SeisNet: A simple CNN for rapidly classifying historic records from the WWSSN

T.P. Nagle-McNaughton1,2\*, A.T. Ringler2, R.E. Anthony2, D.C. Wilson2

\*Corresponding author

1The University of New Mexico

2United States Geological Survey-Albuquerque Seismic Laboratory

# Abstract

# 1. Introduction

At the data of writing, the USGS has \_\_\_,\_\_\_ film chips scans, comprising some 8TB of data. Even with the help of interns, parsing through this archive to identify interesting seismic events would be a daunting and labor-intensive undertaking.

# 2. Methods and data

## 2.X Data pipeline

MORE HERE.

Diagram

Description automatically generated

Figure X. The overall processing pipeline for classifying film chips. 1) A film chip is downloaded and loaded into the processing directory. 2) The edges of the tiles that contain metadata and scan artifacts are masked/cropped out. 3) The film chip is converted from 8-bit greyscale to binary. 4) A random 200x200 tile is cropped out of the masked area. 5) The overall brightness of the tile is evaluated, if it is too dark, it is ignored and a new crop is taken, if the tile is bright enough, it is evaluated. 6) The tile is classified by SeisNet. 7) The results are saved to a database. 8) The cropping and classification process is repeated until the user-defined minimum sample threshold is achieved. 9) An overall label for the film chip is derived from the database of tile classifications.

## 2.X Model architecture

SeisNet is a relatively simple sequential CNN developed using the Python package Keras. SeisNet is composed of three convolutional blocks, one ANN block, a dropout layer, and an output layer (Table X). In total the model has 598,124 parameters, of which all but 384 were trainable. In total, the full model and all of its parameters are only 7.5MB in size.

Table X. SeisNet architecture. Line colors indicate the different blocks.

|  |  |  |  |
| --- | --- | --- | --- |
| **Block** | **Layer type** | **Output shape** | **Number of parameters** |
| Convolution 1 | Conv2D | (None, 200, 200, 32) | 832 |
| MaxPooling2 | (None, 100, 100, 32) | 0 |
| Convolution 2 | Conv2D | (None, 50, 50, 64) | 51,264 |
| MaxPooling2 | (None, 25, 25, 64) | 0 |
| Batch normalization | (None, 25, 25, 64) | 256 |
| Convolution 3 | Conv2D | (None, 13, 13, 128) | 73,856 |
| MaxPooling2 | (None, 6, 6, 128) | 0 |
| Batch normalization | (None, 6, 6, 128) | 512 |
| ANN | Flatten | (None, 4608) | 0 |
| Dense | (None, 100) | 450,900 |
| Dense | (None, 100) | 10,100 |
| Dropout | Dropout | (None, 100) | 0 |
| Output | Dense | (None, 4) | 404 |

## 2.X Training dataset generation

SeisNet was trained on 18,154 200x200 tiles randomly sampled from 460 scanned film chips. These tiles were manually classified by the first author into four classes: 1) no events, 2) minor or micro events, 3) major events, and 4) scan errors and calibration pulses. The class breakdown of the training set is given in Table X.

Table X. Training dataset class counts.

|  |  |
| --- | --- |
| **Class name** | **Number of occurrences in training data** |
| No events | 6,539 |
| Minor events | 8,393 |
| Major events | 3,049 |
| Errors | 173 |

The dataset was randomly shuffled, and then split into a training set and a validation set: 80% for training and 20% for validation during training. The training tiles were augmented by random vertical and horizontal flips. This augmentation helps the model generalize and prevents overfitting [1–3].

## 2.X Model training

During training, the model used the Adam optimizer and a learning rate of 1x10-5 [4]. As the model trained, the model was saved every time it improved its performance on the validation dataset as measured by accuracy. The dropout rate was set to 40% to prevent the model from overfitting. Dropout resets some fraction of the nodes by temporarily making their weights equal to 0 [5,6]. Some fraction of the layer nodes are thus randomly ignored or ‘dropped out’ and the updates to the weights during training are based on a different ‘view’ of the layer. This process adds noise to the training process, forcing nodes to probabilistically take on more or less ‘responsibility’ for the inputs as they are either ignored or relied upon, which greatly helps prevent overfitting [5,6]. The dropout rate of 40% in SeisNet meant that every training iteration, 40% of the ANN output nodes were reset.

The model was trained on high-end consumer-grade hardware: a 12-core 3.8GHz AMD Ryzen 9 3900X CPU, with 64GB of DDR4 3200MHz RAM, and a NVIDIA GTX 1080 GPU. The data were stored on 2TB Samsung 860 EVO SSD. The small size of SeisNet did not warrant parallelization via the GPU, so all training, testing, and applications were done via the CPU.

