Utrecht University

Predicting Peak Ground Velocity of real-time seismic data from Southern California using Machine Learning

Master thesis - Tessa van der Wel
6443765

March 8, 2024

Supervisors - Prof. dr. Jeannot Trampert and dr. Elmer Ruigrok

Abstract

The peak ground velocity (PGV) is the maximum velocity of the shaking ground during an earthquake. Predicting this PGV can be very useful to reduce damage in affected areas. With accurate PGV predictions, an Early Earthquake Warning (EEW) system can be created, which can give people the opportunity to prepare for significant ground shaking. There is a lot of data available in seismology which makes it very suitable for machine learning. Machine learning is a sub-field of artificial intelligence (AI) in which models are created to learn specific relations in data through training. Machine learning is a tool which has been used before for predicting seismic activity, with promising results. In this research, four different machine learning models are trained and tested to predict PGV of real-time seismic data from Southern California. This prediction is based on the first few seconds of seismic data obtained from 5 seismic receivers in the area. This seismic data is combined with the geographical locations of these receivers and of the prediction location. The different models should learn the spatial dependence of wave propagation, the underlying physics and the geological structures in the area. With combining seismic data with spatial information, it is expected that the models can not only predict PGV on receiver locations, but also in arbitrary locations in the training area. For training and testing the models, data is used from events with a magnitude higher than 4.0 between the 1st of January 2016 and the 15th of November 2023. This gives 143 events which are detected by at least 1 of the 269 receivers used in this research in Southern California. The four different models trained in this research are a Random Forest (RF), a Mixture Density Network (MDN) and two Convolutional Neural Networks (CNNs). The MDN is by far the worst performing model and is with the method used in this research not suitable for predicting PGV. Both CNNs are performing very similarly. When they have to predict data from events they have seen during training, their performance is pretty good, however this worsens drastically when they have to predicts new events. This makes them also, with the used method, not suitable for PGV prediction. The RF model does seem to understand how waves are propagating through the surface, when it predicts PGV on receiver locations. However, when the model has to predict PGV on different locations it does not show the expected results. The RF has a structure which can handle outliers of noise in data very well, which is properly why it is performing much better that the other created models. A disadvantage is that this structure also makes it unclear where the decision of the RF are based on and what the model qualifies as important.

Contents

1 Introduction

Earthquakes are a great hazard to humans in seismically active regions. In February 2023, Turkeye and Syria were struck by a devastating earthquake (Oliver Holmes & Sheehy, 2023). Later in September 2023, also large parts of Morocco were destroyed by a heavy earthquake (Times, 2023). In total 2.7 billion people live in regions where seismic activity can cause damage (Marti et al., 2019). This causes earthquake forecasting to be a very important subject. To be able to forecast ground shaking intensity, ground motion models are essential (Mohammadi et al., 2023). When an earthquake occurs, the P-wave is the first seismic wave to arrive at a certain location. This longitudinal wave has a relatively low amplitude, which causes limited damage. After the P-wave arrival, a transverse S-wave or surface wave arrives, which has a larger amplitude and includes the peak ground motion or the peak ground velocity (PGV) (Allen & Kanamori, 2003). The peak ground velocity (PGV) is the maximum velocity of the shaking ground during an earthquake, at a certain location. The P-wave, S-wave and the peak ground motions or PGV are shown in the seismogram of figure 1 from Allen and Kanamori (2003).

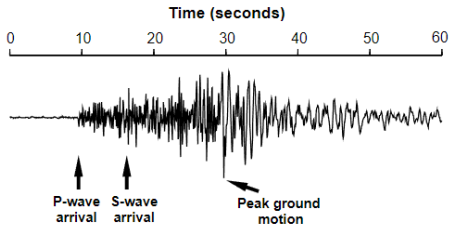


Figure 1: A typical seismic waveform for a local earthquake. On the x-axis time and on the y-axis the measured ground velocity. This horizontal ground motion is measured at a station in southern California, 50 km from the epicenter from an earthquake with magnitude 3.9. The arrows point to the P-wave arrival, S-wave arrival and the Peak ground motion or peak ground velocity (PGV). This figure is from Allen and Kanamori (2003).

Earthquake prediction has been a subject of interest for seismologists since a very long time (Mogi, 1985; Rikitake, 1968; Turcotte, 1991). Almost 60 year ago, Rikitake (1968) already hoped that earthquake prediction would be possible in the near future with new techniques that would develop over time. However, today even with all the new technologies, to predict the location, time and magnitude, real-time earthquake prediction is still not achieved. While it is still not possible, it is interesting to look into the prediction of PGV, after an earthquake happened for example, to enable an Earthquake Early Warning system.

In October 2007, a nation wide Earthquake Early Warning (EEW) system was launched in Japan (Hoshiba et al., 2008). An EEW system is different from an earthquake prediction system. This system provides an advance notice to individuals situated at a specific distance from the earthquake's epicenter before significant ground shaking reaches their location. This system therefore sends a message if an earthquake has already been detected. The time that individuals have before significant seismic energy reaches their location

depends on the distance between the epicenter and their location. However, this time is actually always very short (Hoshiba et al., 2008). The destroying transverse S-wave arrives, approximately 1 second per 4 km from the epicenter after the seismic event. Nevertheless, this time can be enough for individuals to search for a saver spot, or for trains, traffic and for example elevators to react, to prevent accidents and also surgical operations can be stopped or postponed (Hoshiba et al., 2008). The EEW system that is currently active in Japan is based on empirical estimations, where human action is necessary (Hoshiba et al., 2008).

There is a lot of data available in seismology from all over the world, since there is a widespread network of seismic receivers from which data is often freely available for public use. This enormous amount of data makes seismology a suitable tool for machine learning. Machine learning is a sub-field of artificial intelligence (AI), in which computers are able to acquire knowledge autonomously, so without human intervention, and extracting patterns from raw data related to real-world problems (Goodfellow et al., 2016). The performance of a machine learning system enhances when more data is fed into the system. The primary goal is to create a learning algorithm that makes a model from the input data (Zhou, 2021). This learning algorithm forms a so-called neural network. The structure of the neural networks depends on how the model is trained and how the input data is given to the model. A large set of data, which describes the problem, is necessary for the neural network to understand the underlying physics of the problem (Siahkoohi et al., 2018). The very intensive data availability of seismology is very suitable for machine learning. Due to the huge amount of available data, seismology has become, over the years, a field where a multitude of techniques and tools have been devised for earthquake detection and for the study of the earth's structure (Kong et al., 2018). This huge amount of data does also require a different approach from traditional data analysis, described by three V's: volume, variety and velocity (Sagiroglu & Sinanc, 2013). For machine learning, the performance of a model often increases when more data, a bigger volume of data, is available for training the model. In addition, the model will perform better when it has processed a lot of different scenario's, which can be achieved with more variety in the data. Lastly, the faster data can be processed, velocity, the broader machine learning can be applied. The speed in which data is processed is crucial for real-time earthquake detection and an EEW system, because of the given short duration, as mentioned before (Kong et al., 2018).

Machine learning has been used before to estimate seismic activity and different machine learning algorithms have been tested for different seismic purposes. For example, Rouet-Leduc et al. (2017) used a Random Forest (RF) algorithm to predict the timing of the next "labquake", in laboratory slip experiments with high accuracy. The machine learning algorithm learned to identify an acoustic signal emitted by the fault in the laboratory, which was previously regarded to be noise. This research is an example that machine learning algorithms could recognise patterns that humans do not know yet or can not recognise.

A different application of machine learning combined with seismology is reported in the research from Perol et al. (2018). They developed the algorithm *ConvNetQuake*, a convolutional neural network (CNN) which is developed to improve the efficiency and

accuracy of local earthquake detection and location information from a single waveform. Lomax et al. (2019) adapted this algorithm, named *ConvNetQuake_INGV*, to characterize earthquakes at any distance. Their research focuses on using waveform from 1 single seismic station. Both Lomax et al. (2019) and Perol et al. (2018) aim to contribute insights to the potential of a convolutional neural network (CNN) for enhancing earthquake analysis.

Adeli and Panakkat (2009) developed a probabilistic neural network (PNN) to forecast the magnitude of larger earthquakes within a pre-defined future time frame in a seismically active region. This prediction is done based on eight mathematically computed parameters which are known as seismicity indicators, and is done for time periods of 15 days. This neural network could predict earthquakes with a magnitude between 4.5 and 6.0 quite well, however predicting the magnitude of larger earthquakes was not possible with this PNN.

Derakhshani and Foruzan (2019) developed an artificial neural network (ANN), for estimating ground motion parameters. These parameters are peak ground acceleration (PGA), peak ground displacement (PGD) and peak ground velocity (PGV). The models created in their research give reliable estimations of these motion parameters which can be used for different earthquake engineering applications. However, the neural networks trained in their research have as input also information of the event itself. Some input data is for example: the magnitude of the earthquake, the distance between the earthquake and the site and information about the fault plane. These input features are not always readily available, which makes that these models are not suitable for creating an EEW system.

The aim of this study is to train a machine learning model which can predict the peak ground velocity in quasi-real time. The novelty of this project is to use seismic records in real-time to predict peak ground velocity, rather than magnitude, location and other attributes of earthquakes. The peak ground velocity represents the highest recorded shaking speed at a specific location during an earthquake. With machine learning, a model can be trained, which can understand relations between certain variables. This training is done with a large set of training data from seismically records from southern California. This research location is chosen considering the dense receiver network and its history with numerous earthquakes. A large dataset is required for the network to understand the underlying physics, the relationships between variables and to work efficiently (Siahkoohi et al., 2018). The training process can lead to the creation of various types of models. In this research, different types of machine learning models are trained and subsequently compared to investigate which model performs the best for predicting PGV. The selected models are explained in more detail in the method section. The method used in this research is based on Kuijpers (2021), who did a similar research for the Groningen area.

We trained and tested different types of machine learning models, with real-time seismic data. Real-time seismic data can be complex. The study assesses the effectiveness and efficiency of machine learning models to determine which one yields the most accurate predictions for peak ground velocity.

2 Problem Description

California is a western state within the United States of America, located along the Pacific coast with more than 39 million residents. California experiences more than a hundred earthquakes per day. In the region, numerous small earthquakes occur, alongside some larger, destructive earthquakes with a moment magnitude larger than 6.5 (Hutton et al., 2010). In this highly populated area, accurate ground motion prediction can be very important and can be used for an Earthquake Early Warning (EEW) system (Meier, 2017). The initial concept of a system, capable of issuing early alerts for incoming ground shaking comes from Cooper (1868). His idea was never implemented, but it was based on the same concept that many modern EEW systems also follow: Information (the speed of electromagnetic signals is approximately 300,000 km/s) travels faster than seismic waves (just a few km per second) (Satriano et al., 2011). EEW systems can be divided into two groups. The first concept is called 'regional', this approach uses seismic data from the network situated near the epicentral area for rapid earthquake detection. The second concept is called 'onsite warning', where seismic sensors at the target region predict the impact of S-wave based on the P-wave that arrives in the region.

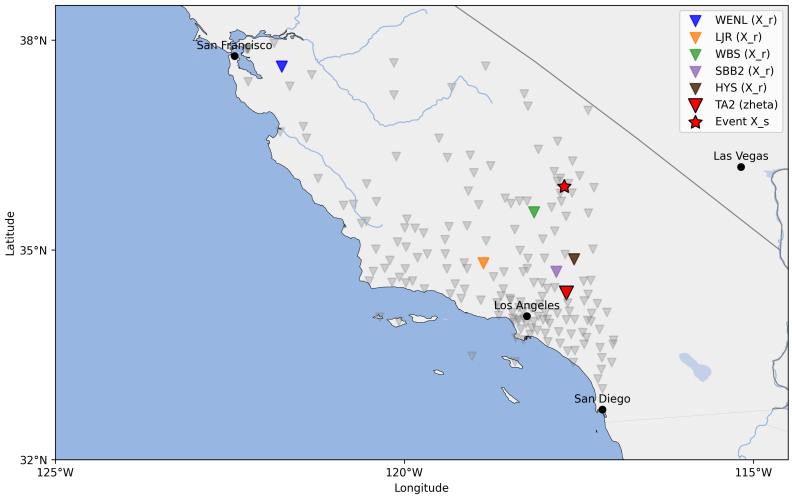
When an EEW system detects an earthquake, possible measures can be taken and residents can be informed about the possible danger. The goal of an EEW system is to estimate the damage potential before the waves reach a certain location (Satriano et al., 2011). Therefore, the system has to estimate this based on the information of the first stations that are reached by the earthquake. While predicting PGV and creating an EEW system is possible, earthquake prediction is not yet possible. Rouet-Leduc et al. (2017) successfully predict earthquakes accurately beforehand in the laboratory, but this not yet done with real-time seismic data. If it ever will be possible, is unknown, so while leaving the earthquake prediction for the possible future, it is also important to look for alternatives such as PGV predictions.

Since 1927, seismic stations are placed in all southern California to register and study local earthquakes. Today there are over 160 stations in the receiver network (Hutton et al., 2010). The significant source of major earthquakes in California is the San Andreas fault zone. The San Andreas fault is a transform fault which is orientated from north to south California, shown in figure 2. The upper 15 km of the fault exhibits brittle behavior and is very seismically active (Barruol & Mainprice, 1993). Not all earthquakes are located along the San Andreas fault, also the region around the fault is really active. The vast majority of all earthquakes in the region are not deeper than 15 km (thickness of brittle layer), with sometimes an exception of a slightly deeper earthquake.

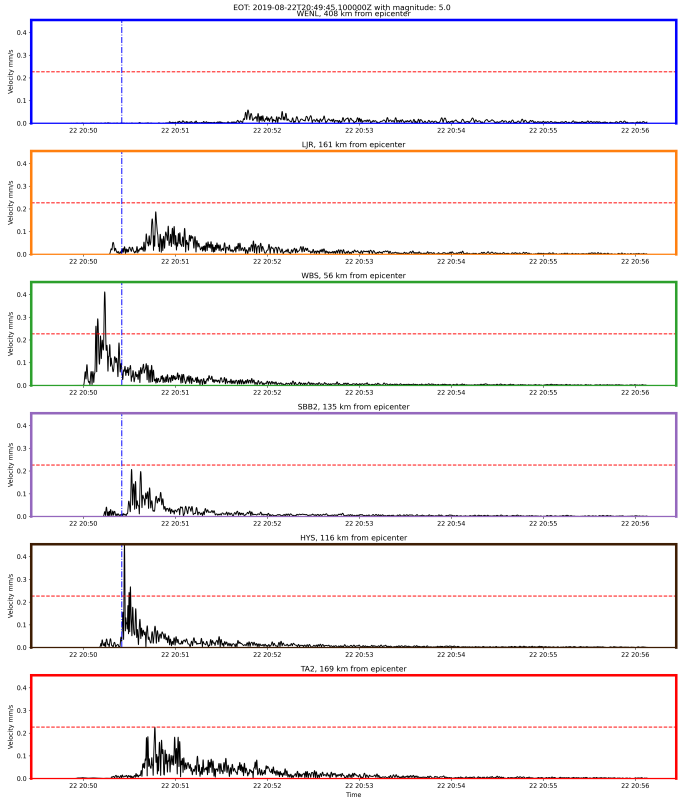
Figure 2: California with the transform San Andreas fault in red, figure by David Lynch (https://geology.com/articles/san-andreas-fault.shtml).

3 Methods

The goal of this study is to predict peak ground velocity (PGV) utilizing a small segment of real-time seismic data. The predictive approach focuses on estimating PGV at a specific location, denoted as $\boldsymbol{\zeta}$. The prediction relies on the initial 35 seconds of seismic data obtained from 5 receiver locations. The locations of the receivers are denoted as $\boldsymbol{X_r}$. An example for the seismic input data is given in figure 3. With on the top a geographical visualisation of an event with 5 random stations at locations $\boldsymbol{X_r}$ and 1 station at location $\boldsymbol{\zeta}$. On the bottom, the measured velocity data is shown, everything before the blue vertical line is input data and the PGV of the 6th station is presented as a red horizontal line, this value should be predicted by the models. Note, that this is not the maximum at every station. This can be caused by different geological structures in the ground, attenuation and the position of the receivers. In figure 3 the colors of the stations of the upper figure correspond to the colors outlining the seismic data in the bottom figure.



(a)



(b)

Figure 3: An example of an event that took place on August 22th 2019 with magnitude 5.0. *Top*: The event is shown as a red star, 5 random stations (all in a different color) and a 6th station ($\boldsymbol{\zeta}$, in red) at which the PGV should be predicted. *Bottom*: The seismic data before the vertical blue line of the upper 5 figures is input for the models. The red horizontal line is the PGV of the bottom figure (station on location $\boldsymbol{\zeta}$), this is displayed in every figure. The colors outlining these figures are corresponding to the colors of the stations in the top figure.

