sensors

MDPI

Article

GroningenNet: Deep Learning for Low-Magnitude Earthquake Detection on a Multi-Level Sensor Network

Ahmed Shaheen [1,*], Umair bin Waheed [1], Michael Fehler [2], Lubos Sokol [3] and Sherif Hanafy [1]

[1] Department of Geosciences, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia; umair.waheed@kfupm.edu.sa (U.b.W.); sherif.mahmoud@kfupm.edu.sa (S.H.)
[2] Earth Resources Laboratory, Department of Earth, Atmospheric and Planetary Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, USA; fehler@mit.edu
[3] Seismik s.r.o., 12800 Prague, Czech Republic; lubos@seismik.cz
* Correspondence: ahmed.afify.shahin@gmail.com

Abstract: Automatic detection of low-magnitude earthquakes has become an increasingly important research topic in recent years due to a sharp increase in induced seismicity around the globe. The detection of low-magnitude seismic events is essential for microseismic monitoring of hydraulic fracturing, carbon capture and storage, and geothermal operations for hazard detection and mitigation. Moreover, the detection of micro-earthquakes is crucial to understanding the underlying mechanisms of larger earthquakes. Various algorithms, including deep learning methods, have been proposed over the years to detect such low-magnitude events. However, there is still a need for improving the robustness of these methods in discriminating between local sources of noise and weak seismic events. In this study, we propose a convolutional neural network (CNN) to detect seismic events from shallow borehole stations in Groningen, the Netherlands. We train a CNN model to detect low-magnitude earthquakes, harnessing the multi-level sensor configuration of the G-network in Groningen. Each G-network station consists of four geophones at depths of 50, 100, 150, and 200 m. Unlike prior deep learning approaches that use 3-component seismic records only at a single sensor level, we use records from the entire borehole as one training example. This allows us to train the CNN model using moveout patterns of the energy traveling across the borehole sensors to discriminate between events originating in the subsurface and local noise arriving from the surface. We compare the prediction accuracy of our trained CNN model to that of the STA/LTA and template matching algorithms on a two-month continuous record. We demonstrate that the CNN model shows significantly better performance than STA/LTA and template matching in detecting new events missing from the catalog and minimizing false detections. Moreover, we find that using the moveout feature allows us to effectively train our CNN model using only a fraction of the data that would be needed otherwise, saving plenty of manual labor in preparing training labels. The proposed approach can be easily applied to other microseismic monitoring networks with multi-level sensors.

Keywords: induced seismicity; micro-earthquakes; deep learning; convolutional neural networks

check for updates

Citation: Shaheen, A.; Waheed, U.b.; Fehler, M.; Sokol, L.; Hanafy, S. GroningenNet: Deep Learning for Low-Magnitude Earthquake Detection on a Multi-Level Sensor Network. *Sensors* **2021**, *21*, 8080. https://doi.org/10.3390/s21238080

Academic Editor: Filippo Ubertini

Received: 27 October 2021
Accepted: 20 November 2021
Published: 2 December 2021

1. Introduction

Automatic detection of low-magnitude earthquakes has been a longstanding research problem in seismology. Although earthquakes are known to be clustered in time and space, the underlying processes that connect one earthquake to the next are still poorly understood. The challenge comes from our limited capability to detect small earthquakes reliably [1]. Given the sharp rise in induced seismicity during the past decade, the study of these low-magnitude events is critical for waste-water injection, geothermal monitoring, carbon capture and storage (CCS), and hydraulic fracturing in shale gas reservoirs. Moreover, the exponential growth of seismic data being collected due to a huge increase in the number of recording stations over the years makes automatic detection a necessity. Therefore, a robust and accurate detection algorithm is essential to get the full potential of the available data and to understand the subsurface processes better.

One of the earliest approaches to automate seismic event detection was the STA/LTA algorithm [2]. The algorithm computes the ratio of a short-term average of the amplitudes (STA) to a long-term average of the amplitudes (LTA). This ratio yields a high value when an event is present. If the ratio exceeds a preset threshold value, a trigger is activated, indicating the presence of a seismic event. The algorithm shows robustness in detecting events with high signal-to-noise ratio (SNR) data. However, the performance of STA/LTA drops significantly when used to detect events for low SNR data. For these cases, STA/LTA fails to differentiate between time-varying noises and seismic events.

To enhance the detection of events for low SNR waveforms, the template matching technique [3] has been quite successful. Template matching cross-correlates a template waveform of a prior master seismic event to continuous seismic records. A high correlation between the waveforms of the records and those of the template indicates the presence of an event. The technique has shown to be efficient in detecting events buried under noise. However, its major drawbacks are insensitivity to events with waveforms that are dissimilar to the master event and high computational cost.

Several other techniques have been proposed over the years to tackle the problem of automatic event detection. Yoon et al. [4] proposed the FAST algorithm, which is a data mining method to detect uncataloged events with good computational efficiency. Poliannikov and Fehler [5] stacked the instantaneous phases of several traces to detect seismic events in the vicinity of a master event. They applied the known moveout of the master event on the traces before stacking. Since the instantaneous phases of noise form a Gaussian distribution, an event will appear on this stacked data as an outlier from this distribution. Mukuhira et al. [6] used the polarization property of the P-wave to detect events in low SNR data. Machine learning techniques have also been developed in recent years to detect low-magnitude events robustly [7,8]. However, there is still a need for improvement in pushing the envelope further to detect more of such small magnitude earthquakes reliably.

Deep learning has emerged as a powerful technique to tackle longstanding problems in various data-intensive fields such as computer vision and speech recognition. The ability of deep learning to recognize complex structures in high-dimensional data makes it attractive over other conventional algorithms [9]. In image classification, for instance, deep learning algorithms have proved significantly more accurate than any preceding technique [10,11]. Moreover, deep learning can extract features directly from the data, requiring little or no feature engineering of the raw data.

Improved performance over conventional methods and ease of implementation have encouraged applications of deep learning on various seismological problems, including passive seismic event detection [12]. Perol et al. [13] trained a convolutional neural network (CNN) to detect earthquakes from seismic records in Guthrie, Oklahoma. The region is known for problems with earthquakes caused by induced seismicity. The alteration of stresses in the subsurface due to waste-water injection has led to the increasing number of earthquakes in Oklahoma [14]. They used around 834,000 labeled samples of events and noise to train the network. The trained network was tested on a month-long continuous seismic record. The network detected ten times more events than those present in the catalog with 94% precision, where precision is the number of true detected events to the total number of event detections. Other studies have also shown the efficacy of deep learning in reliably detecting passive seismic events [15–17].

Detection and study of these, predominantly low-magnitude, events caused by induced seismicity help in understanding subsurface mechanisms and assessing hazard probabilities. Another region that suffers from the problem of induced seismicity is Groningen, the Netherlands. Groningen has suffered from induced seismicity over the past three decades, caused by production from the Groningen gas field—the largest gas field in

Western Europe. Continuous gas production since the 1960s led to reservoir compaction, which has been causing induced seismic events since 1991 [18].

Since the turn of the century, these events have significantly increased in both the frequency of occurrence and magnitude. These seismic events pose a danger to the lives and properties of residents in Groningen and neighboring areas. The recurrence of these events forced the Dutch government to announce the stopping of production by 2022 [19], causing huge financial loss to the country and operators. Therefore, understanding the underlying mechanisms of these earthquakes is vital to minimize further human and financial losses in Groningen and, more importantly, to prevent problems from escalating in other regions of the world. The first step to study these mechanisms is having a robust and accurate algorithm to detect low-magnitude events.

