighting parameters would have scaled to 78,400 (28 x 28 x 100; $\mathcal{O}(m\ x\ n)$). In contrast, the convolutional operation optimizes for the values inside the kernel only. For our image example, we using a CNN, we could pass over a 3x3 kernel to extract features, and then pass the output to a fully connected layer, for the final case, the total number of free parameters is then (9 x 100; $\mathcal{O}(k\ x\ n)$). This sparse features mapping force the network to compose extracted features starting with simple relationships. Often k $\ll$ n, as only few parameters are required to define these simple relationships, providing a huge decrease in the number of operations required for forward/backward-pass iteration. Another beneficial feature of CNNs when compared against the traditional feedforward architecture, is that the extracted features have equivariance. This is a property of the convolutional operator, it means that changing the input to the convolutional operation produces an equivalent change in the produced output. Such a property means that small shifts in the input do not greatly affect the produced features. The same features could be present at different points in the input image and the convolutional kernels are still able to recognize and extract such features irrespective of their position.

Pooling

With sparse-representations key to improving the performance of the traditional neural network, further steps were then made to expand upon this process. Following the application of some convolutional layer, the output can be replaced with a statistical summary metric. This operation is termed *Pooling*, and is commonly applied in tandem with the convolutions. Max-pooling [156], is one of the most common, and replaced the maximum value within a given N-D region. Variants of this can include a range of typical statistical averages within of the region. This step ensures that the derived features are invariant to translations, meaning that a small change in the inputs does not produce a large change in the outputs. It also further reduces the required number of operations by a pre-defined factor which is directly proportional to the size of the pooling region - to improve computational efficiency.

We have now covered the subset of ML algorithms which have been utilized in this thesis. These methods have been incorporated into novel routines for seismic event detection. Chapter 3 utilises a CNN architecture to perform supervised classification of seismic phases. Chapter 4 uses an iterative application of non-linear LSQRs regression, adapted within a model fitting logic to handle low signal-to-noise problems. Chapter 5 integrates a range of state-of-the-art of deep learning models in seismology for seismic phase classification; these are made available through a software toolbox for unified access. Chapter 6 integrates a subset of the different introduced ML components for seismic phase classification and event association and provides a benchmarking analysis of how they perform both against each other, and also against traditional manual techniques.

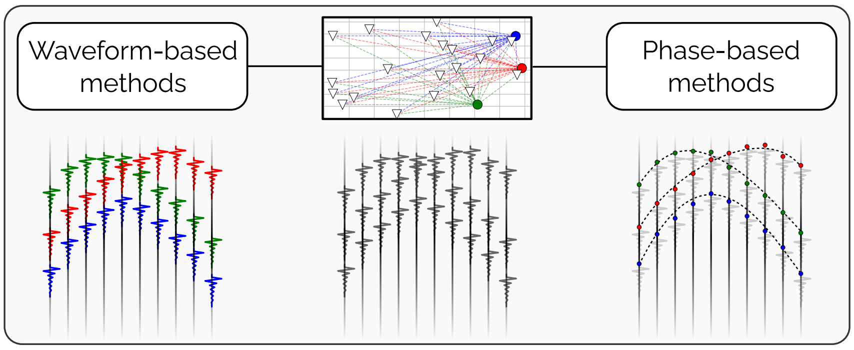


Figure 2.10: Schematic of the different methods to detect seismic events from seismometer timeseries recordings. The centre map inset shows the synthetic propagation of 3 events across the network, color-coded based on the underlying event. Waveform events take this information and identify the presence of a seismic event based on the similarity or coherence of the waveforms themselves; phase association-based workflows first identify the deterministic onsets of seismic phases, then correlate these seismic phases using either known physical laws related to event propagation, or unsupervised clustering-based routines assuming phase detections locally close to one another belong to the same event.

2.4 Methods of seismic event detection

With a theoretical grounding provided for the ML methods present in this thesis, we now turn back to the seismic event detection task, to explore it in more detail. First introducing how it has been performed historically and then exploring how ML is now being leveraged to improve performance for each individual aspect of the task. As the scale and density of seismic arrays have evolved over time, various detection methods have been proposed to identify events across seismic networks. These methods to perform the event detection task can be split into two main categories; detection methods which operate on the waveforms, *waveform-based methods*, and methods which operate on deterministic identifications of arriving seismic phase energy termed pick, or *phase association-based methods*. A schematic overview of what information is used to detect and correlate the presence of seismic events across a range of seismometer timeseries recordings is displayed in Figure 2.10.

For waveform-based methods, the waveform information is typically combined in two different ways for event detection. Waveform-based migration approaches use the coherency of the seismic wavefield as it propagates to detect events. Waveforms from different stations are shifted in time and stacked to identify the focus point of the seismic energy in time and space [52, 42, 29, 50] - corresponding to the seismic source. This is also known as the 'delay-and-sum' concept. The stacking step boosts the signal-to-noise ratio, allowing for the detection of smaller magnitude events. It is, however, a compute intensive undertaking, and whilst variants of this approach are now being proposed which can utilise the parallelized nature of deep learning [102, 79], the traditional approach will not scale well in this new era of large nodal seismic arrays.

The second major way in which waveform information can be used is through template matching. Waveform-based template matching methods use the similarity of waveform signature to cluster events belonging to the same source into groups dependant on the region of origin. Predetermined example waveform templates are cross-correlated to identify events [36, 126], but this biases the detections to be of a similar nature to the reference template, and again, it is computationally expensive. Potentially, templates could instead be selected automatically e.g. [13], but this scales as $\mathcal{O}(N^2)$ and so cannot be applied extensively. Variations of the traditional correlation function have more recently been proposed to overcome the compute limitations, which also aim to improve performance through the use of ML [102, 88, 79].

The second major approach to detect seismic events, *phase association-based methods* split the event detection stage up into two steps. Firstly, the impulsive arrivals of seismic energy (e.g. the arrival of a P- or S-wave) are detected in the continuous timeseries - often called the 'picking' stage. These 'picks' are then independent deterministic detections of seismic energy which are correlated across neighbouring stations to identify the underlying source - often termed the 'association' stage.

For decades the 'state-of-the-art' method for the picking stage was for a human expert to manually identify the seismic phase onsets. Naturally, this is a slow, and tedious process compared to automated approaches, but historically provided exceptionally accuracy. Traditional automated picking methods derive characteristic functions (CFs), or functional approximations of the probability of a seismic phase onset to infer pick onset times. These CFs are built using combinations of statistical properties such as Kurtosis [7, 119], frequency information [75], moving averages (STA/LTA) [4, 3] amongst others. From the wide variety of 'traditional' automated picking methods, the moving average STA/LTA approach proved the most popular. The pick rates for 'traditional' methods, based on manual feature extraction, massively under-represent the true seismicity when deployed during highly active seismic periods, even when applied over dense seismic networks [107]. This poses a problem, as the pickers determine the threshold of information to correlate. Fewer accurate picks mean that less information is available to identifying seismic events. More recently, a new suite of picking algorithms have been proposed which apply deep learning to automatically infer the characteristic properties of seismic phase arrivals [159, 86, 111]. This new type of technique exhibits accuracy similar to, or event greater than, that of a manual expert and have huge performance benefits over other methods. The performance benefits enable the exceptional standard of picking accuracy of a human expert to be applied across the entire sequence of continuously recorded seismic data. This has not been previously possible and has resulted in pick catalogs orders of magnitude larger than obtain from previous studies.

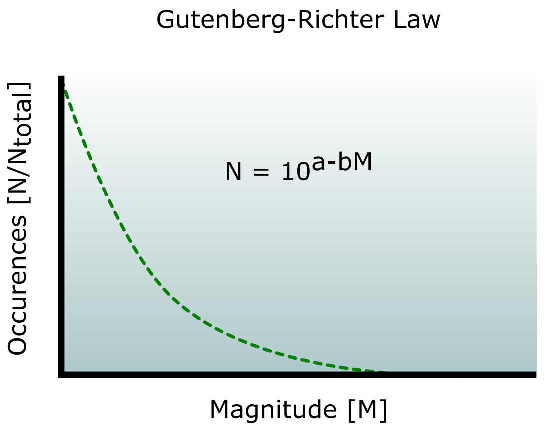


Figure 2.11: Schematic of the Gutenberg-Richter Law, displaying how smaller events occur orders of magnitude more frequently for each integer step along the magnitude scale (x-axis).

The secondary step of an phase-based detection pipeline the 'association' step aims to take the pick information and correlate this to an underlying source. The fact that events can overlap in time has already been highlighted (Figure 1.2), which greatly influences the association stage. With smaller magnitude events occurring an order of magnitude more frequently (Gutenberg Richter Law; figure 2.11), correlating smaller scale seismicity to its source its not a trivial task.

When it comes to detecting seismic events, decreased detection thresholds, such as what has been shown with the latest advancements in seismic picking, greatly complicate the detection process due to the power-law scaling of information to correlate. Nevertheless, the ability to identify events in such extensive datasets, has huge implications regarding the ability to better infer the physical processes of earthquakes, resolving faulting networks and imaging rupture processes.

To provide a direct of example, Figure 2.12 displays the results of running a cross-correlation method to detect seismic events throughout the entire S Californian region on all recorded seismicity over the period 2008 - 2017 [113]. This result was a step change in the number of events that can be feasibly identified with traditional methods, detecting over a 10-fold increase of total number of events compared to the previous catalog, with $\sim$1.81 million seismic events detected in total. The catalog was obtained using standard methods in seismology, but the authors had to utilize $\sim$ 200 Nvidia GPU cards, running on highly efficient super-computing architectures, which is an expensive undertaking. Can the data-driven techniques of ML can generate a similar solution without the associated cost in terms of compute power and time?

This is why the seismological community has a major recent focus on developing novel event association algorithms to efficiently detect events in the latest pick catalogs. ML has again been applied for this task in many forms, from graph theory [80], unsupervised clustering [160], recur-

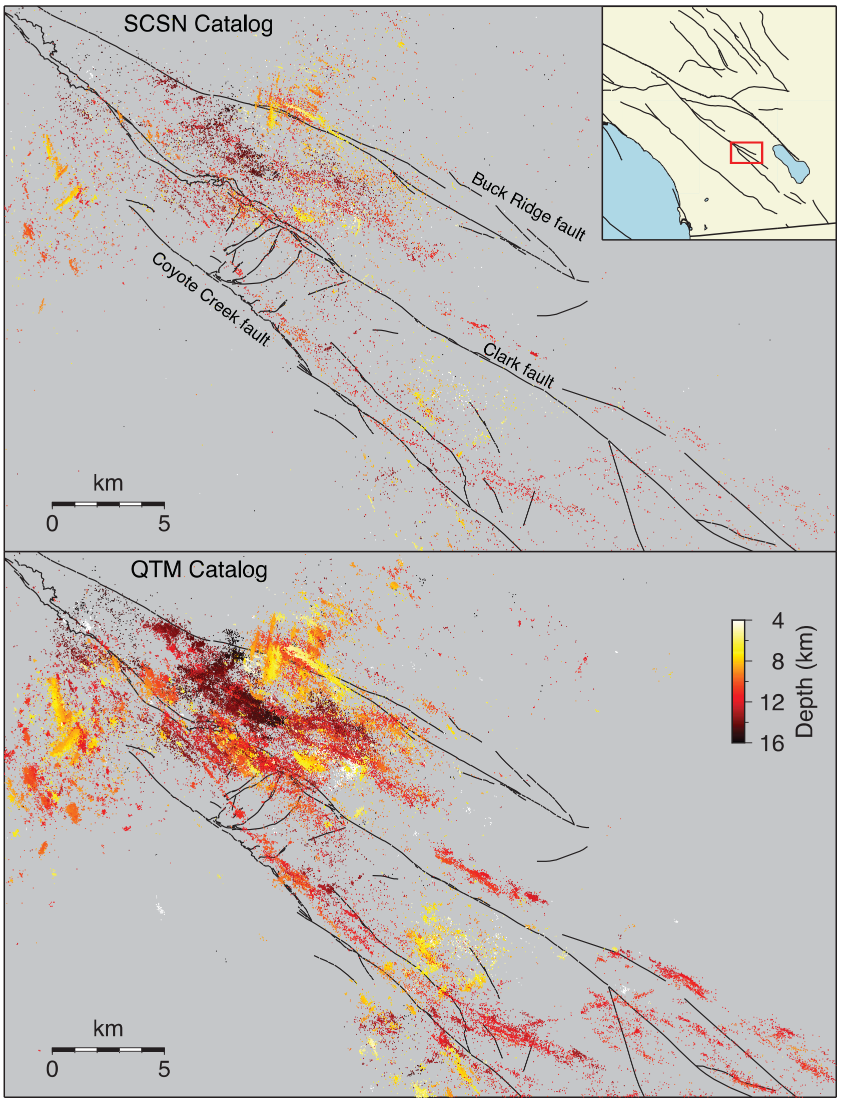


Figure 2.12: Detected seismicity of the 2019 seismic event detection study [113]. The comparison plot displays the improved resolution of faulting structures around the San Jacinto fault zone. This has been obtained by detecting smaller magnitude events, which much higher resolution in the new QTM catalog (bottom panel), compared against the previous Southern Californian Seismic Network Catalog (top map). The inset map shows the study location in S. California Original figure obtained from [113].

rent neural networks [112], amongst others. It also motivates the work of this thesis, where we now introduce ML-driven components to both identify and correlate seismic events.

3 Deep learning for seismic picking - application of a convolutional neural network

This chapter presents a machine learning approach for detecting seismic phases. We build a supervised classification pipeline, training a simple CNN architecture on a limited training dataset for deep learning purposes. We choose to evaluate a simpler model for two reasons. Firstly, our training dataset contains only $\sim$ 11,000 training examples, a simpler architecture, therefore, limits the number of free parameters of our model, reducing the potential for overfitting. Secondly, to date, the potential for smaller neural network architectures to be trained on a case-by-case basis in new regions has not been explored. There have been studies which explore the potential of deep learning for seismic phase picking trained on $\sim$ millions of training examples. In many environments, however, we do not have this scale of training data available a-priori. We frame the task as a supervised multi-class classification problem for classifying seismic timeseries recordings. For each point in the timeseries, our CNN architecture predicts the probability that the seismic trace is recording the presence of either seismic phases (P-phase or S-phase), or noise. As detecting seismic phases is the first part of any phase-based event detection pipeline, and will motivate the methods we introduce in the later sections. The chapter is based upon the following publication:

- **J. Woollam**, A. Rietbrock, A. Bueno, and S. De Angelis. "Convolutional neural network for seismic phase classification, performance demonstration over a local seismic network". in *Seismological Research Letters* 90:2A, 2019, pp. 491–502. DOI: `http://dx.doi.org/10.1785/0220180312`.

3.1 Introduction

Accurate detection of earthquake signals generated within the Earth is a fundamental and challenging task in seismology. Traditionally, the optimal method of identifying seismic phases involves a trained analyst manually inspecting seismograms and determining individual phase arrival times. Continuous developments in data acquisition and storage have resulted in vast, unprecedented increases in the volume of available seismic data. For such large-scale datasets, traditional manual picking methods are rendered unfeasible due to the required investment of time and resources; in addition, manual picking incorporates the subjectivity of different analysts which can bias pick accuracy. Further development of reliable automated picking methods are therefore essential to assist seismologists in their efforts to process large-scale datasets.

3.1.1 Historic Auto-pickers

The pressing need for a reliable automatic phase picker is not new, and numerous methods have been proposed to detect P- and S- wave onsets automatically. The most commonly used method for automatic phase picking is still the STA/LTA approach [4, 3, 30], which measures the ratio between the energy of the seismic signal over a short-term and a long-term window; any values of the STA/LTA ratio above a defined cut-off threshold represent a phase arrival. Baer and Kradolfer (1987) modified the STA/LTA incorporating an envelope function and a dynamic signal threshold into the characteristic function. There are numerous other approaches, including those based upon higher-order statistics [119, 120, 63], autoregressive methods [68, 128, 105], shallow neural networks [144, 21, 22, 155, 34], methods which utilise wave polarisation [7], and those which utilise pickers in tandem [95]. Whilst there has been extensive development of auto-picker routines, automated picking algorithms cannot currently match the accuracy of an experienced analyst. This is attributed to the complex nature of earthquake source and propagation, with multiple physical processes affecting the wavefield; variations in attenuation, noise-interference, source mechanism and energy-partitioning at interfaces all affect the observed waveform.

3.1.2 Why historic auto-picking routines are typically inferior compared to human analysts

Traditional automated picking methods are manually optimized for individual networks and/or even on a station-by-station basis, fine-tuning the 'characteristic functions' to distinguish body wave phases from noise. E.g., triggers can be based on the frequency content of a trace, kurtosis, or some other combination of manually extracted features. One common problem is that S-wave phases are more difficult to pick as their onset is often masked by the coda of P-waves and manually extracted features will often struggle to identify the S-wave in such instances [39].

3.1.3 Advancements in deep-learning

Rather than extracting individual features, deep-learning-based algorithms focus on learning representations of data, where multiple layers of processing provide varying levels of abstraction [65, 123]. Recent advancements in deep learning techniques have yielded a suite of procedures which demonstrate 'super-human' performance when applied to solve problems in fields ranging from computer vision [62], to speech-recognition [46]. Convolutional Neural Networks (CNNs), are a form of supervised machine learning that achieves exceptional results in classifying multi-dimensional inputs such as images, videos, and audio [62, 54, 64]. CNNs apply repeated convolutional and pooling operations to the input data, resulting in a set of learnable filters which automatically 'engineer' the appropriate features for classification. The appropriate features are extracted by fine-tuning of the network's internal parameters (or weights), via a computer-based optimisation process. The intrinsic properties of CNNs make them an ideal method for natural signal classification [65]. Natural signals often demonstrate local connections between samples, an example being the higher amplitudes observed immediately following an impulsive phase arrival. The major advantage of a CNN approach is how such features are then optimised. Shared weights throughout the network result in the systematic optimisation of decision boundaries to find the best weighted combination of local features to classify phase onsets. Another major factor behind the success of deep-learning methods is that the only required input is a large dataset of labelled examples for training. Within the seismological community, large datasets of labelled data are readily available in the form of manually picked earthquake catalogues for many regions. We are now starting to see the adoption of deep-learning-based methods to solve problems in seismological processing [102, 111, 159, 137]. Preliminary results indicate such methods can match or even surpass human levels of performance in seismic phase classification. So far, CNN approaches have been trained over extensive catalogues of ($\sim$ millions) labelled examples collected over decades [111, 159]. We now investigate the dependency of the input data on classification performance by applying a CNN to classify seismic phases, where the network is trained over a relatively small catalogue of events ($\sim$ 11,000 P- & S-phase pairs). Can a relatively simple CNN architecture display similar performance improvements in the absence of an extensive training dataset? If a feature engineering approach demonstrates generalisation capabilities when trained over a small local dataset with inherent biases, this will further validate the potential of deep-learning-based methods over traditional techniques for seismic phase classification.

3.1.4 Data

The dataset used in training the CNN is a manually picked catalogue of 411 events containing approximately 11,000 P- & S-phase pairs, located throughout the Iquique region of Northern Chile. The training catalogue has also been used to perform a minimum 1D velocity inversion, presented in tandem with the results. Events occurred between March-May 2014 and are recorded over a network of 65 broadband and short-period stations distributed throughout Northern Chile and Southern Peru; all stations use a sampling frequency of 100Hz (Figure 3.1).

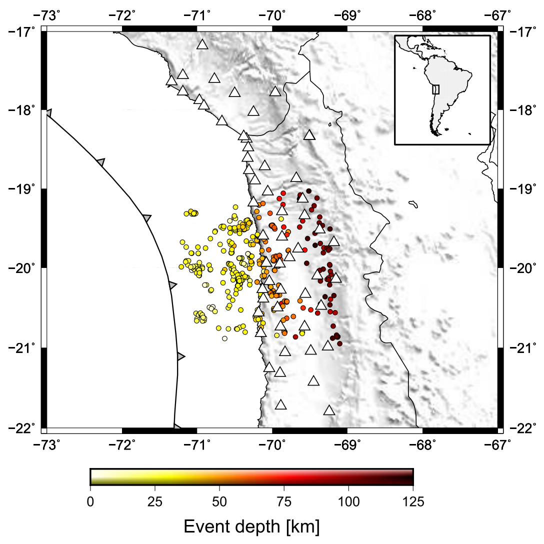


Figure 3.1: Distribution of manually picked events throughout Northern Chile, stations are indicated by white triangles, event hypocentres are plotted as a function of depth.

Manual picking of events was performed using Seismic Data eXplorer (SDX) software: `http://doree.esc.liv.ac.uk:8080/sdx/`. We process the dataset by applying a linear detrend. Whilst the CNN approach is shown to learn the characteristics of P-phases, S-phases, and noise [159], due to our limited training dataset, the CNN network will only be presented with a small portion of noise examples. To limit the potential for the CNN to erroneously identify noise it has not been trained on as phases, and to homogenize the data set due to different instrumentation; we bandpass filter the data between 2 - 25 Hz, a frequency range which lies in the passband of all instruments deployed.

Manual picks are represented probabilistically as a Gaussian function ($\sigma = 1s$, Figure 3.2),

reducing the bias associated with erroneous picks. The σ parameter was determined through manual parameter testing. Larger σ values resulted in the network acting more as an event 'detector' where the output probabilities were not impulsive enough to obtain a definitive phase-onset. Values lower than 1 second resulted in a high proportion of 'miss-picks' as manual pick errors not captured by the classification vector had a detrimental effect on engineering the appropriate features for phase classification. The dataset is split into training, validation and test batches (with ratios of 80:10:10 respectively).

3.1.5 Dataset augmentation and training

Deep learning-based classifiers contain a significant number of trainable parameters in the solution space, therefore, an extremely large number of examples are needed to prevent overfitting of

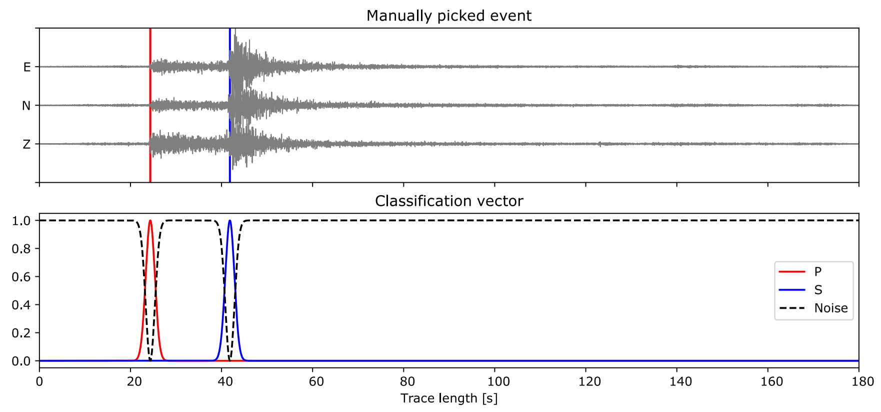


Figure 3.2: An example of input data (top) and classification data (bottom), inputs to the CNN are 3-component traces, linear-detrended, bandpass filtered between 2 - 25 Hz. The associated classification vector for P-pick and S-pick are represented probabilistically as a Gaussian with $\sigma = 1s$.

the training dataset and to enhance generalisation. Our dataset is relatively small for deep learning purposes. To overcome the limitations associated with a small training dataset we perform several additional processing steps. Events are scaled by multiplication of a value drawn from a lognormal distribution, the ends of the segmented event are tapered to limit impulsive amplitude spikes generated by processing, varying levels of Gaussian noise are then added to each batch, resulting in greater variations of signal vs. background noise. The training events are therefore modified to show a range of arrival types, rather than the high-magnitude, well-recorded events that are typically seen in a small catalogue of manually selected earthquakes for further studies. The input window size for the CNN is 6 seconds. To train the CNN, a given input batch is sequentially windowed with a timestep of 0.4 seconds. The windows are randomly shuffled before being used in training, preventing the CNN from learning any unnecessary temporal order. A small time step is used to increase the total number of events during training; also, having the network learn to recognise the presence of phases at any point in the input window will help the network generalise beyond the training dataset. Formatting the input data in such a way reduces the biases associated with our small dataset and enhances the capability of the network to pick varying types of arrival.

3.2 Methodology

3.2.1 Network architecture

The input to the network compromises three one-dimensional windows (x) where each window samples an individual component. For this given input, the network outputs the probability of either P-phase, S-phase, or Noise for each time sample within that window (Figure 3.3).

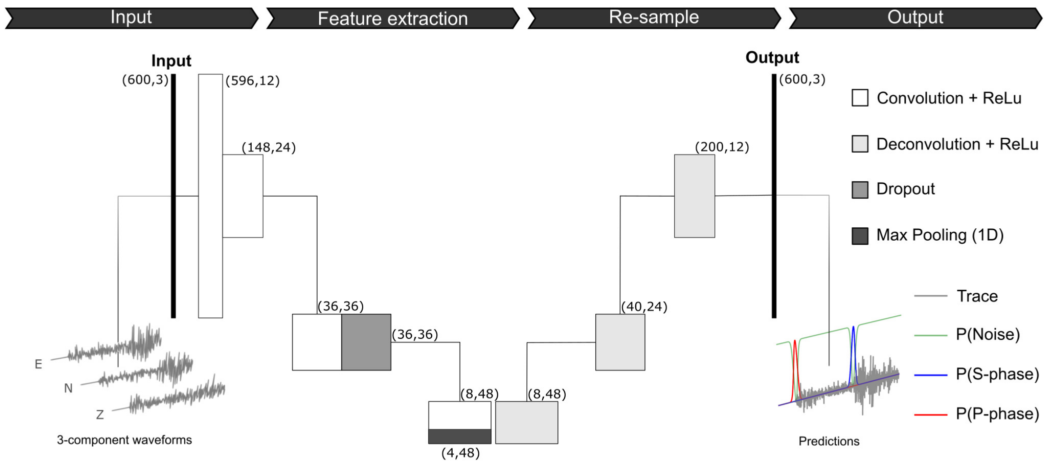


Figure 3.3: Schematic of the overall CNN architecture, displaying the sequential convolution and re-sampling operations applied to the input window.

Probabilities are output by applying the 'softmax' or normalised exponential function to the final layer

$$p(Y = i|x) = \frac{e^{\alpha_i(x)}}{\sum_{j=1}^{3} e^{\alpha_j(x)}}. \tag{3.1}$$

Where j = 1,2,3 represents the P-phase, S-phase and Noise classes, x contains the associated weights for the final layer. The input data are passed through repeated transformations; convolutional operations initially extract the appropriate features to characterise each class, the extracted features then go through repeated re-sampling stages, to output per-class probabilities. At each stage, a Rectified Linear Unit (ReLU) activation function is applied [94]. The cost function used to train the CNN is given by the negative log-likelihood $NLL(x, \theta)$ a multi-class classification problem, where each class is characterised as a series of discrete probability distributions, $NLL(x, \theta)$ is also termed the cross-entropy loss function,

$$NLL(x, \theta) = -\sum_{k=1}^{3} \sum_{n=0}^{n-N} log(p(c_k|x_n, \theta). \tag{3.2}$$

N represents the total number of training instances, c_k corresponds to the class label assigned to the input (x_n) and the network weights θ. Eq. 3.2 is minimised using Adaptive Moment Estimation (ADAM, [56]) along with batch training. The network weights are therefore updated at the end of each batch, over n training instances. Hyperparameter optimisation is the derivation of the optimal network parameters and is a major challenge when designing neural network architectures [9]. Parameters such as number of layers, regularisation of layers, convolutional kernel shape,

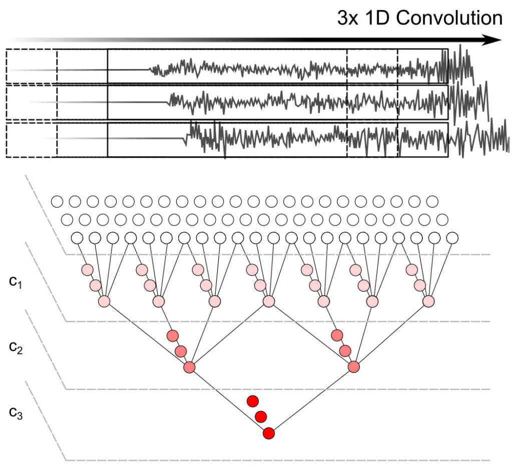


Figure 3.4: Schematic displaying how strided 1D convolutions quickly incorporate the long-term temporal dependencies of the input data into the convolution kernel.

and the learning rate can all be optimised. Methods to solve this problem consist of, grid search, random search, and manual estimation. As our study aims to demonstrate that a robust CNN can be trained on small datasets, the focus is on efficient implementation over more time-consuming systematic search methods. Once a robust network architecture is derived, a constrained search is performed for the best combination of hyper-parameters

Our final network architecture consists of 3 convolutional layers, followed by 3 layers of up-sampling (Figure 3.3). Again, due to the limited nature of the training dataset, the focus for the network architecture is to limit the potential for overfitting. To localise the features corresponding to different classes, convolutional layers apply strided 1D convolutional filters along each component (Figure 3.4.).