Since the predicted velocity is a real number, the problem in this research can be seen as a regression problem. In this study, machine learning is used to create different models. Different types of machine learning models are developed which potentially can predict this PGV very rapidly. To be able to use the created models for an EEW system this rapid prediction is very important, since an earthquake does not have a long time span. The input of the models is known, this is the few first 35 seconds of real time seismic data. The output is also known, this is a prediction of the PGV. The models have to give this output thus without knowing the complete seismic data. What the models are actually knowing and learning in between the input and output is unknown. In this research 3 different neural networks are trained and a random forest algorithm. When training a neural network, layers are created, the layers between the input and output layers are called the hidden layers. A neural network is trained during epochs, in each epoch the neural network learns from the entire dataset of training data. Therefore, the amount of epochs determines how many times the neural network processes the data before the final model is made. The hidden layers of a neural network consist of nodes. The nodes in different layers are connected with weights. During all the epochs, these weights are optimized to an optimum set of weights. This is done during each training iteration as follows: first, a prediction is generated by forward propagation of the input data trough the neural network. Next, the difference between this predicted value and the actual target values is computed to calculate the loss. In the next step, the gradients of the loss with respect to the model weights are computed, and this is propagated backwards. In the last step, an optimizer is used to update the weights of the model based on the calculated gradients in each training iteration. These steps describe in short how a machine learning neural network works. In this research is, besides the three neural networks, also a random forest algorithm developed. A random forest model is not a neural network. The development of this model involves distinct steps and their structure differ from those of neural networks. This is explained in more detail in section 3.2.1. The main objective of machine learning is to predict the probability of a target value, is this case the PGV, with a given input vector, in this case the first few seconds of real seismic data. The model utilizes this probability to generate an output, which is the predicted target value. This probability is conditionally distributed since the output depends on the provided inputs to the models. In regression problems, such as the described research problem, the typical assumption is that the conditional distribution follows a Gaussian distribution. When a model is trained, the inference stage commences. During this stage, the models encounter previously unseen data and are expected to predict the corresponding target value using the acquired knowledge of the conditional distribution. In this research four different models are trained to predict PGV. Each model is trained with the same dataset and with the same parameters as much as possible to be able to make a fair comparison between the performance of the four different models.

When the different models have processed data and made predictions for the PGV a coefficient can be computed. This coefficient is called the R^2 and is a coefficient of determination for regression. It quantifies the proportion of the overall variance in the observed data that the model is capable of explaining. This coefficient is computed by

comparing the predictive target values with the real target values. This R^2-score is 1 when the predicted values match the target values perfectly, and this score is 0 when the predictions are imperfect (Legates & McCabe Jr, 1999).

The different models that will be tested in this research are: Random Forest (RF), a Mixture Density Network (MDN) and two different Convolutional Neural Networks (CCNs): a CCN-MDN and a CNN-MSE, where MSE stands for Mean Squared Error. All the models are created, trained and tested in *Python* with packages from *PyTorch* and *scikit-learn*. In addition, all the plots in this research are made with *Python*.

3.1 Data collection and preparation

Before the models can be trained, the collected data needs to be prepared. All the data used in this research is collected from the website of the Incorporated Research Institution for Seismology (IRIS): *iris.edu* (see Acknowledgements), all the data is publicly available. This research utilizes seismic data collected in Southern California (USA). This data is collected with a Python script, so a lot of data can be collected at once. 269 recorders are present in the study area which recorded at least 1 of the 148 used events. These events and receivers are situated within a geographic region bounded by a latitude of 33 and 38 degrees and a longitude of -123 and -117 degrees. This is roughly an area of 555 km from north to south and 540 km from west to east. Notably, the west-east distance diminishes, due to the curvature of the earth, by 35 km from the southern to the northern extent of the study area. All the events under consideration have registered a minimum magnitude of 4.0 and took place between the 1st of January 2016 and the 15th of November 2023. In addition, all the events that are detected at less than 5 stations are not taken into account. This is not necessary, this filtering step is implemented to ensure a diverse set of seismic data during training. Although data from a single station can and still is included multiple times during training, retaining events with measurements from multiple station result in a more varied dataset, and since there are a lot of events, this is not limiting the data. All the receiver and earthquakes used to train the different models in this paper are shown in figure 4.

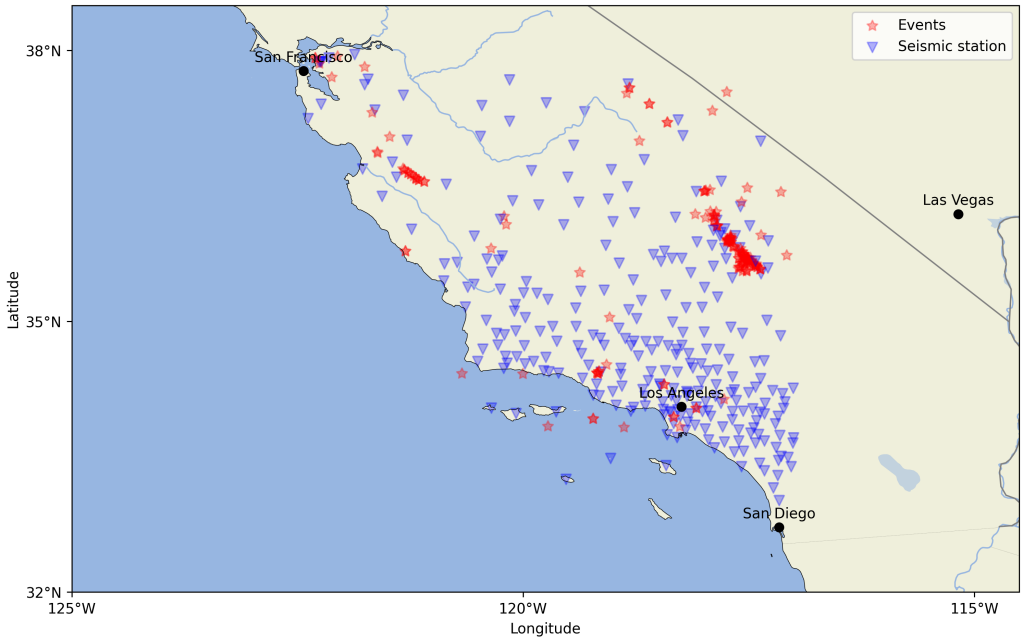


Figure 4: All the events (red stars) between the 1st of January 2016 and the 15th of November 2023 and the receivers that detected the events (blue triangles).

In this research, 3 Broadband channels, BHZ, BHN and BHE of seismic data are used in order to have data in all three directions: Z for vertical motion, E for east-west motion and N for north-south motion. Once the events are downloaded and filtered based on the criteria mentioned above, the data needs to undergo manipulation to become suitable for machine learning. First, for each event the closest station which recorded the concerned event is determined. Determining this station is independent of the input data determined later. The three channels of the seismic data of this closest station are stacked along a common axis to create a unified data frame. The stacked data frame provides a comprehensive representation of seismic activity recorded at this closest station. This data is detrended, bandpass-filtered between 0.05 and 2 Hz and the receiver response is removed. Subsequently, the P-wave arrival time of the closest station is computed with the Python package *TauPyModel*. This P-wave arrival determines the positioning of the time window specific to each event. The input for the models consists of the data recorded 5 seconds before the first P-wave arrival and approximately 30 seconds after this arrival. This uncertainty comes from the fact that when the start or end time does not correspond exactly to a node in the velocity data array, the array could be a few nodes longer or shorter. As mentioned before, all the computations are done in Python. To be able to make a data frame in Python, to store all the data for later use, all the data need to have the same length. The start time is determined and then the data is sliced until it has a certain length, this is, however, always very close to 30 seconds. Successively, for each event and all the recorders that recorded that event, the same merging of channel, detrending and bandpass filtering processes are applied, and the receiver response is removed. For all the data, the time window is made a lot larger than that of the input for the models, which

is 35 seconds, to ensure the PGV is present in the data. Next, the PGV is computed according to equation 1. In this function, PGV is the peak ground velocity, calculated at a specific location $\boldsymbol{X}$, with the velocity data on that location, v_c in a determined time frame t. This location can be a receiver location which is for example location $\boldsymbol{X_r}$. The PGV at a specific location, $\boldsymbol{\zeta}$, will be the target value for the models. The velocity data of the 3 channels is squared, summed and subsequently square rooted. The resulting data will become the input data in the specific time window. The PGV is determined by finding the maximum value of this data. Consequently, all the data and targets are represented as positive values.

$$PGV(\boldsymbol{X}) = max\sqrt{\sum_{c=1}^{3} v_c^2(t, \boldsymbol{X})} \tag{1}$$

3.1.1 Scaling data

Figure 5 shows that the original target distribution (left) is very narrow and concentrated around zero. This is not suitable for training the models, because some of the models give a Gaussian distribution as output. This is explained for each model in section 3.2. To make the distribution wider and flatter a (natural) logarithmic transformation is applied, shown in the center of figure 5. On this point logarithmic target values which are smaller than -6 are filtered out. This is done because these smaller values were just a few target values compared to the frequency of target values creating the distribution in figure 5. Without this filtering there are targets with a logarithmic value of -10, with between -10 and -6 a line which is very close to 0. This will cause that the models will be trained very poorly on this lower values of the data. On the right hand side of figure 5 the wider and flatter target distribution is normalized. The targets are normalized between 0 and 1 to make the training process easier.

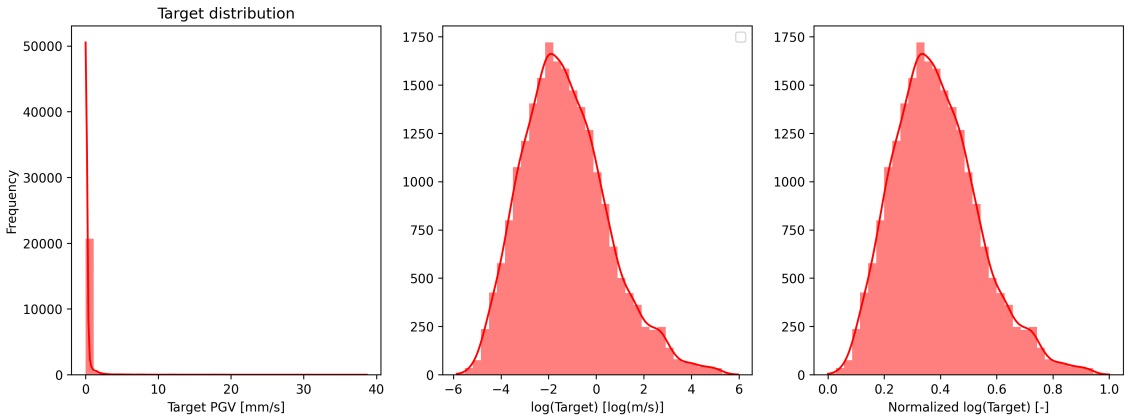


Figure 5: The distribution of PGV values as target values for training the models. With on the y-axis for all the subplots the frequency of PGV values and on the x-axis the target values. *Left:* The target distribution with the original PGV values. *Center:* The target distribution of the natural logarithm from the PGV values. *Right:* The target distribution from the normalized natural logarithm of the PGV values. The targets are normalized between [0,1]. These are the targets used for training the models.

The seismic data that will serve as input, is scaled with its own maximum value. In addition the geographical locations of all the stations are scaled. In this way, the range of values of the latitude and longitude are closer to that of the scaled target values and scaled seismic data. The geographical locations are scaled between 0 and 0.4, this makes them still distinguishable from the other data.

3.1.2 Input data

When all the data is prepared and ready for being input, the process continues to the next stage. This will be determining which data is going to be input. During the training, it is important that the input data remains consistent. In this research, the input will be a random selection of $\boldsymbol{K}$ receivers (with location $\boldsymbol{X_r}$) from a selection of the $\boldsymbol{L}$ closest receivers for each event. In all the results presented in this study: $\boldsymbol{L} = 80$ and $\boldsymbol{K} = 5$. Not all the stations have measurements of all the events. This has to be taken into account when the stations are chosen, only stations that have seismic data from an event can be chosen when that event is going through training. This can vary per event. However, taking this into account now ensures that the created models also have the opportunity to grow along the growth of the seismometer network in the future. The time window is based on the P-wave arrival time of the closest station for each event. With the random choice of stations this closest station is not naturally present in all the training steps. Next, a 6th station is randomly chosen, which location is denoted as $\boldsymbol{\zeta}$. At this station the PGV should be predicted. The 5 random stations with location $\boldsymbol{X_r}$ and the station with location $\boldsymbol{\zeta}$ are chosen with the help of a before created *csv* file. This file contains the numbers from all the 6 receivers and determines the combination in which these receivers are chosen. For each event, it can be chosen how many combinations of receivers the file will contain. It is crucial for the models to have the geographical locations of the 5 random

chosen stations ($\boldsymbol{X_r}$) and the location of 6th station ($\boldsymbol{\zeta}$). Without this information, the models would never be able to make a prediction of PGV in the region which is related to geographical location. The input data contains 35 seconds of the scaled seismic data from 5 random receivers, the geographic locations $\boldsymbol{X_r}$ and $\boldsymbol{\zeta}$, the time between the EOT and the first P-wave arrival at the closest station, t_{PW}, and finally the scaling factor with what the seismic input data is scaled, d_{max} . Equation 2 describes the prediction the models represent with the corresponding input data. In this function $\boldsymbol{d_K}$ is the seismic input data of the $\boldsymbol{K}$ stations, which is 5 in this research and are chosen of the $\boldsymbol{L}$ closest stations the event is measured at, which is 80 in this research. $\boldsymbol{X_r}$ is the location of the $\boldsymbol{K}$ stations and $\boldsymbol{\zeta}$ the location of the 6th station.

$$p(log(PGV)|\boldsymbol{d_1}, .., \boldsymbol{d_K}, \boldsymbol{X_{r1}}, .., \boldsymbol{X_{rK}}, \boldsymbol{\zeta}, t_{PW}, d_{max}) \tag{2}$$

3.2 Model description

3.2.1 Random Forest

The Random Forest (RF) algorithm is introduced by Breiman (2001) and is an effective tool in machine learning predictions. RFs are a merge of tree predictors, where each tree relies on the values of an independently sampled random vector. The random vectors share the same distribution across all trees within the RF algorithm (Breiman, 2001). RFs are very popular machine learning algorithms, this is because they are widely applicable for real-life prediction problems and are simple to use. Beyond this, RFs are user-friendly and the method is also well known for its accuracy (Biau & Scornet, 2016). In addition, with the *Scikit-learn* package in Python, which includes a RF machine learning method, creating a RF model is easily made in a few lines of code in Python (Hao & Ho, 2019).

Figure 6 shows a schematic diagram of the RF algorithm. The RF algorithm randomly picks subsets from the complete input dataset. Each tree formulates a decision based on the given subset of data. These individual decisions serve as predictions for the target value. Subsequently, the predictions from all trees are averaged to produce a final RF prediction for the target value (Sahour et al., 2021). The algorithm grows N distinct (randomized) trees, shown as a combination of blue and orange dots in figure 6. Each tree is constructed with randomly drawn observations from the original dataset. This selection can be done with selecting the same observation multiple times or select each observation only once. Only these observations, with possible repetitions, are used to construct a tree (Biau & Scornet, 2016). This means each tree is based on a different part of the training data only. All trees consist of cells, at which the algorithm makes a decision. This seems simple, however theoretical understanding of random forests is less convincing, there is limited knowledge about the method's mathematical properties. This understanding is challenging because it is a black-box type algorithm (Biau & Scornet, 2016). Each tree can make their own network of splits where they make a decision, which are difficult to control. Biau and Scornet (2016) give an overview of the algorithm, with a focus on the theory behind RF.

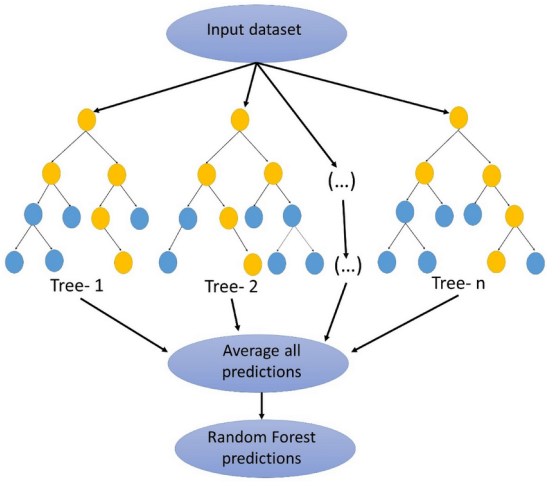


Figure 6: Schematic diagram of the random forest algorithm from Sahour et al. (2021)

As mentioned before, a RF algorithm can be made with *Scikit-learn* in a few lines of code. The input for this code should be: the dataset, the target values, the fraction of test data, the amount of decision trees and the maximum depth of each of these trees. The contents of the dataset is explained before in section 3.1.2. In this research, the amount of decision trees in each RF model is 1000 and they all have a maximum depth of 20. Given a maximum depth to each tree prevents the model from overfitting the data. Without this maximum, the trees would have such a depth that each part of the training data with its respective target has it's own cell at the end of a tree. If this is the case the model would not be able to handle new, never seen before data. This would make the model unusable, since real-time seismic data is never the same. After the input for the RF training phase is defined, the first step is dividing the data into training and test data. Subsequently, a Random Forest Regressor is used to train a RF model using the training data. Following this training phase, the model predicts the target values of the test data. On this point, this is new data for the model. Finally, the model is saved and the performance of the model can be tested on unseen data. This is done in the result section of this research.