In this study, we use data recorded by the shallow borehole network in Groningen, known as the G-network. The network consists of borehole stations that contain multi-level geophones at increasing depths of 50 m. Since CNNs are well-known for identifying spatial features in data, we use the moveout pattern of the energy traveling across the borehole sensors at a single station as the main distinguishing factor between events originating in the subsurface and noise coming from the surface. This is often the most challenging problem in detecting low-magnitude earthquakes. On a single sensor, earthquake energy coming from below or energy coming from a local noise source on the surface may look the same, leading to difficulty in classification. In contrast, the two sources of energy will show opposite moveout patterns on a multi-level sensor network. Just as a human interpreter could visually separate true events from coherent noise by picking the moveout, i.e., up-down (noise) or down-up (event), we train a CNN for this classification task. Recognizing these moveout patterns increases the network's resolution in detecting events of lower magnitudes and in minimizing false detections.

We test the trained CNN model on two months (December 2017–January 2018) of continuous records at five stations near Zeerijp—the site of a major event (M_L = 3.4) that occurred on 8 January 2018. We compare the findings of the CNN model to the outputs of the STA/LTA and template matching algorithms on the same two-month continuous record. We demonstrate that despite being trained on a relatively small amount of data, the CNN model shows significantly better performance than STA/LTA and template matching in detecting new events missing from the catalog and minimizing false detections. In total, we managed to increase the number of detected events by about 100% compared to those in the original catalog. The trained CNN model and associated scripts can be found at: https://github.com/AhmedShaheen01/GroningenNet (accessed on 27 October 2021).

The rest of the paper is organized as follows: We begin by explaining the methodology used to train and test the CNN model. This is followed by the results obtained using the proposed method and its comparison with conventional techniques. Finally, we discuss the key findings of this study and their implications.

2. Methodology

In this section, we begin by outlining details of the seismic network used in the study. Next, we illustrate the pre-processing steps applied to the seismic records before training the CNN. Then, we shed some light on CNN's key features, followed by a description of the CNN model's architecture. Lastly, we describe how the model's performance on continuous data is evaluated and compared to other methods.

2.1. Training Data Selection and Pre-Processing

The increase in the number and strength of induced seismicity connected to the Groningen gas field resulted in the densification of the monitoring network over the years. In particular, after the largest event in the region (M_L = 3.6) that occurred in 2012 near Huizinge, a total of 70 stations were added to cover the Groningen gas field with an average spacing of 4–5 km. Together, these 70 stations form the G-network that is comprised of shallow borehole stations [20]. Each borehole station contains an accelerometer at the surface and four velocity geophone sensors (3-component). These velocity sensors are located at depths of 50 m, 100 m, 150 m, and 200 m, respectively, from the surface. In this study, we use records from the four velocity geophones for training and testing the CNN model. Figure 1 shows the configuration of a typical G-network station.

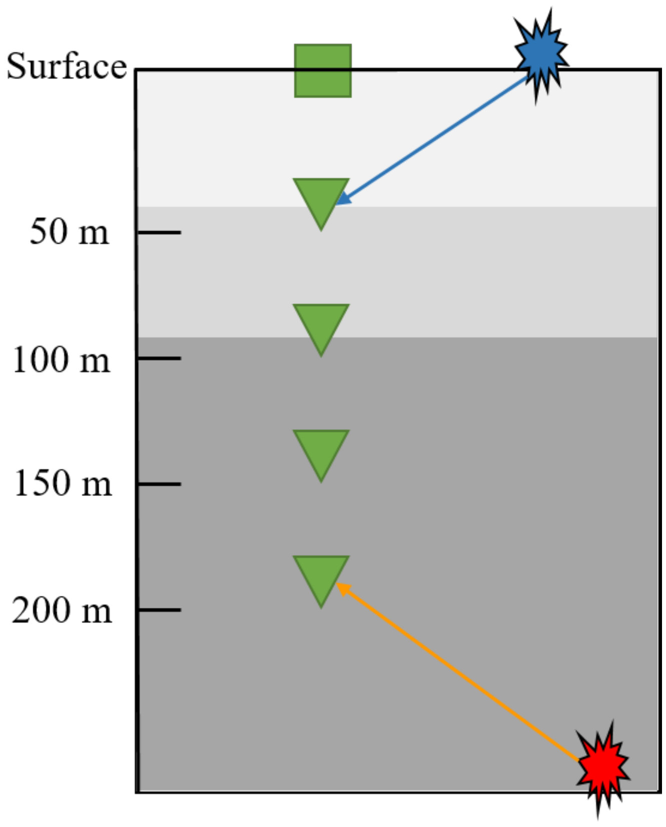

Figure 1. A typical G-network station consisting of an accelerometer at the surface (green square) and four geophones (green triangles). This set up allows the differentiation between genuine seismic events (red) and noise coming from the surface (blue) by observing the moveout pattern across the sensors. Adapted from ref. [20].

We harness this borehole station set up to improve the robustness of our CNN model in distinguishing events from local sources of noise. True seismic events will arrive first at the deepest sensor, while noise coming from the surface arrives first at the shallowest sensor. This results in a moveout pattern—bottom-up for true events in contrast to up-down for coherent noise sources on the surface. We teach a CNN model, using carefully selected waveforms, to distinguish between such moveout differences and improve classification accuracy. Classification algorithms that use data only at a single sensor are likely to suffer from inaccurate predictions in the presence of strong coherent noise as they cannot effectively distinguish coherent noise from true events if observed only at a single sensor level. Figure 2 shows the moveout pattern of a genuine seismic event across the four velocity geophones of a G-network station, and Figure 3 shows an example of the moveout of noise coming from above.

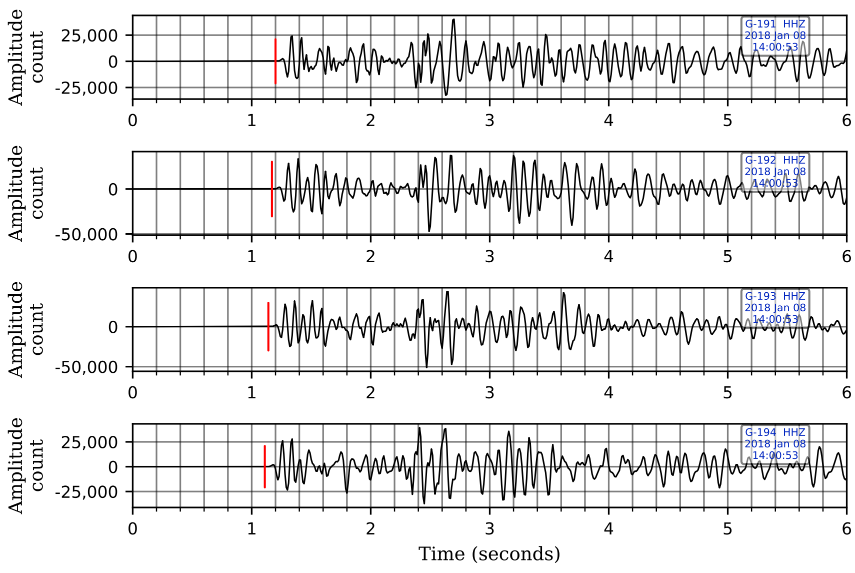

Figure 2. The moveout pattern of a genuine seismic event observed across the four geophone levels. The red vertical lines denote the first-arrival times of the seismic wave at each sensor. The wave arrives first at the deepest sensor and propagates upwards. A bandpass filter between 5–25 Hz has been applied to the records. Note that only vertical components are shown here for illustration. The waveforms are displayed in sequence with the waveform from the shallowest sensor at the top and the one from the deepest sensor at the bottom.