The stride for the convolutional window is set to 4, this down-samples the time series by a factor of 4 for each layer, reducing the overall number of free parameters and allowing for quicker incorporation of long-term temporal dependencies into the convolution kernel. A dropout parameter is added to the second convolutional layer. Dropout is a regularisation technique which randomly drops weights during training, reducing model complexity [131]. One-dimensional max-pooling is applied to the final convolutional layer, further reducing the overall number of network weights.

3.2.2 Picking Phases

To obtain P- and S-phase onsets from the CNN output probabilities, we use knowledge of the simple temporal relationships between P- and S-phases to determine onset times (Figure 3.5). For the P-phase probability distribution $\boldsymbol{p} = [p_1, p_1, \ldots, p_n]$ and the S-phase probability distribution $\boldsymbol{s} = [s_1, s_1, \ldots, s_n]$ if P-phase probabilities exceed a defined cut-off threshold p_{cut} the P-

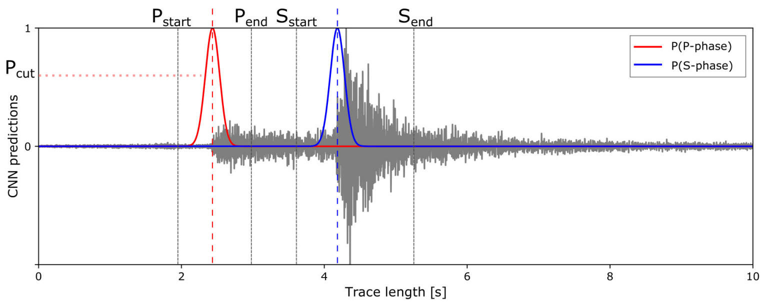


Figure 3.5: Displaying how the temporal relationship between P- and S-phases is used to identify phase onsets from the output CNN probabilities. Solid lines correspond to the output P-/S-phase probabilities; vertical dashed lines indicate phase onsets and the phase type is labelled above each vertical dashed line. Vertical dotted lines indicate the start or end of a P-/S-phase search window, where the corresponding labels are again presented at the top of each line. The horizontal dotted line represents the p_{cut} parameter used in determining phase onsets.

phase onset is searched for within the window $[p_{start}, p_{end}]$. The onset is set at the index of the maximum P-phase probability within this range. If the P-phase criterion is met, the corresponding S-phase is searched for within the searched window $[s_{start}, s_{end}]$, if $\sum_{s_{start}}^{s_{end}} s_i > s_{cut}$ then the S-phase is set at the index of the maximum S-phase probability within search window. Both conditions must be satisfied for an event to be picked, consequently, the ratio of P:S phase picks using these criteria is 1:1. The parameters used in detecting phase onsets are provided in Table 3.1, note that all index values are relative to the initial p_{cut} index.

3.3 Results

3.3.1 Predictions

The trained network takes a 6 second input window for 3-component data and makes phase predictions for each time sample within the window. Figure 3.6 displays a sample of the output phase probabilities for events in the test dataset. The predictions display a clear distinction between P-phases and S-phases, further confirming that deep-learning-based classification methods

p_{cut}	0.75	s_{cut}	5
p_{start}	-200	s_{start}	500
p_{end}	200	s_{end}	4000

Table 3.1: Parameters applied to the autopicker function, which takes advantage of the temporal relationship between phases to identify phase onsets, all start/end indexes are given in samples (where sampling rate for all instruments = 100Hz).

engineer the appropriate features to accurately categorise P, S, and noise classes. This presents a major advantage over historic auto-picking methods which utilise manual feature extraction and often struggled to identify the S-phase.

To obtain P/S phase onsets, we apply our autopicker function, with input parameters of Table 3.1, taking advantage of the simple temporal relationship between P- and S-phases to assign phase onsets (vertical lines on Figure 3.6). The phase onsets are then compared against the original manual picks and the residuals are plotted (Figure 3.7).

The residual distribution for the test dataset displays a good agreement in the centre of both the P- and S- residual distribution; however, the CNN has also picked extra events/phases in some waveforms. These extra phase picks may be accurate; however, any additional events are not represented in our classification vectors as a detailed association of individual phases to specific events arriving simultaneously is beyond the scope of this work. This negatively affects the residual distribution and is responsible for several of the large outliers observed.

3.3.2 Relocation testing

To overcome the issue of extra picked arrival times from simultaneously occurring events biasing our residual comparison, we perform an additional test to remove arrival times from any events overlapping in time. This additional test provides a more consistent assessment of auto-picker performance. We perform an iterative inversion procedure, relocating both the original manual picks and the CNN picks for the initial dataset. The catalogues are relocated using the VELEST routine [59], which applies a minimum-1D velocity model along with station corrections to solve for hypocentre locations. Hypocentral parameters are solved for all events within the catalogue. When using VELEST, all phase picks within a segmented trace are assigned to a single event during relocation. The large outlier residuals a significant distance (+3s) from the trend are attributed to multiple picked events in the same segmented trace being erroneously classified as a single event in VELEST. We, therefore, reject events with RMS residual larger than 3s to remove any picked events overlapping in time. Statistics of the residual distribution for the original manual picks compared against the CNN picks is provided in Table 3.2. The residual distribution indicates that manually picked P-phases are slightly more accurate than CNN P-phase picks (σ decreased by 0.051s); however, S-phase picks of the CNN approach achieve similar performance to manual picking (σ decreased by 0.019s). We recognize that our training and test data set used for the earthquake location data set are not independent; however, the residual distribution obtained from the CNN methodology is similar to that of the manual picks of an expert seismologist.

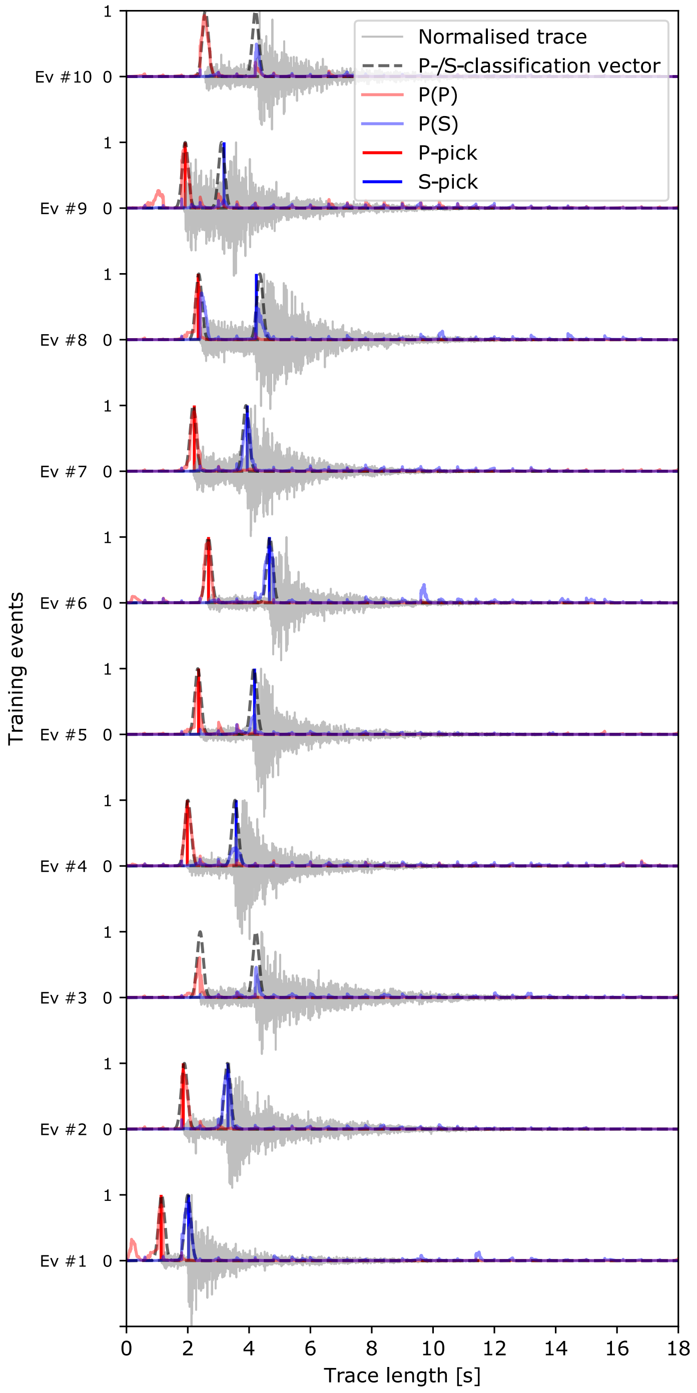


Figure 3.6: Output CNN prediction probabilities when applied to identify phase onsets for the test dataset, phase onsets are indicated by vertical lines.

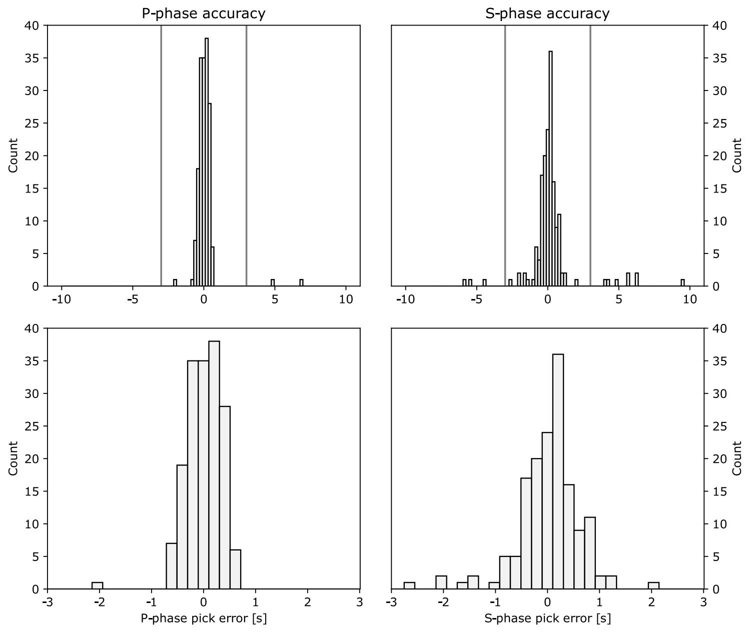


Figure 3.7: Residual of CNN predicted phase onsets vs. original manual picks for the test dataset.

GAP < 220°	P		S	
	μ_p	σ_p	μ_s	σ_p
CNN	-0.261	0.445	0.282	0.749
Manual	-0.124	0.394	0.390	0.730

Table 3.2: Statistics of the pick residual distribution of the original manual phase picks and the CNN phase picks. Statistics computed assuming a normal distribution where μ_p, μ_s denote the average P-phase and S-phase residual, and μ_p, μ_p denote the P- and S-phase residual standard deviation respectively.

	STA/LTA	CNN
P	72,655	77,623
S	63,353	77,623
Total	136,008	155,246

Table 3.3: Overall number of automatic picks on a separate catalogue of new events throughout Northern Chile.

3.3.3 Autopicker comparison

To further test the CNN picker, we apply the CNN methodology in predicting phase-onsets for a separate catalogue of events throughout Northern Chile, on the same temporary seismic network. Events were initially segmented using an iterative approach based on a STA/LTA trigger [107] and provides a useful test case for the CNN method. The relocation procedure is again applied to compare performance. The initial number of phase picks for both methods is provided in Table 3.3. Figure 3.8 displays an event from the new catalogue picked using the CNN method, multiple event arrivals are again present in the traces. To limit the effect of this issue on our residual comparison, we set both the STA/LTA and CNN method to only pick a single P-/S-phase pair per trace and again use the iterative relocation procedure to assess residual. The relocated hypocentre distribution for both methods are displayed in Figure 3.9 and Figure 3.10.

It can be clearly observed that locations are more clustered in the CNN approach and are better concentrated along the plate interface, indicating the greater consistency in phase picks for the CNN approach. Phase residuals for the relocated events are displayed in Figure 3.11; we show residuals for both the final catalogues (minimum azimuthal gap $< 220°$) and for only the best-located events (minimum azimuthal gap $< 160°$). Statistics for the residual distributions are displayed in Table 3.4.

Assuming a normal distribution, the CNN method exhibits decreased variance in phase residual for both P- and S- phases when compared to the optimised STA/LTA approach. The relocation residuals (Figure 3.11) are not just dependent upon the accuracy of detected phases, but also on velocity variations not captured in the 1D model or station corrections affecting the residuals. As both catalogues were relocated with the same iterative re-location procedure using the same 1D velocity model and station delay terms, discrepancies in residual distributions should directly reflect the relative consistency of picks in each catalogue. Investigating the residual distribution, the CNN approach has markedly improved both the overall relocation residual (Figure 3.9 vs. Figure 3.10), and the variation in residual for both P- and S-phases. In addition to this, the difference in σ for the well-located events is shown to be more accurate for the CNN approach, with σ improving by 0.230s for P-phases and 0.326s for S-phases when compared against the optimised

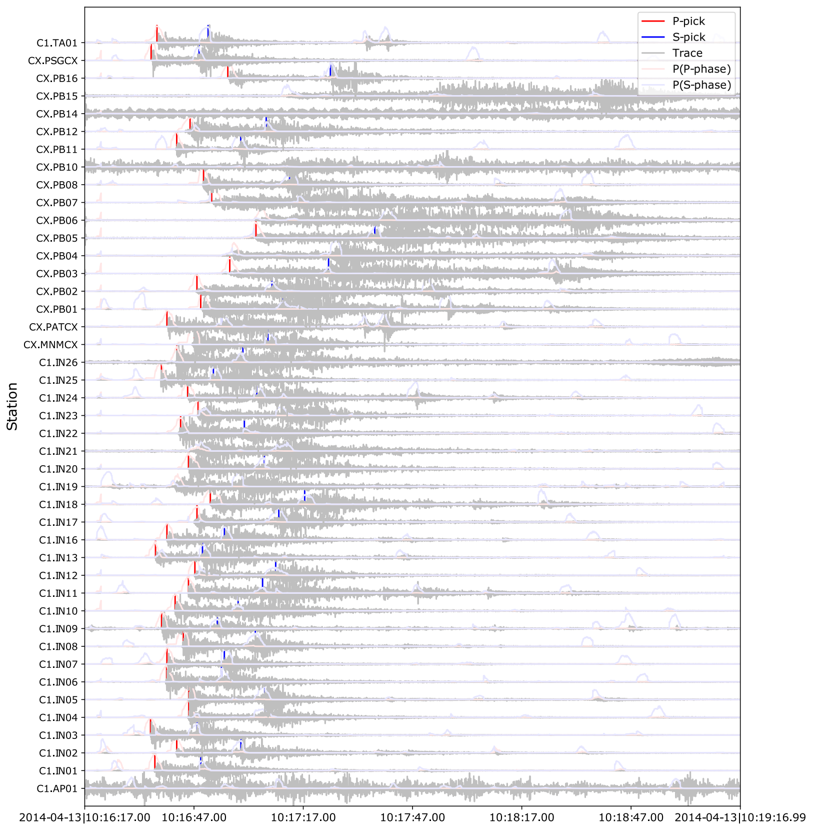


Figure 3.8: Demonstrating the CNN auto picker performance on a new dataset for Northern Chile, where events were segmented using an STA/LTA trigger [107]. We only allow the auto picker to pick the presence of a single P-/S-phase per trace, to prevent relocation errors.

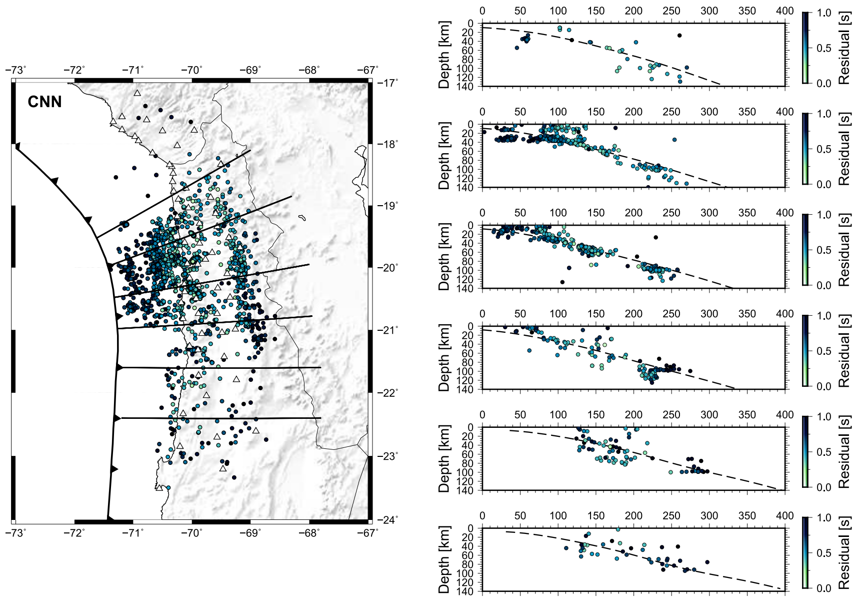


Figure 3.9: Hypocentre relocation comparison for the CNN auto-picked catalogue. Event relocations are plotted as a function of RMS residual, slab profile is provided by [44].

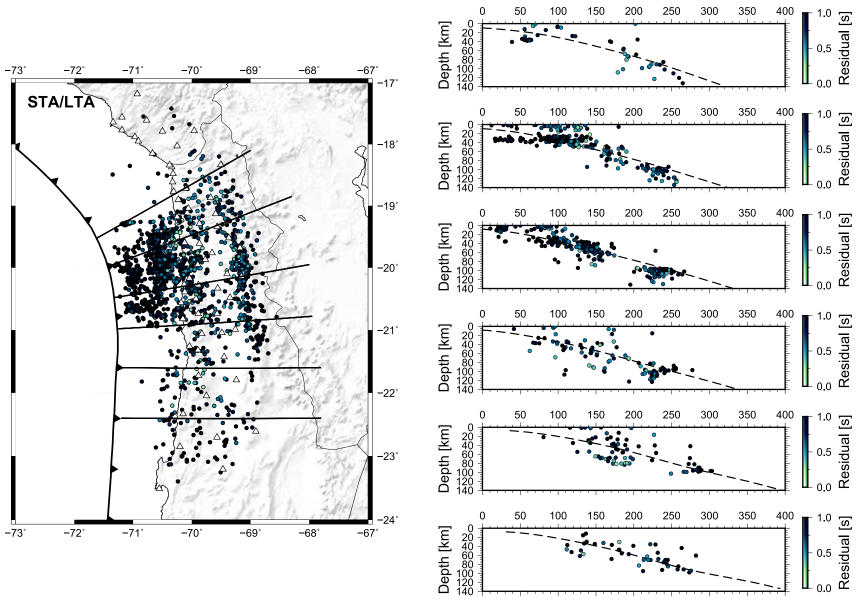


Figure 3.10: Hypocentre relocation comparison for the STA/LTA auto-picked catalogue. Event relocations are plotted as a function of RMS residual, slab profile is provided by [44].

STA/LTA picking approach respectively. The statistics of the residual distribution are also in a similar range to that of the manual picks (see Table 3.2).

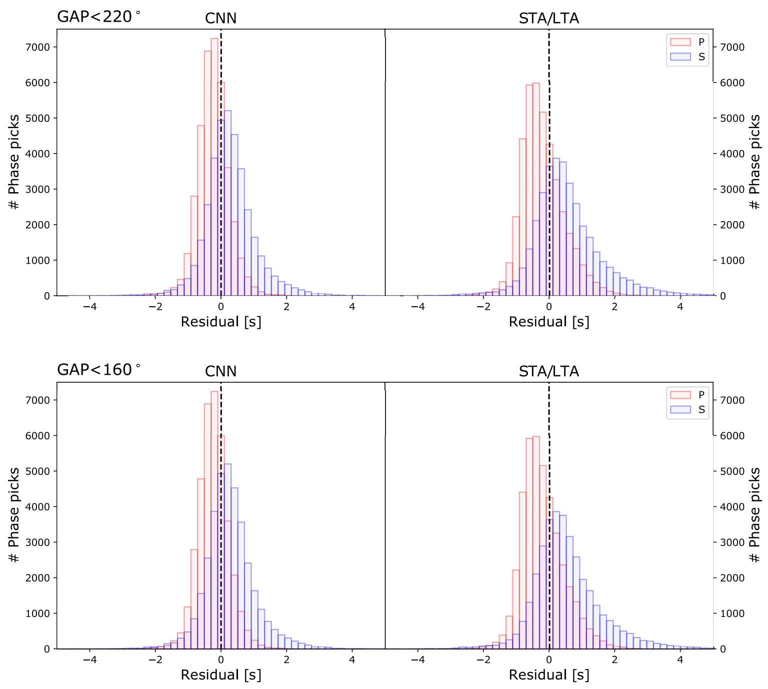


Figure 3.11: Both auto picking methods phase residuals following hypocentral relocations, plotted for well-located events (minimum azimuthal gap 160°) and the for entire relocated catalogues (minimum azimuthal gap 220°)

	GAP $< 220°$		GAP $< 160°$	
	CNN	STA/LTA	CNN	STA/LTA
μ_p	-0.238	-0.216	-0.247	-0.333
σ_p	0.487	0.696	0.393	0.623
μ_s	0.277	0.539	0.270	0.435
σ_s	0.780	1,081	0.596	0.922

Table 3.4: Statistics of residual distribution for both the CNN and STA/LTA derived catalogs. GAP $<$ 220° and GAP $<$ 160° relate to the filters applied to the catalogs to only keep events with an azimuthal gap less than the threshold in degrees. $\mu_p, \mu_s, \sigma_p, \sigma_s$ are the average P- and S-phase residual, and the standard deviation of the P- and S-phase residual distribution respectively.

3.4 Discussion

3.4.1 CNNs as an efficient method of feature engineering

This chapter has investigated the potential for deep learning techniques to improve the performance of the first stage of any phase-based event detection pipeline - seismic phase picking. CNNs, are an optimal technique for this task, they can be interpreted as an automatic feature engineering approach when compared against traditional methods which use manual derivation of 'characteristic functions' for detecting the presence of a seismic phase. This investigation shows that even when the data are scarce, a simple CNN approach still significantly outperforms traditional techniques such as the STA/LTA. This is evidenced by the improved σ of the P- and S- pick relocation residuals, for the well-located events (minimum azimuthal gap $< 160°$), resulting in a decrease of 0.230 s, and 0.326 s respectively when using the CNN over the STA/LTA picking approach. The results from this work add to the literature on supervised learning-based methods for seismic phase classification and demonstrate that with appropriate considerations regarding overfitting and generalization, such methods can improve seismological processing workflows, not just for large well-recorded catalogs, but for varying study regions and datasets.

3.4.2 Future work

Future applications of deep learning techniques for seismic phase picking could therefore include deploying such pre-trained systems on poorly monitored areas of interest resulting in improved data recovery (recall), and more efficient automation of seismic workflows. The results of this work also raise interesting questions, inline with other initial investigations of deep learning for seismic event detection [102, 159]. The improved recovery of seismic phases, especially the improved recovery of the S-wave, has huge implications for improving the quality of identified events. As deep learning-based pickers such as CNN architectures are also potentially able to operate at lower signal-to-noise ratios, this will result in many more individual deterministic onsets to correlate to an underlying source. We have highlighted the power-law scaling of the Gutenberg-Richter law (Figure 2.11). This relationship means that that CNNs, along with other related deep learning techniques, could potentially generate orders of magnitude more picks when compared against traditional approaches to-date. The next step of phase-based seismic event detection is to correlate these phase identifications to some underlying source. This is now a much more difficult problem in light of recent developments in seismic phase detection, and provides a natural avenue for future work. Can ML/data-driven techniques also be leveraged to correlate this increased amount of information, in an efficient manner? This is what we will explore in Chapter 4.

4 Seismic Phase Association Through Machine Learning

After having explored how ML approaches are transforming the amount of phase information obtained from seismic streams, this chapter presents a novel method for associating this new generation of seismic phase catalogs in an efficient manner. Again, we focus on how to best exploit the information contained in such catalogs through the use of 'data-driven' techniques. We adapt a parametric model fitting approach, applying an iterative hypothesis testing logic to enable extracting of true events even in very low signal-to-noise ratio environments. Our parametric model is based on the physical properties of the moveout of seismic energy with distance, this contrasts the highly non-linear functional approximators, such as neural networks which are strongly dependant on the training data. With this approach, we can provide a generalized first-order approximation for the predicted arrival of seismic phases across arbitrary local to regional seismic networks. This is agnostic to the station network geometry, where other techniques could require retraining on a case-by-case basis to deploy to new regions. To systematically test the performance of our approach, we construct a series of synthetic tests to see where the performance of our algorithm breaks down. The chapter is based upon the following publication:

- **J. Woollam**, A. Rietbrock, J. Leitloff, and S. Hinz. "HEX: Hyperbolic event extractor, a seismic phase associator for highly active seismic regions". in *Seismological Society of America* 91:5, 2020, pp. 2769–2778. DOI: http://dx.doi.org/10.1785/0220200037.

4.1 Introduction

The automatic detection of seismic onset from radiated energy caused by an earthquake or other sources has been studied intensively over the last decade. However, the correlation of onset times to a common source, termed phase association, has not gained much attention until recently. With the continued increase in station density, along with hugely improved automatic detection algorithms the amount of detected seismic phases has increased enormously. This has led to a renewed focus on phase association methods to correlate the seismic energy contained within the latest, extensive phase catalogues.

Since the inception of phase association, the predominant approach to solving this problem typically involves the application of a grid-based search procedure. Whilst individual nuances exist, these algorithms can be summarized as seeking to parameterise the region of interest and back-project arrivals to detect those which originated from the same coherent source [109, 37, 118, 27]. Often requiring large-numbers of parameters to tune, along with complex logic to prevent mis-associations, backprojection methods struggle to associate events during periods of intense, frequent seismicity, struggling when events occur close to each other in space and time, or if events are not well imaged throughout the seismic network. All current software packages for automatic earthquake association currently apply some form of the backprojection methodology to correlate events (e.g. Earthworm, SeisCompP3, Early-Est, Antelope, GLASS3).

Separate from associating deterministic phase picks, seismic events can also be correlated with template-matching techniques, which correlate events based on similarity of neighbouring waveforms [36]. These methods work well for small networks, performing accurate event correlations for regions of low signal to noise [126, 101, 113], but the expensive computational cost reduces generalizability. In addition, matched templates are highly dependent on the 'master' template selected and therefore might "ignore" a vast number of small earthquakes for which no template exists, potentially misrepresenting the seismicity in a given region.

Seismologists are continuously seeking the identification of smaller, more frequently occurring events as they provide further insight into the complex physical processes occurring within the subsurface. Dense catalogues of well-located seismic arrivals are, therefore, of vital importance within seismology. The data can be used for creating higher resolution tomographic models[127, 45] and the improved hypocentral locations develop the understanding of a variety of seismogenic processes; examples include monitoring of seismicity rate changes [81], anthropogenic induced seismicity [47, 73] and fault imaging [55]. As classical seismic association routines cannot correlate events during the most intense seismic episodes, the crucial processes occurring during these periods remain undetected. For example, the post-seismic period of a major earthquake contains extensive seismicity. Accurately resolving the complex, small-scale seismicity occurring during such periods could hold the key to further understanding of an entire range of physical processes associated with earthquake ruptures.