3.2.2 Mixture Density Network

The Mixture Density Network (MDN) was introduced by Bishop (1994). He obtained the network by combining a conventional neural network with a Gaussian mixture model. This results in a multi-layer neural network, with an input layer, a single hidden layer and an output layer of parameters for each Gaussian distribution in the mixture. A schematic representation of a MDN, with the three layers, is shown in figure 7. As shown in the figure, the input layer is 1 vector and the output layer is a set of Gaussian output parameters, α, μ and σ in this figure. The output distribution is shown in the figure as Mixture Distribution which is a combination of the trained Gaussian mixture components. The output from the model forms a probability distribution that characterizes the likelihood of the random variable adopting different values within the target range. Therefore, this

probability distribution serves as a comprehensive description of potential outcomes. In the Gaussian MDN from Bishop (1994) , the Gaussian mixture model established a general structure for representing any conditional density function for any given value of x, which is the input vector. This input vector contains the components described in section 3.1.2 all as one concatenation. This is shown in equation 3 from Zhang et al. (2020)

$$p(y|x) = \sum_{i=1}^{m} \pi_i(x) N_i(y|\mu_i(x), \sigma_i^2(x)) \tag{3}$$

In this function, m is the amount of constituents in the mixture model, $\pi_i(x)$ are the mixing coefficients, $\mu_i(x)$ the mean of the i-th Gaussian and $\sigma_i^2(x)$ the variance of the i-th Gaussian. In addition, the sum of mixing coefficients $\sum_{i=1}^{m} \pi_i(x) = 1$, because its output is dominated by a normalized exponential function (Zhang et al., 2020). In function 3, y symbolizes the the target value for the model, which is the normalized natural logarithm of the PGV in this research. During training, this becomes equation 4 due to a trainable parameter $\boldsymbol{w}$, the weights and biases (Bishop, 1994).

$$p(y|x, \boldsymbol{w}) = \sum_{i=1}^{m} \pi_i(x, \boldsymbol{w}) N_i(y|\mu_i(x, \boldsymbol{w}), \sigma_i^2(x, \boldsymbol{w})) \tag{4}$$

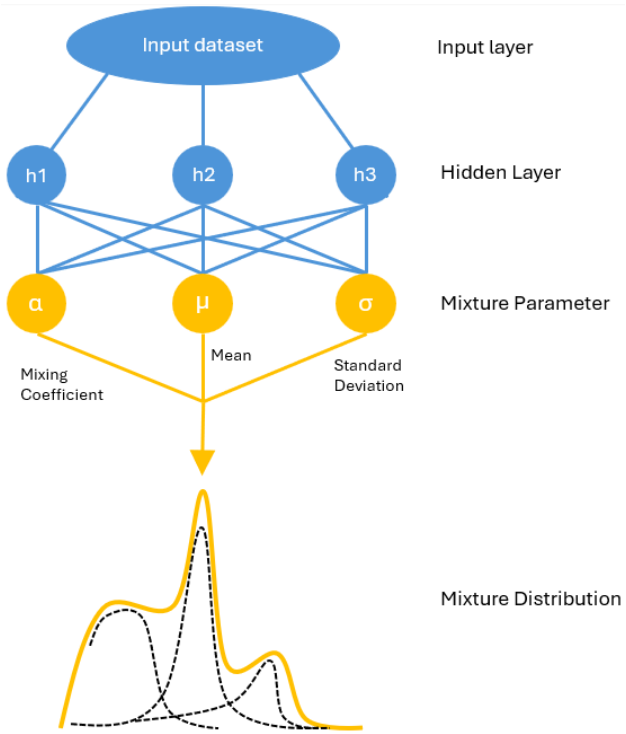


Figure 7: Schematic representation of a Mixture Density Network. (Figure adapted from Zoutendijk and Mitici, 2021)

The MDN is made in python with *PyTorch* which is a framework for building machine learning models. First, the dataset is split into fractions of training, testing and validation data. The initial learning rate (lr) for the model is set to $1e{-}4$ which is adapted by an Adam optimizer during the learning process. Kingma and Ba (2014) introduced this method for optimizing stochastic objective functions, functions that involve randomness or variability, using gradient-based techniques. For this optimizer, the weight decay parameter is set to $1e{-}8$. The model is trained during 250 epochs. During each epoch the model is set to train mode, which ensures the model knows that it is in the training phase and it is able to adapt the structure of the network. Next, the Adam optimizer is used to clear out the gradients of all optimized tensors, to prevent accumulation from previous interactions. Next, for every batch of training data the Gaussian output parameters, π, σ and μ in this research, are obtained via a forward pass through the network. Subsequently, the loss is calculated with a MDN loss function. This loss is the difference between the predictions of the MDN and the actual target values. Then again the gradients are cleared out, before the loss is propagated backwards trough the network. Finally, the model parameters are updated. These steps are repeated for all epochs. The MDN gives the output as a probability density distribution, so a possibility of a value for a target. This is not just one absolute value of the predicted PGV as in the case of RF. To be able to compare the MDN with the other neural networks, the PGV is chosen to be the maximum value of the probability density distribution, so the most possible target according to the network.

3.2.3 Convolutional Neural Networks

Fukushima and Miyake (1982) did an attempt to synthesize a neural network model which should have the same capabilities for pattern recognition as the human brain, which was called the neocognitron. This neocognitron proposed by Fukushima and Miyake (1982), could be seen as a predecessor of the modern Convolutional Neural Network (CNN). This neocognitron was inspired by the findings of Hubel and Wiesel (1968), who made a hierarchy model of the visual nervous system and discovered the role of cells in the visual cortex of animals, revealing their capacity for detecting light within specific receptive field. The performance of the neocognitron from Fukushima and Miyake (1982) has been improved over the years and has developed into a well-known deep learning architecture, CNN. Training a CNN considers to find the global optimum, the best fitting set of parameters (Gu et al., 2018). Figure 8 shows a schematic representation of a CNN with a MDN output layer at the end of the network. In this research, the performance of two different CNNs are tested, one with a MDN and one with an MSE output layer. The CNN-MSE network has the same structure as in figure 8, only with an MSE output layer at the end instead of a MDN output layer, which means the last layer consist of just 1 node, since a MSE predicts just 1 target value. A MDN gives the output in multiple nodes as a probability density distribution. This is the same as described in section 3.2.2 where the MDN is discussed more extensively. As visible in the figure, in contrast to the RF and MDN, the input of a CNN is a matrix with separate input values. In a CNN model convolution is applied to the input matrix. The separation of the data is not only in the input part, but also the segments of input data follow different steps through the network. This should make it

easier for the network to understand the input data and to find patterns in the data, since it does not have to sort out which data belongs together. In the MDN, all the layers are fully connected to each other by the connection of all the nodes between the layers. In a CNN, the nodes are not fully connected, which causes the amount of parameters in a CNN to be lower than in the MDN.

A CNN has a convolutional layer in which convolution is applied to a part of the input data. This convolution causes the input matrix to decrease in size but increase in depth. This is also visible in figure 8. This process aims to learn from important features that are present in the input data (Gu et al., 2018). After convolution, an activation function is applied, which is ReLu in this research. This function brings in non-linearities to the CNN, a desirable aspect for enabling multi-layer networks to detect nonlinear features (Gu et al., 2018). Next, max Pooling is applied. Pooling is an operation which usually involves a sum, an average, a max or or occasionally another commutative combination rule (Boureau et al., 2010). In this research, a max Pooling is applied and the Pooling factor is set to two, which means that during Pooling only the maximum value of 2 adjacent values if kept. This reduces again the size of the input data. The process of convolution, applying ReLu and Pooling, which is in this research applied to the seismic data, is repeated and is done twice in total. A 1D convolution is applied to the input locations of all the receivers which are inputted as a 2D matrix. After the convolution processes, the four different input parameters are flattened and concatenated, as shown in figure 8. This then forms a fully connected layer. Before this layer can be the input to the MDN or MSE output layer, a last ReLu activation is applied. Then depending on it being a CNN-MDN or CNN-MSE, the output layer is created with respectively a probability distribution or a predicted target value.

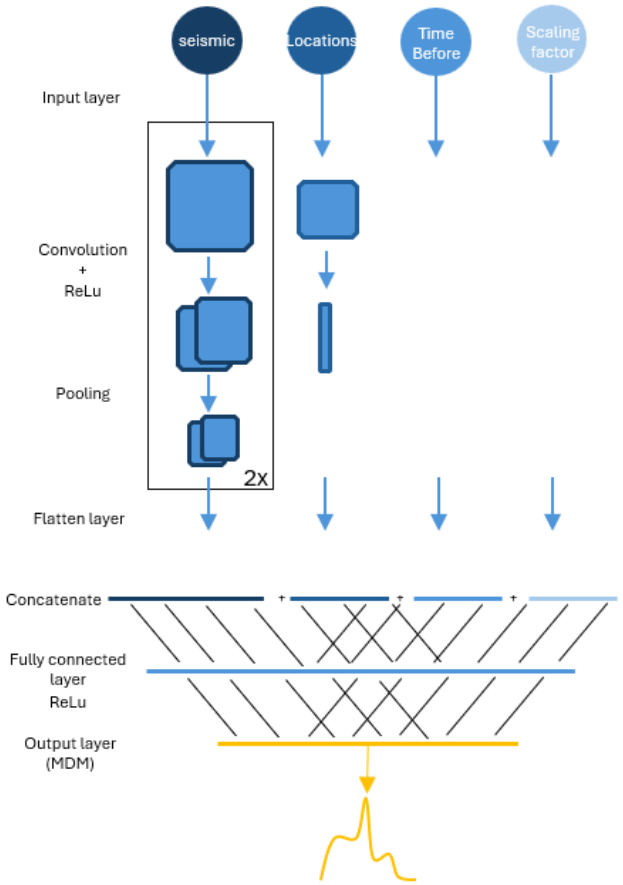


Figure 8: Schematic representation of a Convolutional Neural Network with a Mixture Density Network layer.

Both CNNs are made in Python with *PyTorch*. The implementation is very similar to that of the MDN. Only the forward pass through the model has a completely different form. The dataset is split into fractions of training, testing and validation data. The initial learning rate (lr) is set to the same value as for the MDN, $1e-4$. During training of the CNNs, also the Adam optimizer from Kingma and Ba (2014) is used with again a weight decay of $1e-8$. Both CNNs are trained during 250 epochs. During each epoch the models are set to training mode so the neural network can be adapted and the the Adam optimizer is used to clear out the gradients of all optimized tensors. Next, a forward pass through the network is applied, in which the process of convolution, applying ReLu and Pooling takes place. For the CNN-MDN, this gives the Gaussian parameters, for CNN-MSE this gives a prediction. Next, the loss is calculated with an MSE loss function or a MDN loss function depending on the CNNs output layer, which is backward propagated through the network and finally the parameters of the predictions are updated. For the CNN-MDN, the prediction for the PGV is chosen the same as for the MDN, the maximum value of the probability density distribution.

4 Results

Various strategies were tested to obtain the most optimal models possible. In this result section, all different approaches are described and explained. The used parameters and associated values are described per section. However, all the models are trained with a random combination of 6 receivers for each event. From 5 of these receivers, 35 seconds of seismic data is provided (5 s before P-wave arrival, 30 s after P-wave arrival) and at 1 of these receivers, located at $\boldsymbol{\zeta}$, the PGV is predicted. The models are trained with the longitude and latitude of real coordinates. This means that the models are structured based on spatial coordinates, enabling the input of any longitude-latitude pair into the models, and not only the location of receivers in the region. This is a great advantage of the models, since not all vulnerable areas have a receiver on that exact location. Testing the ability of predicting the PGV at a coordinate on a location different from that of a receiver is done in section 4.3. When the different models have processed the test data and made predictions for the PGV, the R^2-score can be computed. As mentioned before, this R^2-score is 1 when the predicted values match the target values perfectly, and this score is 0 when the predictions are completely off (Legates & McCabe Jr, 1999). When the R^2-score is negative this means that the model is performing worse than the mean of the target vales and is thus not suitable for predicting the requested task. This score is also used to make decisions on which models perform the best and should be continued with. To be able to make fair comparisons, all the models that are be compared have been shown the same dataset during training and testing.

4.1 Different combinations of training and testing datasets

First, the predictive results are shown for the receiver locations $\boldsymbol{X_r}$ at which the concerned event is also detected. All the different models are compared. Since the PGV is known at the prediction location $\boldsymbol{\zeta}$, the models can not only be compared to each other, but also to the real PGV target value. Initially, all the different kinds of models were trained with the complete dataset of events between the 1st of January 2016 and the 15th of November 2023, with per event 1000 random combinations of 6 out of the 80 closest receivers. This dataset consists of 269 different receivers, which recorded at least 1 of the 148 events which are used in this area. For this training, the scaling of the target, seismic data and locations was done as described in the method section. After training, the performance of all the different models was tested. The first test was done with a testing dataset which consists of 100 random combinations of receivers from all the events the models have been trained with, also from the complete dataset of events between the 1st of January 2016 and the 15th of November 2023. This makes the target distribution of the training and test data equal to each other, since they are from the same events. However, this does not mean that the input is the same. The combinations of receivers is chosen randomly, this makes that the combination in which the data is shown to the model is not the same as during training. 94.8 percent of the combinations shown during testing are new for the model. It is expected that the models see the different combinations of seismic data as new data. If this is indeed the case, then this makes that even with data from the same events, the models

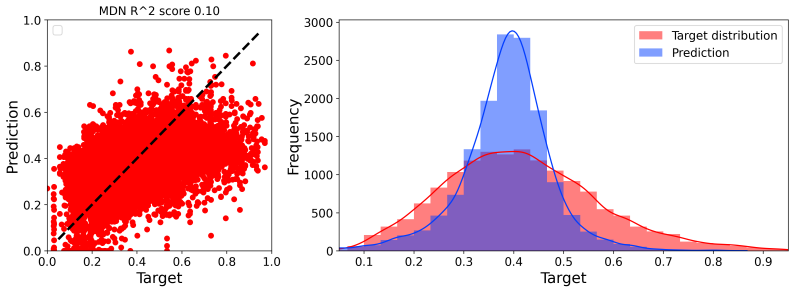
could be tested on their performance to combine seismic velocity data with geographical data to make an accurate PGV prediction in a specific location. In this study area, there is a lot of data available, which ensures that this testing method is not necessary. However when doing the same kind of research in different areas where less data is available, this could be an essential testing method. Since there is an abundance of data in this area, this testing method can be compared to testing the models with data from unseen events.

Figure 9 shows the performance of the four different models trained with data from the complete dataset with all the events (1000 random combinations) and tested with events from the same dataset with all the events (100 random combinations). A complete dataset means the dataset containing all the events with a light or dark blue color in figure 12. All these events are outlined in dark blue. The distinction between the light and dark blue stars will be explained later. Each dot in figure 9 on the left hand side represents the predicted value shown on the y-axis and the actual target value on the x-axis. So, when dots are closer to the black striped line, $x = y$, the predictions of the model are better. The right hand figures show in red the actual target distribution of the dataset given to the models during testing. Note that these are the same for all the models since they are tested with the same dataset. The scale of the y-axis varies for each model due to the differences in frequency of the PGV predictions. Those are plotted in blue in the same figure. The difference in performance of all the models is clearly visible. The R^2-score above the left figure of each plot quantifies the difference in the plots.

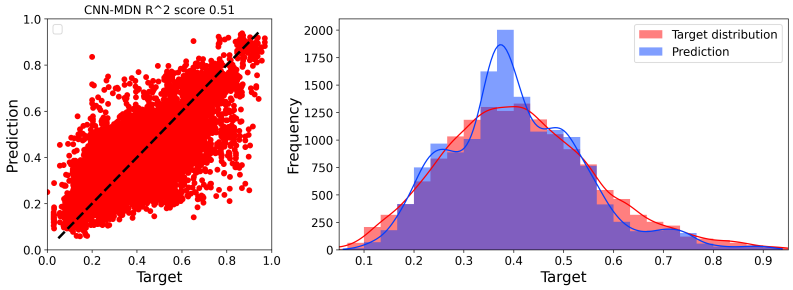
Starting with figure 9a, which shows the MDN model, this model makes by far the worst predictions with the given test data. The scatter plot shows a lot of predictions that are not close to the target values. This is also visible in the distribution plot, the MDN model has a very high peak, while the actual target values are way more flattened out.

Next, the CNN-MDN and CNN-MSE models, displayed in figure 9b and 9c respectively. These models exhibit a similar R^2-score. The distributions of the predicted target values however do show a different shape. The CNN-MDN has 3 peaks in the prediction while the CNN-MSE has 1. The single peak of the CNN-MSE looks similar to that of the actual target value distribution. The peaks in the distribution plot are also visible in the scatter plot, as outlier dots. Compared to the single layer MDN model, the CNN-MDN, which includes an output layer of MDN, is performing significantly better. When using a CNN model, the input for the models is separated into the different parts of the input. It could be that this improves the performance of the MDN layer in a CNN.

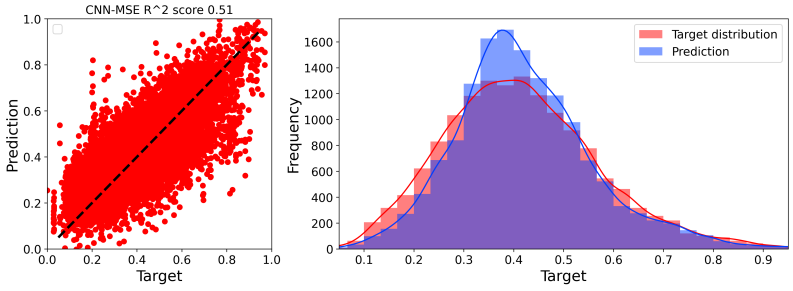
Lastly, figure 9d shows the same plots for the RF model. The R^2-score is considerably higher than for all the other models. The predictive distribution looks very similar to the actual target distribution. This model actually makes a pretty good fit with only a few outliers. The R^2-scores of these 4 models are also displayed in the first column of table 1.