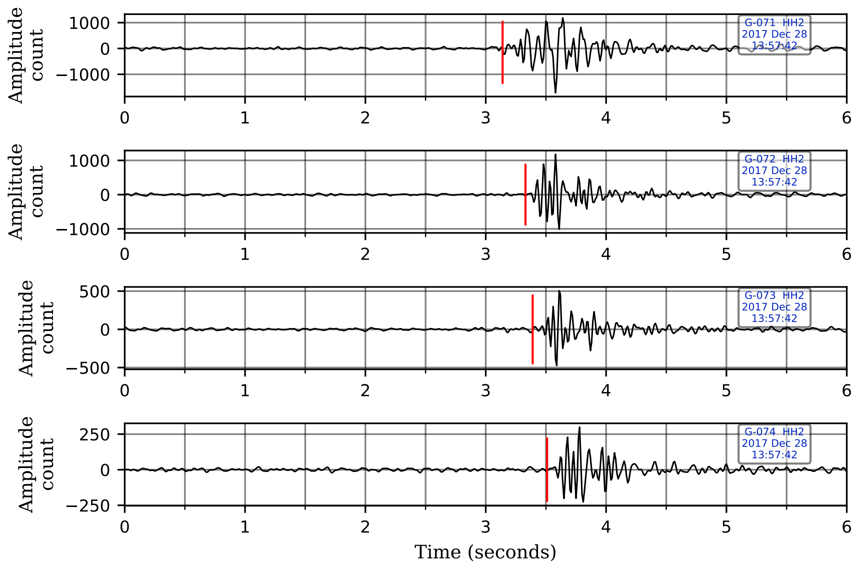

Figure 3. The moveout pattern of noise coming from the surface observed across the four geophone levels. The red vertical lines denote the first-arrival times of the seismic wave at each sensor. The wave arrives first at the shallowest sensor and propagates downwards. A bandpass filter between 5–25 Hz has been applied to the records. Note that only one of the horizontal components is shown here for illustration. The waveforms are displayed in sequence with the waveform from the shallowest sensor at the top and the one from the deepest sensor at the bottom.

Seismic records of the G-network are made available online by the Royal Netherlands Meteorological Institute (KNMI) at its seismic and acoustic data portal [21]. In particular, we are interested in the time period around the large event (M_L = 3.4) in the Zeerijp region that occurred on 8 January 2018. Therefore, we consider data from all the 47 events listed in the KNMI catalog between 1 October 2017, and 28 February 2018, for training a CNN model. For each of the 47 listed events, we retrieve four-minute long records at all the 70 G-network stations starting from three minutes before the listed origin time until one minute after the origin time. The time-windows for the event class are taken from the interval after the origin time, and the time-windows for the noise class are taken from the interval before it. Moreover, additional time-windows containing coherent noise coming from the surface were searched for and added as noise time-windows. The length of these event and noise windows are taken to be 30 s long, which we empirically found to be the most suitable length for most events in the region.

All time-windows undergo pre-processing before being used to train the CNN model. The time windows are detrended, demeaned, and bandpass filtered between 5–25 Hz. In agreement with the analyses of Wyer et al. [22] and Mukuhira et al. [6], we find this band of frequencies to be most suitable in removing ambient noise without much effect on the signal of interest, making the events more obvious to detect. Next, we down-sample the data from 200 Hz to 100 Hz to reduce the data size by a factor of 2. This helps improve the efficiency of the training process as fewer time samples are needed to be processed by the CNN model. Finally, these time-windows are manually verified to remove any window that does not fit its corresponding label.

Furthermore, to help the CNN model generalize better to variations in the test data, we augment our training event windows by adding time shifts such that the first-break time is varied with respect to the start of the window. This step is necessary to avoid any potential bias towards events that have first-breaks at particular time-samples within an event window. For each event window, we generate an additional sixteen windows by moving the window back and forth with a randomly generated time-shift. Figure 4 shows a signal time-window with three of the seventeen total windows coming from a single original event window. Without data augmentation, this bias may affect the performance of the CNN model when testing on a continuous record since, in a continuous record, the signal can arrive at any time sample in a given window. The same data augmentation technique is applied to noise windows as well. Finally, the waveforms from each of the four sensor levels are normalized individually to have a maximum amplitude of unity.

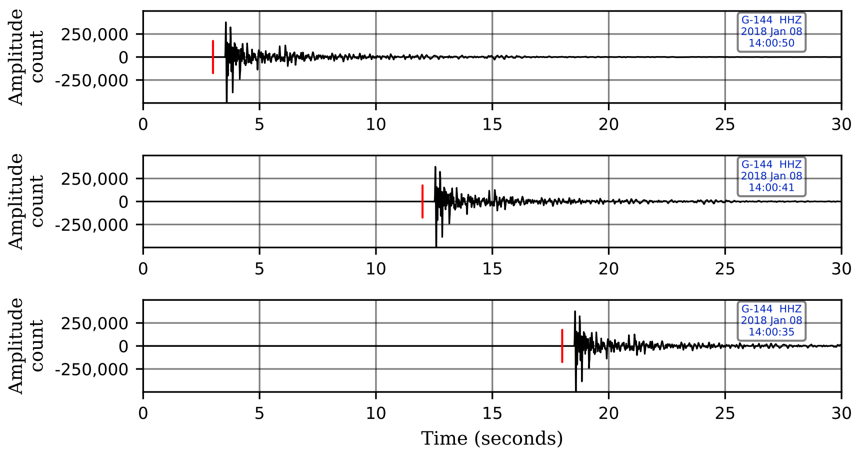

Figure 4. Three examples of the shifts done on an event time-window. Each event window is taken 17 times with 17 different placements of the arrival time to the window start: 3 s shift (**top**), 12 s (**middle**), 18 s (**bottom**). Red lines indicate the first-arrival times.

The output after the afore-mentioned pre-processing workflow is a 30 s time-window labeled into either signal or noise. The shape of each training data is a 3D matrix whose dimensions are: 4 (number of velocity geophones) $\times$ 3001 (time-window length $\times$ sampling rate) $\times$ 3 (the 3-components of each sensor). This is an analogous set up to the one used in the classification of color images, where the first two dimensions are the image length and width, and the third dimension represents the three color channels (red, green, and blue).

The total number of labeled examples is 67,847, with 21,624 belonging to the event class and 46,223 belonging to the noise class. These labeled examples are split according to the following distribution: 60% for training, 20% for validation, and 20% for testing.

2.2. CNN Architecture

Convolutional neural networks (CNNs) are inspired by studies on the visual cortex of cats and monkeys [23,24]. A CNN is a neural network with multiple hidden layers that uses convolutions to generate features hierarchically. Its main advantage over a feed-forward neural network is that it does not require tedious feature selection and engineering of raw data before feeding to the network. Moreover, a CNN can work with inputs in their original shape (whether it is 2D, 3D, or even 4D) without the need to flatten them into a 1D vector as required by feed-forward neural networks. This helps in locating spatial variations in multi-dimensional data better. Furthermore, due to the filters in its convolutional layers, a CNN shows higher efficiency in detecting local features within data compared to a feed-forward neural network.

A convolutional layer is a key part of a CNN, where feature extraction takes place. The layer extracts features from data by convolving filters of small dimensions with the input data. Then, the result of this convolution process is passed through a non-linear activation function. The output after the activation function is referred to as a feature map. A feature map carries important information from the input data after being filtered. A convolution layer can produce tens of feature maps, each containing a certain trait of the data that helps in efficiently performing the task at hand, be it a classification or a regression problem.

The convolutional layer produces a huge amount of data that need to be scaled down for computational efficiency without losing important information before proceeding further. Therefore, each convolutional layer is followed by a max-pooling layer. This layer reduces data dimension by taking the maximum value of neighboring inputs in the feature maps. Getting the maximum ensures that the most important values are passed on. The reduced output of the pooling layer is passed to the subsequent convolutional layer to obtain more abstract features, followed by another pooling layer, and so on. This sequence of convolutional and pooling layers continues as many times as necessary to build highly complex features. Then, the output of the final pooling layer is flattened and passed into fully connected layers before passing on to the final layer that outputs the result. The convolutional filter coefficients and network weights are learned through the process of backpropagation [25].