With the continued advancements in instrument deployment, increasingly smaller magnitude events are now being detected. These events often lie close to or below the background noise level and are more likely to occur simultaneously. Seismic picking algorithms have also progressed significantly in recent years [7, 63], accurately and reliably picking such larger volume datasets. In particular, the recent application of deep learning-based techniques to seismic picking has itself resulted in a step-change in overall volume of picks now being made, with these techniques demonstrating state-of-the-art performance at increasingly lower signal-to-noise ratios (SNR) [111, 159, 147]. Such factors further increase the number of phases to simultaneously correlate, rendering traditional association methods unsuitable as they are overwhelmed by the vast portions of information contained in such datasets. In light of such developments, event-association methods have seen a renewed focus, where the application of machine learning-based techniques again demonstrate significant performance improvements over traditional methods. The single-station waveform similarity detection algorithm, FAST [152] applies a datamining approach to encode compact representations of continuous waveform data for efficient clustering of similar waveforms. This removes the need for a 'master' template and greatly reduces computational cost. [112] apply a recurrent neural network to model the temporal and contextual relationship between sequential picks to discriminate which belong to the same event. [80] improve upon previous back-projection methods by approaching the problem in a graph-theory context, solving for sources and phase assignments simultaneously. As developments in seismic sensor resolution and picking methods continue, we can expect the seismic phase association task to inevitably involve associating increasing amounts of simultaneously occurring seismic events, in the presence of numerous false impulsive signals.

Considering this continued growth in seismic data volume, we seek to leverage an approach to seismic phase association which is computationally efficient and performs estimation well in the presence of outliers (low SNR). We adopt the logic of the RANdom SAmple Consensus (RANSAC) [32] algorithm, a machine learning technique widely applied within the computer vision community [124, 96] and apply this to the phase association task. This approach to parameter estimation is specifically designed to work in the presence of a high proportion of outliers, a common problem for temporary seismic networks where stations are often installed in imperfect seismic recording conditions. By incorporating such logic, we mitigate many of the problems associated with traditional association methods and are able to demonstrate a minimum event spacing resolution of ∼15 s through synthetic testing. Accurate event associations at this scale are beyond the typical detection rates for regional seismic operational and research catalogues [127]. We present a computationally efficient phase association routine which can be applied in real-time, throughout all typical seismic environments.

4.2 Hyperbolic Event eXtractor (HEX) associator logic

The problem of associating seismic phases can be cast as a parameter estimation problem, where we seek to identify - to first order - the hyperbolic relationship in a catalogue of automatic picks generated by an earthquake with unknown origin time. This problem can be treated analogous to the Normal-Moveout (NMO) correction in reflection seismology. For a homogeneous half-space, the arrival time of the pick ti at station i with location (x_i, y_i) can be expressed by

$$t_i^2 = t_0^2 + \frac{\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}}{v^2} \tag{4.1}$$

With a seismic velocity v, location of the epicentre (x_0, y_0) and zero offset time t_0 which in our case incorporates the unknown origin time of the earthquakes. Therefore, all picks associated with the same event can be approximated by a travel time hyperbola. It is well known from reflection seismology that equation 4.1 provides a good approximation for a one-dimensional velocity model when replacing v with the root mean square average velocity, v_{RMS}; even for 3D velocity models the hyperbolic approximation still acts as a very good zero order approximation [41]. There are two major advantages to applying the hyperbolic moveout to the event association problem. Firstly, by only inverting for a combination of zero offset travel time (therefore depth) and origin time the strongest trade off in the event localisation problem is circumvented. Secondly, we do not have to pre-define a velocity model as v in equation 4.1 is used in the sense of a hyperparameter acting as a first order control over the curvature of the hyperbola.

4.2.1 RANSAC for seismic phase association

The RANSAC algorithm is a parameter estimation approach which iteratively draws random minimal sets of data points from a data distribution (γ_D). Each set of samples (γ_S) form a candidate solution for the model parameters (m^n). For every iteration, the candidate model is evaluated according to some cost-function, with the standard being an inlier/outlier-based threshold. After n trials, the best scoring candidate model is saved as the final model. This conceptually simple, but powerful approach is easily generalisable and so is extensively used throughout the computer vision community due to its proven ability to deal with datasets containing a high proportion of outliers [115].

Framing the RANSAC logic in the context of the event association problem; the data distribution (γ_D) represents the catalogue of deterministic phase picks. As we seek the association of phases corresponding to the same underlying seismic source, we apply the RANSAC algorithm to iteratively sample subsets of picks (γ_S; refer to the black points in Figure 4.1).

Each γ_S is used in constructing a parametric model of event moveout (hyperbola, Figure 4.1), where the model is projected across all stations, providing arrival time predictions. Any "inliers" are, therefore, phases which lie within the models predicted arrival time plus some threshold (clear points, Figure 4.1), and the "outliers" are any other phases. All models are evaluated based on their weighted inlier count, with the best scoring model saved. RANSAC assumes that once a subset of picks belonging to the same underlying event is found, the fitted model provides an optimum solution to explain both the physical moveout parameters, and to associate any phases related to that event. Figure 4.1 displays a summary of the HEX associator logic.

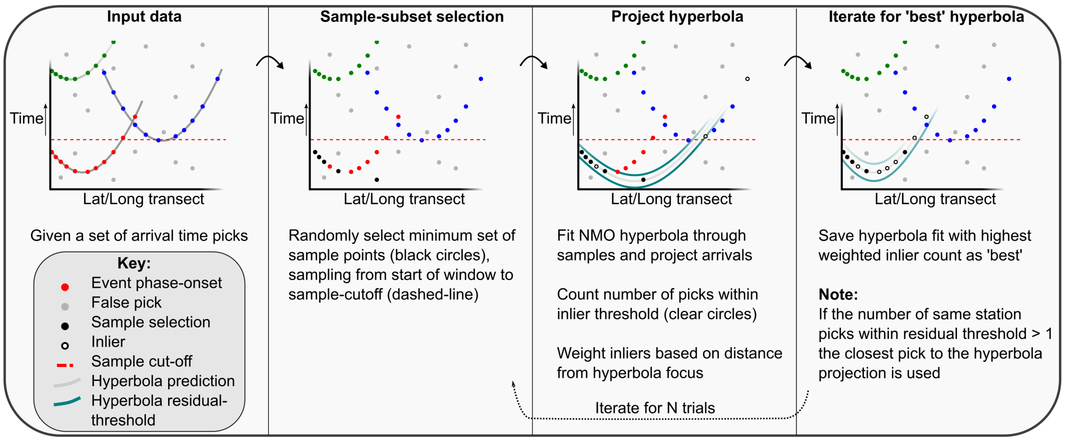

Input data
Sample-subset selection
Project hyperbola
Iterate for 'best' hyperbola
Time
Lat/Long transect
Given a set of arrival time picks
Key:
Event phase-onset
False pick
Sample selection
Inlier
Sample cut-off
Hyperbola prediction
Hyperbola residual-threshold
Randomly select minimum set of sample points (black circles), sampling from start of window to sample-cutoff (dashed-line)
Fit NMO hyperbola through samples and project arrivals
Count number of picks within inlier threshold (clear circles)
Weight inliers based on distance from hyperbola focus
Iterate for N trials
Save hyperbola fit with highest weighted inlier count as 'best'
Note:
If the number of same station picks within residual threshold > 1 the closest pick to the hyperbola projection is used

Figure 4.1: Schematic demonstrating the logic of the HEX association routine.

Algorithm 1 Extract hyperbolae using RANSAC

```
Require: $n \geq 0$
  Set starting number of inliers $N_{best} = 0$
  for $n$ trials do
      Select subset of $\alpha$ traveltime observations randomly from $[t_w, t_{w+n}]$
      Using $\alpha$ samples, perform least-squares estimation of Eq. 4.1 to get hyperbola parameters
  $(x_0, y_0, t_0, v)$
      Create candidate hyperbola using model parameters and project arrival times $(t_s)$
      Count $N$ traveltime observations within projection $\pm$ residual threshold $(t_s = t_i \pm r)$
      if number of inliers $N$ greater than previous best $(N > N_{best})$ then
          Save new inlier count as best $(N_{best} = N)$
          Save candidate hyperbola parameters as best $(p_{best} = [x_0, y_0, t_0, v])$
      end if
  end for
```

We solve for the hyperbolic parameters through constrained least-squares [12]. Bounds can be set for each parameter during optimisation, to ensure physicality in the inversion process. By framing this parameter estimation problem within the RANSAC logic (pseudo-code provided in Algorithm 1), any false picks are easily disregarded, as the algorithm discriminates based on a physical model of wavefield propagation. Many extensions to the standard RANSAC framework exist. Proposed adaptions improve specific aspects of either; computation time, accuracy, or robustness. These factors inherently trade-off against one-another. Probabilistic extensions such as MLESAC (Maximum Likelihood SAC) [139], MAPSAC (Maximum A Posterior Estimation SAC) [138] modify the cost-function to maximise the likelihood, improving accuracy at the expense of computational cost [18]. Other proposed adaptions attempt guided sampling (GASAC, [108]; PROSAC, [78]) in the sample selection stage to accelerate performance, but the additional

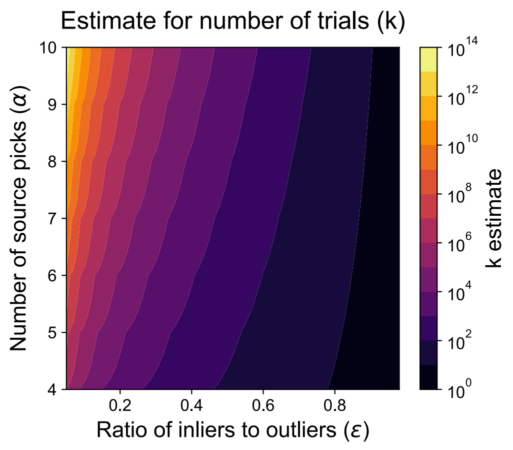


Figure 4.2: Trade-off plot for estimated number of trials to select a full inlier subset (k) as a function of inlier-outlier ratio and number of source picks

computation step can again increase the computation time [18]. If the true distribution is known, the computational cost of RANSAC can be simply estimated. With the number of trials required to select a full inlier subset (k) defined as

$$k = \frac{log(1-p)}{log(1-\epsilon^{\alpha})}, \tag{4.2}$$

where p is the probability that at least one of the subsets is outlier free (typically set to 0.99; e.g. [25]), ϵ is the ratio of inliers to outliers, and α is the number of observations to select. The α value is an upper-bound for the number of required iterations as it assumes samples are selected independently. Figure 4.2 displays the trade-off plot for k as a function of both ϵ and α, demonstrating how the ratio of inliers to outliers in the distribution exhibits the strongest control over the estimated minimum number of trials.

For the problem of event-association, dense pick catalogues contain an unknown number of events. An unknown number of false picks further decreases the inlier-outlier ratio, greatly increasing the estimate for k (Figure 4.2). Considering such factors, our application of the RANSAC logic seeks to optimize computational performance, whilst remaining robust. To optimise the computational performance, we apply the traditional inlier/outlier count as the cost-function. Applying a 'hard-bounded' cost function additionally allows for accommodation of any potential uncertainty in the model, such as the non-hyperbolic moveout due to a heterogenous velocity half-space. We also perform our own form of 'guided' sampling. When sampling candidates, only picks within a pre-defined window $[t_w, t_{w+n}]$ are selected (dashed line, Figure 4.1). This increases the inlier-outlier ratio (ϵ) decreasing k. Factors such as spherical divergence and attenuation decrease the likelihood of a phase onset being detected with distance from source. We, therefore, apply a distance-based weighting schema to any inliers, focusing hyperbola extraction on local spatial regions where there is the greatest likelihood of detecting seismic phases linked to the same event.

4.3 Synthetic testing

When associating phases in practice, the true number of sources will be unknown. To quantitatively test the performance of our algorithm, we adopt a synthetic testing workflow. This approach mainly follows the rationale of the latest event association studies [112, 80]. It should be noted that whilst adopting a synthetic testing workflow in line with previous works allows for general comparisons with other association techniques, the forward models and underlying station network geometries differ for each study. Direct quantitative comparisons cannot, therefore, be performed, as the testing regions are fundamentally different. General performance comparisons can, however, be made as all methods follow a common testing approach. All coordinates are expressed in terms of distance from the centre of gravity of the seismic network. We initially generate a grid of 3,146 events throughout northern Chile, all sources within the event grid have spacing 20 x 20 x 2 km in latitude, longitude, and depth respectively (Figure 4.3).

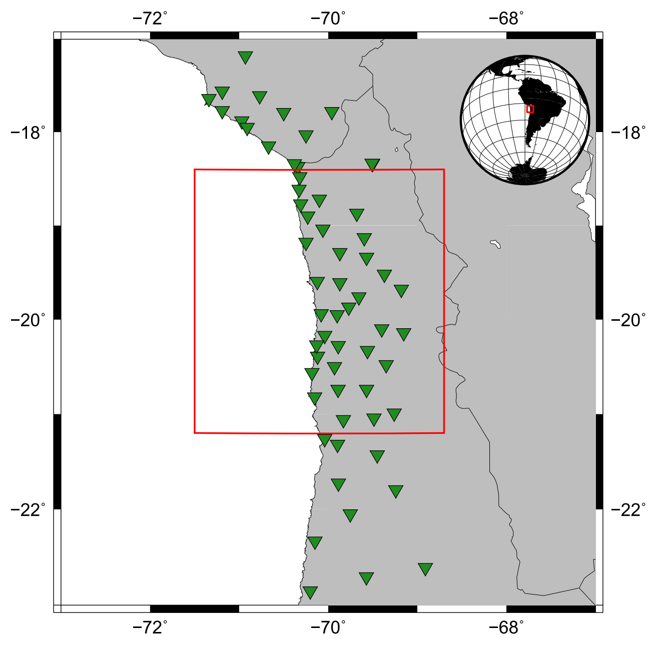


Figure 4.3: Displaying station network throughout northern Chile used in synthetic testing. Triangles represent seismic stations; the grid of potential event sources is delineated by the boxed region.

Forward modelling of the P-phase traveltimes for each source are calculated using the 1D velocity model of [49]. Each traveltime is assigned a label corresponding to the synthetic source from which it originated. The aim is to cluster or correlate phase arrivals which correspond to the same event source. As the routine is associating P-phase hyperbolae, for all of the following tests the optimization bounds for Equation 4.1 are set as $(x_{min} = -400km, x_{max} = 400km, y_{min} = -400km, y_{max} = 400km, t_{min} = -30s, t_{max} = 40s, v_{min} = 5km/s, v_{max} = 12km/s)$. It should be noted that in our test a source might be located outside the station network, a phenomenon commonly encountered within subduction zone environments.

4.3.1 Single event stress test

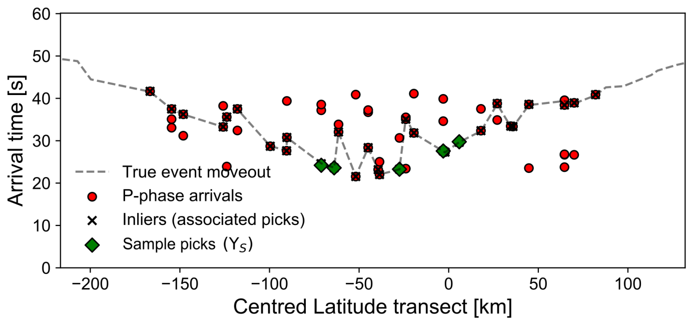
To highlight the unique advantages afforded by the RANSAC framework, the HEX associator is tasked with first associating a single event in the presence of increasing proportions of false picks (Figure 4.4). For this test, one source is selected from the event-grid and the forward travel-times are calculated. A ratio of M false picks are added (termed 'noise factor' in Figure 5). All false arrival times are sampled from a uniform distribution $U \sim (tt_{min}, tt_{max})$. If the weighted inlier count is less than a minimum value γ, the association to a single common event is rejected. These parameters are then systematically tested, varying the ratio of false picks, the number of sample points used for fitting (α), and the residual cut-off threshold (γ). Testing for a single event only provides a simple guide as to where the algorithms association performance breaks down. The results of this test also serve as a general outline for HEX input parameter selection.

Figure 4.5 displays the statistics for the single event stress test when the number of sample points (α) is set to 5. Any cases where HEX has detected false additional events are plotted as crosses.

The precision and recall metrics [103], assess the association performance for this simple test case (Equation. 4.3).

$$Precision = \frac{TP}{TP+FP} \quad , \quad Recall = \frac{TP}{TP+FN} \quad , \quad F1 = \frac{2(PXR)}{P+R} \tag{4.3}$$

Precision is defined as the ratio of true positives (TP) (true event picks correctly associated) to true positives plus false positives (FP) (false picks incorrectly associated); the recall is the ratio of true positives to true positives plus false negatives (FN) (true event picks the associator missed); the F1 score is simply the harmonic mean of both the precision and recall. For any of the HEX input parameter combinations, no spurious event associations are made when up to 50 $\%$ false picks are present in the input window (noise factor = 1). Increasing the minimum number of sample points to use in the model selection stage (α) allows for accurate event correlation of events at greater noise levels, but such an approach decreases the probability of detecting smaller magnitude



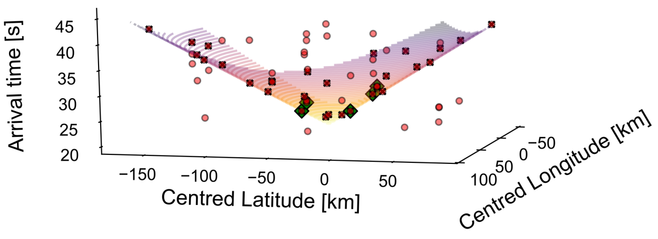


Figure 4.4: A result of the single-event stress test. Top plot compresses all the arrivals onto a latitude transect, plotting arrivals as solid circles, picks then associated by HEX are overlain by crosses, the true event moveout is indicated by the dashed line, source picks for hyperbolic parameter estimation are diamond markers. Bottom plot displays the same association hyperbola in 3D.

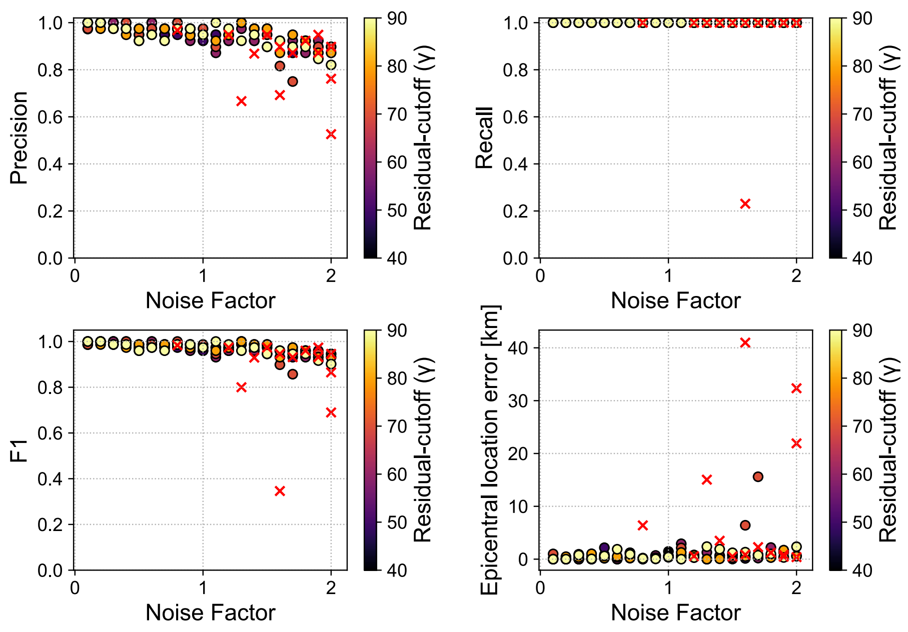


Figure 4.5: Results of the single event stress test where number of sample points (α) was set to 5. Crosses represent iterations where the association routine found extra, false events.

events, which are typically only picked across a limited number of local stations. Increasing α also raises the computation time (refer to Figure 4.2, Equation 4.2). Figure 4.5 serves to highlight the inherent suitability of the RANSAC framework to the phase association problem. The approach associates phases even in cases of significant noise. As a physical model of seismic event moveout is used to correlate phases, once a full inlier subset is detected, the remaining picks corresponding to this event are typically found, indicated by the recall metric in Figure 4.5.

4.3.2 Continuous stream simulation stress test

For future applications, we envisage that the HEX association algorithm will be applied on real-time data streams. To test the performance under such conditions, a workflow is constructed for the simulation of multiple sources arriving simultaneously. The initial catalogue of 3,146 events are randomly shuffled and sampled. For each iterative sample taken from the catalogue, the event onset time t_i is set to the previous event onset time plus a given offset c. Where c is drawn from a uniform distribution U within a specified range (Equation 4.4). The onset time of the first event t_0 is set to 0.

$$t_i = t_{i-1} + c \quad where \quad c \sim U(c_{min}, c_{max}) \tag{4.4}$$

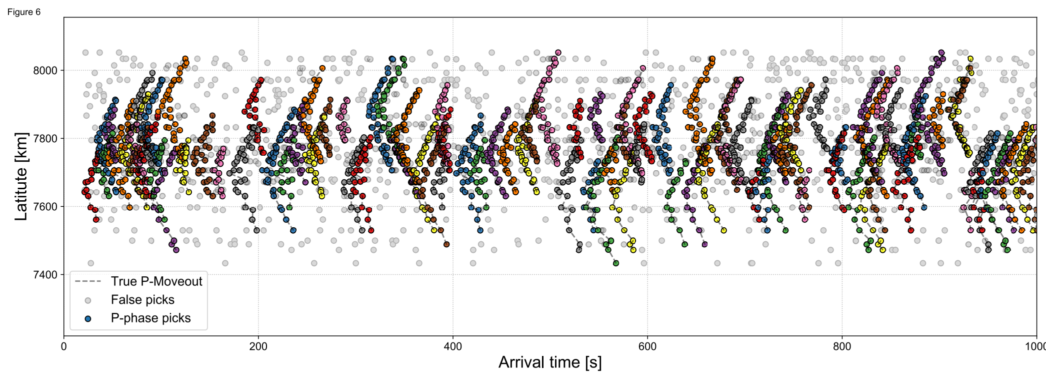


Figure 4.6: Displaying the P-phase arrivals for the continuous stream simulation. Events are coloured are function of true event label, the true event moveouts are indicated by dashed lines.

The maximum source receiver distance is varied for each event to simulate heterogenous propagation effects, again sampling from a uniform distribution $U \sim (80, 250)km$. The ratio of false picks is set to 0.4 with false picks onset times drawn from $U \sim (0, max(tt))$. Figure 4.6 displays an example of the synthetic event data where $c_{min} = 0s$ and $c_{max} = 20s$. The synthetic data catalogue is then sequentially windowed and passed to the associator (window-size=150s, timestep=10s). To ensure the association routine fully captures all phases corresponding to an event, the window size should be large enough to capture the full hyperbolic moveout of arrivals across the network. We also recommend setting a timestep $<$ inlier search window size to allow the algorithm to fully capture any event foci which lie at the edge of the search window. HEX is then applied to each window to search for potential events with the following input parameters $(n = 1000, \alpha = 5, \gamma = 50, r = 2)$. If the number of phases present in the sampling window is less than α, or if an associated hyperbolas weighted inlier count is less than γ then all events in the window are assumed to be found, and the routine continues to the next window.

Figure 4.7 displays an example of how the synthetic events are sequentially associated and removed from the synthetic catalogue. For this test, a metric to assess the performance of HEX in terms of associating events is also required. For each association made (A_i), the Event Precision (E^p_i) is calculated between it and the c true events

$$E^p_i = \max_{j=1}^{c} \frac{A_i \cap B_j}{|A_i|} \tag{4.5}$$

As the maximum intersection identifies the true event (B_j) with the most common picks to A_i. We assume that A_i is attempting to associate this true event. E^p_i is simply theratio of correct picks over total picks associated. We calculate the recall metric similarly; for each true event B_j, the Event Recall $(E^r_j)$is calculated by comparing over all d event associations

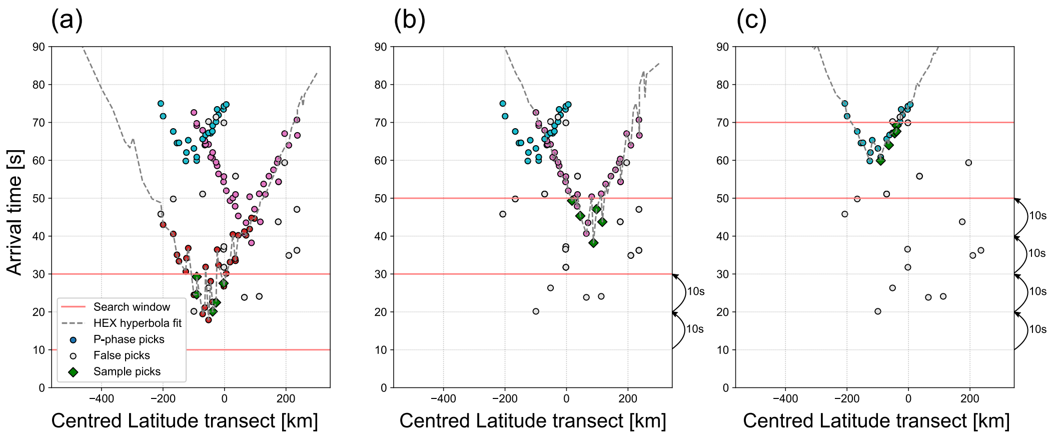


Figure 4.7: Example of how the HEX algorithm sequentially associated 3 hyperbolae from the synthetic catalogue in a 'greedy' manner, where once an event is associated the phases are removed from the catalogue. Events are coloured a function of true event label, dashed line denotes HEX association moveout.

$$E_j^r = \max_{i=1}^{d} \frac{A_i \cap B_j}{|B_j|} \tag{4.6}$$

E_j^r represents the ratio of true picks recovered within an association. Similar to [112], we determine that any associations with $E_i^p > 0.5$ are correct. This is where more than half the picks in a given association are common with a single true event.

4.4 Results

We systematically stress test the HEX algorithm by setting $c_{min} = 0$ and varying c_{max} from 10, 15, 20, 25, 30, 60, 100 to 120s. For each test, 1000 events are created. To account for the randomness of the synthetic generator, each test was repeated 5 times and the average values were taken as final performance statistics. Figure 4.8 displays the results of the continuous stream synthetic tests. The performance is exceptional up to ~ 15 s average spacing. However, we do note that there is no clear drop off in the event precision and recall curves, so this performance assessment is subjective.

Beyond groupings of picks, HEX also estimates epicentral parameters for each association. This information can be used in further evaluating the accuracy of any detections. For every HEX association, we again use the maximum intersection of A_i and B_j to identify which true event the algorithm has attempted to correlate. For this event, the distance between true and predicted epicentre provides a further estimate of association accuracy. We show the proportion of events within r km of the true epicentre, as a function of the average event time separation (Figure 4.8).

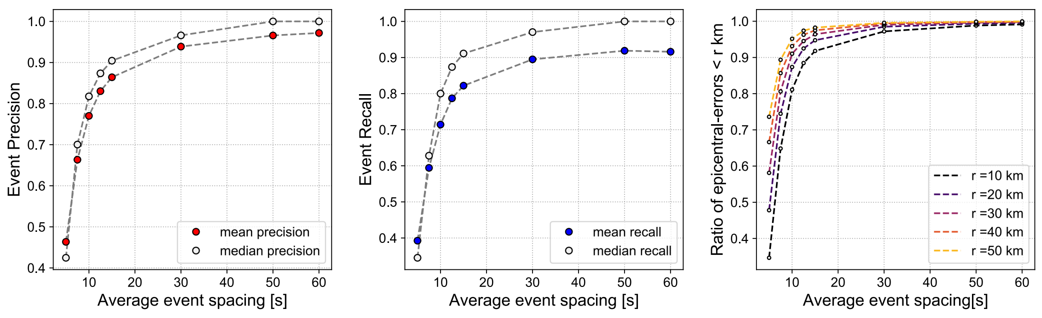


Figure 4.8: Displaying the Event Precision, Event Recall and epicentral error for the continuous stream synthetic tests.