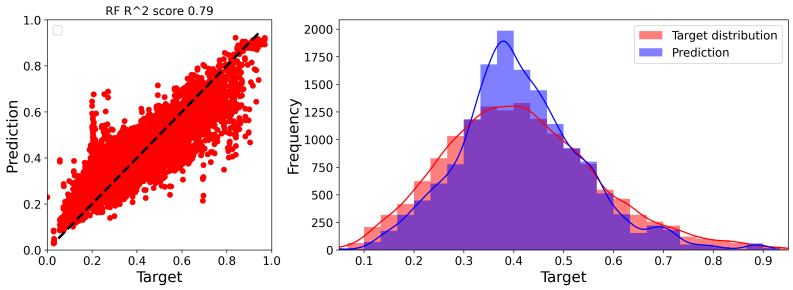
(a) MDN



(b) CNN-MDN



(c) CNN-MSE



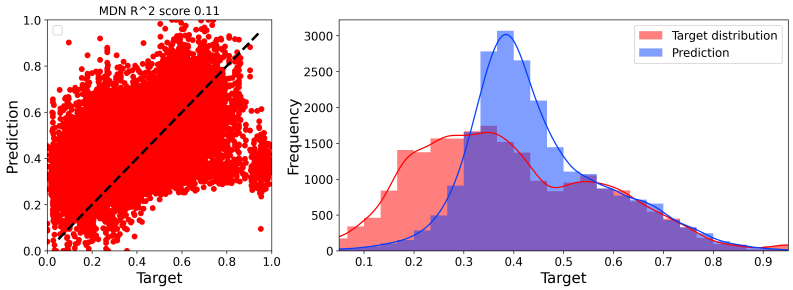
(d) RF

Figure 9: Performance of the four models, trained with 1000 combinations of random receivers of the complete dataset and tested with 100 random combinations of the the same complete dataset, consisting of 269 different receivers, and 148 events. a) MDM, b) CNN-MDM, c) CNN-MSE, d) RF shown with: *Left:* scatter plots with the target values on the x-axis and the predicted PGV on the y-axis. The R^2-score is shown above the figures. *Right:* Distribution of the predictive PGV values in blue plotted over the distribution of the target values in red.

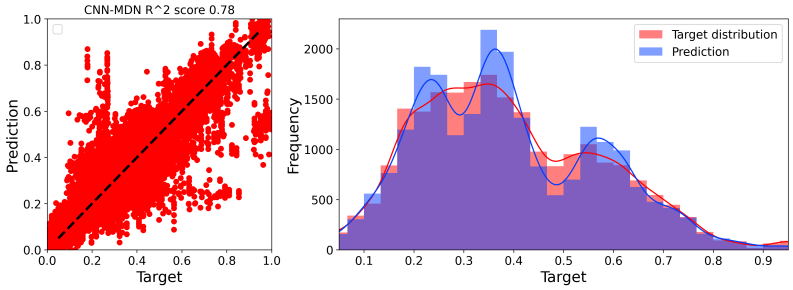
Next, models are trained with data of only 36 stations which was downloaded from events between the 15th of November 2015 and the 15th of November 2023. Note, that this time range is a little bit longer, however these are just 2 events in the downloaded data. The 36 stations makes for a very low concentration of receivers in a big area, which was initially not the reason to choose for Southern California. However, fewer stations showed some interesting results. In addition, in Southern California there are a lot of receivers, but this is not the case for every seismically active region on Earth, so analysing also these results can be very interesting.

Figure 10 shows the performance of the four different models trained with data from all the events from the complete dataset (1000 random combinations) and tested with mainly new combinations of data from all the events (100 random combinations). This are all the light and dark blue events in figure 12. The difference in performance of all the models trained with just a few receivers is clearly visible. Also in this plot, the R^2-score confirms the difference in performance. Just like in figure 9 the MDN model, shown in figure 10a, has by far the lowest score relative to the other models. Next, the CNN-MDN and CNN-MSE models, displayed respectively in figure 10b and 10c, again show a similar R^2-score in this plot. The RF model is displayed in figure 10d. It is again for this training dataset, which just a few receivers, the best performing model, based on the R^2-score.

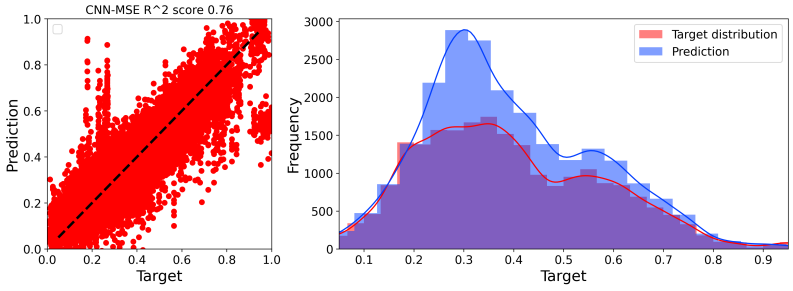
Figure 10 shows that using less receivers improves the R^2-score significantly. Decreasing the total number of receivers while maintaining the same number of random combinations of these receivers will increase the frequency at which the models see data from those receivers location during training. The R^2-score of these four models are displayed in the first column of table 2.



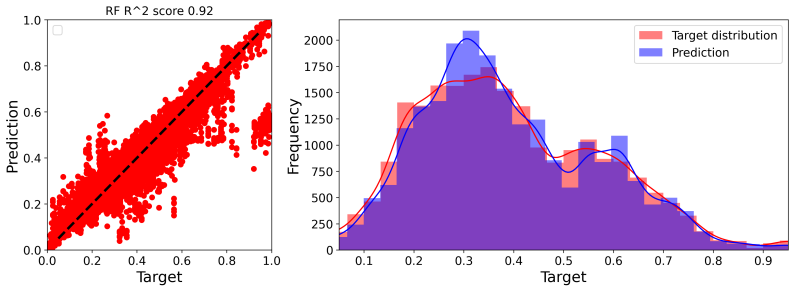
(a) MDN



(b) CNN-MDN



(c) CNN-MSE



(d) RF

Figure 10: Performance of the four models, trained with 1000 combinations of random receivers of the complete dataset and tested with 100 random combinations of the the same complete dataset, consisting of 36 different receivers and 150 events. a) MDM, b) CNN-MDM, c) CNN-MSE, d) RF shown with: *Left:* scatter plots with the target values on the x-axis and the predicted PGV on the y-axis. The R^2-score is shown above the figures. *Right:* Distribution of the predictive PGV values in blue plotted over the distribution of the target values in red.

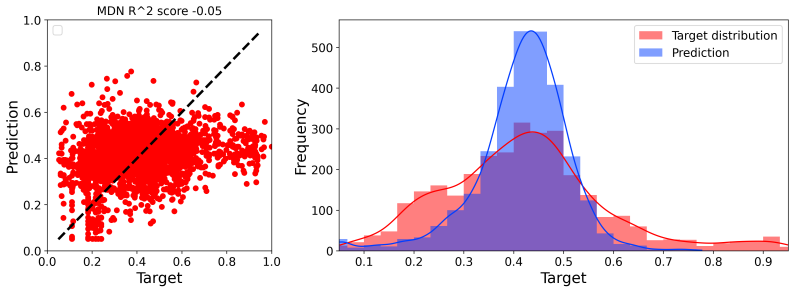
Next, to test the models ability to handle new data, the models are tested with a dataset consisting of events the models have never seen before. This dataset is made with events between the 1st of January 2014 and the start date of the training dataset. This is the 1st of January 2016 for all the receivers and the 15th of November 2015 for just 36 receivers. These events are shown in red in figure 12. All these events took place before the events used in the training data. All the events in these new datasets are still in the same region and also have a magnitude bigger than 4.0. These new datasets do not necessarily have the same target distribution. However, the targets are scaled with the same method as the training data. So, all the target values of the new datasets are also between zero and one. Out of this new dataset consisting of older events, a combination of 100 randomly chosen receivers is made. Note, that the receivers themselves do not change in the new dataset. It could be that some receivers are from after 2015, however as mentioned before, this should not be a problem for the models. Unexpectedly, the performance of all four models with this new dataset is very bad, for both the models with all the receivers and less receivers. For the models trained with all the receivers all the R^2-scores are below zero. This score is displayed in the second column of table 1. For the models trained with just 36 receivers also all the R^2-scores are below zero. This score is displayed in the second column of table 2. This dataset with older events, shown in red in figure 12, is from now denoted are old testing dataset.

This very bad performance could result from two things: the trained models do not understand the data or the testing dataset is bad. The testing dataset is all from events earlier than those of the training dataset. It is thus relative old data (8-10 years old) and it could be that some filtering of the data is too hard for the models, such as filtering out the receiver response. To be able to test if the testing dataset with old data is bad, a new training and testing dataset is made. The datasets are made with the events from the original complete training dataset. To be able to test the ability of the models to process data from new events, this original dataset is split randomly into two. 80 percent of the events in this dataset form a new training dataset and the remaining 20 percent from a new test dataset. The distribution is made randomly to ensure that there are old and new events in both datasets. When there is actual something wrong with the older data, it is unknown for which time period this is the case. It may be part of the dataset the models were trained with initially. With this random distribution, both the training and testing datasets may have data that has a contribution of bad data and it will then thus have less effect on the results. Figure 12 shows which events are now part of the new training dataset, light blue, and which are now part of the new testing dataset, dark blue.

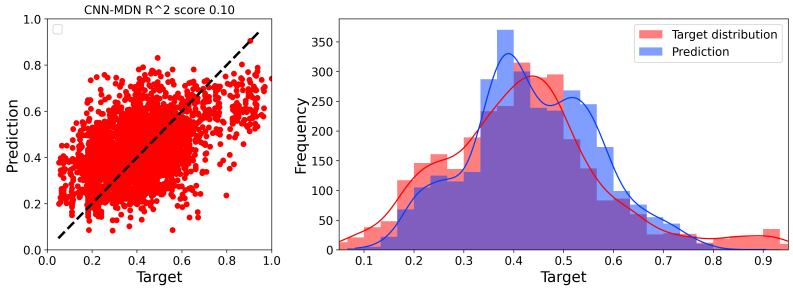
This new train and test dataset will not only test if there is something wrong in the older testing event, but will also show if the models understand the data, since with this method also new data is fed into the model. However, in addition to changing the train and test dataset, also some parameters are changed to be sure that the original method is the best. This is described in the next section, 4.4.

Figure 11 shows the performance of the four different models which are trained with 1000 random combinations of all the receivers of 80 percents of the events from the complete dataset (light blue events in figure 12) and tested with 100 combinations of receivers of 20

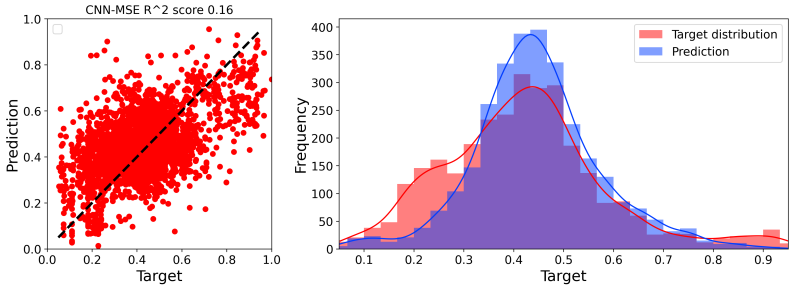
percent of the events from the complete dataset (dark blue events in figure 12). The R^2-scores are also displayed in table 1. This figure shows that all the models have a bit more trouble with predicting the new target distribution. The target distribution of the new training data has a very similar shape to that of the distribution of the complete dataset which is shown in figure 5. The actual target distribution of the testing dataset, shown in red in figure 11, shows a different shape. It shows a higher frequency of smaller target values. However, these are not values the models have not seen during training. Despite this the R^2-scores of the models have decreased a lot when testing the models with data from new events. The same testing is done with the models trained with less receivers, the R^2-score of this test are shown in the third column of table 2. Also these models were trained and tested with respectively the light blue and dark blue events in figure 12. For both the different receiver numbers the R^2-score of the RF decreases the least. The R^2-scores of both the CNNs has decreased significantly. To check if these lower scores are caused by the different dataset, for example the fewer events the models are trained with, this model is also tested with combinations of the events they were already trained with. So trained and tested with the events shown in light blue in figure 12, which are the events from the new training dataset. The testing is done with a majority of new combinations of receivers. The R^2-scores of these models are shown in the 4th column of tables 1 and 2. Compared to when they were trained and tested with the complete dataset (all light and dark blue events in figure 12), this does not change the R^2-scores a lot. Next, the models that were trained with the new training dataset (all light blue events in figure 12) were also tested with the complete dataset. So, the models were tested with partially new events. The R^2-scores of these models are shown in the 5th column of tables 1 and 2. This score show that the performance of all the models is a bit lower than when they were only shown events they already had seen. The R^2-scores suggest that training and testing it with the same dataset does have a positive influence on the R^2-scores. Also the performance of the models with the new test dataset obtained trough splitting the complete dataset is much better than with the test dataset with old events. In section 4.2 the performance of the models when predicting the PGV on a 6th location ζ co-located with a receiver position is described in more detail, without only looking at the R^2-score.



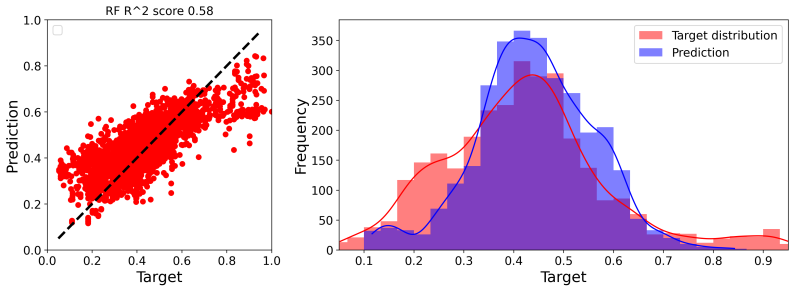
(a) MDN



(b) CNN-MDN



(c) CNN-MSE



(d) RF

Figure 11: Performance of the four models, trained with 1000 combinations of random receivers of the new training dataset and tested with 100 random combinations of the new test dataset, consisting of 269 different receivers, and 148 events. a) MDM, b) CNN-MDM, c) CNN-MSE, d) RF shown with: *Left:* scatter plots with the target values on the x-axis and the predicted PGV on the y-axis. The R^2-score is shown above the figures. *Right:* Distribution of the predictive PGV values in blue plotted over the distribution of the target values in red.

Train file	Complete	Complete	New Train	New Train	New Train
Test file	Complete	Old test	New Test	New Train	Complete
Model	R^2-score				
MDN	0.10	-2.32	-0.05	0.08	0.04
CNN-MDN	0.51	-2.42	0.10	0.49	0.37
CNN-MSE	0.51	-1.97	0.16	0.50	0.39
RF	0.79	-2.12	0.58	0.78	0.72

Table 1: R^2-score for different models that are trained with the 1000 combinations of 269 receivers from 148 events of the train file mentioned in the box corresponding to *Train file* and tested with 100 combinations of the test file mentioned in the box corresponding to *Test file*. In this table, train file *complete* stands for the complete dataset (all blue stars in figure 12) and train file *New Train* (all light blue stars in figure 12) for the random 80 percent of the complete dataset. Test file *complete* stands for the complete dataset, *Old test* for the test data from events before the complete dataset (all red stars in figure 12), *New Train* for the random 80 percent of the complete dataset and *New test* for the rest 20 percent of this dataset (all dark blue stars in figure 12).

Train file	Complete	Complete	New Train	New Train	New Train
Test file	Complete	Old test	New Test	New Train	Complete
Model	R^2-score				
MDN	0.11	-2.97	0.03	0.04	0.03
CNN-MDN	0.78	-1.74	0.29	0.79	0.69
CNN-MSE	0.76	-3.34	0.24	0.78	0.67
RF	0.92	-2.25	0.80	0.96	0.93

Table 2: R^2-score for different models that are trained with the 1000 combinations of 36 receivers from 150 events of the train file mentioned in the box corresponding to *Train file* and tested with 100 combinations of the test file mentioned in the box corresponding to *Test file*. In this table, train file *complete* stands for the complete dataset (all blue stars in figure 12) and train file *New Train* (all light blue stars in figure 12) for the random 80 percent of the complete dataset. Test file *complete* stands for the complete dataset, *Old test* for the test data from events before the complete dataset (all red stars in figure 12), *New Train* for the random 80 percent of the complete dataset and *New test* for the rest 20 percent of this dataset (all dark blue stars in figure 12).