Our proposed CNN architecture consists of 3 convolutional layers. Each of them is followed by a max-pooling layer. The output of the last pooling layer is flattened and passed into two fully connected layers, followed by the output layer, which consists of a single neuron outputting whether the input is an event (1) or noise (0). Table 1 shows a summary of the CNN architecture, and Figure 5 shows the network pictorially. This architecture was chosen empirically after testing several architectures.

Table 1. A detailed summary of the CNN architecture, showing the number and dimensions of filters and neurons at each layer of the CNN.

Layer	Input Dimension	Number of Filters	Kernel Dimension	Output Dimension
Conv Layer 1	$4 \times 3001 \times 3$	16	1×8	$4 \times 3001 \times 16$
Pool Layer 1	$4 \times 3001 \times 16$	-	1×2	$4 \times 1500 \times 16$
Conv Layer 2	$4 \times 1500 \times 16$	16	4×8	$1 \times 1500 \times 16$
Pool Layer 2	$1 \times 1500 \times 16$	-	1×8	$1 \times 187 \times 16$
Conv Layer 3	$1 \times 187 \times 16$	64	1×4	$1 \times 187 \times 64$
Pool Layer 3	$1 \times 187 \times 64$	-	1×2	$1 \times 93 \times 64$
Flatten	$1 \times 93 \times 64$	-	-	5952×1
Dense 1	5952×1	500	-	500×1
Dense 2	500×1	80	-	80×1
Output Layer	80×1	1	-	1

The activation function used in all layers, except for the final layer, is the rectified linear unit function (ReLU). The final layer uses the sigmoid activation function. We use the Adam optimizer, and the loss function used to compute the error is the binary cross-entropy. The CNN model is implemented using the Tensorflow library [26] in Python. The model's performance is evaluated by calculating the accuracy of its predictions on all datasets (training, validation, & testing). The network stops training when the validation accuracy ceases to improve for eight consecutive epochs. The CNN model then gets the weights of the epoch with the best validation accuracy.

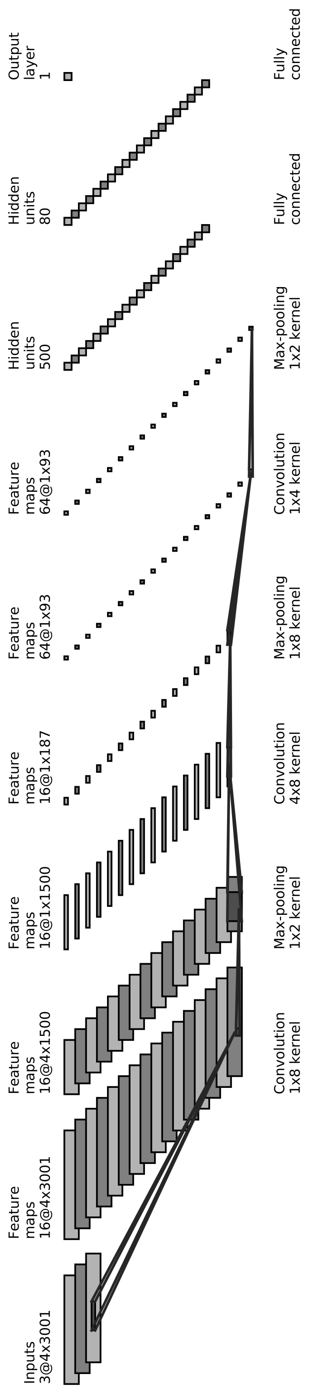

Figure 5. The proposed CNN architecture starting with 3 convolutional layers, each followed by a max-pooling layer. The output of the final pooling layer is flattened and passed into two fully connected layers before outputting the result: event (1) or noise (0).

2.3. Performance Evaluation on Continuous Data

To evaluate the performance of the trained model on continuous data, the model is tested on recordings from five stations (G09, G10, G14, G18, G19) during the months of December 2017 and January 2018. In these two months, 23 events are recorded in the catalog, including a major event near Zeerijp (M_L = 3.4) on January 8. The chosen five stations are the closest to the epicenter of this major event. Figure 6 shows locations of the cataloged events that occurred during the two-month period and the G-network stations. We highlight, in red, the five stations whose records are used for performance evaluation.

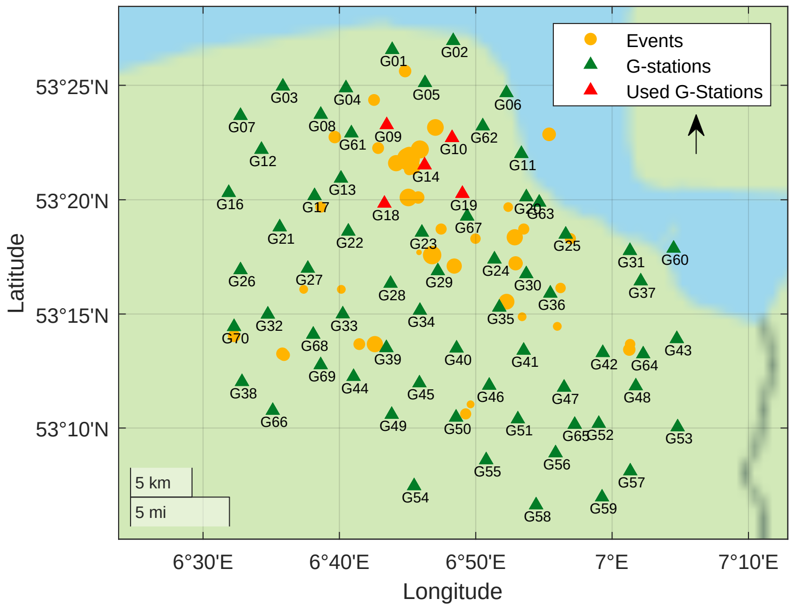

Figure 6. Map of the G-network stations (green and red) and the 23 cataloged events (orange) that occurred during December 2017 and January 2018. The selected five stations (shown in red) are used to test the trained CNN model on a two-month continuous record.

The two-month continuous record is cut into overlapping 30-s time-windows. Each subsequent 30-s window starts 10 s after the start time of the preceding time-window, resulting in a 66.67% overlap with the preceding window. This ensures that the CNN model does not miss any event that may get split between two consecutive windows. Every time-window undergoes the same pre-processing steps that were used for the training data (detrending, demeaning, bandpass filtering, downsampling by a factor of 2, and amplitude normalization to unity). We require a given time-window to be flagged as an event class on at least two of the five stations for it to be classified as an event. This helps us reduce the possibility of false alarms and ensures the robustness of the CNN predictions. We use the available event catalog and manual verification to analyze the accuracy of these predictions.

For comparison, we run STA/LTA with several detection thresholds on the same continuous data coming from the selected five stations. The four velocity geophones at each station are stacked to enhance the SNR. This stacking is done to improve the STA/LTA's detection accuracy. Similar to CNN, to be classified as an event, we require at least two of the five stations to cross a given threshold. The findings of the STA/LTA are classified into true events, uncertain events, and false detections. The uncertain class refers to time-windows that are categorized as events by STA/LTA but are unverifiable visually due to low SNR data.

Template matching is also run on the same stations for the two-month period. It uses the 23 cataloged events during the two-month period as master events to search for undetected events. Template matching results are also compared to the CNN's results in the following section.

3. Results

In this section, we provide training details of the CNN architecture, outlined in Section 2.2, and its performance on training, validation, and test datasets. This is followed by the analysis of the trained network's performance on a two-month continuous record and its comparison with the STA/LTA and template matching algorithms.