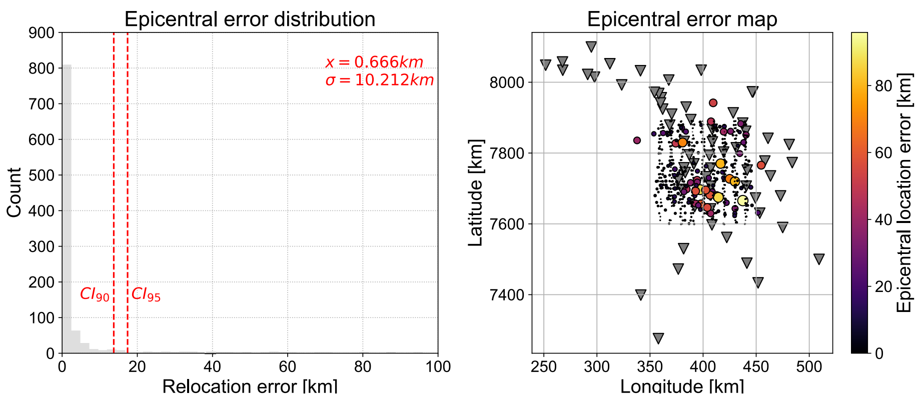


Figure 4.9: Epicentral error distribution. Left hand side plot is a histogram of the epicentral relocation error, confidence intervals are indicated by vertical dashed lines. Right hand side plot is the map view of epicentres, the size and scale of the marker is a function of the epicentral error.

Again, at 15s average spacing 92% of the events have an epicentral error < 10 km, indicating correct associations. The epicentral coordinates provided by HEX should be used as starting locations for automatic earthquake locations to account properly for uncertainties and location error estimates. Figure 4.9 shows the epicentral error distribution of one of the continuous stream tests when setting $c_{max} = 30$. The median ($\hat{x}$) is used as a measure of central tendency.

4.5 Discussion

4.5.1 Where ML can enhance seismic phase association

Whilst earthquake detection is a well-established concept within seismology, recent developments underline the need for improved association algorithms. Improved picking routines [159, 111, 147] and instrumentation mean that there are now significantly greater volumes of picks which need correlating. Seismic association algorithms are, therefore, seeing a renewed focus [112, 80, 153]. We present HEX as an efficient routine to associate events under extreme noise conditions. Applying a framework specifically designed to deal with a large proportion of outliers in the data distribution, greatly enhances the robustness of the routine. HEX differs from traditional back-projection based phase association methods in that the parameterization and subsequent grid-searching steps are not required. From a computational perspective this is more tractable. The association problem is essentially reduced to iteratively fitting the best hyperbolic model to explain the data, assuming that the full set of samples is contaminated by noise. The hyperbolic fitting concept has been used for decades in reflection seismology. Any phase associations are made based upon the underlying physical model of wavefield propagation, and errors in either the picks, or the model itself can be accommodated with a hard bounded cost function. HEX can deal with scenarios where over 50% of the picks are false, opening up the possibility to have highly sensitive pickers which operate at lower signal-to-noise ratios but create a significant amount of false picks.

4.5.2 Extrapolating performance to realtime continous data streams

In this work, we perform associations over a regional network, however, the routine itself can be applied to detect on regional, local or even micro-seismic scale. With our stress tests performed for our sparse, regional network, a greater inter-station distance makes event correlations inherently more difficult. As events are better recorded across denser networks, we expect performance to even increase when applied to such scenarios. Sampling the maximum source to station distance from a uniform distribution also represents an upper bound on seismicity rate. As event frequency and magnitude obey a logarithmic relationship in practice, large magnitude events will not overlap as significantly as shown in Figure 4.6. Finally, simultaneously occurring arrivals will interfere and only be detected as a single arrival in practice; all these factors indicate that the stress tests performed in this study serve as an extensive example as to the limitations of the HEX routine.

Whilst this study introduces HEX as viable approach to phase association, we expect future adaptations of this logic to further improve performance and generalisability. For example, the selection of γ is intrinsically linked to optimal event associations. The optimal γ value will vary based on the underlying station distribution. Practical applications of the HEX associator should explore a general guide for the γ parameter as a function of station geometry. Other adaptions to the HEX routine are introduced in later sections to improve it's applicability in practical applications. The S-wave information is also integrated to improve the amount of information belonging to one underlying event which needs correlating. This has a significant impact on the signal-to-noise ratio of true phase picks related to false picks associated with either separate, unrelated events, or false picks picking some form of impulsive noise. These new integrations are applied in a practical case study in chapter 6.

4.5.3 How to evaluate competing ML event detection algorithms

After having introduced two novel techniques to both detect (chapter 3) and associate seismic phases (this chapter), what remains is to quantitatively evaluate how these methods perform on real data. To achieve this, systematic benchmarking routines are vital. Being able to compare state-of-the-art routines easily across the same data is a key component of advancing the state-of-the-art for any task. Our initial presentation of the new ML-based components has performed some benchmarking against traditional methods but we can greatly improve the way in which benchmarking is achieved. The following chapter (chapter 5) will explore this concept. We will introduce software specifically designed for iterating and comparing state-of-the-art methods for seismic event detection and more general tasks in seismology.

5 The SeisBench framework: A Toolbox for Machine Learning in Seismology

Given the rapid increase in interest in ML methods throughout recent years, numerous workflows now exist to apply ML to notable tasks in seismology. For fields where ML is more well-established (e.g. computer vision and natural language processing), the rapid advancement of the state-of-the-art algorithms for solving tasks can be attributed to two major factors; firstly, the open availability of benchmark training data for assessing performance; and secondly, the development of general-purpose toolboxes for accelerating research and development. An added benefit of such general purpose toolboxes is that they often provide high-level interfaces to the latest state-of-the-art research. For new researchers in the field, this can greatly reduce the 'barrier for entry' - which helps grow the research community. Such toolboxes can, therefore, act as a reference point for productionizing the research process and quickly comparing the benefits and limitations of the latest models.

In this chapter, we present SeisBench as such a toolbox for ML research. It is a general purpose toolbox for accessing both benchmark training data, and the latest state-of-the-art models for ML tasks in seismology. This chapter is based upon the following publication:

- **J. Woollam**, J. Münchmeyer, F. Tilmann, A. Rietbrock, D. Lange, T. Bornstein, T. Diehl, C. Giunchi, F. Haslinger, D. Jozinović, A. Michelini, J. Saul, and H. Soto. "SeisBench—A Toolbox for Machine Learning in Seismology". in Seismological Research Letters, 2022. DOI: http://dx.doi.org/10.1785/0220210324.

It is also based upon

- J. Münchmeyer, **J. Woollam**, A. Rietbrock, F. Tilmann, D. Lange, T. Bornstein, T. Diehl, C. Giunchi, F. Haslinger, D. Jozinović, A. Michelini, J. Saul, and H. Soto. "Which picker fits my data? A quantitative evaluation of deep learning based seismic pickers". in *Journal of Geophysical Research: Solid Earth*, 2022, http://dx.doi.org/10.1029/2021JB023499.

Where the benchmarking works were conducted through the SeisBench toolbox. Here, both Jannes Münchmeyer (J.M) and myself contributed equally to both publications, in particular, we both worked on assimilating the different benchmark datasets, converting the waveforms and metadata from the differing data formats found at individual research centres into the standardized SeisBench format. We jointly built the unified interface in PyTorch [99] to access the models, converting any pre-trained weights. The remaining components of the software also saw joint implementation, including any augmentation pipelines For the picking benchmark, J.M performed the computation of the benchmarking itself, with the project outline and metric definition a communal work Both majorly contributed to the write up and review process for either paper, along with additional inputs from the other co-authors. Both publications, therefore, have an 'equal contribution' footnote appended to the author list - to denote contributions.

5.1 Introduction

Seismology has always been a 'data-rich' field. With the continued advances in computational power, along with the increased use of high-density density deployments of nodal geophones, the seismic wavefield is now recorded with increasing resolution and fidelity. Such advances are not just exclusive to seismology; within science in general, larger, more detailed datasets are being compiled. Machine Learning (ML) has risen to prominence as a set of techniques to best exploit the information contained in such extensive datasets. Often termed 'data-driven' methods, ML tools probabilistically model the statistical properties of a given dataset to perform inference for a given task. As datasets get larger, and the inference step becomes a more tractable problem, these techniques are now achieving state-of-the-art performance across the entire spectrum of scientific fields. In many areas performance is outpacing the human analyst.

Although some pioneering works harnessed neural networks for seismological applications [145, 140], for many years such techniques did not find wider usage in seismology until approximately three years ago. The possibility to assemble large datasets, massive parallelisation on commodity hardware through GPU computing, algorithmic improvements and, importantly, the availability of software frameworks such as PyTorch [99] and Tensorflow [1] has driven a wave of applications of ML techniques to classical seismological problems, including earthquake phase identification [159, 111, 147, 157, 143], earthquake detection [85, 86, 157, 102, 26, 98], magnitude estimation [74, 84, 31, 91], and earthquake early warning [61, 71, 92], amongst others.

From these initial works, a natural question arises: which ML techniques perform best for each task? Answering this question is not trivial, as each study uses different data, different ML frameworks for algorithm development, and different assessment metrics. Benchmarking and comparison studies are, therefore, inherently difficult. The varying data used during training is a particular problem, as the variable nature of earthquake source, propagation medium and site conditions mean that the performance of a model trained on one region or environment might not be directly compared to a model trained in a different region. To enable fair comparisons of models and model architectures over a range of possible environments, benchmark datasets are essential.

Labelled benchmark datasets have been vital to rapid-progress in various classic ML application domains, most prominently computer-vision (MNIST, Deng, 2012; ImageNet Deng et al., 2009), and natural language processing (Sentiment140, Go, Bhayani, and Huang, 2009), as they allow for easy assessment of which ML algorithms perform best. Creating such quality-controlled datasets takes, however, a significant amount of time. Benchmark datasets perform this step for users ensuring comparability of different studies, greatly accelerating the development and testing of novel ML algorithms. With ML methods only recently being widely adopted in seismology, historically, there were no benchmark datasets available for comparison works. This situation is now changing with the value of such datasets widely recognised. The seismological benchmark datasets now emerging (e.g. LenDB, Magrini et al., 2020; INSTANCE, Michelini et al., 2021; NEIC, Yeck et al., 2021; STEAD, Mousavi et al., 2019) already cover a wide-range of potential seismic environments (e.g. global, regional, local), essential factors for training robust algorithms.

However, the availability of new benchmark datasets does not completely solve the comparison problem. Remaining issues include the differing data formats employed by different benchmarks, and the specific framework libraries ML researchers use to implement their models e.g. PyTorch, Tensorflow, Keras [19], Sklearn [100], add complexity to any comparison work. Any benchmarking must, therefore, check that operations applied within each library are directly comparable, with no discrepancies in implementation.

The easy availability of both benchmark datasets, and standardised access to the latest models, are crucial ingredients for advancing the state-of-the-art. As this problem is common to any application based on ML [57], tools have been developed in other fields to provide researchers with easy access to models and benchmark datasets (e.g. FLAIR, Akbik et al., 2019, natural language processing toolbox). These continue to be widely used, evidence of their ability to aid development.

To date, we are unaware of the availability of such software in seismology. The outlined bottlenecks affect a wide range of potential users of ML. For the 'practitioner', who wishes to apply ML models to their seismic data, they are currently facing significant hurdles, as they would have to learn specific frameworks to integrate the latest ML algorithms into their workflows. For the 'expert' interested in developing novel techniques, they currently have to integrate various models, testing over varying datasets', which may be in differing formats. Without any frameworks or toolboxes to help with these problems, researchers must construct such comparison pipelines from scratch. This is a significant undertaking. These factors are currently hindering more widespread ML adoption in seismology and are limiting progress in the development of the next generation of ML methods. Tackling these problems is key if the seismological community is to accelerate the development of ML techniques for seismic tasks and promote further adoption of ML within the field. We have built the SeisBench open-source software package to address these issues.

5.2 The SeisBench ML framework

SeisBench provides a unified point-of-access for ML development and application within the seismological community. Built in Python, it integrates both state-of-the-art models and datasets in a single framework. Figure 5.1 visually highlights this concept, introducing the core components of SeisBench. The range of datasets presented in the initial release include currently published seismological benchmark datasets from the literature, directly integrated into the package.

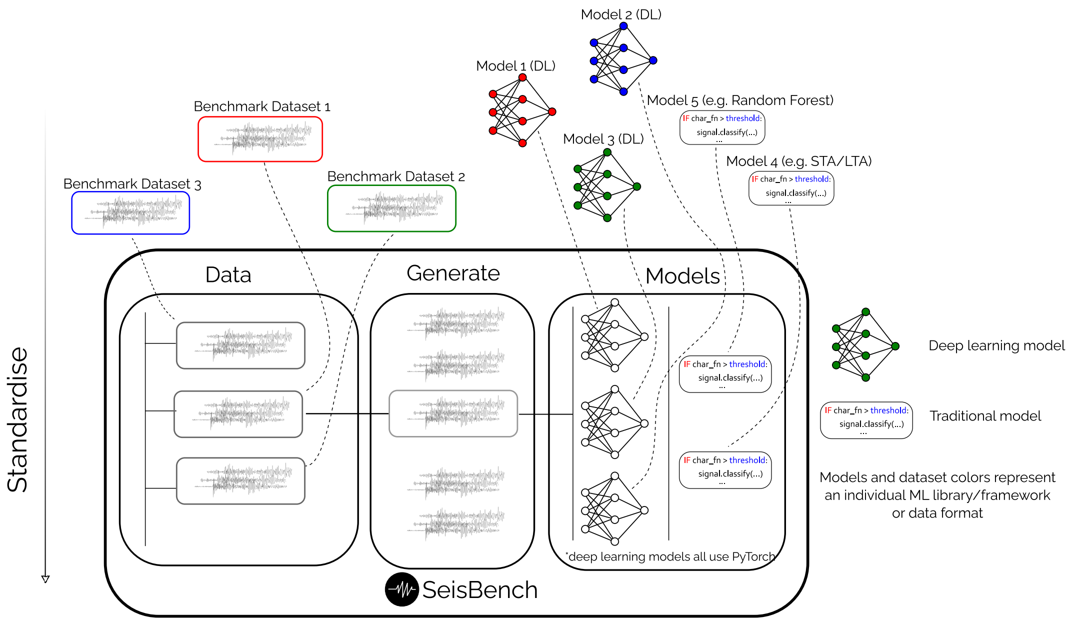

Model 2 (DL)
Model 1 (DL)
Model 5 (e.g. Random Forest)
Model 3 (DL)
Model 4 (e.g. STA/LTA)
Benchmark Dataset 1
Benchmark Dataset 2
Benchmark Dataset 3
Standardise
Data
Generate
Models
IF char_fn > threshold: signal.classify(...)
*deep learning models all use PyTorch
SeisBench
Deep learning model
Traditional model
Models and dataset colors represent an individual ML library/framework or data format

Figure 5.1: Schematic diagram to show the motivation behind SeisBench. SeisBench acts as a unifying framework for developing models and applying them to seismic data. The differing packages used for model development, and the differing benchmark dataset formats are represented by varying colours. The *data*, *generate*, and *model* tags highlight the different modules available within SeisBench.

SeisBench also provides access to additional custom benchmark datasets which are made newly available in the initial release of the software. As all datasets within SeisBench adhere to a common format, users can compare their algorithms across a range of seismic environments, from detecting global signals to local settings. Models are accessed through a unified interface – enabling easy comparison of differing approaches. Whilst the model interface is designed towards integrating various deep learning models, the types of models that can be built and compared in

SeisBench are not just limited to deep learning-based routines; traditional methods can also be directly deployed and integrated into comparison workflows. Finally, typical data augmentation and pre-processing steps are provided through an augmentation API. With seismologists, and general ML practitioners often, re-implementing the same operations for data pre-processing and augmentation, inclusion of many of the standard processes and augmentations in SeisBench will further facilitate faster model development.

SeisBench is designed to be generally applicable to the entire spectrum of general seismological tasks, such as source parameter estimation, magnitude estimation, ground motion prediction. Whilst the currently included models relate specifically to picking and event detection, SeisBench is suitable for many other seismological tasks based on waveform analysis. The extensible nature of the API means that any parameter from a datasets' metadata can be used as a label (target variable), enabling the construction of any supervised classification pipeline.

5.2.1 Data - Standardising access to Benchmark datasets

A standardised format for seismic waveforms and metadata information

The SeisBench *data* module contains functionality to read and process seismological datasets which have been converted into the SeisBench standardised format. Using a standardised framework enables the construction or conversion of varying benchmark seismological datasets. The dataset format follows a typical approach encountered within the ML community (Figure 5.2), where the waveforms (training examples) are included in a single file. We use Hierarchical Data Format 5 (HDF5) to store the raw waveforms [33]. Each multi-component waveform example is indexed by a lookup key. For all datasets, the required parameter *'trace_name'* is used as the lookup key. The labels/metadata associated with each training example are then stored in a simple table-structure (.csv). To ensure compatibility across datasets, metadata parameter names should follow a common naming schema 'CATEGORY_PARAMETER_UNIT' where: category defines the object which the parameter describes (i.e., path, source, station, trace); parameter describes the provided information e.g. latitude or longitude; and unit provides the unit of measurement e.g., m, cm, s, counts, samples.

Where several entries are required, such as trace start time and station name and location, such a data structure leaves the freedom to include additional specialised metadata only available for selected datasets. The metadata information is read into memory with the popular, high-level data-analysis library Pandas [106]. With such a format, users can easily create their own custom pipelines to query and extract metadata information associated with the waveforms. Providing a common framework for data storage is key to any proposed benchmarking works. Imposing restrictions on both the format and naming schema ensures that any newly defined parameters

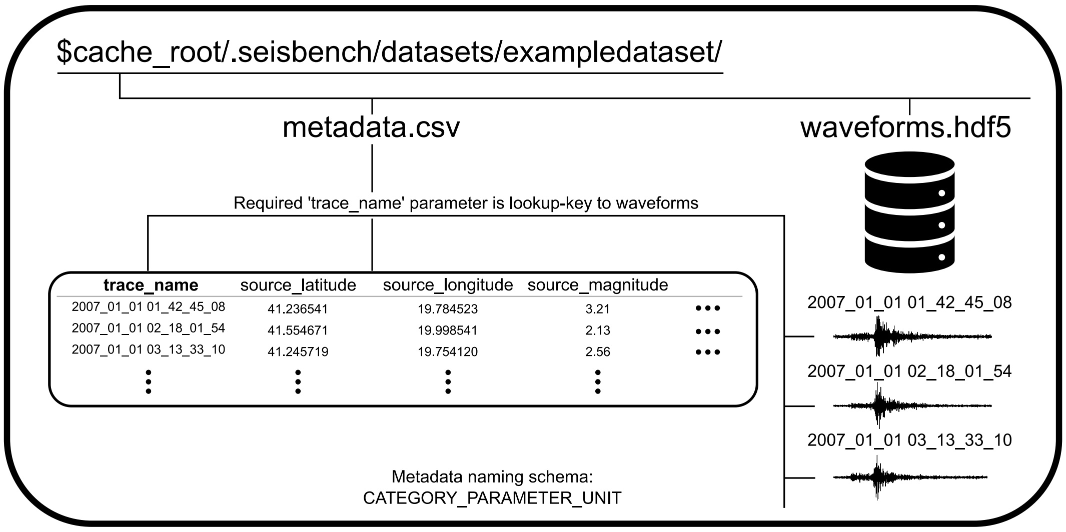


Figure 5.2: Example of data structure for SeisBench. Waveforms are stored in a HDF5 file, indexed by trace name. The metadata for each waveform example is stored in a table format as a .csv file. The trace name is required as a column, as this is then used as the lookup key to the raw data. This schematic diagram displays the overall concept, with the implementation slightly more complex to optimise performance. For more information see the technical documentation (`https://seisbench.readthedocs.io/en/latest/`).

are still standardised across datasets. This greatly the aids extensibility and comparability across datasets. Data throughput can be a major factor in the efficiency of training and application of ML models. SeisBench therefore introduces additional performance optimizations to the data structure that enhance IO read/write speed.

Once a dataset has been converted to the SeisBench format, it is integrated into the SeisBench API by extending the base dataset interface, providing a unique class for the dataset. Ordering the datasets into a class-based hierachy naturally reflects the dataset format. Common operations such as filtering metadata and obtaining waveforms are all available via the base dataset interface. Further individual properties of each dataset can then be encapsulated in the dataset class. Tools are available to help scientists to convert their own datasets into benchmark datasets and contribute them to the SeisBench repository, if desired.

Providing a common endpoint for benchmark datasets

We have converted a range of seismological benchmark datasets (Table 5.1; Figure 5.3) into the SeisBench data format. These datasets contain various types of seismic arrivals from local to global scales (Figure 5.4). All the datasets were either compiled from publicly available seismic data and metadata, or were directly converted from a published benchmark dataset from the literature. SeisBench thus provides easy access to data and model interfaces. All users have to do upon in-

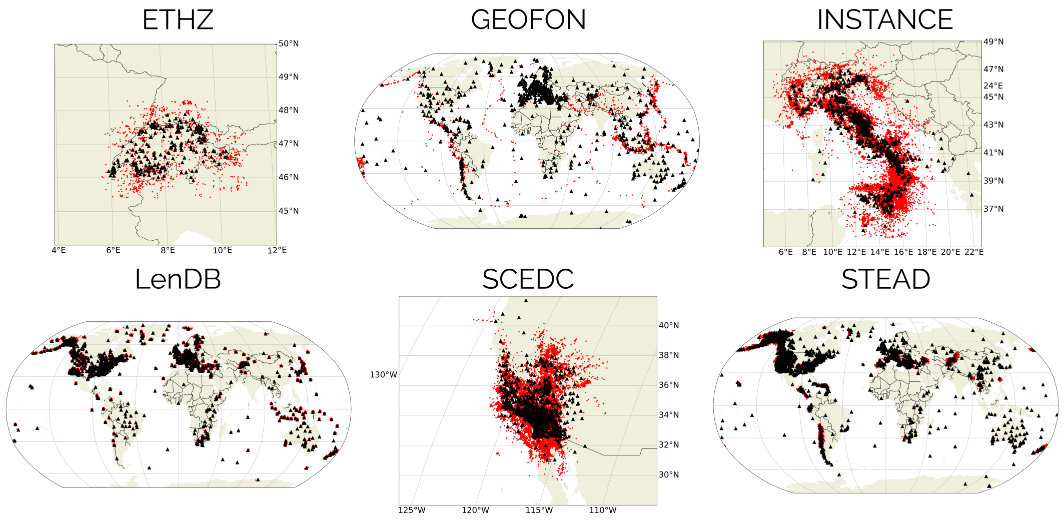


Figure 5.3: Benchmark datasets integrated into SeisBench with the initial release of the software; seismic sources are circles, stations are triangle markers. Not shown are some additional datasets which are included in the SeisBench initial release dataset collection, but are either missing source information (NEIC, GPD, Ross2018JGRPick, Ross2018JGRFM, Meier2019JGR), or have minimal number of events for plotting (the local Iquique dataset).

stallation of the package is to instantiate their preferred data/model object; the data will then be downloaded and cached for repeat use. Within each benchmark dataset, training, validation, and testing splits are pre-defined to reduce variability of benchmark comparisons resulting from randomness or different choices for dataset splitting approaches. Of course, it remains possible to define custom splits for specialized applications.

Here, we summarise the benchmark datasets integrated into the first release of SeisBench. The benchmark datasets can be separated into two groups, datasets that are missing some common metadata such as station location information, and those that contain all typical metadata information such as the station location and source parameters. Table 5.1, and Figure 5.3 and 5.4 generally only show those datasets of the first group, where all the common metadata are present. The following dataset descriptions provide further information on the included metadata.

Table 5.1: Overview of the datasets. The noise column indicates the number of dedicated noise traces. Note that it is still possible to extract noise examples from datasets without dedicated noise traces by selecting windows before the first arrival. For distances, the datasets cover local(L, $0 \leq \Delta < 150$ km), regional (R, $150 \leq \Delta < 600$ km), and teleseismic (T, $\Delta > 600$ km) records. The datasets with variable trace length contain considerably more than 60s of data for most examples. f_s denotes the sampling rate. When this parameter varies within a dataset, the range of sampling rates is listed. The GPD, JGRPick, JGRFM, Meier2019JGR datasets are omitted from this table because these datasets do not contain source property information. NEIC is included because it is used for the benchmark comparison in [93].

	Traces	Events	P picks	S picks	Noise	Region	Dist	Tr. length	f_s [Hz]
ETHZ	36,743	2,231	35,227	18,960	0	Switzerland	L/R	variable	100 - 500
INSTANCE	1,291,537	54,008	1,159,249	713,883	132,288	Italy	L/R	120 s	100
Iquique	13,400	409	13,327	11,361	0	N. Chile	L	variable	100
LenDB	1,244,942	303,902	629,095	0	615,847	various	L	27 s	20
SCEDC	8,111,060	378,528	7,571,970	4,364,155	0	S. California	L	variable	40 - 100
STEAD	1,265,657	441,705	1,030,231	1,030,231	235,426	various	L/R	60 s	100
GEOFON	275,274	2,270	284,240	2,847	0	global	R/T	variable	20 - 200
NEIC	1,354,789	137,424	1,025,000	329,789	0	global	R/T	60 s	40

ETHZ

The ETHZ benchmark dataset is a manually compiled dataset for SeisBench. It contains local to regionally recorded seismicity throughout Switzerland and neighbouring border regions. The data are recorded on the publicly available networks: [134, 16, 132, 133, 5], operated by the Swiss Seismological Service (SED) at ETH Zurich. To construct this dataset, we obtained both the waveform recordings and the corresponding metadata information via SED's FDSN web service (`http://www.seismo.ethz.ch/de/research-and-teaching/products-software/fdsn-web-services/`). Any detected seismic event from this network has had the phases manually labelled, including the discrimination of first, and later phases (e.g. Pn vs. Pg). In addition to the typical phase identification, the magnitude and polarity information is also available. In total, there are 57 metadata variables available for this dataset. We select all $M > 1.5$ events from the period of 2013 - 2020 for integration. In total there are 2,231 events containing 36,743 waveform examples. The traces are all in raw counts.

We split training examples for this dataset into training, validation, and testing example splits by setting all events before August 1st 2019 as training examples (61.6%), all events between this date and the 4th September 2019 are set as the validation split (9.9%), and all the remaining events later than this date are the testing split (28.5%). Please note that the validation set can also be called the development set. These terms are interchangeably used throughout the literature.

GEOFON

The GEOFON monitoring service acquires and analysis waveforms from over 800, globally distributed seismic stations worldwide. The GEOFON benchmark dataset has been compiled from these recordings. It is a teleseismic dataset which includes 2270 events containing ∼275,000 waveform examples occurring between 2009 – 2013. Events have been picked automatically initially, with manual analysis and onset re-picking performed routinely whenever necessary to improve the location quality. The magnitudes range from ∼M 2 - 9. With the bulk of events compromising intermediate to large events (M 5-7; Figure 5.4). Any regional events with smaller magnitudes are predominantly from the regions of Europe and northern Chile. 54 metadata variables are included with this dataset, the trace units are in raw counts.

For the GEOFON dataset, please note the varying class distributions of picked phase types for this dataset. For local and near-regional events S onsets have been picked and for a small fraction both Pn and Pg are included. For teleseismic events, almost no S onsets have been picked. Depth phases have been picked occasionally but not comprehensively

For the training, validation and testing splits, we set all events occurring before 1st November 2012 as training examples (58.6%), all events between this date and 15th March 2013 as the validation examples (10.1%), and any remaining events past this date as the testing examples (31.3%).

INSTANCE

The INSTANCE benchmark dataset [83] comprises ∼1.3 million regional 3-component waveforms from the Italian region, containing ∼50,000 earthquakes M 0 – 6.5 and also including ∼130,000 noise examples. Within SeisBench, we provide separate access to the individual partitions of this dataset. The noise examples and signal examples are available as their own distinct dataset; the seismic events are further subdivided into datasets with waveforms in counts, and with waveforms in ground motion units. A combined dataset containing all noise examples and waveform examples in counts is also available. A total of 115 metadata variables are provided. In addition to the standard metadata variables, this dataset includes a rich set of derived metadata, e.g. peak ground acceleration and velocity, assigned pick label uncertainty in seconds.