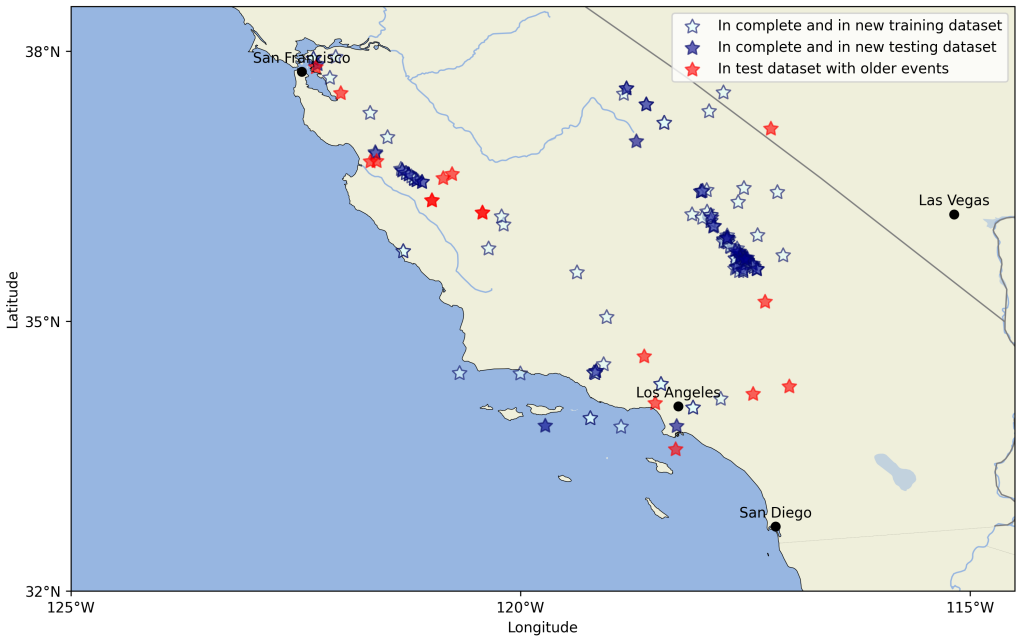


Figure 12: All the events between the 1st of January 2014 and the 15th of November 2015. The red stars are events in the old testing dataset with events between the 1st of January 2014 and 1st of January 2016. All the dark blue outlined stars (filled with dark of light blue) are events from the complete data set with events between the 1st of January 2016 and the 15th of November 2015. This dataset is randomly divided in the new training dataset which contains the events filled in light blue and the new testing dataset which contains the events filled in dark blue.

4.2 Location $\boldsymbol{\zeta}$ at a receiver location $\boldsymbol{X_r}$

In this section, the performance of the different models is analysed in more detail for the situation where the prediction location $\boldsymbol{\zeta}$ is located on a receiver location, which was not part of the input of the model. This is the same situation as described in section 4.1 in which different datasets for training and testing are described. Figures 13, 14, 15 and 16 show on the left hand side all the receivers colored in their respectively target value, so the actual determined PGV detected at the receiver location. In the middle, the MDN, CNN-MDN, CNN-MSE and RF predictions in the same scale as the target values of the figures at the left hand side. The right hand side shows the same prediction in a scale determined by the models themselves. Note, that this is different for each model depending on the predictions. This 3rd column is displayed to be able to see how the models combine the geographical locations of the receivers to the velocity data. When the models do not give the correct range of PGV values, these figures can show if the models maybe do recognise the wave propagation pattern. In this figure, the input receivers are outlined in red. In the right 2 columns of these figures the PGV of the input receivers is also predicted. So, for these locations the prediction location $\boldsymbol{\zeta}$ is co-located with an input location $\boldsymbol{X_r}$. This is never done during training, so it is not necessarily expected that these locations have accurate predictions.

Figure 13 shows the predictions of the models trained with 1000 combinations of all the 269 receivers and a random event which the models have never seen before. The actual target values displayed on the left hand side does have generally a lower PGV with an increasing distance from the epicenter. When comparing the left and the middle figure to each other, it shows that in every model predicts values which underestimate higher PGV values and overestimate lower PGV values. This makes that all the PGV values are very close to each other. Figure 11 already suggested this. When comparing the distribution of the actual targets (in red) with the predictions (in blue), all the models have a lack of predictions in above 0.8 and below approximately 0.3. All the models predict PGV with a bias towards the mean value. On the right hand side of figure 13 the scale is determined by the models themselves. In this figure, it is visible that the RF model does show a gradient pattern in predicting the PGV and does seem to understand where the event is located, since it predicts a bright yellow color almost at the location of the event. The other three models predict PGV values in which no clear pattern can be distinguished.

Figure 14 shows the same models but now they have to predict an event that was already in the training data. Again the actual target values of the left hand side show a gradient pattern, with a decreasing PGV when the distance from the epicenter increases. For this event clear color differences can be observed between the predictions for the different receiver locations in the middle figures. The RF model shows a color pattern pretty close to that of the actual target values. The other three models over estimate the PGV values in almost every receiver location. On the right hand side, when the the scale is determined by the models, it is visible that again the MDN and the CNNs do not show a clear pattern. The scaling of the RF model does not change a lot comparing with the actual target values scaling. This means the model does predict PGV values close to the actual target values.

Figure 15 shows the prediction of the model trained with 1000 combination of 36 receivers and a random event which the model has never seen before. On the left hand side of the figure, it can be seen that with fewer stations the distribution of the actual target has a few seemingly unexpected measurements. Such as the bright yellow station at the top (which does not seem to be closest station) or the lighter colored stations in the right bottom. This unexpected measurement could be confirmed or contradicted when there are more measurements in the region. A station with an outlying measurement has more impact when there are less receivers available, since there is less information available of the surroundings of the station. Although there are just a few stations with measurements in figure 15, it looks like the CNN-MDN and the RF do understand that the PGV decreases when the distance from the epicenter increases. Both the CNN-MDN and the RF do underestimate the PGV of the closer stations, which is also visible in the scale of the right figures. The predictions of the MDN are seemingly arbitrary. The CNN-MSE also underestimates the PGV, and predicts a less clear pattern between distance and PGV. Figure 16 shows the same models but now they have to predict an event that was already in the training data. These models, especially the RF, had very high R^2-score (0.96). In this figure, the MDN model again predicts very different values from those actual measured, the predicted values for higher PGV are too low and the other way around. The MDN causes all the values to be more around the mean. It can be seen that the CNN-MDN also does

this to a lesser extent. The CNN-MSE is predicting the PGV too high almost everywhere. The RF is making an almost perfect prediction, there is almost no color difference between the the left and middle figures.

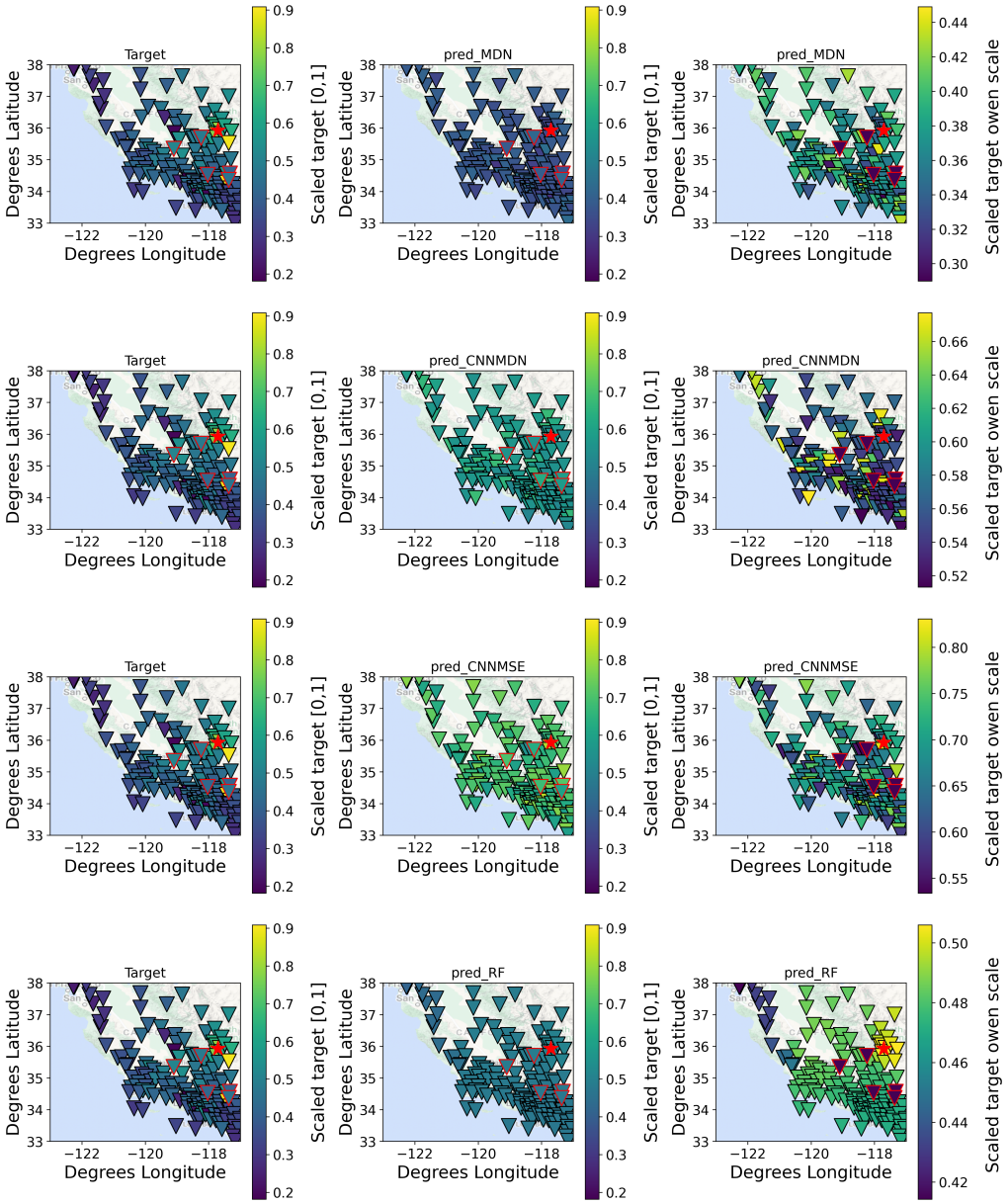


Figure 13: From top to bottom, the MDN, CNN-MDN, CNN-MSE and RF prediction for the PGV on every station location where the event is detected. The models are trained with 1000 combinations of 269 receivers of the training dataset. The highlighted event (red star) is new for the models. The 5 input receivers are outlined in red. *Left*: Receivers colored in their target value. *Middle*: Receivers colored in the prediction of the models in the same scale as on the left. *Right*: Receivers colored in the prediction of the models in a self determined scale.

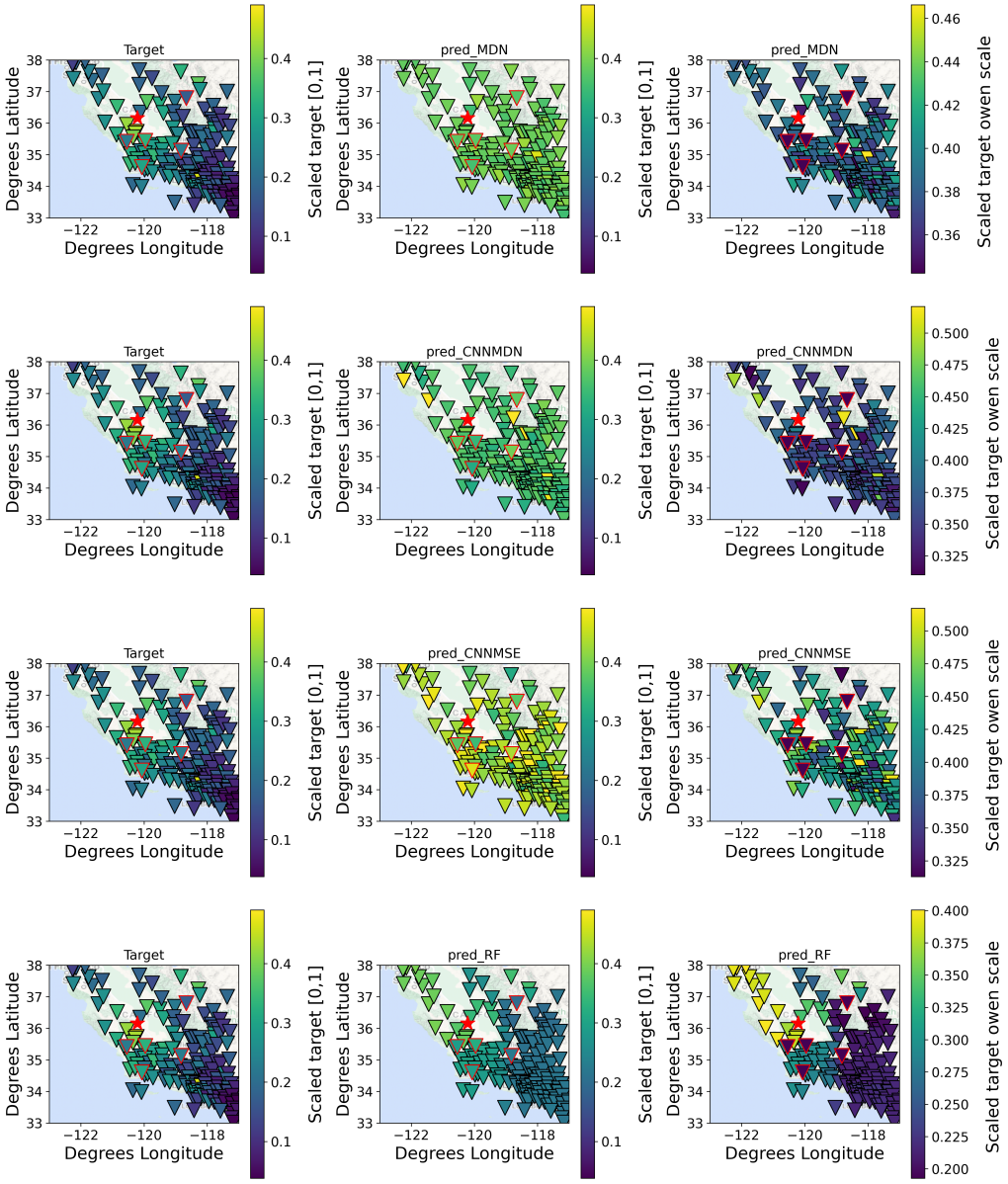


Figure 14: From top to bottom, the MDN, CNN-MDN, CNN-MSE and RF prediction for the PGV on every station location where the event is detected. The models are trained with 1000 combinations of 269 receivers of the complete dataset. The highlighted event (red star) was part of the dataset during training. The 5 input receivers are outlined in red. *Left*: Receivers colored in their target value. *Middle*: Receivers colored in the prediction of the models in the same scale as on the left. *Right*: Receivers colored in the prediction of the models in a self determined scale.

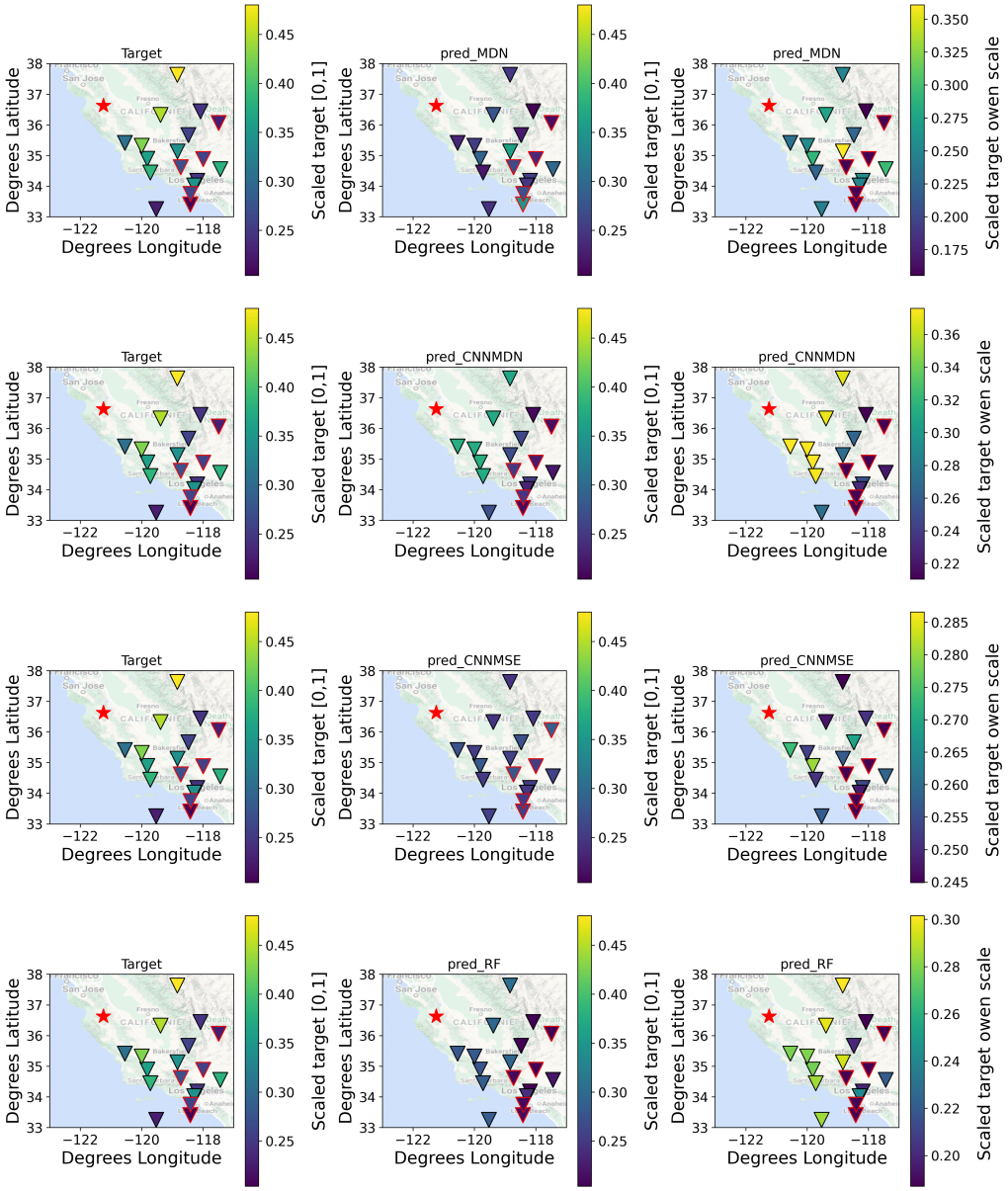


Figure 15: From top to bottom,, the MDN, CNN-MDN, CNN-MSE and RF prediction for the PGV on every station location the event is detected. The models are trained with 1000 combinations of 36 receivers of the train file. The highlighted event (red star) is new for the models. The 5 input receivers are outlined in red. *Left*: Receivers colored in their target value. *Middle*: Receivers colored in the prediction of the models in the same scale as on the left. *Right*: Receivers colored in the prediction of the models in a self determined scale.