The network is trained with an Adam optimizer using mini-batch optimization with a batch size of 64. The training stops after 14 epochs as validation accuracy ceases to improve for eight consecutive epochs. Figure 7 shows the accuracy and the loss of the training and validation datasets versus the training epochs. We observe high accuracy for both the training and validation datasets even after the first epoch as we use a small batch size to help with the convergence speed of the network. We obtain 100% accuracy on the training, validation, and testing datasets.

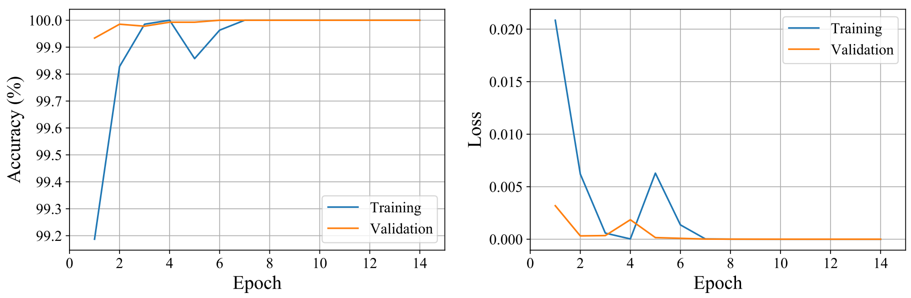

Figure 7. The accuracy and loss of training and validation datasets vs. epochs. The network converges after a few epochs to the optimum solution.

The network is then tested on two months of continuous data at the five stations near Zeerijp, indicated with red triangles in Figure 6. The trained CNN model flags a total of 45 time-windows as events. Upon manual verification and comparison with the existing catalog, we find 40 of those to be true events while 5 were found to be false detections. It is worth noting that all the false detections were identified as events on only two of the five stations, which is the minimum threshold set for identifying an event. It is possible to reduce the number of false detections by increasing the threshold; however, it will result in missing a few true events. This observation also points to the direction that a human expert may be used to manually verify only the borderline cases. This may allow using the human attention efficiently without compromising on the accuracy of detections.

The trained CNN model picks 20 of the 23 cataloged events that were reported during the two-month period. The three unpicked cataloged events are actually very low-magnitude events (M_L = 0.4, 0.5 & 0.7) with epicenters considerably far away from the selected stations. Upon manually looking at the five selected stations at these reported time-windows, we find the signals to be undetectable visually as they were buried under noise. Missing these events using the selected five stations is understandable since our CNN model was trained using events that we can visually confirm, and therefore, it is unable to identify such events. However, the trained network picks 20 more low-magnitude events that were not cataloged. These events were inspected visually and found to be true events. Figure 8 shows the Z-component records of two of these uncataloged events with low SNR data that were picked by our CNN model. Table 2 details the 45 picked time-windows and their classifications.

Table 2. CNN detections with cataloged events highlighted in green color, uncataloged in yellow, and false alarms in red.

Time Windows	Number of Stations That Picked the Event	Class	Comments
2017-12-01 11:33:30	5	Event	Cataloged
2017-12-01 21:05:40	5	Event	Cataloged
2017-12-02 09:00:30	5	Event	Cataloged
2017-12-02 11:23:20	2	Event	Not Cataloged
2017-12-03 16:52:50	2	False Alarm	-
2017-12-06 23:29:00	3	Event	Cataloged
2017-12-10 15:55:30	2	False Alarm	-
2017-12-10 16:48:30	5	Event	Cataloged
2017-12-13 00:43:50	3	Event	Not Cataloged
2017-12-14 03:38:50	4	Event	Not Cataloged
2017-12-15 20:55:40	5	Event	Cataloged
2017-12-16 02:17:50	2	Event	Not Cataloged
2017-12-17 07:01:30	5	Event	Not Cataloged
2017-12-20 18:26:40	2	Event	Not Cataloged
2017-12-21 19:39:00	4	Event	Cataloged
2017-12-22 19:40:20	5	Event	Cataloged
2017-12-22 20:06:10	5	Event	Cataloged
2017-12-22 23:41:00	5	Event	Not Cataloged
2017-12-24 07:05:10	4	Event	Not Cataloged
2017-12-24 17:49:50	3	Event	Cataloged
2017-12-25 12:52:10	3	Event	Cataloged
2017-12-28 14:00:30	4	Event	Cataloged
2017-12-28 18:02:30	4	Event	Not Cataloged
2017-12-29 19:59:50	5	Event	Not Cataloged
2017-12-29 20:53:30	2	Event	Not Cataloged
2017-12-29 23:15:40	5	Event	Cataloged
2017-12-31 02:52:10	2	False Alarm	-
2017-12-31 03:03:20	2	False Alarm	-
2018-01-01 14:46:50	4	Event	Cataloged
2018-01-02 09:34:30	4	Event	Not Cataloged
2018-01-08 14:00:50	5	Event	Cataloged
2018-01-08 16:58:10	4	Event	Not Cataloged
2018-01-09 15:46:40	5	Event	Cataloged
2018-01-12 16:20:10	3	Event	Not Cataloged
2018-01-13 10:29:30	4	Event	Not Cataloged
2018-01-13 10:30:00	3	Event	Not Cataloged
2018-01-17 04:37:40	4	Event	Cataloged
2018-01-19 06:49:00	2	Event	Not Cataloged
2018-01-20 08:19:20	2	Event	Cataloged
2018-01-21 06:06:20	5	Event	Not Cataloged
2018-01-22 07:23:10	3	Event	Cataloged
2018-01-24 17:04:10	3	Event	Not Cataloged
2018-01-27 12:54:20	2	False Alarm	-
2018-01-29 02:57:10	2	Event	Not Cataloged
2018-01-31 13:11:00	5	Event	Cataloged