The training, validation, and testing sets are performed by randomly selecting 'event-wise' for this dataset. All waveform examples belonging to the same event are, therefore, in the same split group. The final proportion of waveform examples for each class are 60.3% for training, 10% for validation, and 29.7% for testing respectively.

Iquique

The Iquique benchmark dataset is a benchmark dataset of locally recorded seismic arrivals throughout northern Chile originally used in training the deep learning picker in [147]. It contains 13,400 waveform examples with 13,327 manual P-phase picks and 11,361 manual S-phase picks. All waveform units are in raw counts, there are 23 metadata variables associated with this dataset.

For this dataset, the training, validation and testing splits are selected through randomly sampling the training examples, returning 60%, 30% and 10% for the training, validation, and testing splits respectively.

LenDB

The LenDB benchmark dataset [77] is a published benchmark dataset containing local earthquakes recorded across a global set of 1487 broad-band and very broad-band seismic stations. It comprises ∼1.25 million waveforms. The dataset is split into 629,095 local earthquake examples and 615,847 noise examples. The data were processed using a bandpass filter between 0.1 - 5∼Hz and the instrument response was deconvolved to convert the recordings into physical units of velocity. Unlike the other datasets, only automatic P-phase picks are provided for LenDB. In total there are 23 metadata variables for this dataset.

The training, validation, testing split is performed by selecting all examples with waveform start times before 16th January 2017 as training examples (60%). Any examples between this date and the 16th August 2017 form the validation split (9.5%), and the remaining examples past this date form the test split (30.5%).

SCEDC

The Southern Californian Earthquake Data Centre (SCEDC) benchmark dataset has been constructed from publicly available waveform data [122]. The waveforms and associated metadata are obtained via the Seismic Transfer Programme (STP) client [121]. For the obtained seismic arrivals, all events have been manually picked. We select all publicly available recordings of seismic events in the Southern Californian Seismic Network, over the period 2000 - 2020. Only local recordings of seismic events (∼M -1 – 7) are included, with source to station paths spanning up to a maximum distance of ∼200 km. The dataset comprises ∼8 million waveform examples, which contain ∼7.5 million P-phases and ∼4.3 million S-phases. This dataset also contains a range of seismic instrument types including: extremely short period, short period, very broadband, broadband, intermediate band and long period instruments - both single and 3-component channels are also present. Units for the examples are raw counts.

The split for this dataset is set randomly, with 60%, 10%, and 30% of the data compromising the training, development, and testing splits respectively. For the magnitude metadata information, please note the increase of M = 0 in events in comparison to the overall trend (Figure 5.4) which suggests some data cleaning is still required for this dataset for the purposes of magnitude prediction.

STEAD

The STanford EArthquake Dataset (STEAD; Mousavi et al., 2019) published benchmark dataset, contains a range of local seismic signals – both earthquake and non-earthquake – along with noise examples. The dataset includes ∼1.2 million waveforms, of which ∼200,000 are noise examples and the remaining contain seismic arrivals from ∼450,000 earthquakes (∼M -0.5 - 8). The units for the waveform examples are raw counts and there are 40 metadata variables associated with this event.

For the split, we use the same test set as defined in [86] which randomly set 10% of the examples as testing examples, we then add a validation set by randomly sampling from the remaining samples. The final ratios of the training, validation, and testing split are again 60%, 30%, 10% respectively.

The following datasets include cases where the publicly available waveform data, along with corresponding metadata was available for training ML models, but some common metadata is missing.

NEIC

The National Earthquake Information Centre (NEIC; Yeck et al., 2021) published benchmark dataset comprises ∼1.3 million seismic phase arrivals with global source-station paths. As information on the trace start-time and station is missing for this dataset, it is stored in the SeisBench format, but without this normally required information.

For the training, development and testing split, the original publication presented randomly sampled splits, based on event-id. This random splitting approach is implemented in the SeisBench conversion of this dataset, again at 60%, 10%, and 30% for the training, development, and testing examples respectively.

There are additional datasets integrated into SeisBench which were originally used in training notable deep learning algorithms in seismology. Typically, the waveforms for these datasets were already pre-processed for training, including windowing and labelling, so the original station metadata for each training example is unavailable for these datasets. As many of the datasets also use picked waveforms from the SCEDC network, this results in potential common overlap between the following listed datasets, for both metadata parameters and waveforms. The only differences being potentially different metadata variables across datasets (e.g. picked phase labels, vs. first motion labels).

The deep learning training datasets converted into SeisBench format include: the 'GPD' training dataset [111] containing 4,773,750 examples of 4s waveforms, sampled at 100 Hz; the 'Ross2018JGRFM' dataset used for training the deep learning-based first motion polarity detection routine in the [112] study, containing 6∼s Z-component waveform samples from 100 Hz instruments; the 'Ross2018JGRPick' dataset used for training the deep learning-based picker presented in the same work; The 'Meier2019JGR' dataset, which contains the S. Californian component of the training examples from the [82] work.

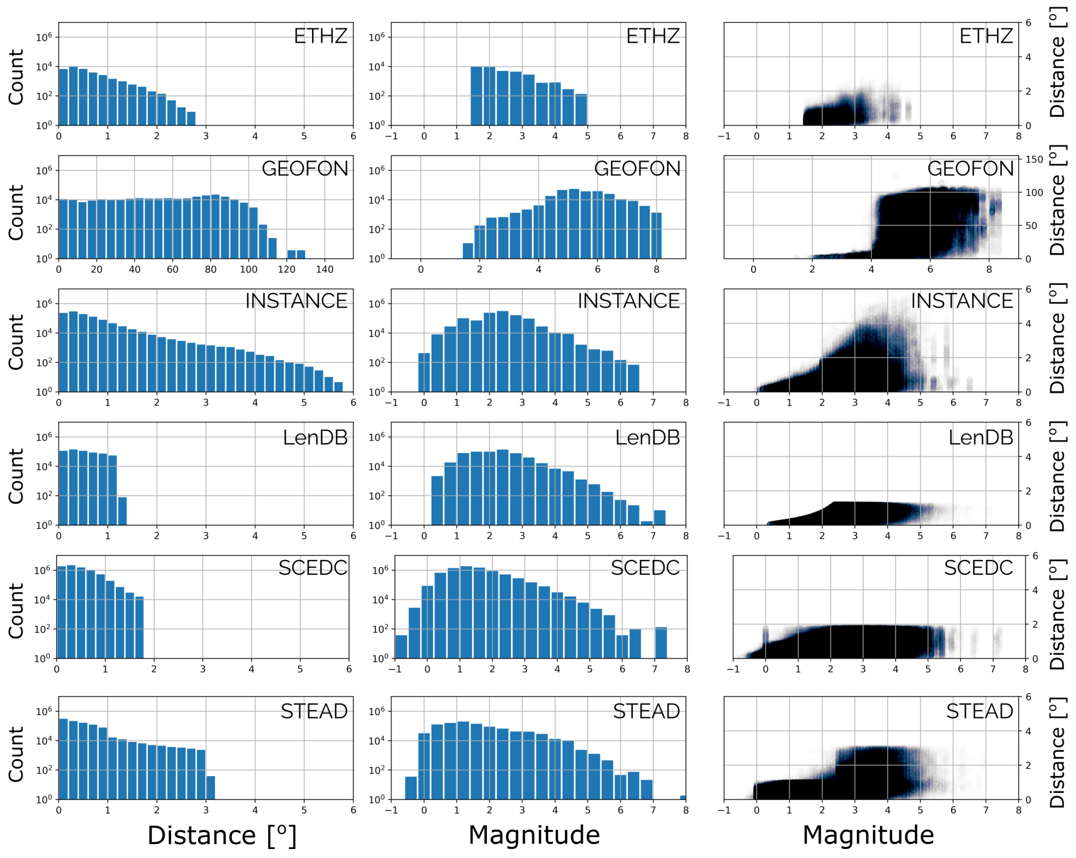


Figure 5.4: Logarithmic histograms of epicentral distance and magnitude distributions for the datasets with source and station information. For the two-dimensional scatterplot in the last column, all points are plotted with transparency to highlight the overall distribution. The Iquique, NEIC, GPD, Ross2018JGRPick, Ross2018JGRFM, Meier2019JGR datasets are not shown because they are lacking either, magnitude, or source and station location information.

5.3 Models

The SeisBench *model* interface is an extensible framework which encompasses the application of all types of models to seismic data. It is designed to be generalizable to arbitrary seismic tasks which operate on waveform data. A range of deep learning models from the literature are provided through SeisBench (Table 2). All deep learning models are integrated with the PyTorch framework [99]. Where possible, models integrated into SeisBench have the corresponding weights from the original training procedure integrated. We also provide weights for each of the models trained on each of the included datasets (see the companion paper to this work, [90]).

5.3.1 Initially integrated models

The initial set of models integrated into SeisBench are listed below, where the acronyms CNN and RNN relate to Convolutional Neural Network, and Recurrent Neural Network respectively. For a more detailed description, refer to [90].

	BascicPhaseAE	CRED	DPP	EQT	GPD	PhaseNet
# Params	33,687	293,569	199,731/ 546,081/ 21,181	376,935	1,741,003	23,305
Type	U-Net	CNN-RNN	CNN/RNN/RNN	CNN-RNN-Attention	CNN	U-Net
Training set	N. Chile	S. California	N. Chile	STEAD	S. California	N. California
Orig. weights	N	Y	N	Y	Y	N
Reference	[147]	[85]	[130]	[86]	[111]	[159]

Table 5.2: Description of the models studied. The number of parameters refers to the total number of trainable parameters. Note that these numbers might deviate slightly from the ones published by the original authors due to differences in the underlying frameworks. For DPP, information delimited by slashes indicate Detector/P-Picker/S-Picker networks. The row "Orig. weights" indicates whether original weights were published and are available in SeisBench. For PhaseNet, weights were published by the authors, but these weights could not straightforwardly be integrated into SeisBench due to technical issues.

- BasicPhaseAE [147], basic CNN U-Net, initially applied to regional aftershock sequence in Chile.
- CRED [85], CNN-RNN Earthquake Detector, initially trained on 500,000 training signal and noise examples from Northern California.
- DPP [130], DeepPhasePick, is a combination of a CNN for phase detection and two RNNs for onset time determination. Like BasicPhaseAE, the networks were designed for detect-

ing and picking local events, with an initial application on a regional seismic network in Chile.

- EQT [86], EarthQuake Transformer, an Attention-based Transformer Network to both detect and pick events.

- GPD [111] Generalised Phase Detection, CNN algorithm to detect seismic phases.

- PhaseNet [159], CNN autoencoder algorithm, adapts the U-Net segmentation framework to the 1D problem of classifying seismic phases.

5.4 Training data generation pipeline

A common task for training ML models in seismology is building data generation pipelines. First, some pre-processing is usually done; for example, traces need to be truncated to the correct length and possibly normalized, labels need to be encoded. Furthermore, often it is beneficial to augment the data to increase the variability on training examples, for example by adding noise to the waveforms. To standardize this task, reduce the required coding amount and reduce errors in the training pipeline, SeisBench provides the *generate* API (cf. Figure 5.1).

The *generate* API provides individual processing blocks, e.g., window selection, label definition, or normalization, which can be combined into a data generation pipeline in a flexible way. While many standard augmentations are already implemented, custom routines can be added easily. As the *generate* API only relies on the abstract *data* API, the same set of augmentations can be applied to any SeisBench compatible dataset with minimal changes in the code. In addition, since the *generate* API is integrated with PyTorch, it can facilitate efficient data generation with PyTorch's built-in multi-processing.

5.5 Example workflows - Using SeisBench benchmark datasets and models

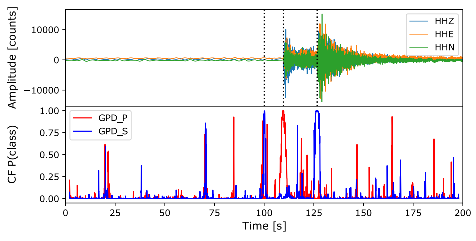
Here we highlight how the features and functionality provided through SeisBench can support users with their tasks, from practitioners just looking to use an ML model to experts wishing to conduct extensive, in-depth, comparison and benchmarking pipelines.

5.5.1 Workflow 1 - Use pre-trained models for picking new seismic streams

This workflow is relevant for practitioners who seek to leverage ML techniques on seismic data, but do not necessarily have the in-depth domain knowledge to do this through ML frameworks. This example demonstrates how to pick seismic waveforms with two leading, pre-trained models (EQT and GPD) via the SeisBench API. The commands to do this are displayed in Figure 5.5. The high-level functionality allows users to apply ML models to seismic data with just a few commands. If not previously downloaded, the pre-trained model weights are downloaded and subsequently cached for repeat use. The annotate and classify methods of the SeisBench models integrate with stream objects from the obspy package [11], widely used within the seismological community. We omit the plotting code for brevity. Users can easily expand upon this example workflow to conduct seismic detection and picking pipelines. In terms of computational performance, we test the EQTransformer implementation on a K80 GPU and annotate 24 hours of 100 Hz data from a single station in 6 s. Scaling this process results in a months worth of data being labelled in $\sim$3 minutes.

```
[1]: from obspy.clients.fdsn import Client
     from obspy import UTCDateTime
     import matplotlib.pyplot as plt


     # Get a seismic stream
     client = Client("GFZ")

     t = UTCDateTime("2007/01/02 05:48:50")
     st = client.get_waveforms(network="CX", station="PB01", location="*",␣
      ↪channel="HH?", starttime=t-100, endtime=t+100)
```

```
[2]: from seisbench.models import GPD

     # Load pre-trained weights into model
     gpd_model = GPD.from_pretrained("original")

     # Apply GPD to stream
     gpd_annotations = gpd_model.annotate(st, stride=5)
     gpd_picks = gpd_model.classify(st, P_threshold=0.95, S_threshold=0.95)
```



```
[2]: from seisbench.models import EQTransformer

     # Load pre-trained weights into model
     eqt_model = EQTransformer.from_pretrained("original")

     # Apply EQT to stream
     eqt_annotations = eqt_model.annotate(st)
     eqt_picks = eqt_model.classify(st)
```

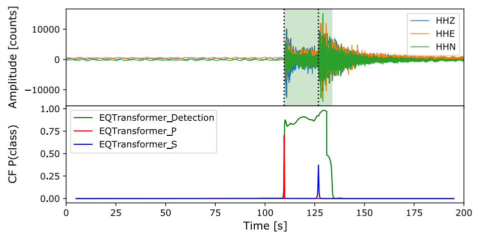


Figure 5.5: Example code-blocks which download a seismic waveform [1], then loads a pre-trained deep learning picking model and applies the model to predict on the seismic stream using either one of two ML architectures (GPD and EQTransformer) [2]. Resulting picks and characteristic functions from the output probabilities are displayed beneath the code blocks. Characteristic function is abbreviated to "CF". Picks are represented by dotted lines, event detections for the EQT case are the shaded regions. The GPD picker makes a spurious S -pick before the onset of the event but as the original model weights have been incorporated into the pickers to pick on new, unseen data, this example may not be representative of the optimum performance of the respective model architectures, which could be achieved by training on data matched to the application case.

5.5.2 Workflow 2 - Training models

Training a deep learning model

For those wishing to train a deep learning model, Figure 5.6 provides a run through of how this workflow can be built in SeisBench. This workflow highlights how the *data*, *generate*, and *model* modules combine to help users perform all the typical tasks required in such a pipeline. Any loading of the required models and data is performed initially. In this example, we train PhaseNet on the INSTANCE dataset. Once the dataset and model are loaded, the *generate* module can be used to perform typical pre-processing and data augmentation steps on the waveforms. The generator object accepts a suite of augmentations which will be applied to each batch during training. In this example, we randomly window the waveforms, normalise the amplitudes using the maximum amplitude present in the window, change the datatype to 32bit floats, finally creating a probabilistic vector representation of P-picks, S-picks, and noise examples in the waveform. These steps can be achieved in 10 lines of code through SeisBench (see the *Preprocessing and augmentations* code block, Figure 5.6). The waveforms following processing are displayed in Figure 5.6. The augmented waveform data then form a training sample for PhaseNet. We also show the standard PyTorch syntax to iterate through a DataLoader object and train the model as the last step (see the *Train* code block, Figure 5.6).

Transfer learning

Rather than train a new model from scratch, transfer learning forms another common workflow users may require. Transfer learning, involves using a pre-trained model, initially trained for some given task - for example detecting seismic phases on a regional scale throughout S. California - and subsequently training the model to solve a related task - such as detecting teleseismic arrivals. This is often a useful as the knowledge learned during the initial training phase results in relatively less data being required to optimize the model for the new task.

The modular nature of the API means that to switch any dataset or model for another, all that is required is to change data or model imported (indicated by the dashed lines in Figure 5.6). So, to load a pre-trained version of a given model, all users have to do is call the *from_pretrained* method. The syntax to perform this step is also displayed in workflow 1. Datasets can also be swapped easily. For the purposes of this example, any dataset containing P-, and S-picks could be loaded in place of the INSTANCE dataset in workflow 2, and the training would then be performed on this alternative dataset, using the PhaseNet model initially trained on regional seismic waveforms in California as initialization for the training.

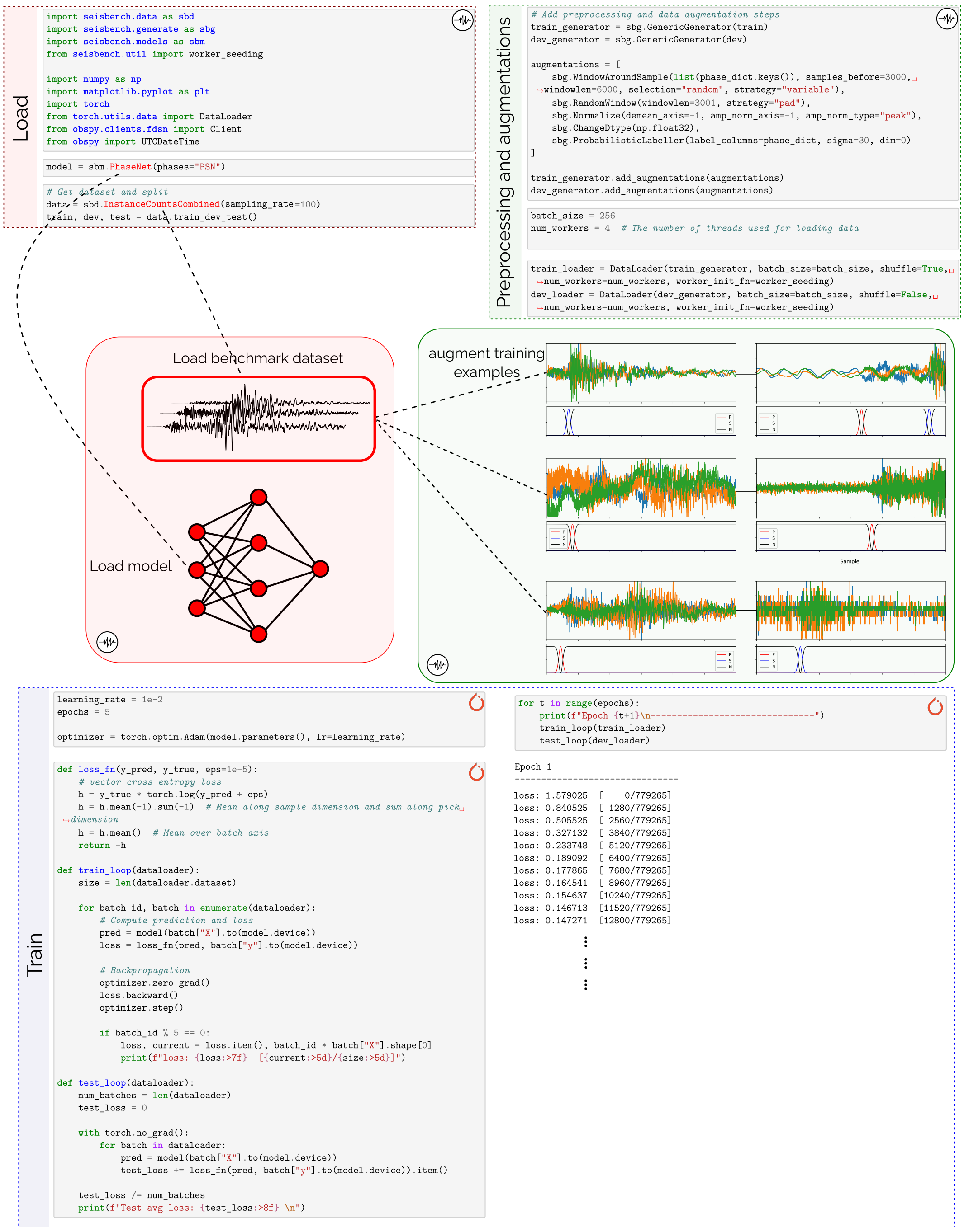
5.5.3 Workflow 3 - Benchmark differing models across differing datasets

Beyond training for a single model or dataset, SeisBench allows for comparison pipelines to be easily constructed. Having an objective measure of the performance of newly proposed algorithms against current state-of-the-art routines is fundamental to progress in any field, and standard procedure in traditional ML domains such as image recognition. As ML is a recent adoption within seismology, it could be argued that this step has not yet been carried out extensively. A detailed benchmarking study of various published ML picking models was carried out by us with the SeisBench framework and is presented with the companion paper to this work [90]. The code used for this benchmark study is made available and can serve as a template for future benchmarking studies[1].

5.6 Extensibility

The SeisBench API is published with an open-source license (GPLv3). The software is designed to be extensible, and we encourage the seismological community to contribute. If users wish to integrate their own benchmark datasets or models to the package for public download, we ask that they get in touch with the project through GitHub (`https://www.github.com/seisbench/`

[1]Available at `https://github.com/seisbench/pick-benchmark`



```python
import seisbench.data as sbd
import seisbench.generate as sbg
import seisbench.models as sbm
from seisbench.util import worker_seeding

import numpy as np
import matplotlib.pyplot as plt
import torch
from torch.utils.data import DataLoader
from obspy.clients.fdsn import Client
from obspy import UTCDateTime
```

```python
model = sbm.PhaseNet(phases="PSN")
```

```python
# Get dataset and split
data = sbd.InstanceCountsCombined(sampling_rate=100)
train, dev, test = data.train_dev_test()
```

```python
# Add preprocessing and data augmentation steps
train_generator = sbg.GenericGenerator(train)
dev_generator = sbg.GenericGenerator(dev)

augmentations = [
    sbg.WindowAroundSample(list(phase_dict.keys()), samples_before=3000,␣
 ↪windowlen=6000, selection="random", strategy="variable"),
    sbg.RandomWindow(windowlen=3001, strategy="pad"),
    sbg.Normalize(demean_axis=-1, amp_norm_axis=-1, amp_norm_type="peak"),
    sbg.ChangeDtype(np.float32),
    sbg.ProbabilisticLabeller(label_columns=phase_dict, sigma=30, dim=0)
]

train_generator.add_augmentations(augmentations)
dev_generator.add_augmentations(augmentations)
```

```python
batch_size = 256
num_workers = 4  # The number of threads used for loading data
```

```python
train_loader = DataLoader(train_generator, batch_size=batch_size, shuffle=True,␣
 ↪num_workers=num_workers, worker_init_fn=worker_seeding)
dev_loader = DataLoader(dev_generator, batch_size=batch_size, shuffle=False,␣
 ↪num_workers=num_workers, worker_init_fn=worker_seeding)
```

```python
learning_rate = 1e-2
epochs = 5

optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

```python
def loss_fn(y_pred, y_true, eps=1e-5):
    # vector cross entropy loss
    h = y_true * torch.log(y_pred + eps)
    h = h.mean(-1).sum(-1)  # Mean along sample dimension and sum along pick␣
 ↪dimension
    h = h.mean()  # Mean over batch axis
    return -h

def train_loop(dataloader):
    size = len(dataloader.dataset)

    for batch_id, batch in enumerate(dataloader):
        # Compute prediction and loss
        pred = model(batch["X"].to(model.device))
        loss = loss_fn(pred, batch["y"].to(model.device))

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if batch_id % 5 == 0:
            loss, current = loss.item(), batch_id * batch["X"].shape[0]
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")

def test_loop(dataloader):
    num_batches = len(dataloader)
    test_loss = 0

    with torch.no_grad():
        for batch in dataloader:
            pred = model(batch["X"].to(model.device))
            test_loss += loss_fn(pred, batch["y"].to(model.device)).item()

    test_loss /= num_batches
    print(f"Test avg loss: {test_loss:>8f} \n")
```

```python
for t in range(epochs):
    print(f"Epoch {t+1}\n------------------------------")
    train_loop(train_loader)
    test_loop(dev_loader)
```