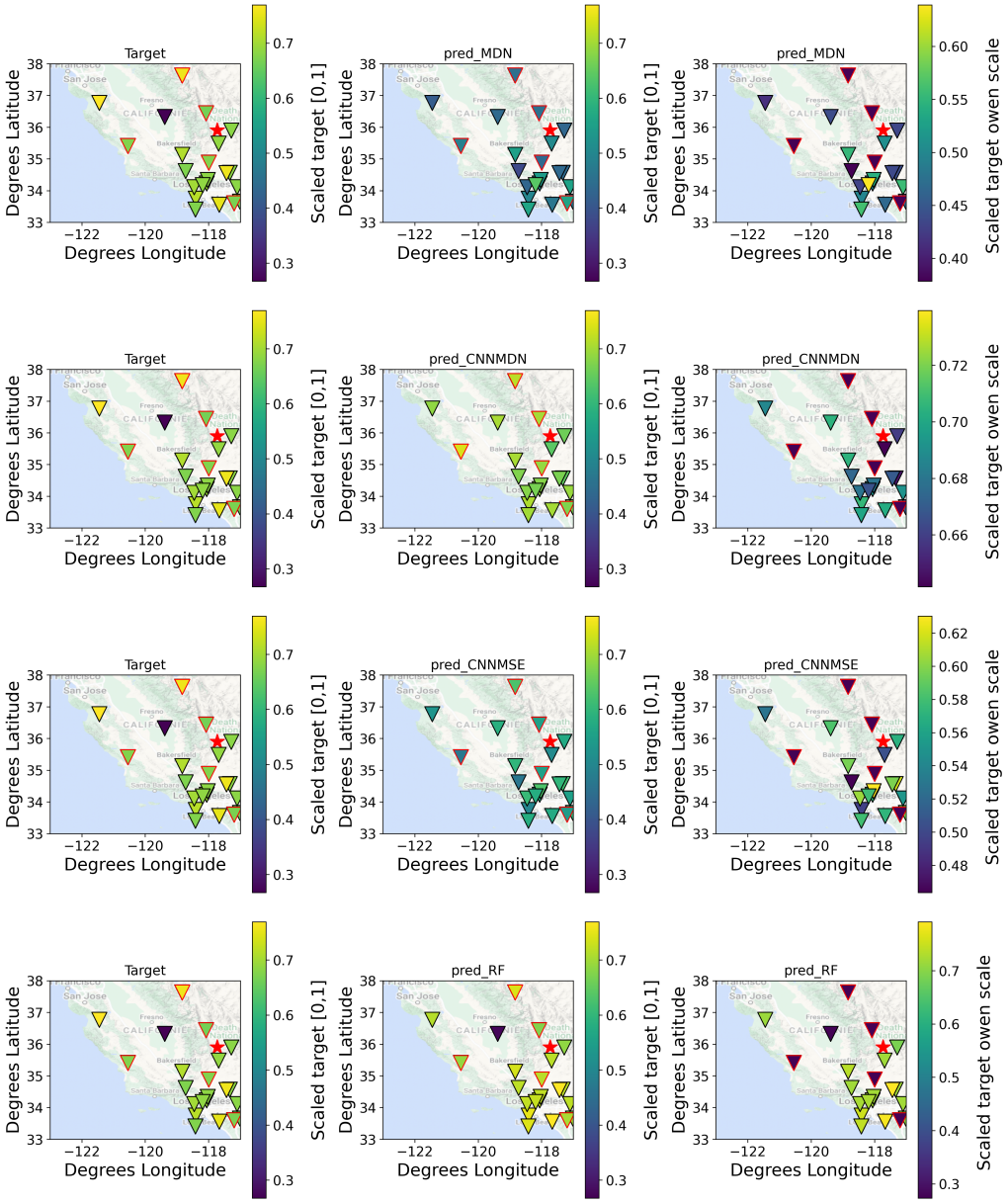
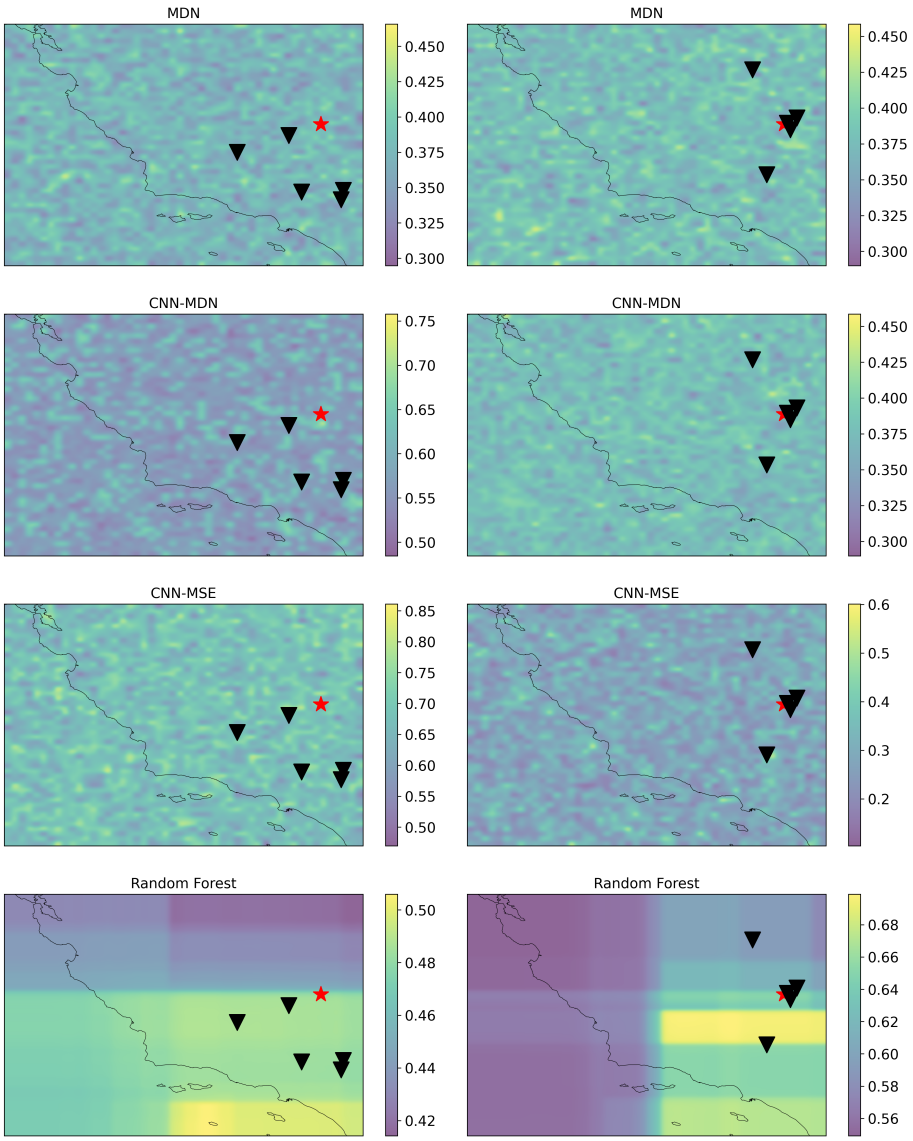


Figure 16: From top to bottom, the MDN, CNN-MDN, CNN-MSE and RF prediction for the PGV on every station location the event is detected. The models are trained with 1000 combinations of 36 receivers of the train file. The highlighted event (red star) was part of the dataset during training. The 5 input receivers are outlined in red. *Left*: Receivers colored in their target value. *Middle*: Receivers colored in the prediction of the models in the same scale as on the left. *Right*: Receivers colored in the prediction of the models in a self determined scale.

4.3 Location ζ at a random location in southern California

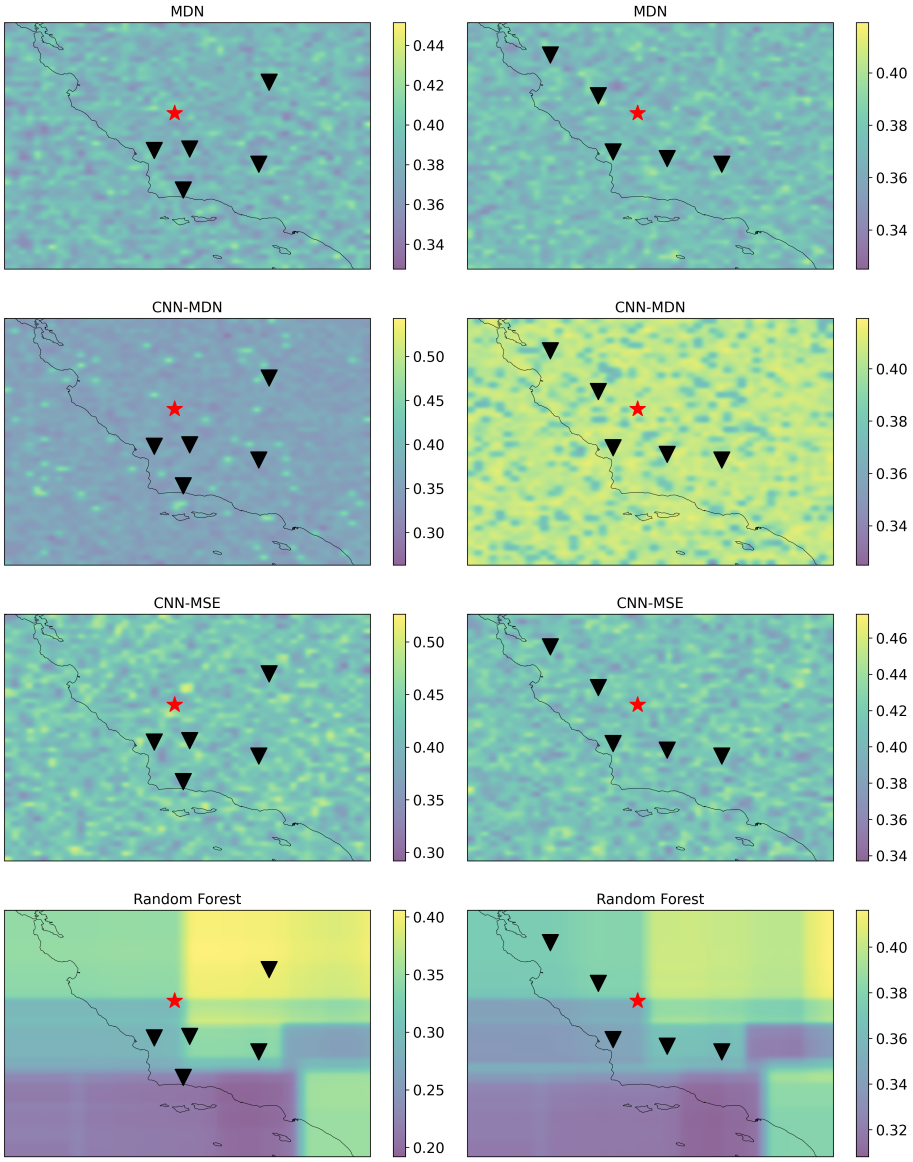
From the results in section 4.2 it seems like some of the models have actually learned how the PGV is evolving trough the area based on receiver locations and can predict the value of the PGV sometimes very close to the actual value. The models are trained with spatial coordinates, so they should be able to predict the PGV at a random location in the domain. To test this, a grid is created and the input location $\boldsymbol{\zeta}$ will now be any position on this grid. Seismic data of 5 receivers is given as input to the models with the accompanying longitude-latitude pairs.

Figure 17 shows the PGV prediction from all the models with the grid points as prediction location. Both sides of the figure show the prediction for the same event, but with a different set of input receivers. This figure shows that only the RF model does return something other than a noisy dotted pattern. Both predictions are, however, very different from each other, while they are predictions for the same event. On th left hand side, the highest PGV is predicted mostly in the ocean. On the right hand side the prediction of the PGV is higher almost at the event location. The PGV is also decreasing with increasing distance from the epicenter especially towards the north. However, it does not look like the pattern which was expected after the results in figure 13. The left side of figure 17 is made with the same model and uses the same input receivers for the same event as in figure 13. The uniform output pattern of the MDM and both the CNNs in figure 17 is also visible in the right hand figures of 13, where also no gradient pattern can be seen. The RF model in figure 13 does show a prediction that seems to understand the event location. However, the prediction on the left side of figure 17, which is made from the same input receivers, does not seem to understand this. For figures 14 and 18, 15 and 19 and figures 16 and 20 the same applies that they are made with the same models and the input receivers used in section 4.2 are the same as used for the left hand side of the figures in this section. The right hand side of the figures in this section are from a different combination of receivers to show the influence of the input receivers on the PGV predictions. Figure 18 shows two predictions for 2 different sets of 5 of the 269 receivers for an event which was already part of the training data. Here too, the predictions of the MDN and both CNNs are just noisy where every location has a prediction around the average. The RF model shows two predictions that do not differ a lot from each other. However, in both predictions, the highest PGV is more to the right of the actual epicenter. Figures 19 and 20 show the prediction of the model where less receivers were used during training, for respectively an event that was not seen during training and what was seen during training. Figure 19 is the only prediction for the CNN-MDN which shows a pattern different from a noisy dotted pattern. However, the pattern is unexpected and the model does not seem to understand the coordinate system in 2D. In both figure 19 and 20 the RF models does not return an expected prediction. The RF models which are trained with 36 stations are only capable of predicting the PGV at a receiver location like in figures 15 and 16.



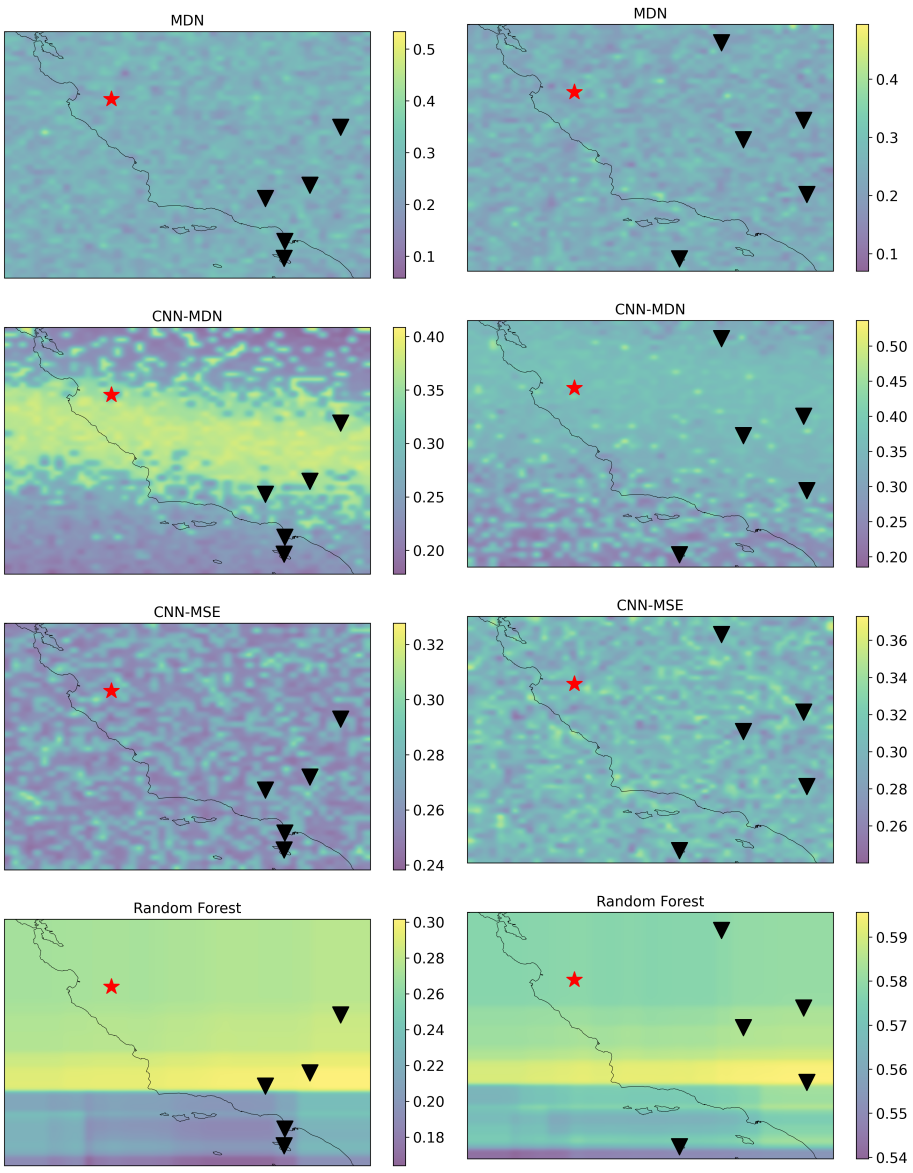
(a) (b)

Figure 17: PGV prediction of all locations on a grid of 50 by 50 in Southern California for two different combinations of random receivers (black triangles) on the left and right hand side, both for the same event (red star). The models are trained with 1000 combinations of all the 269 receivers and this is an event they have not seen during training.