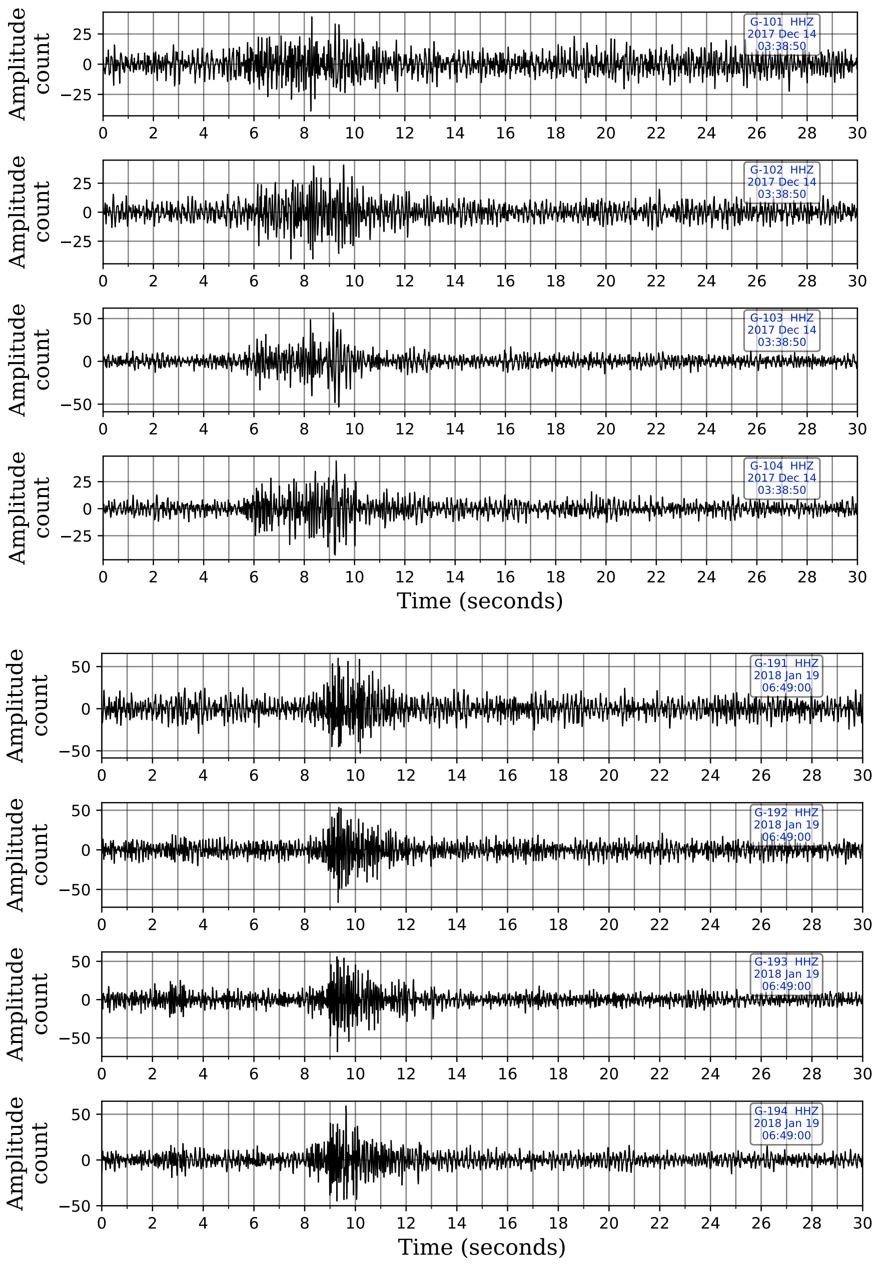

Figure 8. Z-component waveforms at the four sensor levels for stations G10 and G19 from two of the uncataloged events picked by our trained CNN model. These two events were neither picked by STA/LTA nor template matching. Picking of these two events, despite their low-magnitude and relatively low SNR data, shows the efficacy of the trained CNN model. The waveforms are displayed in sequence for each station, with the waveform from the shallowest sensor at the top and the one from the deepest sensor at the bottom.

Moving on to the comparison with conventional techniques, it is well-known that the sensitivity of STA/LTA predictions is highly dependent on the chosen threshold value. Low threshold values lead to a lower risk of missing true events at the cost of getting a higher number of false detections. On the contrary, a threshold value that is too high may avoid those false detections but leads to missing true events. Therefore, we apply STA/LTA on the two-month continuous record for a range of threshold values. Figure 9 shows the number of true events, uncertain events (not verifiable manually), and false detections for a range of threshold values. For the threshold value of 60, we get 39 true events, 14 uncertain events, and 917 false detections. Despite using a low threshold value, STA/LTA was able to pick only 14 out of the 23 cataloged events. Besides the cataloged events, STA/LTA also picks 8 true events that were also identified by CNN and 16 weak events that were not picked by the CNN model. However, the extremely high number of false detections renders this approach to be unacceptable.

On the other end of the spectrum, i.e., for a threshold value of 120, we manage to reduce the number of false detections to only three, but the true picks are also substantially reduced to only six, missing many cataloged events. Figure 10 shows the detailed classification of the STA/LTA event detections at each threshold value, highlighting cataloged or uncataloged events and whether these events were picked by our CNN model or not.

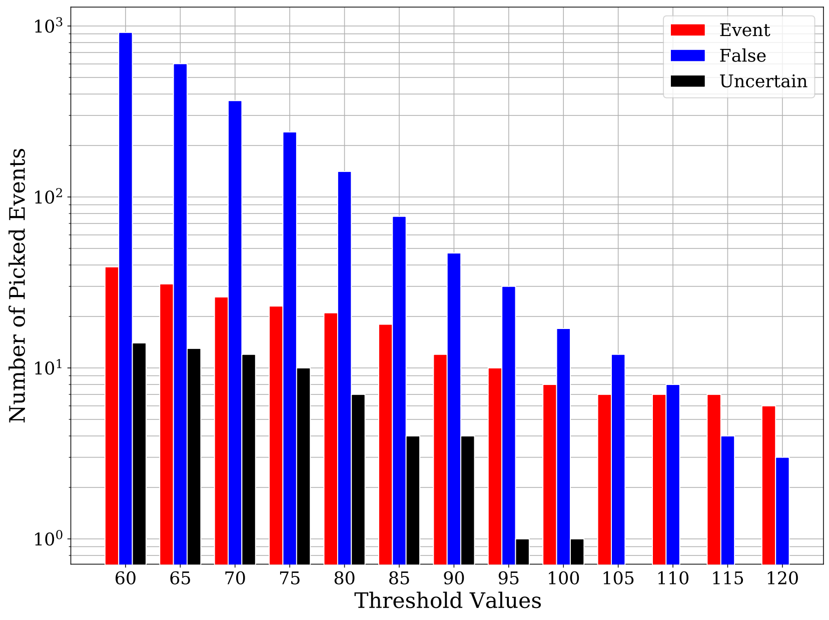

Figure 9. A barplot of STA/LTA detections with their classification at different threshold values. Using low threshold values STA/LTA picks numerous false detections, while stricter threshold values result in missing lots of true events.

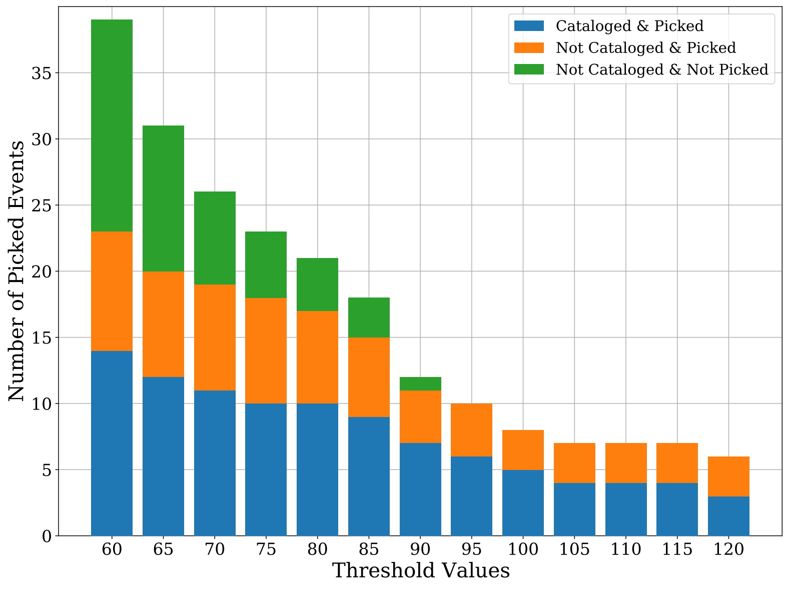

Figure 10. A barplot showing the detailed classes of events picked by STA/LTA for different threshold values. The categorization shows how many of these events have been cataloged/uncataloged and whether they were picked by our CNN model or not.

To compare the CNN and STA/LTA detection performance, we compute the precision value of the CNN and STA/LTA detections. Precision is the ratio of the true positive findings to the summation of the true and false positives. For STA/LTA, we exclude the uncertain events from computing the precision, which is computed using the following formula:

$$Precision = \frac{True\ Events}{True\ Events + False\ Events}.$$

We evaluate the precision of the CNN model on the two-month dataset to be 88.9%, while the STA/LTA detection precision does not exceed 67% on any of the threshold values tried. STA/LTA only reaches the 67% precision value with the highest threshold used, where it detects only six events. Figure 11 shows the precision of STA/LTA together with the number of the detected events at each threshold value. Since STA/LTA relies on sudden changes in amplitude to differentiate an event from noise, the under-performance of the algorithm in detecting low-magnitude earthquakes is understandable. Since weak events and coherent noise yield similar waveform signatures, the STA/LTA algorithm ends up either detecting many false alarms or missing true events, depending on the chosen threshold value. Moreover, as opposed to the CNN model, it can neither recognize the polarization of the seismic events on the 3-component data nor the difference between moveout patterns across multi-level geophones.