```
Epoch 1
------------------------------
loss: 1.579025  [    0/779265]
loss: 0.840525  [ 1280/779265]
loss: 0.505525  [ 2560/779265]
loss: 0.327132  [ 3840/779265]
loss: 0.233748  [ 5120/779265]
loss: 0.189092  [ 6400/779265]
loss: 0.177865  [ 7680/779265]
loss: 0.164541  [ 8960/779265]
loss: 0.154637  [10240/779265]
loss: 0.146713  [11520/779265]
loss: 0.147271  [12800/779265]
                  ⋮
                  ⋮
                  ⋮
```

Figure 5.6: Example code-block with additional schematic diagrams displaying syntax required to perform full training of a deep learning model in SeisBench. PhaseNet is used for training, with the INSTANCE dataset being used as training data. Further workflow examples demonstrating the functionality provided by SeisBench can be found at https://github.com/seisbench/seisbench/tree/main/examples.

seisbench); where further information on the contribution guidelines can be found. In particular, we encourage inclusion of already published models and datasets. The code-base has extensive test coverage to reduce the risk of coding errors.

With the picking and detection problems having been widely explored in recent years with ML approaches, more complex problems are now being tackled with these techniques. We envisage that the models incorporated into SeisBench will expand to include such tasks. For example, hypocentre determination, source parameter estimation, etc., can all be constructed with SeisBench. All that is required is that the labels for a supervised learning task are present in the metadata. Once the state-of-the-art ML models for a given task are available in SeisBench - as shown with the picking example above - the major advantages of integrating new models within this framework become apparent. The initial processing routine set up for a model can be directly used to compare against existing state-of-the-art models. This ease of testing will hopefully promote further innovation of ML in seismology.

5.7 Discussion

In this chapter we have introduced the SeisBench toolbox as an open-source Python package, built to aid users in their application of ML techniques to seismic data. SeisBench minimizes common barriers to development for both practitioners looking to apply ML methods to seismic tasks, and experts who wish to benchmark and train leading algorithms. The software provides access to recently published benchmark datasets for machine learning in seismology, downloadable and accessible through a common interface. SeisBench extends this concept to provide a common access point to ML models, with state-of-the-art models and corresponding weights for seismic tasks directly integrated. We provide access to a range of picking models from the literature in the first iteration of the software but the framework is applicable for many seismological tasks based on waveform analysis such as location and magnitude estimation. By tackling some of the common bottlenecks encountered when developing ML algorithms, we hope that SeisBench will help practitioners iterate and deploy their models, advancing the development of the next generation of ML technique within seismology. As an anecdotal point, since its open-source release, and publication of the software paper [150], SeisBench has seen over 1,200 individual users access the documentation for the package, from over 70 countries. This is testament to the growing requirement for such general purpose ML tooboxes in seismology.

Such toolboxes can facilitate rapid examination of ML algorithms on curated benchmark datasets, but previous chapters have already alluded to the heterogeneity of seismic wave propagation. This property can result in major differences in seismic data across different regions. It also poses challenges when deploying ML tools in practice to new seismic environments. With data-driven techniques so strongly dependant on the training data, practical performance of ML algorithms on new out-of-sample data may still differ when applied to new regions - even when trained using millions of labelled examples. This is an open question in the research community, and practical investigations of the latest ML tools are to-date an active area of research. The next chapter will focus on this question, where we again benchmark and deploy some of the latest ML algorithms (often using the high-level API provided through the SeisBench toolbox), but here we are evaluating their performance in a completely new environment, containing highly active seismicity following the M_w 6.4 2019 Durrës earthquake in Albania.

6 Combining machine learning components for event detection: A case study in Albania

This chapter combines a selection of the latest methods in seismic event detection to analyse an 18-day period of aftershock seismicity for the M_w 6.4 2019 Durrës earthquake in Albania. We test two phase association-based event detection workflows methods, the EarthQuake Transformer (EQT) end-to-end seismic detection workflow [86], and the PhaseNet picker [159] with the Hyperbolic Event eXtractor associator [148]. Both ML approaches are benchmarked against a data set compiled by two independently operating seismic experts who processed a small subset of events of this 18-day period. Both ML methods demonstrate good generalization performance when applied to new regions with the larger catalog (PhaseNet & HEX) achieving a magnitude of completeness of $\sim$1. By relocating the derived catalogs with the same minimum 1D velocity model, we calculate statistics on the resulting hypocentral locations and phase picks.

We find that the ML methods yield results consistent with manual pickers, with a bias that is no larger than that between different pickers. The achieved fit after the relocation is comparable to that of the manual picks but the increased number of picks per event for the ML pickers, especially PhaseNet, yields smaller hypocentral errors. The number of associated events per hour increases for seismically quiet times of the day, and the smallest magnitude events are detected throughout these periods, which we interpret to be indicative of true event associations. Such practical pipelines will serve as a useful comparison when used in conjunction with the results of previous chapters' work on designing and benchmarking solutions to the seismic event detection problem.

Elements of this work form part of the Masters Thesis of Van der Heiden (VH) [141]. Denoting contributions, JW (thesis author) implemented the automated event detection pipeline, including both picking and association components. VH extracted the matching events from the catalogs and determined the minimum 1D velocity model for the region; JW analysed the matched event locations, writing the results and discussion within this chapter.

6.1 Introduction

6.1.1 The seismic event detection problem: How is it tackled?

Throughout recent years there has been an explosion in interest in the application of Machine Learning (ML) methods in seismology. Driven by increases in computational storage and compute capacity, these techniques are proving effective in solving a variety of tasks across the field. One such area which has received a particular interest is the task of seismic phase and event detection. Numerous approaches utilising ML are now attempting to detect seismic events at ever lower-signal to noise ratios. With this task forming a fundamental step to many seismological workflows, even minor increases in the event detection rate could greatly affect the subsequent understanding of any underlying physical-processes.

As the detection of seismic events is an intrinsic point of many workflows in seismology, numerous methodologies have evolved for performing this task over decades. These varying approaches to detect the coherent energy of seismic events can be typically split into three categories: migration-based, cross-correlation-based, and phase association-based methods. The first two groups operate directly on time series, compromising either the (usually) filtered waveforms or characteristic functions, with the final group operating on pre-determined arriving phase information.

Migration-based approaches detect events through coherency of the seismic wavefield, back-propagating waveforms or characteristic functions derived from them to find the focusing energy point in time and space as the source location and origin time [52, 42, 28]. Whilst this approach enables the detection of smaller magnitude events due to the underlying stacking process, it is computationally expensive. Depending on the frequency range used and whether oscillatory waveforms, or unsigned characteristic functions are back-projected, this can also impose strict demands on the quality of the assumed velocity model. Additionally, if the seismicity rate increases, distinguishing between multiple events and their associated P- and S-wave radiated energy becomes challenging because many secondary maxima can occur due to interference of waves from different events, making it hard to understand which maxima correspond to real events.

Cross-correlation-based event detection routines use the similarity of waveform signature to cluster waveforms belonging to the same seismic source region into groups. The predominant technique is template matching [36, 126] where a correlation function is used as the similarity metric, but more varied approaches using the same concept also exist [102, 88, 79]. However, due to bias towards templates of existing seismicity, events at 'hot-spots' can be detected well but the analysis might be blind to earthquakes not located in already known clusters.

Phase association-based event detection involves splitting the event detection pipeline into two distinct stages, highlighted in Figure 6.1. Firstly, the impulsive onsets of seismic phase arrivals

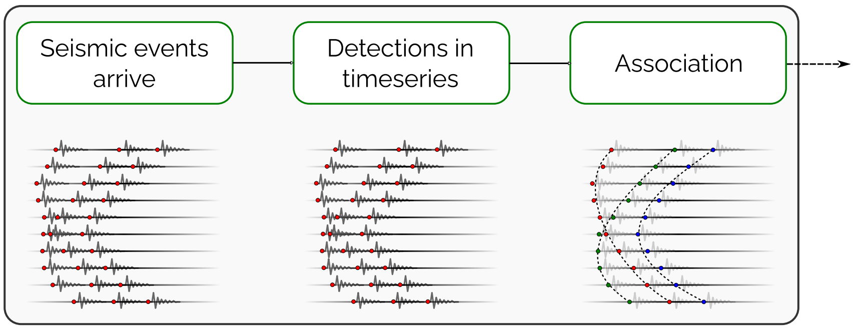


Figure 6.1: Schematic overview of a typical event detection pipeline.

are detected (the seismic picking stage, Figure 6.1) and classified via wave type (e.g. P- or S-wave). These independent detections of seismic energy are then correlated to their underlying source (the association stage, Figure 6.1). The association step is non-trivial, as other sources of seismic energy arriving in the continuous data can result in false picks, complicating the association process.

In this work we focus on the this specific approach, comparing the relative merits of phase association-based event detection workflows for analysing continuous data from a temporary aftershock deployment.

For many years, due to the heterogeneous nature of seismic wave propagation, the most accurate method was for a human expert to perform the picking stage. The natural drawbacks of having to employ a human expert is that the task becomes time-intensive, rendering the approach impractical in the current era of extensive data [70]. In contrast, automated phase association pipelines offer orders of magnitude faster processing speeds, historically at the expense of event detection and location accuracy.

In terms of methods, again there are a wide variety of techniques proposed. Automated picking algorithms can encompass traditional characteristic function-based approaches, applied for decades in real-time detection pipelines [4, 3, 6, 75], but more recently, deep learning based picker have emerged as the leading automated picking method [159, 111, 86, 129]. Deep learning routines are typically trained on millions of labelled phase examples to automatically infer the characteristic properties of seismic phase onsets. The latest deep learning routines now display accuracy levels similar to - or even exceeding - the performance of a human expert [93]. These methods are massively parallelized, and can be applied via GPU architectures, enabling rapid processing. They are also able to detect significantly more picks compared to traditional approaches [147].

Any increase in the phase detection rate renders the second component of an event association pipeline more difficult. Ignoring the potential for false picks, the simple power-law scaling of the Gutenberg-Richter relationship highlights that, as the minimum detectable event size decreases, orders of magnitude more events should be detected. Due to this factor, compute intensive phase association algorithms (e.g. backprojection-based, or cross correlation-based methods) will require significant compute time if they are to associate smaller magnitude events - which are key for enhancing the physical understanding [113]. This problem is further exacerbated by the increasing number of sensors present in the latest seismic deployments.

To cope with increased level of information to correlate, there has also been a recent influx of novel association algorithms, which have been designed with scalability in mind. These can incorporate traditional ideas combined in a new way e.g. Rapid Earthquake Association and Location (REAL) [153], a hybrid approach which takes a typical waveform backprojection methodology and applies it in a 'sparse' way through using the coherency of detected phases instead of the waveform itself. There are also a more prominent general suite of techniques being applied to tackle the scalability problem; ML-based methods, which can utilise the information contained within extensive datasets to perform inference. Newly proposed phase association methods utilise a variety of ML techniques, from graph-theory [80], Bayesian Gaussian Mixture Models for unsupervised clustering [160], recurrent neural networks [112], and also RANdom SAmple Consensus (RANSAC) [32], a data-driven ML technique to fit a parametric model to a data distribution [148, 158]. These techniques aim to improve both accuracy and performance of the phase association task, decreasing the minimum threshold of detectable events given the new conditions of densely recorded picks in time and space.

Accurate event association algorithms are therefore a key tool for seismologists aiming to better image and interpret the processes occurring within the subsurface. This work analyses 18 days of continuous data from a short period aftershock seismic network following the M_w 6.4 Durrës event. We apply a selection of the latest ML-based methods for both seismic picking, and phase association to test how they perform associating a physical aftershock sequence recorded over a dense local array of recording instruments. Our analysis tests the PhaseNet [159] and EQ-Transformer (EQT); [86] pickers. The EQT implementation also comes packaged with its own associator. This associator, along with the Hyperbolic Event eXtractor (HEX) [148] approaches are integrated into the detection pipelines to associate events. All detection pipelines are benchmarked against the manual event picks of two seismic experts who worked independently to analyse randomly selected events of the aftershock sequence. Such a workflow is similar in scope to [20], where we seek to evaluate the potential improvements ML algorithms hold over traditional methods for event detection.

6.1.2 Seismic setting

The continental collision of the Adriatic micro-plate with Eurasia generates the compressional tectonic structures observed throughout western Albania. Faulting structures in the region have been highly active, with notable events recorded in historical records. The largest of these being

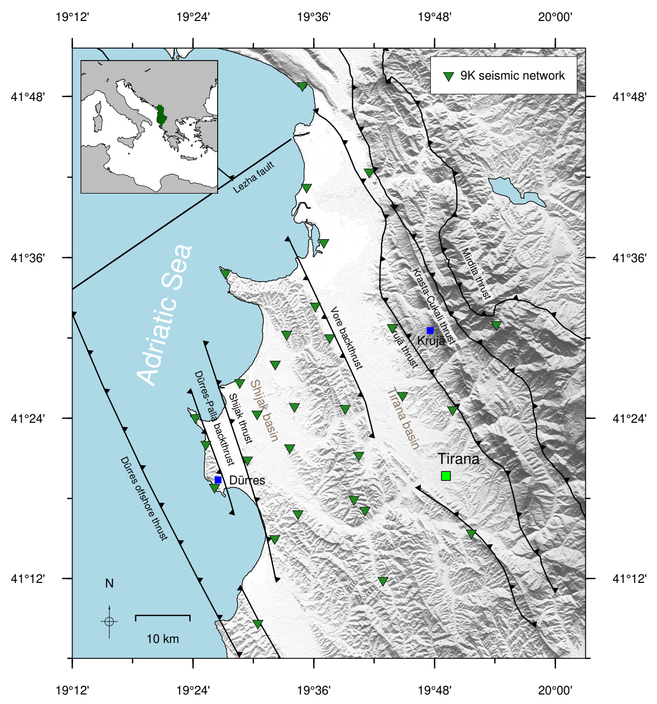


Figure 6.2: Seismic deployment following the M_w 6.4 Durrës earthquake (2019-11-26). Temporary network shown with inverted triangles; stations are mix of 1 Hz and 4.5 Hz instruments. The main faulting structures are highlighted, geo-referenced from [136, 142]. The Albania region is higlighted in the inset map in green.

.

the 1979 M_w 6.9 event occurring ~100km north of the coastal city of Durrës [8]. In recent years, the occurrence of numerous moderate-large magnitude events (2016, 2017, 2018) indicated a re-activation and changing stress dynamics within these faulting structures. The M_w 6.4 mainshock ruptured ~10 km north Durrës at a depth of 24 km [35] on 26 November 2019.

The Geoforschungs Zentrum Postadam (GFZ) and Karlsruhe Institute of Technology (KIT) in conjunction with Polytechnic University of Tirana deployed 30 3-component short-period seismic instruments (a mixture of 1Hz and 4.5 HZ sensors) to record the aftershocks sequence (Figure 6.2; Schurr et al., 2020). Stations were deployed from approximately two weeks after the mainshock. Stations were deployed close to houses to facilitate rapid deployment and ensure security of the recording devices which also means that an increased noise level had to be accepted. All stations were operated offline and the data was collected in regular visits of about 3 months, which became more complicated as the COVID-19 pandemic took hold in Europe.

6.2 Methodology

6.2.1 Event detection methods

To benchmark our various approaches, all methods are applied on the same period of continuous waveform data, 2019-12-13 to 2019-12-31. This covers an ~18-day period directly following the deployment of the first instruments. With the first instruments from the temporary network only recording from 16 days after the mainshock, the recorded data does not capture the aftershock seismicity occurring immediately after the main rupture.

The comparison of workflows for processing the 18 days of continuous waveform data are displayed in Figure 6.3. In total, there are three stages to our event association workflow: picking, association, and location. We apply three workflows, one combined manual approach, and two automatic approaches, resulting in three independent event catalogs over the analysis period. For the manually analysed control data set we applied a standard STA/LTA (short time average over long term average) arrival and event detection procedure as outlined in [107]. The main reason behind this was to quickly compile a basic event list from which about 300 events were selected for manual analysis in order to gain a quick impression of aftershock activity and maybe identify the main causative fault. The magnitude range of selected events is roughly between M_L 1 and 4, providing a good range of magnitudes for the benchmarking exercise. For the automated procedures, our choice of algorithm for the association step was motivated by the results from the picking stage. Below, we first state the overall performance of the different picking methods.

The total number of picks for each method are displayed in Table 6.1. In total, analyst A picked 9,761 total phases, and analyst B picked 2,980 phases; for the automated approaches, EQT [86] picked 78,778 phases, and PhaseNet [159] picked 561,326 phases. The total number of picks be-

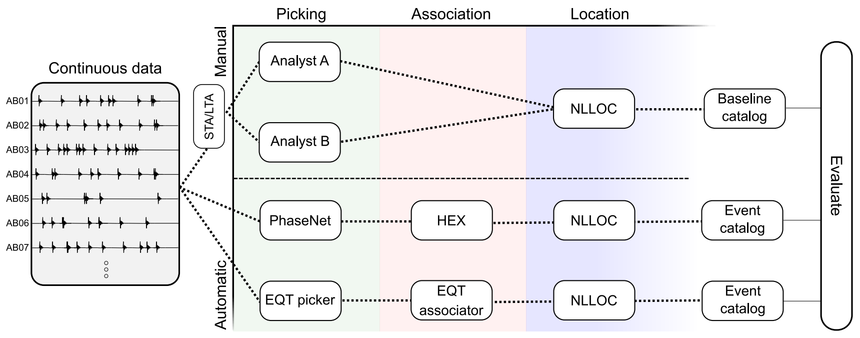


Figure 6.3: Outline of workflows for generating seismicity catalogues for comparing automated and manual approaches. All derived catalogs are evaluated against one another. See text for references to the different methods.

Table 6.1: Picking statistics following the application of the various manual and automated approaches in processing all continuous waveform data over the recording network from 2019-12-13 to 2019-12-31. Values denote the total number of picks made in the first stage of the event detection pipeline, before any association stage.

Parameter	Manual		Automatic	
	Analyst A	Analyst B	EQT	PhaseNet
P-picks	5,758	1,680	39,918	289,509
S-picks	4,003	1,300	38,860	271,817
total	9,761	2,980	78,778	561,326
#P/#S	1.438	1.292	1.027	1.065

tween manual approaches and automated approaches cannot be compared directly, however, as the manual picks these were not picked in a comprehensive manner. Both EQT, and PhaseNet output probabilistic Characteristic Functions (CFs), which are a proxy for the probability of the occurrence of each phase. Picks are obtained from the output CFs by defining a cut-off value past which the sample of the maximum will be set as a pick onset. This threshold is a user-definable input parameter, for both picking algorithms we use the default value from the original publications.

As the EQT pick catalog contains almost an order of magnitude less picks than the PhaseNet approach, irrespective of which cut-off probability threshold used, EQTs simple built-in association routine can deal with this level of pick information (one pick arriving on average every 20 s across the network), whereas we use the machine learning-based HEX routine [148] to associate the greater number of pick information generated by PhaseNet (one pick arriving on average every 3 s, across the network) In the final stage of each workflow the probabilistic location software NonLinLoc (NLLOC) [76]) is applied to locate any associated events.

As the goal of this study is to verify the quality of automatic event catalogs made with the latest ML methods, any events located by each approach should, therefore, exhibit well-located hypocentres. To ensure all compared events are well-located, all final catalogs are filtered according to the following criteria. The minimum acceptable number of phases for a location are set as 6 for P-phases, and 4 for S-phases. As poor S-phase constraints are known to lead to incorrect focal depth estimates [40], for the relocation step, we only relocate events with phase information satisfying the following criteria [43]; azimuthal gap $< 180°$ (e.g. [58]); at least 1 S-wave arrival must be at an epicentral distance less than the focal depth (e.g. [17]).

6.2.2 Evaluation

We first compare the results of the two manual event catalogues against one another to see how they differ amongst themselves. This provides an assessment of how the baseline catalogs vary between one-another acts as a reference point of consistency for which the automatic procedures can be evaluated against. Following this comparison, the two manual event catalogues are merged to one combined manual event catalogue (without duplicates). Merging of matched events from manual analysts is done as follows: first, all picks and phases from both analysts are merged into a single combined manual catalog; then duplicated events are determined as events from each respective manual analyst with an onset time within 5s; finally, we fit an linear trend through the combined catalog events RMS relocation residual and total number of detected phases. As residual is strongly correlated to an event size and moveout across the network, factoring in this trend allows for determination of the 'better-quality' events when comparing matched events (discussed further in section 6.3.3). The event with the smallest 'corrected' RMS is then taken as the event to use. Only the phases of the 'best quality' manual events are then used in the combined catalog. This combined manual catalog is then compared to the EQT and PhaseNet catalogs (top workflow, Figure 6.3).

The quality of all automatic event locations are assessed by comparing against this suite of 'base' benchmark events. Evaluating the latest events association methods against this approach provides a relative estimate against the 'traditional' state-of-the-art, when performing event association in practice. Once the varying manual and automated events catalogs are obtained, the workflow to pairwise compare events between catalogues is the following:

1. Find commonly detected events in both catalogues (using origin time). We consider events with an origin time difference of less than 2 s to be the same event.

2. Find common phase arrivals between pairs of matched events (using station and phase information) and calculate their residuals.

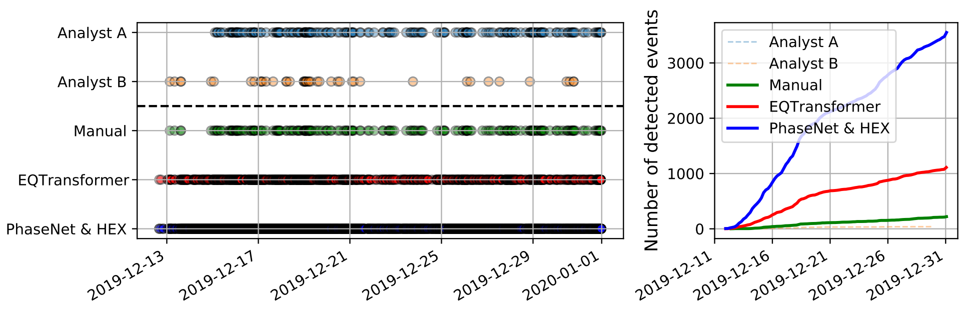


Figure 6.4: Event distribution in time. On the left each event is marked by a point, on the right the growth of cumulative number of events is shown.

Table 6.2: Comparison of the statistics of various automatic and manual event catalogs. Compared statistics are related to the matched events only and include: the number of matched events; the detected picks within matched events, and the mean ratio of the number of P picks and the number of S picks per matched event. For the manual analyst comparison, percentages are stated as fraction of picks of analyst A. For the automatic vs. combined manual catalog comparisons, the percentages are stated as a fraction of the combined manual baseline catalogs picks. EQT. denotes the EQTransformer event detection method; PhN. denotes the PhaseNet & HEX detection method; An. A and An. B denote Analyst A and analyst B, respectively; Man. denotes the combined manual baseline event catalog.

Comparison	Manual			EQTransformer			PhaseNet & HEX		
Catalog	An. A	An. B	match	Man.	EQT.	match	Man.	PhN.	match
events	209	43	33	219	1,110	206 (94.06%)	219	3,351	218 (99.54%)
picks	1,318	1,640	1,278 (96.97%)	7,799	7,892	6,494 (83.27%)	8,221	11,107	7,937 (95.90%)
#P/#S	1.41	1.18		1.36	1.05		1.35	1.09	

3. Analyse the comparison statistics of matched events and phases.

We determine events with an origin time difference of less than 2s to be the same event.

6.3 Results

6.3.1 Benchmarking automated event detection workflows

The overall event detection results over the benchmarking period are displayed in Figure 6.4, and Table 6.2. From the combined manual baseline catalog, the EQT approach detects 94.1% of the manual events, and the PhaseNet & HEX approach detects 99.5 % of the manual events. EQT finds 1,110 events in total, with PhaseNet & HEX finding 3,551 events (Figure 6.4).

Table 6.3: Comparison of statistics for common events between catalogs. For the manual events, the differences are calculated as the derived results from Analyst A compared against the results of Analyst B. Subscripts 'eqt', 'phn', 'base' refer to EQTransformer, PhaseNet & HEX, and manual baseline, respectively.

Comparison	Manual		EQTransformer		PhaseNet & HEX	
	μ_{A-B}	σ_{A-B}	$\mu_{base-eqt}$	$\sigma_{base-eqt}$	$\mu_{base-phn}$	$\sigma_{base-phn}$
Depth (km)	-0.2530	0.5764	0.2146	0.6235	0.0161	0.4339
Latitude (km)	-0.0046	0.0031	-0.0005	0.0030	-0.0007	0.0028
Longitude (km)	-0.0006	0.0056	-0.0010	0.0045	-0.0005	0.0034
RMS (s)	-0.1148	0.0517	-0.0033	0.0576	-0.0243	0.0496
number of picks	-10.9394	5.1109	-0.2233	10.0491	-13.0229	7.9616
number of P picks	-3.2424		2.1456		-4.8211	
number of S picks	-6.5455		-2.5971		-8.4174	

6.3.2 Statistics of matched events and phases

The results of the comparison routine are shown in Table 6.3, where we calculate statistics of the residual differences of matched events between catalogs, determined for a set of common event parameters. For example, the mean number of P-picks for the PhaseNet vs. manual baseline comparison ($\mu = -4.8211$) means that on average, PhaseNet picks 4.8211 more P-picks when comparing matched events of the PhaseNet catalog with the manual baseline catalog We calculate statistics for the following parameters; depth, latitude, longitude, Root-Mean-Square location error (RMS), and the Number of Picks (NPS). We compare the events detected by both manual analysts against one another, to provide a rough reference point for the consistency of the manual seismic phase picking.

For all matched events, the intersection of common picks are then examined. Once common picks are identified between catalogs, the phase pick residuals can be determined, The statistics of pick differences are displayed in Table 6.4 and Figure 6.5. Residuals are calculated for arrival time in seconds, pick weight, and epicentral distance to the station where the phase was picked in kilometers. Again, all metrics describe the nature of the residual distribution when comparing matched phases in both catalogs. Traveltime denotes the difference in pick onset time. The underlying distributions for the traveltime residual comparison of matched phases are displayed in Figure 6.5.

The PhaseNet & HEX, and EQT association approaches display an average difference in pick onset of 0.038, and 0.043 s, respectively, i.e., they pick slightly earlier than the baseline catalogue, which is dominated by the picks from analyst A. This average difference in pick onset between automated approaches and the baseline manual catalog is smaller than the mean difference between the two analysts, where analyst B seems to have picked 0.112 s earlier on average (this estimate is

Table 6.4: Phase pick difference for matched phases. For the 'Manual' column the matched phase picks of Analyst A are compared with those of Analyst B; for 'EQT' and 'PhaseNet' columns, the phase picks of EQTransformer and PhaseNet are compared against the manual baseline catalog, respectively. 'NLLoc weight' refers to the difference in pick weighting assigned in the NLLoc relocation procedure for matched picks.

Comparison		Manual		EQTransformer		PhaseNet & HEX	
	phase	μ_{A-B}	σ_{A-B}	$\mu_{base-eqt}$	$\sigma_{base-eqt}$	$\mu_{base-phn}$	$\sigma_{base-phn}$