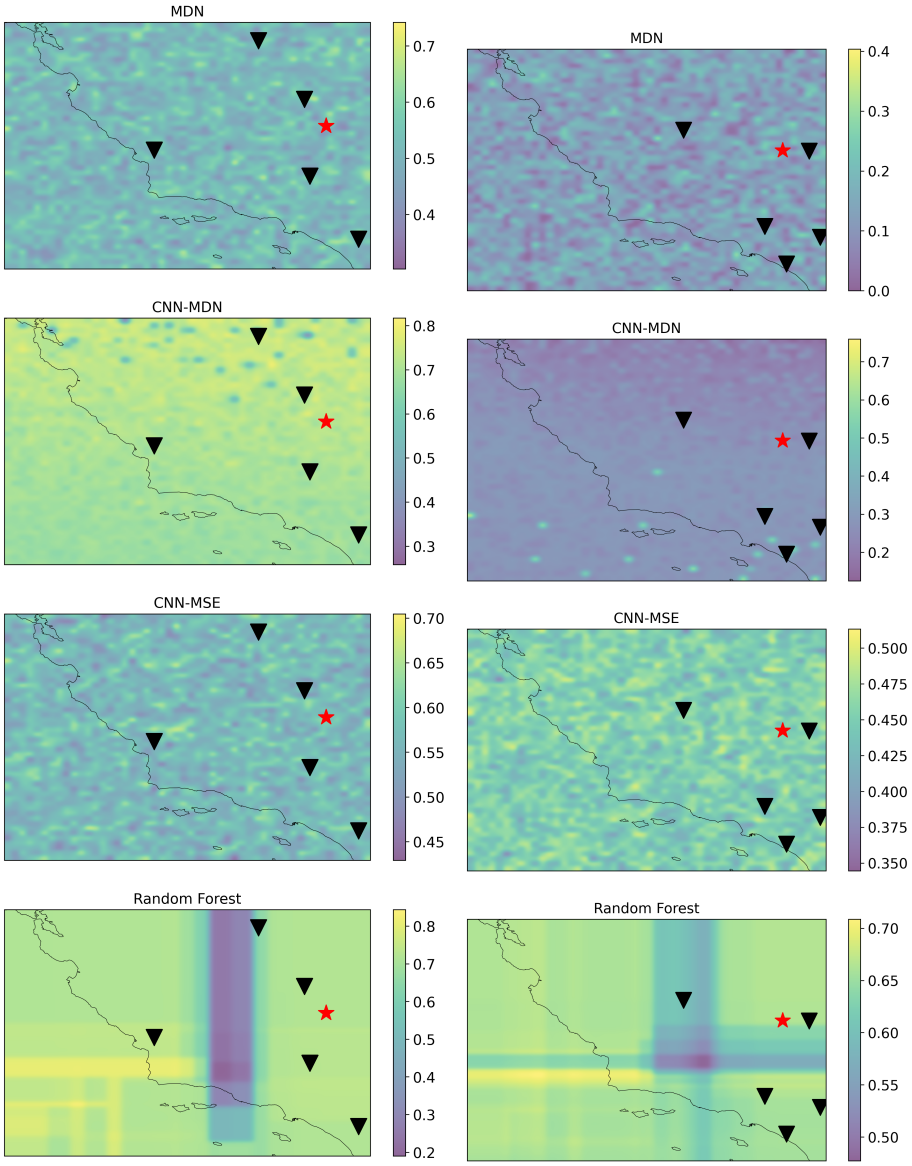
(a) (b)

Figure 18: PGV prediction of all locations on a grid of 50 by 50 in Southern California for two different combinations of random receivers (black triangles) on the left and right hand side, both for the same event (red star). The models are trained with 1000 combinations of all the 269 receivers and this is an event they have seen during training.



(a) (b)

Figure 19: PGV prediction of all locations on a grid of 50 by 50 in Southern California for two different combinations of random receivers (black triangles) on the left and right hand side, both for the same event (red star). The models are trained with 1000 combinations of the 36 receivers and this is an event they have not seen during training.



(a) (b)

Figure 20: PGV prediction of all locations on a grid of 50 by 50 in Southern California for two different combinations of random receivers (black triangles) on the left and right hand side, both for the same event (red star). The models are trained with 1000 combinations of the 36 receivers and this is an event they have seen during training.

4.4 Alternative parameters

A few quick trial runs have been made with some parameters adapted. First, the scaling of the input data is changed from scaling the seismic data with the maximum of the 5 input stations to scaling each seismic data input with its own maximum. Secondly, the locations are changed from longitude-latitude pairs to distance-back azimuth (from the epicenter). Lastly, the method of scaling the original longitude-latitude pairs is changed. Training the models is a time consuming process, since the models have to go through a lot of data. To save some time, testing the different parameter are done with 200 random combinations of receivers for each event. This will not improve the overall performance. However, when the models are trained for each different parameter with the same amount of combinations of receivers, they can be compared to each other. In table 3 the R^2-scores are displayed for the original training method and the 3 parameter changes for the models trained with all the 269 receivers. Table 4 shows the same for the models trained with the 36 receivers. The first remarkable observation is that in both tables the original training method, where the models are just trained with 200 combination of receivers from the training dataset and tested with 50 combination from the test dataset, is not performing significant worse from when they were trained with 1000 combinations (3rd column of tables 1 and 2). For some models the R^2-score is even higher. This despite the fact that the amount of data shown to the models during training is just 1/5 of the data they have seen during the training of the model for which R^2-scores are displayed in tables 1 and 2. When all the different parameters have been changed, the RF model is still the best performing model, regardless of the changes. In addition, the models which are trained with fewer receivers are returning a higher R^2-score. For both the amount of receivers, the RF model is returning the highest R^2-score when the input locations are given as a distance and back azimuth. Also the R^2-scores of both the CNNs increases when the input location is given by a distance and a back azimuth. However, a big disadvantage of defining the location by a distance and a back azimuth is that event information is needed. The distance and back azimuth are determined based on the location of the event. This would thus not be suitable for an EEW system, since the location of an event is not always determined in the first few seconds when an earthquake occurs.

Changing one of the other alternative parameters does not show a significant difference. The difference was too small to justify continuing running a bigger model with these parameters. For future research this could be an option to look into in more detail.

	R^2-score			
Neural Network	Original	Data scaling	Distance/Backazimuth	Location scaling
MDN	0.02	0.03	0.16	0.04
CNN-MDN	0.14	0.10	0.40	0.16
CNN-MSE	0.14	0.26	0.31	0.22
RF	0.63	0.63	0.64	0.61

Table 3: R^2-score for different parameter for models that are trained with 200 combinations of all the 269 receivers from the training file and 50 combinations of receivers of the test file.

	R^2-score			
Neural Network	Original	Data scaling	Distance/Backazimuth	Location scaling
MDN	0.15	0.08	0.16	0.05
CNN-MDN	0.27	0.27	0.30	0.26
CNN-MSE	0.30	0.37	0.36	0.31
RF	0.77	0.78	0.79	0.79

Table 4: R^2-score for different parameter for models that are trained with 200 combinations ofthe 36 receivers from the training file and 50 combinations of receivers of the testfile.

5 Discussion

In this research, four different types of models are trained with different datasets and varying parameters to search for the best possible model for predicting peak ground velocity (PGV) in southern California. This training is done with the first 35 seconds of seismic velocity data of 5 random receivers located at $\boldsymbol{X_r}$ which all detected the event. The peak ground velocity (PGV) is the maximum motion of the ground during an earthquake, at a certain location. This peak ground motion and the S-wave which includes the PGV, can be destroying for inhabitants of a seismic active area. This S-wave has relative to the P-wave a slow propagation velocity. When the distance from the epicenter increases the time between the P-wave and S-wave arrival also increases. In this research real-time seismic data is combined with geographical locations to train the different models. They are trained without event information, such as magnitude and location. In addition to training the models without event information, giving them only the first few second of seismic makes that the results can possibly be used for an EEW system. Derakhshani and Foruzan (2019) already succeeded in estimating PGV among other motion parameters very well, however this was with event information as input.

This research is based on the guided research of Kuijpers (2021). She did a similar research for the Groningen area. With the first 6 seconds of seismically data from 5 of the 79 available receivers in the area she tried to predict the PGV on a 6th location denoted as ζ, just like in this research. However, in the Groningen area, there are no large earthquakes and outside the relative small area, the receiver network is very limited. The study area used in this research is more than 50 times larger than in the study area of Kuijpers (2021) in Groningen, there is a very dense receiver network, and there are lots of earthquakes, including bigger ones. Making the area bigger increases the time an S-wave travels from one side of the domain to the other side of the domain which means that the input of seismic data need to have a longer time range, which was in this research 35 seconds. 5 seconds before first P-wave arrival and 30 second after first P-wave arrival. The 35 seconds is based on the average distance between the events and the receivers which is 172 km. An S-wave has an average velocity of 4.5 km/s and 8.1 km/s for a P-wave (Rajasekaran, 2009), which means that the S-wave should travel up to 135 km and the P-wave up to 243 km in 30 seconds. In practice, not all the stations in a distance of 135 km around the epicenter detect the S-wave, this can be caused for example by a different geological

structure of receiver positioning. In this research, different random combinations are made with receivers. For 35 seconds of data and 1000 combinations of random receivers, 22 percent of receivers present in these combinations contained the PGV. This percentage is calculated such that when a receiver which detects the PGV and is more than once present for 1 event, it counts for the amount of times it is present. So, this percentage is the actual input data containing the PGV. This percentage is probably a bit overrated since sometimes the target was from a station where data was downloaded from, but did not actually measure the event due to noise. The length of the trace of seismic data, that is given to the models during training, has a huge impact on the training time. When a trace is longer (thus a bigger time frame) the amount of parameters in the models increases and so does the training time. When the length of the input time frame is increased, the predictions should also be improved, however with increasing the time frame also the speed of the predictions will be decreased, while the ultimate goal is to make predictions when the wave of the earthquake has not reached all the vulnerable locations yet. In addition, the performance of the models should improve when more data is given to the models during training, however, this also increases the training time. A great advantage of this research and that of Kuijpers (2021) is that the models are trained to combine spatial information with seismic input data, which makes it possible to use the models also on locations without receivers. In addition, the models do not need any input information about the geological structure in the study area, they have to learn themselves based on the physics of wave propagation. This has the benefit that nothing has to be known about Earth's parameters in the research area. However, this does mean that the models have to be retrained if the same research is performed at a different location, since earth's structure differ at every place.

5.1 Description of Datasets and effects on predictions

Real-time seismic data is very complex. For example, it includes noise, different types of wave forms and sometime multiple Earthquakes happen in the 30 second time window. To be able to deal with the noise, it is chosen that only earthquakes with a magnitude larger than 4.0 are taken into account. Earthquakes with this magnitude have a ground velocity significant higher than the noise, which ensures that the noise is clearly distinguishable without using features to filter them out. Larger earthquakes often have a series of aftershocks, however, these can be often used as new events to train the models. The only problem with this is that the PGV is based on the highest velocity in the time window and this will thus not necessary be the corresponding event when this occurs in the same time window. After the data was downloaded from *IRIS* the events were sorted based on EOT. When earthquakes had an EOT less than 100 ms apart, they are seen as the same event. This is is done because there were otherwise a lot less receivers per event and a lot more events, which not differ from each other. In addition, on the website of IRIS events are sorted based on seconds. Events detected at less than 5 receivers were discarded to prevent bias, in practise there were eventually none when using all the 269 receivers. With just the 36 receivers, a few events were filtered out by this. The first models were made with 36 receivers. It is still unclear how this selection of 36 receivers was made. These

receivers are all from the Southern California Earthquake Data Center (SCEDC) network. However, these were the only stations which were downloaded with the MassDownloader tool with only IRIS added as provider. When the providers "SCEDC" and "NCEDC" (Northern California Earthquake Data Center) were added to the MassDownloader, data from the 269 receivers was downloaded, mostly from the SCEDC network. While more data should result in better models, this does give an opportunity to compare a dense receiver network area with a less dense receiver network area. This could be beneficial since not all seismically active regions in the world have a dense receiver network.

For making the random combinations between the receivers including the prediction location, it is chosen to make this combination only between the 80 closest stations. To be able to use the models for a potential EEW system, it is preferred that the models can predict the PGV based on the first stations that detect an Earthquake. However, the models need to train on location further from the epicenter to be able to predict PGV also in those places. The amount of random combinations of receivers has a huge impact on the performance of the models. The more combinations chosen, the more combinations the models have seen during training. When a specific receiver location is present in the data a few times, it is easier for the models to really understand their spatial location. This could also be the reason why the models which are trained with fewer receivers returned such high R^2-scores. When the random combination between receivers is made, it is possible that the data of 1 receiver is multiple times in the input data for 1 target. So, when for example 50 receivers are available, it is possible that the 5 input stations are: 7, 16, 16, 33, 47. This process is completely random. When the station for location ζ is chosen, it is made sure that this was not one of the receivers in the input data for that target. So, the target can never be part of the input data.

The dataset in which just 36 receivers were available does surprisingly well at predicting PGV on at least the receiver locations, even with completely new data. The models have seen the data of each event per receiver many times, at least more often that in the dataset where 269 receivers were present. This dataset is also made with 1000 combinations of random receivers and tested with 100 combinations of random receivers. However, since there are now less receivers, the probability that the same set of receivers is given as input increases. This was also the case, however still 85 percent of the combinations were new. Table 2 shows that when using the same train and test dataset gives very high R^2-scores. This high score does not necessary mean that the models actually learned the physics behind the wave forms. Because every part of data is seen so many times during training, it is possible that the model can remember this, or the overlapping combination contributes for a large part to the high score. However, this dataset is also split into two and tested with completely new data, this is the 3rd column in table 2. The R^2-scores of the CNNs is decreasing a lot, however the RF has still a pretty high score. This suggests the model can handle new data, and thus also new events that are happening. It is expected that when more combinations are used for the dataset with all the receivers, the RF will perform very well and maybe even understand the real physics behind it. This would be interesting to investigate in a future project.

When the models were tested with the data from the older events, the performance of

all the models was very bad. When looking at figure 12, which shows which events are part of the different datasets, it becomes clear that the older events (shown in red) have a different distribution in the area than the newer event (shown in blue). The models that were tested with the red event in this figure were trained with all the blue events (so dark and light blue). In this figure the blue events have 2 locations where a lot of events have occurred. A bit south of San Francisco and a bit west of Las Vegas. Especially in this last area, there is a very high concentration of events. The older events are plotted on top of the younger events, so it is not the case that there are hidden red stars underneath the blue cluster. When looking at the EOTs in the data it becomes clear that this area has become very seismically active since 2019. When looking at the rest of the domain outside this cluster of events, the amount of training data points (all the blue stars) is hardly more that the testing data points (all the red stars). This could be an explanation why the the models are performing so bad with the older testing data. They are simply not trained well enough on data outside these clusters.

5.2 Testing method

The models created in this research are tested with events that were already in the training dataset and with completely new events. Kuijpers (2021) tested her models based on the assumption that different combinations of receivers from the same events are seen as new data for the models. She noted, however, that with this testing method, the testing and training dataset are generated with the same method and thus follow the same underlying distribution. In this research there is a lot of data available for testing and training the models with different event data. The training and testing datasets are made with the same method, however since these are completely new events it does not matter in which distribution they are given to the models. Even with a lot of data, it can be that the distribution of the events in the training and testing data is not evenly divided. This difference in distribution is visible in red in figures 9, 10 and 11. This last figure shows that all the models have a bias towards the training dataset and have a lack of predictions higher than 0.8 and lower than 0.3. The R^2-scores of all the trained models show that training and testing the models with different event data has a significant effect on the performance of the model. Testing the models with only an event that is not new, can give a wrong impression about the quality of the models. To be able to know if a model actually understands the relation between PGV and spatial coordinates, it is essential to make a separate training and testing dataset.