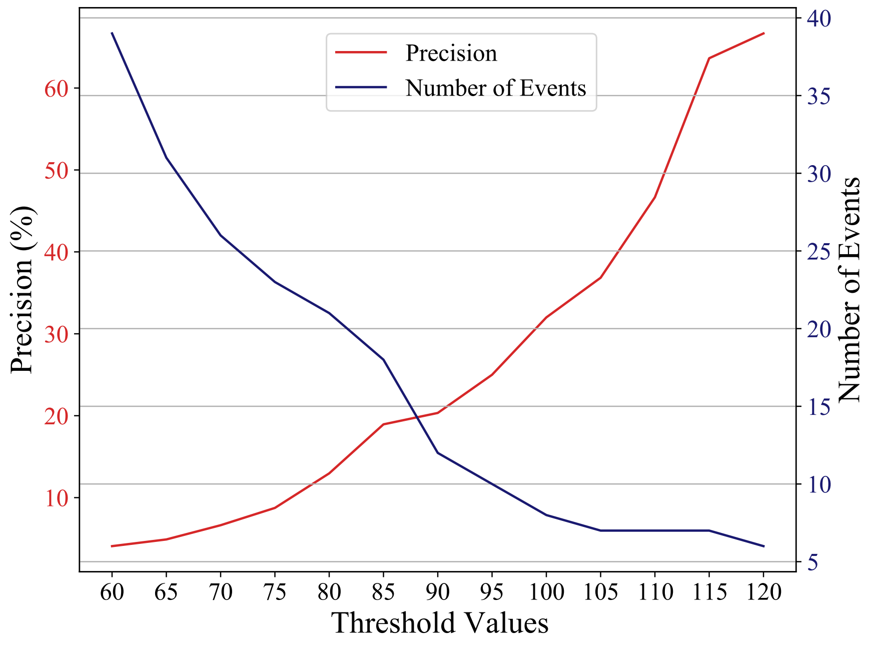

Figure 11. Precision of STA/LTA along with the number of the detected events at each threshold value. Using a higher threshold value increases the precision but at the expense of reducing the number of detected true events. Even with the highest threshold value, the STA/LTA precision (67%) remains less than the CNN precision (88.9%), and it picks only 6 events for this case compared to 40 events picked by the CNN model.

We also analyze the detection performance using template matching. Since it uses all the 23 cataloged events as master events, it finds them all during the two-month period, which is unsurprising. In addition, template matching detects 11 uncataloged events. This is almost half the number of uncataloged detections by the CNN model. Moreover, template matching yields 15 false detections, resulting in a precision value of 69.39%, which is considerably lower than the precision of our CNN model (88.9%).

Table 3 lists the uncataloged events captured by template matching and compares them with the uncataloged events detected by the CNN model. We note that the CNN model detects 9 of the 11 events identified by template matching as well, i.e., template matching picks 2 events that are not picked by the CNN model. These two events are at 06:19:50.809 h on 1 December 2017, and 14:44:32.929 h on 28 December 2017. Investigation of these two events reveals that the first event is not picked by the CNN model because of the two-station requirement to flag an event. This event is visually detectable on only one station and is correctly picked at this station by the CNN. However, it is not flagged as an event because it was buried under noise at other stations. The second event, shown in Figure 12, is buried under noise at all five stations. Since the CNN model was trained only on events that were visually detectable, it is understandable that it missed these events.

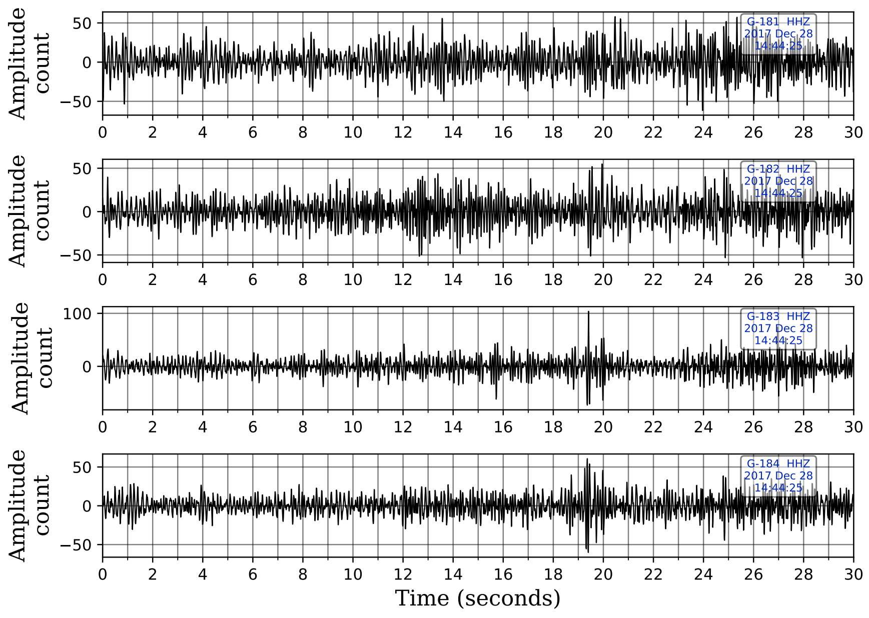

Figure 12. An event detected by template matching and missed by our CNN model at 14:44:32.929 h on 28 December 2017. The event is buried under noise, and the network was not trained on such event waveforms. Only the vertical waveforms are shown for illustration. The waveforms are displayed in sequence with the waveform from the shallowest sensor at the top and the one from the deepest sensor at the bottom.

Table 3. Summary of uncataloged events picked by template matching and the trained CNN model. Events picked by both template matching and CNN are highlighted in green. Events picked by template matching only are highlighted in yellow, and events picked by CNN only are highlighted in blue.

Events	CNN Detected	Template Matching Detected
2017-12-01 06:19:50	No	Yes
2017-12-02 11:23:20	Yes	No
2017-12-13 00:43:55	Yes	Yes
2017-12-14 03:38:50	Yes	No
2017-12-16 02:17:54	Yes	Yes
2017-12-17 07:01:30	Yes	No
2017-12-20 18:26:40	Yes	No
2017-12-22 23:41:05	Yes	Yes
2017-12-24 07:05:17	Yes	Yes
2017-12-28 14:44:32	No	Yes
2017-12-28 18:02:30	Yes	No
2017-12-29 20:00:01	Yes	Yes
2017-12-29 20:53:30	Yes	No

Table 3. *Cont.*

Events	CNN Detected	Template Matching Detected
2018-01-02 09:34:36	Yes	Yes
2018-01-08 16:58:20	Yes	Yes
2018-01-09 20:49:00	Yes	Yes
2018-01-12 16:20:10	Yes	No
2018-01-13 10:29:30	Yes	No
2018-01-19 06:49:00	Yes	No
2018-01-21 06:06:30	Yes	Yes
2018-01-24 17:04:10	Yes	No
2018-01-29 02:57:10	Yes	No

4. Discussion and Conclusions

We developed a CNN-based algorithm for detecting low-magnitude events in Groningen using a multi-level sensor network. We find the performance of the trained CNN model to be considerably superior in comparison with STA/LTA and template matching. We test the performance of these methods on a two-month dataset around the relatively large $M_L = 3.4$ event on 8 January 2018. While many other deep learning based techniques have also been developed in recent times, our proposed methodology takes advantage of the multi-level sensor network in discriminating between true events arriving from the subsurface and local noise coming from the surface by training the network to distinguish between moveout patterns at the borehole sensors. Moreover, unlike template matching, the method is not sensitive to only the waveform signatures of the master events. Furthermore, once the CNN model is trained, it can provide detection results in real-time, which is also an important consideration for hazard mitigation in microseismic monitoring.