Travel time (s)	P	0.112	0.118	0.043	0.322	0.038	0.269
	S	0.058	0.268	0.064	0.483	-0.006	0.347
NLLoc weight	P	0.087	0.134	-0.066	0.156	-0.102	0.133
	S	-0.129	0.165	-0.002	0.196	-0.027	0.177

based on a very small number of events, though). However, the standard deviation of the pick time differences with respect to the manual baseline is larger for the automated methods (0.270 s for PhaseNet & HEX, and 0.321 s for EQT) when compared against the variation observed between different analysts (0.122 s). So, whilst both automated approaches detect more picks per event (Table 6.3), these picks have potentially larger associated uncertainties compared to a manual analyst.

6.3.3 Matched event locations

For evaluating the location consistency, the differences between the manual analysts can again provide a baseline metric which we can compare the automated catalogs against. The locations of compared event catalogs are displayed in Figure 6.6. When comparing the 209 events of Analyst A and 43 events of Analyst B, 33 events are matched, detected across both event catalogues. These intersecting events differ in epicenter by 705 m on average (left panel, Figure 6.6). For the 206 matched events from the EQT catalog (94.1% of the total baseline events), these locations exhibit an average epicenter difference of 424 m when compared against the baseline catalog (centre panel, Figure 6.6). Finally, 218 (99.5%) of the base events are found using PhaseNet and HEX; locations show a difference of 377 m in epicentral distance on average (right panel, Figure 6.6).

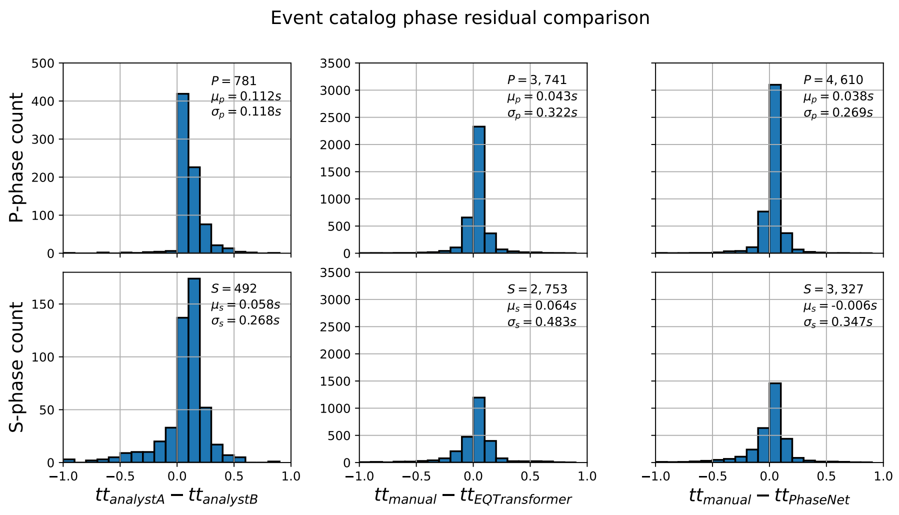


Figure 6.5: Phase arrival differences for matched phase picks of the matched events. The 'manual' comparison of Analyst A vs. Analyst B is the left-column, the EQTransformer vs. manual baseline comparison is the central column, and the PhaseNet & HEX vs. manual baseline comparison is the right column.

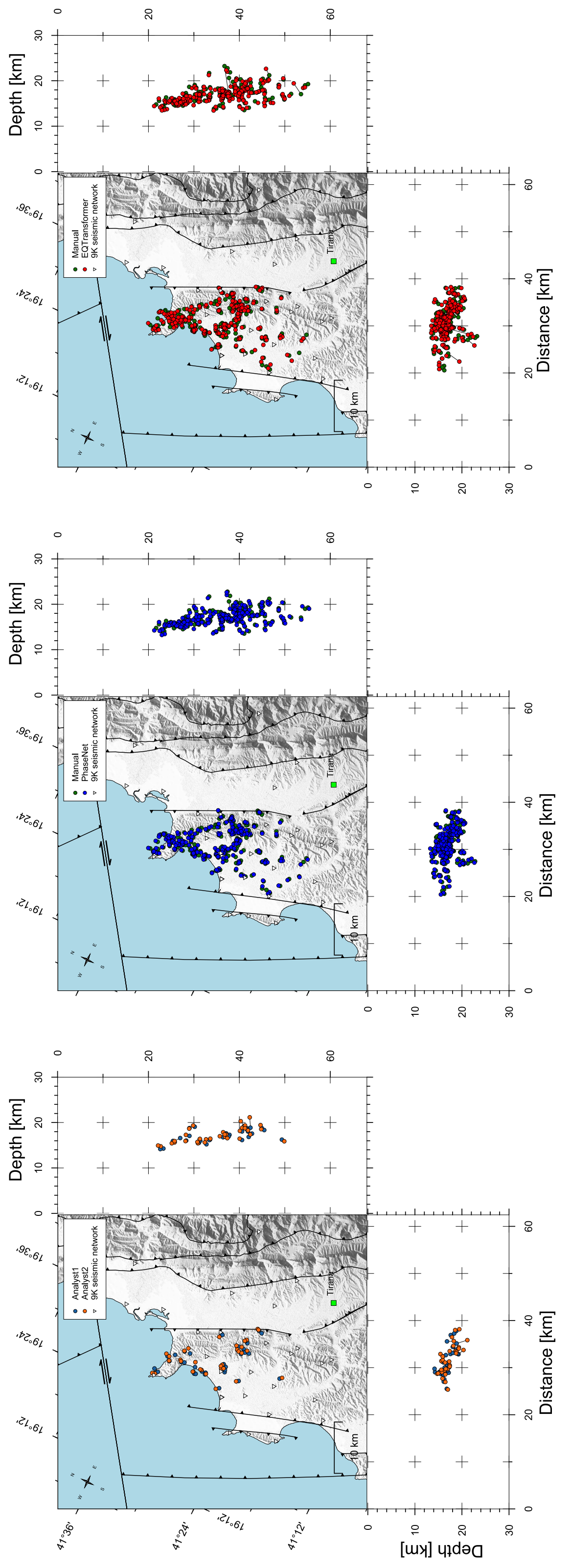


Figure 6.6: Matched event locations determined by Analyst A and Analyst B in the left panel; manual baseline and EQTransformer matched events in the centre panel, and manual baseline and PhaseNet & HEX matched events in the right panel.

Figure 6.7 displays the travel time residual (RMS) plotted as a function of number of picks (NPS) for matched events in each of the event catalog comparisons. Generally, we expect the RMS to increase with an increase in associated number of picks due to larger numbers of picks made at varying distance ranges incorporating higher pick uncertainties. This is easily visualised with the comparison between manual analysts. Analyst B picked more phases than Analyst A for every event, resulting in a larger RMS for every event, and is indicated by the positive slope of connecting line in left panel of Figure 6.7. This signifies how we cannot simply take the standard RMS residual values as a 'gold standard' of uncertainty, as there are other contributing factors influencing the RMS. For example, not accounting for the 3D velocity perturbations in the relocation procedure could potentially account for the increase of RMS as a function of the number of picks; furthermore, picks at larger distances, more frequent for the automatic pickers, might suffer more from this.

The automated EQT approach finds events with a similar RMS and number of associated picks to the manual baseline catalog (see central panel in Figure 6.7). EQT makes ∼0.2 more picks per event on average (Table 6.3). In terms of location RMS, events located by the human analysts range between RMS 0.06 to 0.37 s, and for EQT event RMS ranges between 0.06 and 0.44 s (Figure 6.7). Results for the Manual and PhaseNet & HEX approach comparison are distinctively different. Here, we see that the majority of PhaseNet events have a RMS ranging between 0.11 and 0.31 s, and the number of picks per event ranges from 38 to 58. On average, the PhaseNet & HEX method associate ∼13 more picks per event than the human analysts, increasing the RMS by 0.024 s on average (Table 6.3).

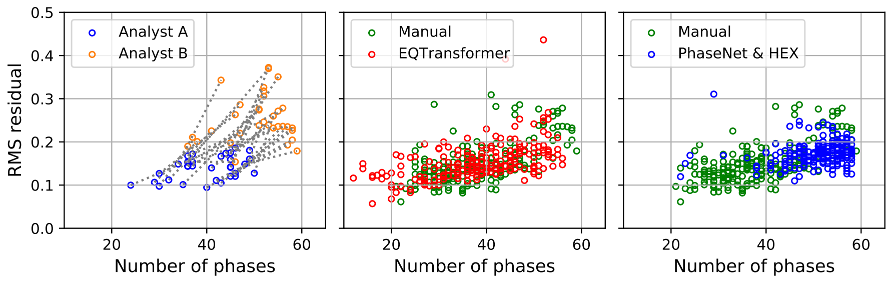


Figure 6.7: Travel time residual (RMS) as a function of number of picks for matched events from the manual Analyst A vs. Analyst B comparison (left), EQTransformer vs. Manual baseline catalog comparison (center), and PhaseNet & HEX vs. manual baseline catalog comparison (right).

6.4 Discussion

6.4.1 How do ML-based event detection pipelines compare to manual detection methods?

The results of the benchmarking analysis indicate that the latest ML event detection methods at least match the reliability of individual analysts for individual events, while potentially being able to analyse very large number of events. More specifically, improvements do not only relate to greater numbers of event detections, but the ML detectors can identify a greater number of phases per event, with the ability to pick at larger distance ranges with relatively minor increases in RMS residual. Both end-to-end automated workflows in the EQT and PhaseNet & HEX approaches take continuous seismic data streams, and return ∼5x and ∼16x more events, respectively, than our manually analyzed catalog. 219 events is a typical number of manually analyzed detections for temporary aftershock campaigns which can be quickly processed over a feasible time period. For a manual analyst to reach the number of detections of the ML workflows, this would be a significant, time-intensive undertaking. ML detected events are well-recorded across the seismic network, returning accurate event locations which can be used in subsequent analysis. Such results are in line with other works now applying ML tools to process seismicity [135, 20], where ML are displaying significant performance improvements over traditional methods. With the increasing popularity of ML event detection components, our results may also help inform future researchers who are seeking to apply such techniques in and end-to-end fashion. Within the community, toolboxes are becoming available to streamline this task [154, 150], testament to their growing usefulness.

Our investigation focuses on benchmarking a selection of fully integrated event detection pipelines, Crucially, the ML workflows presented here require no a-priori knowledge of the subsurface velocity structure to make these accurate detections. As the Albania region is one such area without a well-constrained velocity structure to date, we can see how applying such end-to-end detection workflows will greatly enhance understanding of the seismic structure of yet unexplored regions. This is especially true for the phase association problem, as whilst there might be room for performance increases by incorporating additional information like a region-specific 1D or 3D velocity model; however, the drawback of these approaches are that you are relying on a-priori knowledge, which might not be available for new regions or only of poor quality. The full PhaseNet & HEX event catalog from this work has been used in performing a new 1D velocity model inversion for the Albania region, using only results of the automated detection methods [141].

One remaining question to ask is: Are all of the automatically detected aftershocks real events? To answer this question we investigated in more detail the obtained catalogue by looking into the magnitude and the time distribution of the events. Firstly, we compute magnitude estimates

(M_L) for all the events in both the manual, and the PhaseNet & HEX catalog (Figure 6.8), using the calibration for the Albanian region of [89]. We estimate magnitudes for all the events detected throughout the analysis period, so in this case, the final quality filtering step is not performed. This is as the magnitude calculation only very weakly depends on event depth, so here, we do not need such strict criteria, boosting the total number of associated events to 270 for the manual approach and 5,548 for the PhaseNet & HEX approach. The PhaseNet & HEX automatic catalog is determined to be complete above M_L 1, with the manual catalog complete to M_L 2.5. We note that the magnitude of completeness for the manual catalog should not be used in a direct comparison with the automated procedures, as the manual events were not picked in an exhaustive manner. It does, however, indicate what sort of completeness can be achieved when aiming to perform a standard manual analysis of an aftershock sequence, in a reasonable period of time.

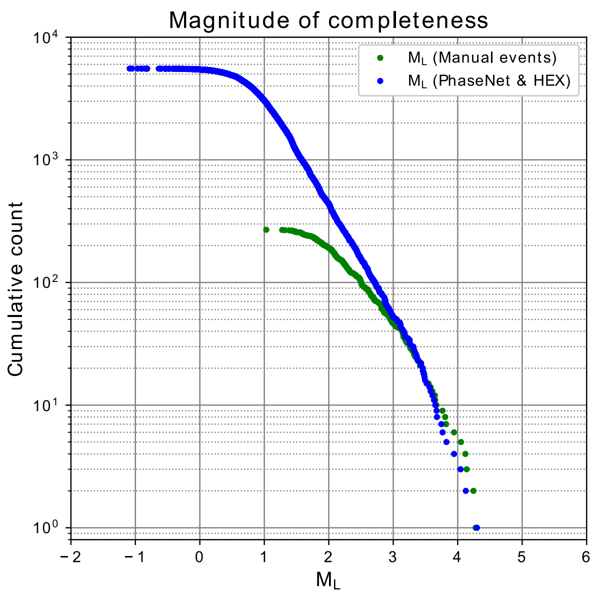


Figure 6.8: Frequency-magnitude distribution for the PhaseNet & HEX detected events and manually detected events over the benchmarking period. The final quality location filtering has not been applied here.

Diurnal variation of detected events Secondly, we investigate the dependence of the number of detected events on the time of day in the benchmarking period. Figure 6.9 displays the number of detected aftershocks throughout the day for both ML-based methods. We also overlay the average amplitude of background noise (between 1 to 10 Hz) for each respective 3-component channel (HHE, HHN, HHZ). A clear diurnal pattern is observed. The noise estimation curve shows background noise increasing during the day and decreasing at night. As the seismic network is in the vicinity of populated areas, we can relate this to anthropogenic activity.

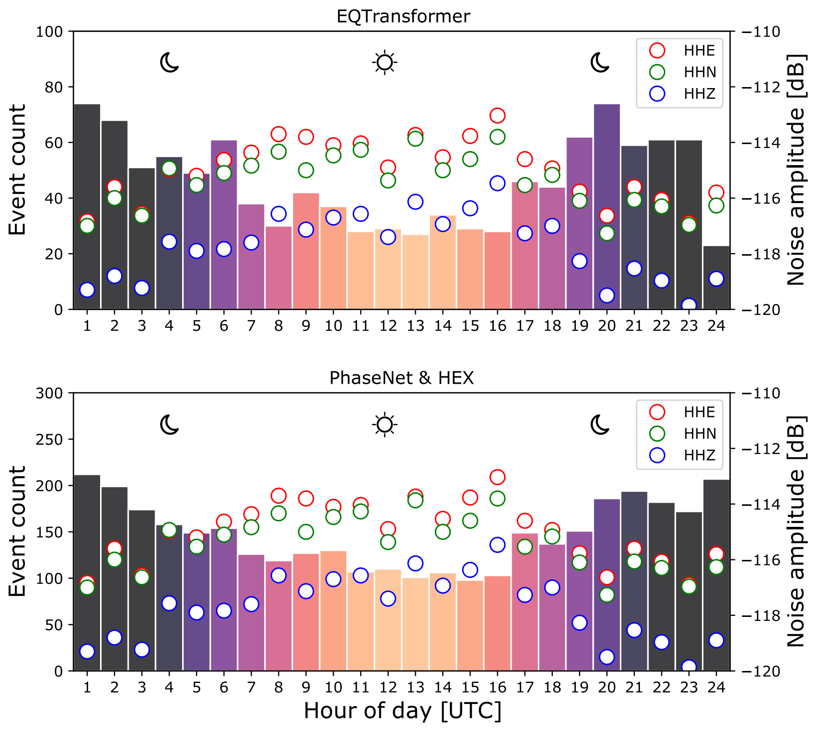


Figure 6.9: Number of detected aftershocks per hour in the day for both ML-based methods overlaid by the 3-component network noise in 1-10 Hz band.

If the workflows were wrongly associating noise, we would expect the distribution of aftershocks throughout the day to match the daily background noise distribution, with more events in the middle of the day, during periods of higher background noise. For true associations, the expected behaviour would be to detect larger numbers of the smaller magnitude aftershocks during the night when the noise level drops (see the noise estimation point for components in the edge portions of Figure 6.9). The number of associated aftershocks through the day exhibits the latter behaviour, indicating true associations (see the hourly event count bars in Figure 6.9). Such a time-of-day dependency on the number of detected events is well-known from the manual analysis. This pattern is further highlighted when plotting the magnitude of events through time (Figure 6.10). Here we can clearly see the local minima of detected event magnitudes roughly aligning with the occurrence of midnight.

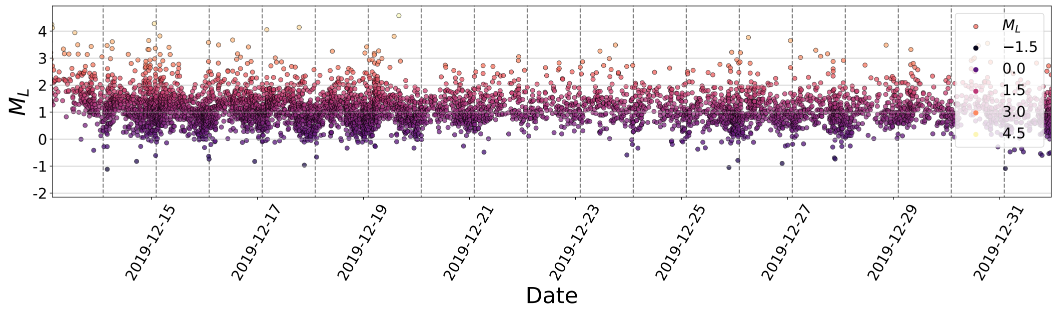


Figure 6.10: Magnitude-time plot of detected events over the benchmarking period for the PhaseNet & HEX detection workflow.

6.4.2 Contrasting results when applying state-of-the-art ML models to new regions

Both methods are making true associations, uncorrelated to the background noise level but, a surprising result is the number of picks made by both ML methods in practice. EQT makes 78,778 picks in total, and PhaseNet makes 561,326 total picks. This result is in contrast to the latest results of benchmarking works using curated benchmark datasets [93], which determined EQT and PhaseNet to detect phases at a similar rate. One potential explanation is the length of the input window required for both algorithms. EQT requires a 60s input window, PhaseNet requires a 30s input window, but due to the nature of the CNN method along with a lack of global connections, its receptive field is $\sim$4 s long. The larger input window for EQT means that longer term temporal dependencies can be optimised for during training. When deployed in practice to a new seismic region not seen during the training stage - such as Albania - these relatively longer-term temporal relationships may now differ due to the natural heterogeneity of wave propagation. Another way in which the learned temporal relationships could differ is due to the potentially higher seismicity rate of a temporary aftershock deployment. In such a network, simultaneously occurring events are more likely, overlapping in time to change the arriving waveform signature. These factors could potentially be contributing to the decrease in pick identifications when compared to PhaseNet, which applies a CNN with a smaller input window to detect local relationships only.

7 Conclusions

This thesis has sought to explore to what extent ML can be leveraged to solve the historical task of seismic event detection. We initially outlined the motivation behind conducting such work. With the scale of seismic data becoming exponentially larger, we outlined how ML techniques are well-suited to exploit the information contained within such datasets. To evaluate this problem, the initial chapters 3, 4 focused on designing and implementing novel solutions to both identify and associate seismic phases. Our initial experimentation was benchmarked against traditional techniques to quantitatively outline the potential benefits and limitations of ML-based event detection. Later chapters then expanded upon this concept of benchmarking, introducing software to act as a reference point for facilitating ML research within the field 5. We finally deployed multiple end-to-end ML-based seismic event detection pipelines in practice in chapter 6. Here, we evaluated performance in practice to see if these latest methods performed as well as was expected from the literature summarizing performance on the latest curated datasets [93]. Again, evaluating all components against the traditional state-of-the-art of a manual expert.

We now summarize our findings in this concluding chapter. We will go through what has been learned from each respective work, commenting on any potential implications and future work related to our investigations, and from there we will conclude with a general outlook for data-driven solutions for the event detection task. We will also outline what the latest results imply for the field of seismology in general.

7.1 Deep learning for detecting seismic phases

In chapter 3 we implemented a novel CNN architecture for detecting the presence of seismic phases. This study was conducted with the algorithm trained on a relatively small dataset of $\sim$ 11,000 P- and S-phase pairs. All training examples were manually labelled following the occurrence of a moderate magnitude seismic event in northern Chile. We showed that even in the absence of extensive training data, ML methods could still outperform traditional automated seismic picking techniques in new regions. CNNs are a natural fit for detecting the presence of seismic phases, as the arrival of seismic energy at a seismometer is associated with local changes in frequency content and amplitude of a seismic timeseries. These features are well-suited for detection by the convolutional kernel. A result validated by other published works applying CNN architectures for seismic picking [159, 111]. CNNs were shown to 'automatically' engineer the appropriate features for classifying seismic phases in continuous seismic streams. This was evidenced by the increased proportion of accurate S-phase identifications compared to traditional approaches such as the STA/LTA. With S-phase identifications often masked in the coda of later arrivals, manually defined 'characteristic functions' struggled to identify S-phases compared CNNs. Such results have huge implications for improving not only the number of picks able to be detected but also the quality of the detected picks. An increased number of S-phases is vital for improved relocations of seismic events [40]. This was evidenced by the increased resolution of hypocentral relocations using our CNN approach, attributed to the improved P-/S-phase pick ratio.

Chapter 3 acted as an initial exploratory work into the potential future applications of ML pickers. The improvements in relocations associated with the higher quality picks of later phases could result in deep learning being leveraged to improve the identification of the entire spectrum of later depth phases (e.g. Pn and Pg phases). This would transform the resolution of crustal tomography studies. More generally, a trained deep learning architecture essentially consists of a set of weights and bias terms are applied to an input, so the runtime complexity of these algorithms is $\mathcal{O}(n)$. This is in contrast to other leading methods to identify seismic events, such as cross-correlation methods, which have a runtime complexity of $\mathcal{O}(n^2)$. As the number of seismic sensors exponentially increases, trained deep learning methods have an advantage over other techniques when it comes to optimizing for the computational cost of such investigations. The fact that they also either match or exceed state-of-the-art accuracy for the seismic event detection task also outlines the positive impact these types of algorithms can have when applied to phase picking.

7.2 HEX as a phase association algorithm for the latest generation of seismic phase catalogs

Following the results from chapter 3, we then focused on how to associate the increased number of seismic phases generated by the latest picking algorithms. Again, implementing a novel data-driven solution which adapts the RAndom SAmple Consensus Algorithm (RANSAC) [32] logic. We employ this logic to fit a parametric model of the seismic wavefield moveout with distance and time - terming the algorithm the Hyperbolic Event eXtractor (HEX) [148]. The RANSAC framework was primarily employed in the computer vision community for tasks such as extracting shapes from point-cloud estimation [25]. Such an approach has numerous benefits compared to other non-parametric solutions. Once a seismic event is found, we can provide a first-order approximation of the seismic wavefield propagation to new points in space. This helps boost the number of seismic phases correlated, which greatly improves the robustness of any statistics during downstream tasks such as relocation or event magnitude estimation. Synthetically stress-testing the performance of the HEX association algorithm, identified that it can still associate events when the signal-to-noise ratio of the event catalogs are up to 50 %. This is crucial for detecting seismicity in high-noise environments.

This new generation of association algorithm has been specifically designed to accurately associate the orders of magnitude larger seismic phase catalogs now being created in seismology with deep learning pickers - as shown in chapter 3. HEX was designed to operate on local to regional seismic networks. On this scale, an enhanced understanding of the physical processes associated with major earthquake ruptures can only be achieved through more detailed, well-resolved seismic catalogs that capture small-magnitude events. This will allow for the imaging of small-scale faulting fabrics [113] and will close the gap between the types of rupture processes observed in laboratory conditions [116], vs. what we see in nature.

7.3 Establishing frameworks for the research of ML in seismology

Having introduced new ML-based methods for detecting and associating seismic events, chapter 5 turns to the question of how to systematically benchmark leading techniques. Here, fields where ML techniques are more well-established, such as computer vision and natural language processing, provide a useful reference point for best practices [2]. Such general-purpose toolboxes for accessing the latest state-of-the-art research and conducting benchmarking workflows are vital to advancing the state-of-the-art in any field. Chapter 5 introduces the open-source SeisBench Toolbox [149], as a framework for performing such tasks.

The functionality introduced with the SeisBench package provides a unified interface for accessing not only state-of-the-art ML models for general tasks in seismology, but also a unified API for accessing benchmark training datasets. The detection of seismic events is a task where ML has been applied most widely to date in seismology [102, 159, 160, 111, 147, 86, 130]. Benchmarking a wide range of algorithms is a time-intensive task without such tools. Users' have to integrate a range of ML APIs in order to reproduce results. More generally, SeisBench will act as a general reference point for the ML community in seismology, as it provides a high-level interface to apply ML techniques to seismic data. This is a step-change compared to having to learn the relevant underlying deep learning frameworks and it means that practitioners, who may not necessarily be experts in ML, can also benefit from these techniques. To date, we have seen great feedback regarding the usefulness of the toolbox within the seismological community, with over 1,000 users from 70 countries. Future expansion and integrations into the toolbox will expand the range of models integrated into the framework to more general workflows in seismology. The API has been designed with this in mind, with the natural linking between metadata and raw training examples perfectly suited to both supervised, and unsupervised pipelines. New benchmark datasets can also be integrated in an extensible way, and whilst we already have over 20 million waveforms recorded in the SeisBench benchmark training datasets, data from other more varied environments is due to be integrated - from volcano seismic data to ocean bottom seismometer datasets. This will further expand the audience for the package and solidify its standing as a general-purpose ML toolbox for the community to use in aiding their research.

7.4 Benchmarking ML event detection workflows in practice

Chapter 6 then concluded with applying the latest seismic event detection components to detect seismicity in practice. Here, we tested two state-of-the-art phase association routines, the EarthQuake Transformer (EQT) end-to-end seismic detection workflow [86], and the PhaseNet picker [159] with the Hyperbolic Event eXtractor associator [148]. For both ML pipelines, we constructed a benchmarking work, comparing the derived events against a dataset compiled by two independently operating seismic experts. The reasoning for constructing such a pipeline was twofold. Firstly, the manual labels act as a reference point for the traditional 'gold standard' of accurately associating seismic events. This provides valuable insight into whether the ML pipelines display general improvements in detecting seismicity in practice - highlighting how far we have come in improving algorithms to solve this task. Secondly, with seismic event propagation so heterogeneous, the performance of seismic events associated in practice may vary largely depending on the region in where they are applied. The Albanian region has had a relative period of seismic quiescence over recent decades. It, therefore, remains relatively under-explored when compared to the training data available in many benchmark datasets (e.g. the Western United States).

Both ML-based approaches processed an 18-day subset of the 2019 Durrës aftershock sequence. From the two approaches, we found that the PhaseNet & HEX method detects 3,551 events in total, and EQT detects 1,110 events in total, compared to the 219 manually analysed events. The difference in the number of identified events between ML methods is attributed to the step change in phase detections generated during phase picking. PhaseNet identifies 561,326 total picks, with EQT identifying 78,778 total picks when applied to detect continuous data in practice. This level of automated phase detections is orders of magnitude above what can feasibly be analysed by a manual expert due to time constraints, displaying a similar level of accuracy to the manual expert. By relocating the events using the same minimum 1D model, we matched events and phases common between seismic catalogs and analysed the resulting statistics. Further analysis indicates that the occurrence of the ML-detected events is inversely correlated to the background noise level, further indicating true associations.

A pertinent finding from the practical case study of chapter 6 was that neural network architectures can potentially display varying recall when compared to their initial performance on curated benchmark datasets. As part of the publication of the SeisBench software package [150], we also conducted an in-depth systematic benchmarking of the leading deep learning seismic pickers [93]. The results of this study found that three neural network architectures performed exceptionally on the supervised classification task of seismic phase picking - across all environments and benchmark datasets. These pickers were the PhaseNet picker [159], the EQT picker [86], and the Generalized Phase Detector picker (GPD) [111]. For our practical investigation in Albania, PhaseNet displayed a higher recall rate compared to EQT in recovering picks; irrespective of the detection threshold. In total, the final PhaseNet picker found over 7 times more picks when compared against the respective EQT architecture. There could be many reasons for this difference. We attributed the lower number of detections for EQT to be due to the longer input window, where potentially longer term temporal relationships need to satisfy conditions seen during training. This is in contrast to the PhaseNet picker, which had a much shorter receptive window, and could abstract the feature extraction process starting on local features, and through nested operations incorporate more global relationships. It is worth noting that the higher recall rate may not necessarily be better, as this could lead to a higher number of false picks. It does, however, indicate a potential discrepancy between the performance of pickers across curated benchmark datasets from well-explored seismic regions, and the practical performance observed in new areas. This finding will need to be validated with other works benchmarking deep learning picking algorithms in practice.

7.5 Outlook

Let us now revisit the main question posed in chapter 1; namely, can we utilize ML to better exploit the information contained within historical catalogs of continuous seismicity to improve the resolution of seismic datasets. Our aim was to highlight and explore how well-suited data-driven methods are for this task. We can summarize our key findings and outlook under 4 key themes.

◇ Machine Learning can greatly improve performance when detecting seismic events.

As highlighted in chapter 3, we have shown that ML techniques can improve recovery rates for the task of seismic phase picking. This agrees with case studies from the literature also exploring this task, recovery of seismic events can be $\sim$ order of magnitude larger [102] when using deep learning over traditional automated techniques. Operating at lower signal-to-noise ratios and able to reliably pick phases in the presence of coda of other arrivals, deep learning pickers can transform the resolution of seismic arrival catalogs, which has huge implications for the processing of all downstream tasks which operate on these data. They also offer better runtime performance than the quadratic scaling of other leading methods such as cross-correlation. This means that event detection can be performed across large-scale regional seismic networks without the associated cost in compute time and resources that would of been required when using template matching [113].

We have also shown how ML can be leveraged to deal with the exponentially increased amount of data generated by these latest techniques. Again, data-driven methods are a natural solution to these sorts of problems, as they offer significant run-time performance improvements over more traditional methods such as backpropogation, but they are also built to withstand higher levels of noise. This factor is crucial when exploring new seismic environments, or for temporary seismic installations where anthropogenic noise can contaminate the recordings of seismic arrivals.

◇ Leveraging unified toolboxes for benchmarking and advancement of the state-of-the-art.

With the seismological field now starting to widely exploit ML solutions for a range of seismic tasks, there are a myriad of practical workflows emerging applying new models to novel datasets and then commenting on their respective performance. This sort of exploratory investigation is great when first seeking out the best techniques used to solve problems, but once past this exploratory phase, refinement of leading models and algorithms should be conducted in a more systematic way. Taking examples from other fields where ML has been well-established (e.g. NLP, computer vision), both benchmark training data, and the use of toolboxes to standardise operations is crucial for advancement of the state-of-the-art. We have presented our own packages in this thesis for performing such tasks in seismology. The key ideas are to streamline the development process, and reduce barriers of entry for new researchers into the field. Toolboxes such as the one presented within this thesis act as a general reference point for the community as a whole, and will facilitate rapid benchmarking for leading models, even to more general tasks outside the scope of seismic event picking and detection e.g. seismic signal denoising, source parameter estimation etc.