5.3 Performance of the models

The four different models that are trained with different datasets in this research all performed very differently. To give a very general and brief overview of the results in this research: tables 1 and 2 suggest that based on R^2-score, both the CNNs perform well when they have to predict PGV from events they already have seen. However, when they have to process new event data their performance clearly worsens. This means that the CNN trained in this research would not be able to process new incoming data properly. It is also clear that the MDN is not able to do this, it is performing very bad with new event data as well as with data from events already seen. For all the tested train-test combinations, the MDN is the worst and the RF the best performing model, see tables 1 and 2. They are discussed in more detail in the next sections.

5.3.1 Mixture Density Network

In this research a single layer Mixture Density Network (MDN) is trained and tested, which schematically looks like shown in figure 7. The performance of the model was very bad with all the tests that have been performed with it. It clearly does not understand the input data and certainly also not the connection between the input data. It seems like the MDN almost predicts arbitrary values somewhere around the mean value. In the plots in section 4.1, it is already visible that the MDN always shows a very high peak in PGV predictions which is correlated to the mean value of the training target distribution. This peak is around 0.4 when the model is trained with all the receivers. This corresponds to the peak in figure 5. When it is trained with 36 receivers, the peak is also around 0.4. The distribution of the target for the 36 receivers can be seen in red in figure 10. The MDN has a very strong bias towards this mean value and is therefore not suitable for this type of prediction. This is very similar to the finding from Kuijpers (2021), where the MDN also failed to predict the PGV of real-time seismic data. However, in that same research the MDN did seem to learn something, in contrast to this research. The MDN in her research did show a gradient PGV, however it did not seem to understand that waves travel in all directions. There are some big differences in especially the data between this research and that of Kuijpers (2021), which may explain the difference in performance. In this research, the models are trained with data from broadband channels, which had a sampling rate of 1/40, which means that the data is sampled 40 times per second. Kuijpers (2021) used long-period channels which are sampled 1 time per second. Broadband channels are beneficial for observing regional earthquakes and P-wave events. They contain a lot more data than long-period channels. Together with the longer period of time for seismic input data, this gives the model a lot more data to process in each step. Maybe it is too hard for the MDN to find the difference in the input data because of this. In addition, due to the bigger amount of data, all the models have seen relative less data per event during training. It could be that the MDN will perform better when it is trained with more combinations of receivers. An advantage of a MDN is that it gives a probability distribution as output. This gives more information about what the model is actually doing and about the uncertainties in the predictions. However, in this research

the predictive value of the MDN models is always chosen to be the most common value in this probability distribution which causes this information to be lost. In this research a single layer MDN is created. The performance of the MDN would probably be better when more layers are added or different MDNs are combined.

5.3.2 Convolutional Neural Networks

The two different Convolutional Neural Networks (CNNs) are the only neural network in this research to which the input data is given as separate matrices or vectors. This has the advantage that the models do not have to figure out how the input data is constructed. However, even with this separation of the input data, both CNNs do not seem to understand the spatial dependence of the PGV. For both CNNs the performance decreases significantly when they are tested with data from new events. The predictions for a target somewhere on a grid which covers the domain are also bad. For almost all the models the CNNs show a dotted noisy pattern, just like the MDN. There is no big difference found in the performance of the CNN-MDN and the CNN-MSE. Both models give very similar R^2-scores for all the test and also in the plots there is no big difference between the two. In the research of Kuijpers (2021) there was a big difference between the models. In her research, the CNN-MDN was the worst performing model for both prediction location $\boldsymbol{\zeta}$ on $\boldsymbol{X_r}$ or at a random location. Her CNN-MSE did seem to understand some of the spatial dependence, but also show some dotted patterns.

A CNN has a very complex structure. When a CNN is applied to a new application, the hyperparameters need to be determined. These are for example the learning rate, kernel size or the number of layers (Gu et al., 2018). For these first two parameters a few different things are tried, all in the same range of values as used by Kuijpers (2021). Typical kernel sizes are for example 3 or 5. During training, a CNN extracts important features from the the data, the kernel size is a parameter which determines how much data the model sees at once. A larger kernel size allows the model to capture more global patterns, while a smaller kernel size focuses on local details in the data. In this research, the models are eventually trained with a kernel size of 3. Also different learning rates are tried, but no significant differences were found.

Jozinović et al. (2020) used a CNN to predict ground motion in central Italy with a similar approach. Their result show that their CNN can predict this ground motion very well, which gave the expectations that it could also work in this research. However, they train their CNN with all the receivers at once. In this research the CNN was only trained with the input of just 5 receivers and it had to figure out itself when combinations of receivers are from the same event. Also in the research from Perol et al. (2018) and Lomax et al. (2019) the created CNNs gave some very promising results. They respectively developed and adapted the algorithms *ConvNetQuake* and *ConvNetQuake_INGV* which are developed for local earthquake detection and location information. Gu et al. (2018) gives an overview of different applications of CNNs. CNNs work very well for different kind of application. So, in this research both of the CNNs did not gave the expected results, however, with more and maybe different training they could be potentially suitable for PGV predictions.

5.3.3 Random Forest

Random Forest (RF) is, as expected from the research by Kuijpers (2021), the best performing model for the prediction of PGV. Although a RF is maybe easy to use, and easy to understand how it is constructed, it is very hard to know and interpret which features in the data the RF uses to make the predictions. It is possible to look into the decision trees in Python, however, as mentioned these are 1000 trees so this is not a quick task to figure out. The total training process of the RF used in this research is by far the most time consuming process. This is in contrast to the research of Kuijpers (2021), which uses the same RF model and more combinations of receivers per event. Apparently, training the RF is taking considerably longer when it has to handle longer traces. In addition, the RF model is the slowest model when it has to predict new data after training. Each tree in a RF has to make a decision which causes the model to take more time than the other models. However, decreasing the amount of trees is not beneficial for the performance of the model. In the research of Kuijpers (2021) the RF predicts the PGV of events it already had seen during training pretty well. During training each tree, it has just seen a portion of the training data during training of which the average is determined for the final prediction. So, when the model is tested with events that were already in the training data it could be that the RF model extract the same patterns. However, in this research the RF model is still performing well when completely new events are given to the model. Therefore, there is a high change that the decisions the model makes are actual because it understands the data. The disadvantage from the RF model is that each tree can not predict something else than the end leaf it has chosen. This causes the model to be very depending on with what data it is trained with. The data has to have a widely variation of all possible situations. Figure 11d shows that the model has a lack of very high and low predictions, even though these values are present in the training data. However, these values are already less common and when the dataset was split into a testing an training dataset it could be that the RF model has seen to few of these cases during training to make a good prediction. When the RF is trained and tested with the same dataset, as in figures 9d and 10d, this problem occurs on a much smaller scale. This distribution difference between training and testing data is not only a problem for the RF model, also for the other trained models. The only difference is that due to the structure of the RF model, it would never be able for the RF model to predict a PGV value outside the training domain. So for example, when a very large earthquake occurs, it could only predict the highest PGV seen during training. The distribution of the dataset with less receivers has a very even distribution with also a lot of lower values over the complete scaled domain from 0 to 1. It is expected that the RF model prediction has a very high R^2-score partly due to this good distribution.

As mentioned before, it is unclear what the decisions of the RF are based on, and thus also unclear why it is performing the best. Using a RF model to predict seismic activity is not new. The RF model from Rouet-Leduc et al. (2017) for example could almost predict perfectly the timing of a "labquake". However, this research did not include real-time seismic data. This is very different from laboratory generated earthquakes, since real-time seismic data is influenced by external factors such as geology. Other research which uses RF

for predicting ground motions are not found. For future research this would be interesting to further investigate.

5.4 Limitations and Future Work

Training the models is very time consuming. Especially the RF model takes very long to train. For example, training the RF model with 1000 random combinations of all the 269 receivers took around 4 days. The training time depends on the amount of parameters used or degrees of freedom. A large number of degrees of freedom will take a lot more time. The MDN was by far the quickest model to train. This model was tested with 5000 combinations of random receivers per event which should improve the model's performance. However, even with this amount of combinations, the model did not meet the expectations. It would be interesting to see the performance of the models with the same dataset, but trained with more combinations of random receivers. To reduce the training time of all the models, a smaller time frame of seismic data can be used, allowing more of the training time to be used for more combinations of receivers.

Different datasets with different distributions can be made to sort out what their effect is on all the models. Will it indeed be the case that the RF model becomes better when the distribution is more evenly divided into testing and training data?

In addition, the alternative parameters could be investigated in more detail and maybe even more variations can be explored. From the R^2-scores in tables 3 and 4, it is suggested that changing the training method could improve the performance, especially for the CNNs.

It is unknown why the RF model performed so much better than the other 3 models. It would be interesting to really understand what the RF model is doing. In Python, there are alternative functions to look better into this.

From other researches, like Derakhshani and Foruzan (2019) for example, it is known that PGV can be predicted from earthquake information. Making a good prediction where no event information is needed does not seem to be impossible. This matches some of the results obtained in this research. This may be interesting for further research.

It is possible to create a lot of different structures for machine learning models, which are by far not all discussed in this research. It could be interesting to look at more different model structures or maybe extend the structures used in this research.

The research area used in this research was very big, which also contributed largely to the training time. An other disadvantage of this big area is that the models probably have to learn more about the effects of different geological structures. In addition, a big part of the domain is ocean, where no data is available, this is another feature the models have to learn but which can be solved with a different shaped domain. A large number of people live in the domain of the study area, and also earthquakes occur throughout the whole domain. This makes it interesting to make the models work throughout the domain.

6 Conclusion

In this research, the performance of four different types of models are tested with different datasets and varying parameters to search for the best possible model for predicting peak ground velocity (PGV) in southern California. The Random Forest (RF) model is the best performing model and seems to be able to combine spatial input data with seismic data, especially when the PGV is predicted on an existing receiver location. This is in line with the findings of Kuijpers (2021). However, the predictions are not perfect, and a lot of variations can be tried to possibly improve the performance of the model. In addition, the model takes a very long time to be trained. The Convolutional Neural Networks (CNN-MDN and CNN-MSE) and the Mixture Density Network (MDN) performed unexpectedly worse. The performance of both the CNNs was decreasing a lot when they were tested with new event data. The MDN model does not give good results in any of the tests performed in this research. All the models gave higher R^2-scores when they were trained with less receivers. However, this high score is not obtained because they have actually learned the relation between the spatial coordinates and the seismic data. The high R^2-scores could result from the models learning or remembering the behaviour of a receiver without learning their dependence on each other. Therefore, training the models with less receivers is not an option to improve predictions.

Acknowledgements

- All seismic data were downloaded through the EarthScope Consortium Wilber 3 system (https://ds.iris.edu/wilber3/), including the following seismic networks: (1) the AZ (ANZA; UC San Diego, 1982); (2) the TA (Transportable Array; IRIS, 2003); (3) the US (USNSN, Albuquerque, 1990); (4) the IU (GSN; Albuquerque, 1988).
- This research is following the method of Kuijpers (2021), some of her Python codes are used or adapted in this research.
- I am grateful to my supervisors Prof. dr. Jeannot Trampert and dr. Elmer Ruigrok for their help with this research.

References

Adeli, H., & Panakkat, A. (2009). A probabilistic neural network for earthquake magnitude prediction. *Neural networks*, *22*(7), 1018–1024.

Allen, R. M., & Kanamori, H. (2003). The potential for earthquake early warning in southern california. *Science*, *300*(5620), 786–789.

Barruol, G., & Mainprice, D. (1993). A quantitative evaluation of the contribution of crustal rocks to the shear-wave splitting of teleseismic sks waves. *Physics of the Earth and Planetary Interiors*, *78*(3-4), 281–300.

Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, *25*, 197–227.

Bishop, C. M. (1994). Mixture density networks.

Boureau, Y.-L., Ponce, J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. *Proceedings of the 27th international conference on machine learning (ICML-10)*, 111–118.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

Cooper, J. (1868). Letter to editor. san francisco daily evening bulletin.

Derakhshani, A., & Foruzan, A. H. (2019). Predicting the principal strong ground motion parameters: A deep learning approach. *Applied Soft Computing*, *80*, 192–201.

Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In S.-i. Amari & M. A. Arbib (Eds.), *Competition and cooperation in neural nets* (pp. 267–285). Springer Berlin Heidelberg.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern recognition*, *77*, 354–377.

Hao, J., & Ho, T. K. (2019). Machine learning made easy: A review of scikit-learn package in python programming language. *Journal of Educational and Behavioral Statistics*, *44*(3), 348–361. https://doi.org/10.3102/1076998619832248

Hoshiba, M., Kamigaichi, O., Saito, M., Tsukada, S., & Hamada, N. (2008). Earthquake early warning starts nationwide in japan. *EOS, Transactions American geophysical union*, *89*(8), 73–74.

Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, *195*(1), 215–243.

Hutton, K., Woessner, J., & Hauksson, E. (2010). Earthquake monitoring in southern california for seventy-seven years (1932–2008). *Bulletin of the Seismological Society of America*, *100*(2), 423–446.

Jozinović, D., Lomax, A., Štajduhar, I., & Michelini, A. (2020). Rapid prediction of earthquake ground shaking intensity using raw waveform data and a convolutional neural network. *Geophysical Journal International*, *222*(2), 1379–1389.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kong, Q., Trugman, D. T., Ross, Z. E., Bianco, M. J., Meade, B. J., & Gerstoft, P. (2018). Machine Learning in Seismology: Turning Data into Insights. *Seismological Research Letters*, *90*(1), 3–14. https://doi.org/10.1785/0220180259

Kuijpers, D. (2021). Predicting peak ground velocity using machine learning.

Legates, D. R., & McCabe Jr, G. J. (1999). Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation. *Water resources research*, *35*(1), 233–241.

Lomax, A., Michelini, A., & Jozinović, D. (2019). An investigation of rapid earthquake characterization using single-station waveforms and a convolutional neural network. *Seismological Research Letters*, *90*(2A), 517–529.

Marti, M., Stauffacher, M., & Wiemer, S. (2019). Difficulties in explaining complex issues with maps: Evaluating seismic hazard communication–the swiss case. *Natural Hazards and Earth System Sciences*, *19*(12), 2677–2700.

Meier, M.-A. (2017). How "good" are real-time ground motion predictions from earthquake early warning systems? *Journal of Geophysical Research: Solid Earth*, *122*(7), 5561–5577.

Mogi, K. (1985). Earthquake prediction.

Mohammadi, A., Karimzadeh, S., Banimahd, S. A., Ozsarac, V., & Lourenço, P. B. (2023). The potential of region-specific machine-learning-based ground motion models: Application to turkey. *Soil Dynamics and Earthquake Engineering*, *172*, 108008.

Oliver Holmes, E. M., & Sheehy, F. (2023). *Thousands dead, millions displaced: The earthquake fallout in turkey and syria*. Retrieved February 20, 2023, from https://www.theguardian.com/world/2023/feb/20/thousands-dead-millions-displaced-the-earthquake-fallout-in-turkey-and-syria

Perol, T., Gharbi, M., & Denolle, M. (2018). Convolutional neural network for earthquake detection and location. *Science Advances*, *4*(2), e1700578.

Rajasekaran, S. (2009). *Structural dynamics of earthquake engineering: Theory and application using mathematica and matlab*. Elsevier.

Rikitake, T. (1968). Earthquake prediction. *Earth-Science Reviews*, *4*, 245–282.

Rouet-Leduc, B., Hulbert, C., Lubbers, N., Barros, K., Humphreys, C. J., & Johnson, P. A. (2017). Machine learning predicts laboratory earthquakes. *Geophysical Research Letters*, *44*(18), 9276–9282.

Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. *2013 international conference on collaboration technologies and systems (CTS)*, 42–47.

Sahour, H., Gholami, V., Torkman, J., Vazifedan, M., & Saeedi, S. (2021). Random forest and extreme gradient boosting algorithms for streamflow modeling using vessel features and tree-rings. *Environmental Earth Sciences*, *80*. https://doi.org/10.1007/s12665-021-10054-5

Satriano, C., Wu, Y.-M., Zollo, A., & Kanamori, H. (2011). Earthquake early warning: Concepts, methods and physical grounds. *Soil Dynamics and Earthquake Engineering*, *31*(2), 106–118.

Siahkoohi, A., Kumar, R., & Herrmann, F. (2018). Seismic data reconstruction with generative adversarial networks. *80th EAGE conference and exhibition 2018*, *2018*(1), 1–5.

Times, T. N. Y. (2023). *Morocco races to dig out survivors after strongest quake in 100 years*. Retrieved September 10, 2023, from https://www.nytimes.com/live/2023/09/10/world/earthquake-morocco-marrakesh-news

Turcotte, D. L. (1991). Earthquake prediction. *Annual review of earth and planetary sciences*, *19*(1), 263–281.

Zhang, H., Liu, Y., Yan, J., Han, S., Li, L., & Long, Q. (2020). Improved deep mixture density network for regional wind power probabilistic forecasting. *IEEE Transactions on Power Systems*, *35*(4), 2549–2560. https://doi.org/10.1109/TPWRS.2020.2971607

Zhou, Z.-H. (2021). *Machine learning*. Springer Nature.

Zoutendijk, M., & Mitici, M. (2021). Probabilistic flight delay predictions using machine learning and applications to the flight-to-gate assignment problem. *Aerospace*, *8*(6), 152.