Prior deep learning studies on the problem of seismic event detection [13,15,27] used hundreds of thousands to millions of labels to train their networks. In this study, we show that remarkably high detection performance can be achieved by using a relatively small amount of data. We used only 3991 labels that were further augmented to create 17 times more training data. This can be attributed to the idea of using the moveout at the multi-level sensor network as a discriminating feature, allowing us to gain maximum leverage from such a relatively small amount of data, which also saves us time in manual labeling and verification of the training data.

However, one downside of our proposed CNN model is its lack of ability to detect events buried under noise. This characteristic is due to the data used to train the network. We only used data that a human interpreter could visually confirm as events and, therefore, it did not include events that were masked by noise. One approach to overcome this limitation is to prepare training data using template matching since it has the capability to detect events even in a negative SNR regime. This also highlights an important direction for deep learning practitioners where existing tools and methodologies can be used to further improve the capabilities of these modern learning algorithms instead of viewing them solely as competing approaches.

Author Contributions: Conceptualization, U.b.W. and M.F.; methodology, U.b.W., M.F. and A.S.; software, A.S., M.F., L.S., and U.b.W.; validation, U.b.W., L.S., M.F., and A.S.; formal analysis, A.S., U.b.W., M.F. and L.S.; investigation, U.b.W. and A.S.; resources, U.b.W.; data curation, M.F. and A.S.; writing—original draft preparation, A.S. and U.b.W.; writing—review and editing, U.b.W., S.H., M.F. and L.S.; visualization, A.S. and U.b.W.; supervision, U.b.W. and M.F.; project administration, U.b.W.. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All the data used in this study is available on the seismic and acoustic data portal of the Royal Netherlands Meteorological Institute (KNMI) http://rdsa.knmi.nl/dataportal/.

Acknowledgments: We acknowledge the support by the College of Petroleum Engineering and Geosciences at KFUPM. We extend gratitude to Stewart Greenhalgh for useful discussions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Brodsky, E.E. The importance of studying small earthquakes. *Science* **2019**, *364*, 736–737. [CrossRef] [PubMed]
2. Allen, R.V. Automatic earthquake recognition and timing from single traces. *Bull. Seismol. Soc. Am.* **1978**, *68*, 1521–1532. [CrossRef]
3. Shelly, D.R.; Beroza, G.C.; Ide, S. Non-volcanic tremor and low-frequency earthquake swarms. *Nature* **2007**, *446*, 305–307. [CrossRef] [PubMed]
4. Yoon, C.E.; O'Reilly, O.; Bergen, K.J.; Beroza, G.C. Earthquake detection through computationally efficient similarity search. *Sci. Adv.* **2015**, *1*, e1501057. [CrossRef]
5. Poliannikov, O.V.; Fehler, M.C. Instantaneous phase-based statistical method for detecting seismic events with application to Groningen gas field data. In *SEG Technical Program Expanded Abstracts 2018*; Society of Exploration Geophysicists: Tulsa, OK, USA, 2018; pp. 2907–2911.
6. Mukuhira, Y.; Poliannikov, O.V.; Fehler, M.C.; Moriya, H. Low-SNR Microseismic Detection Using Direct P-Wave Arrival Polarization. *Bull. Seismol. Soc. Am.* **2020**, *110*, 3115–3129. [CrossRef]
7. Akram, J.; Ovcharenko, O.; Peter, D. A robust neural network-based approach for microseismic event detection. In *SEG Technical Program Expanded Abstracts 2017*; Society of Exploration Geophysicists: Tulsa, OK, USA, 2017; pp. 2929–2933.
8. Qu, S.; Verschuur, E.; Chen, Y. Automatic microseismic-event detection via supervised machine learning. In *Seg Technical Program Expanded Abstracts 2018*; Society of Exploration Geophysicists: Tulsa, OK, USA, 2018; pp. 2287–2291.
9. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
10. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
11. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
12. Kong, Q.; Trugman, D.T.; Ross, Z.E.; Bianco, M.J.; Meade, B.J.; Gerstoft, P. Machine learning in seismology: Turning data into insights. *Seismol. Res. Lett.* **2019**, *90*, 3–14. [CrossRef]
13. Perol, T.; Gharbi, M.; Denolle, M. Convolutional neural network for earthquake detection and location. *Sci. Adv.* **2018**, *4*, e1700578. [CrossRef] [PubMed]
14. Keranen, K.M.; Savage, H.M.; Abers, G.A.; Cochran, E.S. Potentially induced earthquakes in Oklahoma, USA: Links between wastewater injection and the 2011 Mw 5.7 earthquake sequence. *Geology* **2013**, *41*, 699–702. [CrossRef]
15. Mousavi, S.M.; Zhu, W.; Sheng, Y.; Beroza, G.C. CRED: A deep residual network of convolutional and recurrent units for earthquake signal detection. *Sci. Rep.* **2019**, *9*, 1–14.
16. Mousavi, S.M.; Ellsworth, W.L.; Zhu, W.; Chuang, L.Y.; Beroza, G.C. Earthquake transformer—An attentive deep-learning model for simultaneous earthquake detection and phase picking. *Nat. Commun.* **2020**, *11*, 1–12. [CrossRef] [PubMed]
17. Othman, A.; Iqbal, N.; Hanafy, S.M.; Waheed, U.B. Automated Event Detection and Denoising Method for Passive Seismic Data Using Residual Deep Convolutional Neural Networks. *IEEE Trans. Geosci. Remote Sens.* **2021**. [CrossRef]
18. van Thienen-Visser, K.; Breunese, J. Induced seismicity of the Groningen gas field: History and recent developments. *Lead. Edge* **2015**, *34*, 664–671. [CrossRef]
19. Netherlands to halt Groningen gas production by 2022. Avaliable online: https://www.reuters.com/article/us-netherlands-gas-idUSKCN1VV1KE (accessed on 27 October 2021).
20. Dost, B.; Ruigrok, E.; Spetzler, J. Development of seismicity and probabilistic hazard assessment for the Groningen gas field. *Neth. J. Geosci.* **2017**, *96*, s235–s245. [CrossRef]
21. KNMI. *Netherlands Seismic and Acoustic Network. Royal Netherlands Meteorological Institute (KNMI)*; Other/Seismic Network; KNMI: De Bilt, The Netherland, 1993.
22. Wyer, P.; Fehler, M.; Poliannikov, O.; Mukuhira, Y.; deMartin, B.; Nakata, N. Comparative Study of Low-magnitude Earthquake Detection Techniques for Use with Dense Seismic Monitoring Arrays: Applications for Induced Seismicity Monitoring. In *AGU Fall Meeting Abstracts*; American Geophysical Union: San Francisco, CA, USA, 2019; Volume 2019, p. S11F-0391.
23. Hubel, D.H.; Wiesel, T.N. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.* **1959**, *148*, 574–591. [CrossRef] [PubMed]
24. Hubel, D.H.; Wiesel, T.N. Receptive fields and functional architecture of monkey striate cortex. *J. Physiol.* **1968**, *195*, 215–243. [CrossRef] [PubMed]
25. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. *Learning Internal Representations by Error Propagation*; Technical report; California Univ San Diego La Jolla Inst for Cognitive Science: La Jolla, CA , USA 1985.
26. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Avaliable online: https://www.tensorflow.org/ (accessed on 27 October 2021).

27. Ross, Z.E.; Meier, M.A.; Hauksson, E. P wave arrival picking and first-motion polarity determination with deep learning. *J. Geophys. Res. Solid Earth* **2018**, *123*, 5120–5129. [CrossRef]