◇ The need for practical pipelines to augment benchmarking works

With systematic benchmarking pipelines, there may be a tendency to focus on any ’in-sample’ data, which has already been recorded and labelled in some benchmark dataset. This can lead to biases in the modelling process, which in turn can lead to overfitting and poor generalization performance. To further now explore how the next generation of event detection components can be integrated into analysis pipelines and research workflows, a number of practical case studies are required, applying these models to new environments, under different noise conditions to objectively assess performance e.g. [135].

Bibliography

1. Martın Abadi et al. “Tensorflow: A system for large-scale machine learning”. In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, pp. 265–283.
2. Alan Akbik et al. “FLAIR: An easy-to-use framework for state-of-the-art NLP”. In: *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 2019, pp. 54–59.
3. Rex Allen. “Automatic phase pickers: Their present use and future prospects”. *Bulletin of the Seismological Society of America* 72:6B, 1982, S225–S242.
4. Rex V Allen. “Automatic earthquake recognition and timing from single traces”. *Bulletin of the seismological society of America* 68:5, 1978, pp. 1521–1532.
5. AlpArray Seismic Network. *Eastern Alpine Seismic Investigation (EASI) - AlpArray Complimentary Experiment*. 2014. DOI: 10.12686/ALPARRAY/XT_2014. URL: http://networks.seismo.ethz.ch/networks/xt/.
6. M Baer and U Kradolfer. “An automatic phase picker for local and teleseismic events”. *Bulletin of the Seismological Society of America* 77:4, 1987, pp. 1437–1445.
7. Christian Baillard et al. “An automatic kurtosis-based P-and S-phase picker designed for local seismic networks”. *Bulletin of the Seismological Society of America* 104:1, 2014, pp. 394–409.
8. Christoforos Benetatos and Anastasia Kiratzi. “Finite-fault slip models for the 15 April 1979 (Mw 7.1) Montenegro earthquake and its strongest aftershock of 24 May 1979 (Mw 6.2)”. *Tectonophysics* 421:1-2, 2006, pp. 129–143.
9. James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization.” *Journal of machine learning research* 13:2, 2012.
10. James Bergstra et al. “Theano: A CPU and GPU math compiler in Python”. In: *Proc. 9th python in science conf.* Vol. 1. 2010, pp. 3–10.
11. Moritz Beyreuther et al. “ObsPy: A Python toolbox for seismology”. *Seismological Research Letters* 81:3, 2010, pp. 530–533.

12. Mary Ann Branch, Thomas F Coleman, and Yuying Li. "A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems". *SIAM Journal on Scientific Computing* 21:1, 1999, pp. 1–23.
13. Justin R Brown, Gregory C Beroza, and David R Shelly. "An autocorrelation method to detect low frequency earthquakes within tremor". *Geophysical Research Letters* 35:16, 2008.
14. Giuseppe Carleo et al. "Machine learning and the physical sciences". *Reviews of Modern Physics* 91:4, 2019, p. 045002.
15. Earthquake Center. "Southern California earthquake center". *Caltech. Dataset*, 2013.
16. CERN. *CERN Seismic Network*. 2016. DOI: 10.12686/SED/NETWORKS/C4. URL: http://networks.seismo.ethz.ch/networks/c4/.
17. Jean-Luc Chatelain et al. "Microearthquake seismicity and fault plane solutions in the Hindu Kush region and their tectonic implications". *Journal of Geophysical Research: Solid Earth* 85:B3, 1980. Publisher: Wiley Online Library, pp. 1365–1387.
18. Sunglok Choi, Taemin Kim, and Wonpil Yu. "Performance evaluation of RANSAC family". *Journal of Computer Vision* 24:3, 1997, pp. 271–300.
19. Francois Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.
20. S. Cianetti et al. "Comparison of Deep Learning Techniques for the Investigation of a Seismic Sequence: An Application to the 2019, Mw 4.5 Mugello (Italy) Earthquake". *Journal of Geophysical Research: Solid Earth* 126:12, 2021. ISSN: 2169-9356. DOI: 10.1029/2021jb023405. URL: http://dx.doi.org/10.1029/2021JB023405.
21. Hengchang Dai and Colin MacBeth. "Automatic picking of seismic arrivals in local earthquake data using an artificial neural network". *Geophysical journal international* 120:3, 1995, pp. 758–774.
22. Hengchang Dai and Colin MacBeth. "The application of back-propagation neural network to automatic picking seismic arrivals from single-component recordings". *Journal of Geophysical Research: Solid Earth* 102:B7, 1997, pp. 15105–15113.
23. Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
24. Li Deng. "The mnist database of handwritten digit images for machine learning research". *IEEE Signal Processing Magazine* 29:6, 2012, pp. 141–142.
25. Konstantinos G Derpanis. "Overview of the RANSAC Algorithm". *Image Rochester NY* 4:1, 2010, pp. 2–3.

26. Ramin MH Dokht et al. “Seismic event and phase detection using time–frequency representation and convolutional neural networks”. *Seismological Research Letters* 90:2A, 2019, pp. 481–490.

27. Timothy J Draelos et al. “A new method for producing automated seismic bulletins: Probabilistic event detection, association, and location”. *Bulletin of the Seismological Society of America* 105:5, 2015, pp. 2453–2467.

28. J. Drew et al. “Coalescence microseismic mapping”. 195, 2013, pp. 1773–1785. DOI: 10.1093/gji/ggt331.

29. Julian Drew et al. “Coalescence microseismic mapping”. *Geophysical Journal International* 195:3, 2013, pp. 1773–1785.

30. Paul S Earle and Peter M Shearer. “Characterization of global seismograms using an automatic-picking algorithm”. *Bulletin of the Seismological Society of America* 84:2, 1994, pp. 366–376.

31. Martijn PA van den Ende and J-P Ampuero. “Automated seismic source characterization using deep graph neural networks”. *Geophysical Research Letters* 47:17, 2020, e2020GL088690.

32. Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. *Communications of the ACM* 24:6, 1981, pp. 381–395.

33. Mike Folk et al. “An overview of the HDF5 technology suite and its applications”. In: *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*. 2011, pp. 36–47.

34. Stefania Gentili and Alberto Michelini. “Automatic picking of P and S phases using a neural tree”. *Journal of Seismology* 10:1, 2006, pp. 39–63.

35. GFZ German Research Centre for Geosciences. *GFZ earthquake bulletin: Mw 6.4 Albania (26-11-2019)*. http://geofon.gfz-potsdam.de/eqinfo/event.php?id=gfz2019xdig. Accessed: 2022-03-28. 2019.

36. Steven J Gibbons and Frode Ringdal. “The detection of low magnitude seismic events using array-based waveform correlation”. *Geophysical Journal International* 165:1, 2006, pp. 149–166.

37. Steven J Gibbons et al. “Iterative strategies for aftershock classification in automatic seismic processing pipelines”. *Seismological Research Letters* 87:4, 2016, pp. 919–929.

38. Alec Go, Richa Bhayani, and Lei Huang. *Twitter Sentiment Classification using Distant Supervision*. 2009. URL: http://help.sentiment140.com/home.

39. Joan S Gomberg, Kaye M Shedlock, and Steven W Roecker. “The effect of S-wave arrival times on the accuracy of hypocenter estimation”. *Bulletin of the Seismological Society of America* 80:6A, 1990, pp. 1605–1628.

40. Joan S. Gomberg, Kaye M. Shedlock, and Steven W. Roecker. “The effect of S-wave arrival times on the accuracy of hypocenter estimation”. *Bulletin of the Seismological Society of America* 80:6A, 1990. Publisher: The Seismological Society of America, pp. 1605–1628.

41. Vladimir Grechka and Ilya Tsvankin. “3-D description of normal moveout in anisotropic inhomogeneous media”. *Geophysics* 63:3, 1998, pp. 1079–1092.

42. Francesco Grigoli et al. “Automated seismic event location by travel-time stacking: An application to mining induced seismicity”. *Seismological Research Letters* 84:4, 2013, pp. 666–677.

43. Jeanne Hardebeck and Stephan Husen. “Earthquake location accuracy”. en, 2010. Publisher: Community Online Resource for Statistical Seismicity Analysis. DOI: 10.5078/CORSSA-55815573. URL: http://www.corssa.org/export/sites/corssa/.galleries/articles-pdf/Husen-Hardebeck-2010-CORSSA-Eqk-location.pdf (visited on 03/14/2021).

44. Gavin P Hayes, David J Wald, and Rebecca L Johnson. “Slab1. 0: A three-dimensional model of global subduction zone geometries”. *Journal of Geophysical Research: Solid Earth* 117:B1, 2012.

45. Stephen P Hicks et al. “Anatomy of a megathrust: The 2010 M8. 8 Maule, Chile earthquake rupture zone imaged using seismic tomography”. *Earth and Planetary Science Letters* 405, 2014, pp. 142–155.

46. Geoffrey Hinton et al. “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. *IEEE Signal processing magazine* 29:6, 2012, pp. 82–97.

47. Austin A Holland. “Earthquakes triggered by hydraulic fracturing in south-central Oklahoma”. *Bulletin of the Seismological Society of America* 103:3, 2013, pp. 1784–1792.

48. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. *Neural networks* 2:5, 1989, pp. 359–366.

49. S Husen et al. “Accurate hypocentre determination in the seismogenic zone of the subducting Nazca Plate in northern Chile using a combined on-/offshore network”. *Geophysical Journal International* 138:3, 1999, pp. 687–701.

50. Miaki Ishii et al. “Extent, duration and speed of the 2004 Sumatra–Andaman earthquake imaged by the Hi-Net array”. *Nature* 435:7044, 2005, pp. 933–936.

51. Michael I Jordan and Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". *Science* 349:6245, 2015, pp. 255–260.

52. Honn Kao and Shao-Ju Shan. "The source-scanning algorithm: Mapping the distribution of seismic sources in time and space". *Geophysical Journal International* 157:2, 2004, pp. 589–594.

53. George Em Karniadakis et al. "Physics-informed machine learning". *Nature Reviews Physics* 3:6, 2021, pp. 422–440.

54. Andrej Karpathy et al. "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.

55. Aitaro Kato et al. "Foreshock migration preceding the 2016 Mw 7.0 Kumamoto earthquake, Japan". *Geophysical Research Letters* 43:17, 2016, pp. 8945–8953.

56. Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*, 2014.

57. Franz J Király et al. "Designing Machine Learning Toolboxes: Concepts, Principles and Patterns". *arXiv preprint arXiv:2101.04938*, 2021.

58. Edi Kissling. "Geotomography with local earthquake data". *Reviews of Geophysics* 26:4, 1988. Publisher: Wiley Online Library, pp. 659–698.

59. Edi Kissling et al. "Initial reference models in local earthquake tomography". *Journal of Geophysical Research: Solid Earth* 99:B10, 1994, pp. 19635–19646.

60. Qingkai Kong et al. "Machine learning in seismology: Turning data into insights". *Seismological Research Letters* 90:1, 2019, pp. 3–14.

61. Qingkai Kong et al. "MyShake: A smartphone seismic network for earthquake early warning and beyond". *Science advances* 2:2, 2016, e1501055.

62. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". *Communications of the ACM* 60:6, 2017, pp. 84–90.

63. Ludger Küperkoch et al. "Automated determination of P-phase arrival times at regional and local distances using higher order statistics". *Geophysical Journal International* 181:2, 2010, pp. 1159–1170.

64. Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". *The handbook of brain theory and neural networks* 3361:10, 1995, p. 1995.

65. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". *nature* 521:7553, 2015, pp. 436–444.

66. Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. *Neural computation* 1:4, 1989, pp. 541–551.

67. Yann A LeCun et al. “Efficient backprop”. In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

68. M Leonard and BLN Kennett. “Multi-component autoregressive techniques for the analysis of seismograms”. *Physics of the Earth and Planetary Interiors* 113:1-4, 1999, pp. 247–263.

69. Kenneth Levenberg. “A method for the solution of certain non-linear problems in least squares”. *Quarterly of applied mathematics* 2:2, 1944, pp. 164–168.

70. Zefeng Li et al. “High-resolution seismic event detection using local similarity for Large-N arrays”. *Scientific reports* 8:1, 2018, pp. 1–10.

71. Zefeng Li et al. “Machine learning seismic wave discrimination: Application to earthquake early warning”. *Geophysical Research Letters* 45:10, 2018, pp. 4773–4779.

72. Seppo Linnainmaa. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”. PhD thesis. Master’s Thesis (in Finnish), Univ. Helsinki, 1970.

73. Andrea L Llenos and Andrew J Michael. “Modeling earthquake rate changes in Oklahoma and Arkansas: Possible signatures of induced seismicity”. *Bulletin of the Seismological Society of America* 103:5, 2013, pp. 2850–2861.

74. Anthony Lomax, Alberto Michelini, and Dario Jozinović. “An investigation of rapid earthquake characterization using single-station waveforms and a convolutional neural network”. *Seismological Research Letters* 90:2A, 2019, pp. 517–529.

75. Anthony Lomax, Claudio Satriano, and Maurizio Vassallo. “Automatic picker developments and optimization: FilterPicker—A robust, broadband picker for real-time seismic monitoring and earthquake early warning”. *Seismological Research Letters* 83:3, 2012, pp. 531–540.

76. Anthony Lomax et al. “Probabilistic earthquake location in 3D and layered models”. In: *Advances in seismic event location*. Springer, 2000, pp. 101–134.

77. Fabrizio Magrini et al. “Local earthquakes detection: A benchmark dataset of 3-component seismograms built on a global scale”. *Artificial Intelligence in Geosciences* 1, 2020, pp. 1–10.

78. J Matas and O Chum. “Randomized RANSAC with Test”. In: *Proc. British Machine Vision Conf.,* 2002, pp. 448–457.

79. Ian W McBrearty, Andrew A Delorey, and Paul A Johnson. “Pairwise association of seismic arrivals with convolutional neural networks”. *Seismological Research Letters* 90:2A, 2019, pp. 503–509.

80. Ian W McBrearty et al. “Earthquake arrival association with backprojection and graph theoryearthquake arrival association with backprojection and graph theory”. *Bulletin of the Seismological Society of America* 109:6, 2019, pp. 2510–2531.

81. Nicole D McMahon et al. “Spatiotemporal evolution of the 2011 Prague, Oklahoma, aftershock sequence revealed using subspace detection and relocation”. *Geophysical Research Letters* 44:14, 2017, pp. 7149–7158.

82. Men-Andrin Meier et al. “Reliable real-time seismic signal/noise discrimination with machine learning”. *Journal of Geophysical Research: Solid Earth* 124:1, 2019, pp. 788–800.

83. Alberto Michelini et al. “INSTANCE–the Italian seismic dataset for machine learning”. *Earth System Science Data Discussions*, 2021, pp. 1–47.

84. S Mostafa Mousavi and Gregory C Beroza. “A machine-learning approach for earthquake magnitude estimation”. *Geophysical Research Letters* 47:1, 2020, e2019GL085976.

85. S Mostafa Mousavi et al. “CRED: A deep residual network of convolutional and recurrent units for earthquake signal detection”. *Scientific reports* 9:1, 2019, pp. 1–14.

86. S Mostafa Mousavi et al. “Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking”. *Nature communications* 11:1, 2020, pp. 1–12.

87. S Mostafa Mousavi et al. “STanford EArthquake Dataset (STEAD): A global data set of seismic signals for AI”. *IEEE Access* 7, 2019, pp. 179464–179476.

88. S Mostafa Mousavi et al. “Unsupervised clustering of seismic signals using deep convolutional autoencoders”. *IEEE Geoscience and Remote Sensing Letters* 16:11, 2019, pp. 1693–1697.

89. B Muco and P Minga. “Magnitude determination of near earthquakes for the Albanian network”. *Bollettino di Geofisica Teorica ed Applicata* 33:129, 1991, pp. 17–24.

90. J. Münchmeyer and 12 coauthors. “Which picker fits my data? A quantitative evaluation of deeplearning based seismic pickers”. DOI: .... URL: `https://arxiv.org/...`

91. Jannes Münchmeyer et al. “Earthquake magnitude and location estimation from real time seismic waveforms with a transformer network”. *Geophysical Journal International* 226:2, 2021, pp. 1086–1104.

92. Jannes Münchmeyer et al. “The transformer earthquake alerting model: a new versatile approach to earthquake early warning”. *Geophysical Journal International* 225:1, 2021, pp. 646–656.
93. Jannes Münchmeyer et al. “Which picker fits my data? A quantitative evaluation of deep learning based seismic pickers”. *Journal of Geophysical Research: Solid Earth*, 2022, e2021JB023499.
94. Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Icml*. 2010.
95. SEJ Nippress, A Rietbrock, and AE Heath. “Optimized automatic pickers: application to the ANCORP data set”. *Geophysical Journal International* 181:2, 2010, pp. 911–925.
96. David Nistér. “Preemptive RANSAC for live structure and motion estimation”. *Machine Vision and Applications* 16:5, 2005, pp. 321–329.
97. Kazushige Obara. “Hi-net: High sensitivity seismograph network, Japan”. In: *Methods and Applications of Signal Processing in Seismic Network Operations*. Springer, 2003, pp. 79–88.
98. Esteban Pardo, Carmen Garfias, and Norberto Malpica. “Seismic phase picking using convolutional networks”. *IEEE Transactions on Geoscience and Remote Sensing* 57:9, 2019, pp. 7086–7092.
99. Adam Paszke et al. “Automatic differentiation in PyTorch”, 2017.
100. Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. *Journal of machine learning research* 12:Oct, 2011, pp. 2825–2830.
101. Zhigang Peng and Peng Zhao. “Migration of early aftershocks following the 2004 Parkfield earthquake”. *nature Geoscience* 2:12, 2009, pp. 877–881.
102. Thibaut Perol, Michaël Gharbi, and Marine Denolle. “Convolutional neural network for earthquake detection and location”. *Science Advances* 4:2, 2018, e1700578.
103. David MW Powers. “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation”. *arXiv preprint arXiv:2010.16061*, 2020.
104. Maziar Raissi, Paris Perdikaris, and George E Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. *Journal of Computational physics* 378, 2019, pp. 686–707.
105. SJ Rastin et al. “Using real and synthetic waveforms of the Matata swarm to assess the performance of New Zealand GeoNet phase pickers”. *Bulletin of the Seismological Society of America* 103:4, 2013, pp. 2173–2187.

106. Jeff Reback et al. "pandas-dev/pandas: Pandas 1.0. 3". *Zenodo*, 2020.

107. A Rietbrock et al. "Aftershock seismicity of the 2010 Maule Mw= 8.8, Chile, earthquake: Correlation between co-seismic slip models and aftershock distribution?" *Geophysical Research Letters* 39:8, 2012.

108. Volker Rodehorst and Olaf Hellwich. "Genetic algorithm sample consensus (gasac)-a parallel strategy for robust parameter estimation". In: *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPR W'06)*. IEEE. 2006, pp. 103–103.

109. William Rodi and MN Toksoz. *Grid-search techniques for seismic event location*. Technical report. MASSACHUSETTS INST OF TECH CAMBRIDGE EARTH RESOURCES LAB, 2000.

110. Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65:6, 1958, p. 386.

111. Zachary E Ross et al. "Generalized seismic phase detection with deep learning". *Bulletin of the Seismological Society of America* 108:5A, 2018, pp. 2894–2901.

112. Zachary E Ross et al. "PhaseLink: A deep learning approach to seismic phase association". *Journal of Geophysical Research: Solid Earth* 124:1, 2019, pp. 856–869.

113. Zachary E Ross et al. "Searching for hidden earthquakes in Southern California". *Science* 364:6442, 2019, pp. 767–771.

114. Sebastian Rost and Christine Thomas. "Array seismology: Methods and applications". *Reviews of geophysics* 40:3, 2002, pp. 2–1.

115. Gerhard Roth and Martin D Levine. "Extracting geometric primitives". *CVGIP: image understanding* 58:1, 1993, pp. 1–22.

116. Bertrand Rouet-Leduc et al. "Machine learning predicts laboratory earthquakes". *Geophysical Research Letters* 44:18, 2017, pp. 9276–9282.

117. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". *nature* 323:6088, 1986, pp. 533–536.

118. MS Sambridge and BLN Kennett. "A novel method of hypocentre location". *Geophysical Journal International* 87:2, 1986, pp. 679–697.

119. Christos D Saragiotis, Leontios J Hadjileontiadis, and Stavros M Panas. "PAI-S/K: A robust automatic seismic P phase arrival identification scheme". *IEEE Transactions on Geoscience and Remote Sensing* 40:6, 2002, pp. 1395–1404.

120. Christos D Saragiotis et al. "Automatic P phase picking using maximum kurtosis and/spl kappa/-statistics criteria". *IEEE Geoscience and Remote Sensing Letters* 1:3, 2004, pp. 147–151.

121. SCEDC. *Seismic Transfer Program (version 1.4.1)*. 2010. URL: https://scedc.caltech.edu/data/stp/.

122. SCEDC. "Southern California Earthquake Data Center. Caltech. Dataset", 2013.

123. Jürgen Schmidhuber. "Deep learning in neural networks: An overview". *Neural networks* 61, 2015, pp. 85–117.

124. Ruwen Schnabel, Roland Wahl, and Reinhard Klein. "Efficient RANSAC for point-cloud shape detection". In: *Computer graphics forum*. Vol. 26. 2. Wiley Online Library. 2007, pp. 214–226.

125. Bernd Schurr et al. "AlbACa – Albanian Earthquake Aftershock Campaign", 2020. DOI: 10.14470/4X7564679396. URL: https://geofon.gfz-potsdam.de/doi/network/9K/2019.

126. David R Shelly, Gregory C Beroza, and Satoshi Ide. "Non-volcanic tremor and low-frequency earthquake swarms". *Nature* 446:7133, 2007, pp. 305–307.

127. Christian Sippl et al. "Seismicity structure of the northern Chile forearc from> 100,000 double-difference relocated hypocenters". *Journal of Geophysical Research: Solid Earth* 123:5, 2018, pp. 4063–4087.

128. Reinoud Sleeman and Torild Van Eck. "Robust automatic P-phase picking: an on-line implementation in the analysis of broadband seismogram recordings". *Physics of the earth and planetary interiors* 113:1-4, 1999, pp. 265–275.

129. Hugo Soto and Bernd Schurr. "DeepPhasePick: A method for detecting and picking seismic phases from local earthquakes based on highly optimized convolutional and recurrent deep neural networks". *Geophysical Journal International* 227:2, 2021, pp. 1268–1294. ISSN: 0956-540X. DOI: 10.1093/gji/ggab266. URL: https://academic.oup.com/gji/advance-article/doi/10.1093/gji/ggab266/6324012.

130. Hugo Soto and Bernd Schurr. "DeepPhasePick: a method for detecting and picking seismic phases from local earthquakes based on highly optimized convolutional and recurrent deep neural networks". *Geophysical Journal International* 227:2, 2021, pp. 1268–1294.

131. Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". *The journal of machine learning research* 15:1, 2014, pp. 1929–1958.

132. Swiss Seismological Service (SED) At ETH Zurich. *National Seismic Networks of Switzerland*. en. 1983. DOI: 10.12686/SED/NETWORKS/CH. URL: http://networks.seismo.ethz.ch/networks/ch/.

133. Swiss Seismological Service (SED) At ETH Zurich. *Seismology at School Program, ETH Zurich*. 2008. DOI: 10.12686/SED/NETWORKS/S. URL: http://networks.seismo.ethz.ch/networks/s/.

134. Swiss Seismological Service (SED) At ETH Zurich. *Temporary deployments in Switzerland associated with aftershocks and other seismic sequences*. 2005. DOI: 10.12686/SED/NETWORKS/8D. URL: http://networks.seismo.ethz.ch/networks/8d/.

135. Yen Joe Tan et al. "Machine-learning-based high-resolution earthquake catalog reveals how complex fault structures were activated during the 2016–2017 Central Italy sequence". *The Seismic Record* 1:1, 2021, pp. 11–19.

136. Simone Teloni et al. "Seismogenic fault system of the Mw 6.4 November 2019 Albania earthquake: new insights into the structural architecture and active tectonic setting of the outer Albanides". *Journal of the Geological Society* 178:2, 2021. jgs2020-193. ISSN: 0016-7649. DOI: 10.1144/jgs2020-193. eprint: https://pubs.geoscienceworld.org/jgs/article-pdf/doi/10.1144/jgs2020-193/5253113/jgs2020-193.pdf. URL: https://doi.org/10.1144/jgs2020-193.

137. Manuel Titos et al. "A deep neural networks approach to automatic recognition systems for volcano-seismic events". *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 11:5, 2018, pp. 1533–1544.

138. Philip H. S. Torr. "Bayesian model estimation and selection for epipolar geometry and generic manifold fitting". *International Journal of Computer Vision* 50:1, 2002, pp. 35–61.

139. Philip HS Torr and Andrew Zisserman. "MLESAC: A new robust estimator with application to estimating image geometry". *Computer vision and image understanding* 78:1, 2000, pp. 138–156.

140. Andrew P. Valentine and Jeannot Trampert. "Data space reduction, quality assessment and searching of seismograms: autoencoder networks for waveform data". *Geophysical Journal International* 189:2, 2012, pp. 1183–1202. ISSN: 0956-540X. DOI: 10.1111/j.1365-246x.2012.05429.x. URL: http://dx.doi.org/10.1111/j.1365-246X.2012.05429.x.

141. Vincent Van der Heiden. "Analysis of the 2019 M_w 6.4 Albania aftershock sequence: An updated velocity model using AI-based solutions". MA thesis. Geophysical Institute: Karlsruhe Insititute of Technology, 2021.

142. Eutizio Vittori et al. “Geological effects and tectonic environment of the 26 November 2019, Mw 6.4 Durres earthquake (Albania)”. *Geophysical Journal International* 225:2, 2020, pp. 1174–1191. ISSN: 0956-540X. DOI: 10.1093/gji/ggaa582. eprint: https://academic.oup.com/gji/article-pdf/225/2/1174/36567167/ggaa582.pdf. URL: https://doi.org/10.1093/gji/ggaa582.

143. Jian Wang et al. “Deep learning for picking seismic arrival times”. *Journal of Geophysical Research: Solid Earth* 124:7, 2019, pp. 6612–6624.

144. Jin Wang and Ta-Liang Teng. “Artificial neural network-based seismic detector”. *Bulletin of the Seismological Society of America* 85:1, 1995, pp. 308–319.

145. Jin Wang and Ta-liang Teng. “Identification and picking of S phase using an artificial neural network”. *Bulletin of the Seismological Society of America* 87:5, 1997, pp. 1140–1149.

146. Paul Werbos. “Beyond regression:" new tools for prediction and analysis in the behavioral sciences”. *Ph. D. dissertation, Harvard University*, 1974.

147. Jack Woollam et al. “Convolutional neural network for seismic phase classification, performance demonstration over a local seismic network”. *Seismological Research Letters* 90:2A, 2019, pp. 491–502.

148. Jack Woollam et al. “HEX: Hyperbolic event extractor, a seismic phase associator for highly active seismic regions”. *Seismological Society of America* 91:5, 2020, pp. 2769–2778.

149. Jack Woollam et al. “SeisBench - A toolbox for machine learning in seismology”.

150. Jack Woollam et al. “SeisBench—A Toolbox for Machine Learning in Seismology”. *Seismological Research Letters*, 2022. ISSN: 0895-0695. DOI: 10.1785/0220210324. eprint: https://pubs.geoscienceworld.org/ssa/srl/article-pdf/doi/10.1785/0220210324/5567643/srl-2021324.1.pdf. URL: https://doi.org/10.1785/0220210324.

151. William Luther Yeck et al. “Leveraging Deep Learning in Global 24/7 Real-Time Earthquake Monitoring at the National Earthquake Information Center”. *Seismological Society of America* 92:1, 2021, pp. 469–480.

152. Clara E Yoon et al. “Earthquake detection through computationally efficient similarity search”. *Science advances* 1:11, 2015, e1501057.

153. Miao Zhang, William L Ellsworth, and Gregory C Beroza. “Rapid earthquake association and location”. *Seismological Research Letters* 90:6, 2019, pp. 2276–2284.

154. Miao Zhang et al. “LOC-FLOW: An End-to-End Machine Learning-Based High-Precision Earthquake Location Workflow”. *Seismological Research Letters*, 2022. ISSN: 0895-0695. DOI: 10.1785/0220220019. eprint: https://pubs.geoscienceworld.org/ssa/srl/article-pdf/doi/10.1785/0220220019/5565734/srl-2022019.1.pdf. URL: https://doi.org/10.1785/0220220019.

155. Yue Zhao and Kiyoshi Takano. “An artificial neural network approach for broadband seismic phase picking”. *Bulletin of the Seismological Society of America* 89:3, 1999, pp. 670–680.

156. Yi-Tong Zhou and Rama Chellappa. “Computation of optical flow using a neural network.” In: *ICNN*. 1988, pp. 71–78.

157. Yijian Zhou et al. “Hybrid event detection and phase-picking algorithm using convolutional and recurrent neural networks”. *Seismological Research Letters* 90:3, 2019, pp. 1079–1087.

158. Lijun Zhu et al. “A multi-channel approach for automatic microseismic event association using ransac-based arrival time event clustering (ratec)”. *Earthquake Research Advances*, 2021, p. 100008.

159. Weiqiang Zhu and Gregory C Beroza. “PhaseNet: a deep-neural-network-based seismic arrival-time picking method”. *Geophysical Journal International* 216:1, 2019, pp. 261–273.

160. Weiqiang Zhu et al. “Earthquake Phase Association using a Bayesian Gaussian Mixture Model”. *arXiv preprint arXiv:2109.09008*, 2021.