SeisNet was trained for 50 epochs, and the iteration with the best performance on the validation dataset was saved. Each epoch of training took ~170s, or about 85 tiles/second on the CPU. In this case, the model’s performance peaked after 35 epochs, or ~100 minutes and in total, the model trained for 140 minutes. The final model had values of 0.3432 for loss, 0.8658 for accuracy, 0.4726 for validation loss, and 0.8221 for validation accuracy. The small difference between the training set accuracy and the validation accuracy (only 4.37% worse on the validation set) is a strong indication that the model was not overfit and was well-generalized. A large difference between the two values would suggest that the model had overfit. For example, if the training score were 10% greater than the validation score, additional work would be required to prevent overfitting.

## 2.X Single-tile classifications and confidence thresholds

Testing the model on the validation dataset showed that the fully trained model generates classifications at a rate of ~450 tiles/second in a single instance on the CPU. This performance could likely be improved through parallelization on the GPU, but for our purposes this speed was sufficient.

The training metrics outlined above did not account for the model’s confidence in its classification, and performance was improved by implementing a confidence threshold which enable a better understanding of the model’s capabilities. For example, the model may make an incorrect classification with only 60% confidence in that classification, which would be much less concerning than the model making the same classification with 98% confidence. The lower confidence threshold would likely indicate that the tile was ambiguous, perhaps somewhere between no events and minor events. Having low confidence in that case would actually be good even if the classification is incorrect. Conversely, if the model were highly confident in an incorrect classification, that might indicate a much more serious problem with how the model is parsing, analyzing and classifying the images.

The performance of the model is thus best evaluated when the low-confidence classifications are ignored, since they likely represent ‘coin-flip’ scenarios where the tile is ambiguous and could have been subjectively misclassified in the training dataset. By setting a high minimum confidence threshold, we can assess how well the model actually understands the problem in clear-cut instances. A confidence threshold of 85% percent was thus implemented. When the validation dataset was evaluated, the model exceeded that 85% confidence for two-thirds of the validation tiles, meaning that the model was highly confident in 66% of its classifications. A comparison of standard metrics with and without a confidence threshold is presented in Table X. With the 85% confidence threshold, the model’s overall accuracy increased from 82% to 92%, suggesting that the majority of the model’s incorrect classifications were among low-confidence tiles (Figure X).

|  |  |
| --- | --- |
| 1. (32%, 19%, 42%, 7%) | 1. (1%, 42%, 57%, 0%) |
| 1. (0.5%, 49.7%, 49.3%, 0.5%) | 1. (2%, 46%, 51%, 1%) |

Figure X. Examples of low-confidence tiles with confidences for each class: none, minor, major, error. A) and B) both have relatively low-amplitude waveforms, but the samples contain large amplitude variability, which is characteristic of a more significant event. These tiles were misclassified as containing major events when they contain minor events. C) and D) both appear to have longer period microseisms, but D) contains a large irregularity in the third line. C) was correctly classified as containing minor events, and D) was correctly classified as containing the beginning of a major event.

Table X. Model performance with and without a confidence threshold.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | No minimum confidence | | | | Minimum confidence = 85% | | | |
| Precision | Recall | F1 | Support | Precision | Recall | F1 | Support |
| No events | 0.84 | 0.88 | 0.86 | 1,309 | 0.93 | 0.96 | 0.94 | 1,013 |
| Minor events | 0.85 | 0.82 | 0.83 | 1,666 | 0.92 | 0.93 | 0.92 | 1,030 |
| Major events | 0.72 | 0.76 | 0.74 | 618 | 0.87 | 0.82 | 0.84 | 321 |
| Errors | 1.00 | 0.00 | 0.00 | 38 | 1.00 | 0.00 | 0.00 | 16 |
|  |  | | | |  | | | |
| Accuracy | 0.82 | | | 3,631 | 0.92 | | | 2,380 |
| Macro mean | 0.85 | 0.61 | 0.61 | 0.93 | 0.68 | 0.68 |
| Weighted mean | 0.82 | 0.82 | 0.82 | 0.92 | 0.92 | 0.91 |

A confusion matrix for the validation dataset was generated (Figure X). This diagram illustrates misclassifications across the four classes used in this model. There is some confusion between no events and minor events (~6%), and between minor events and major events (17%), but notably there is almost no overlap between no events and major events (<1%). Given that these classifications are subjectively derived from a continuous spectrum of waveform patterns, the overlap between adjacent classes is unsurprising, but it is promising that there is essentially no overlap between the end-members. The model was unable to classify any error tiles and confused them with the other classes.

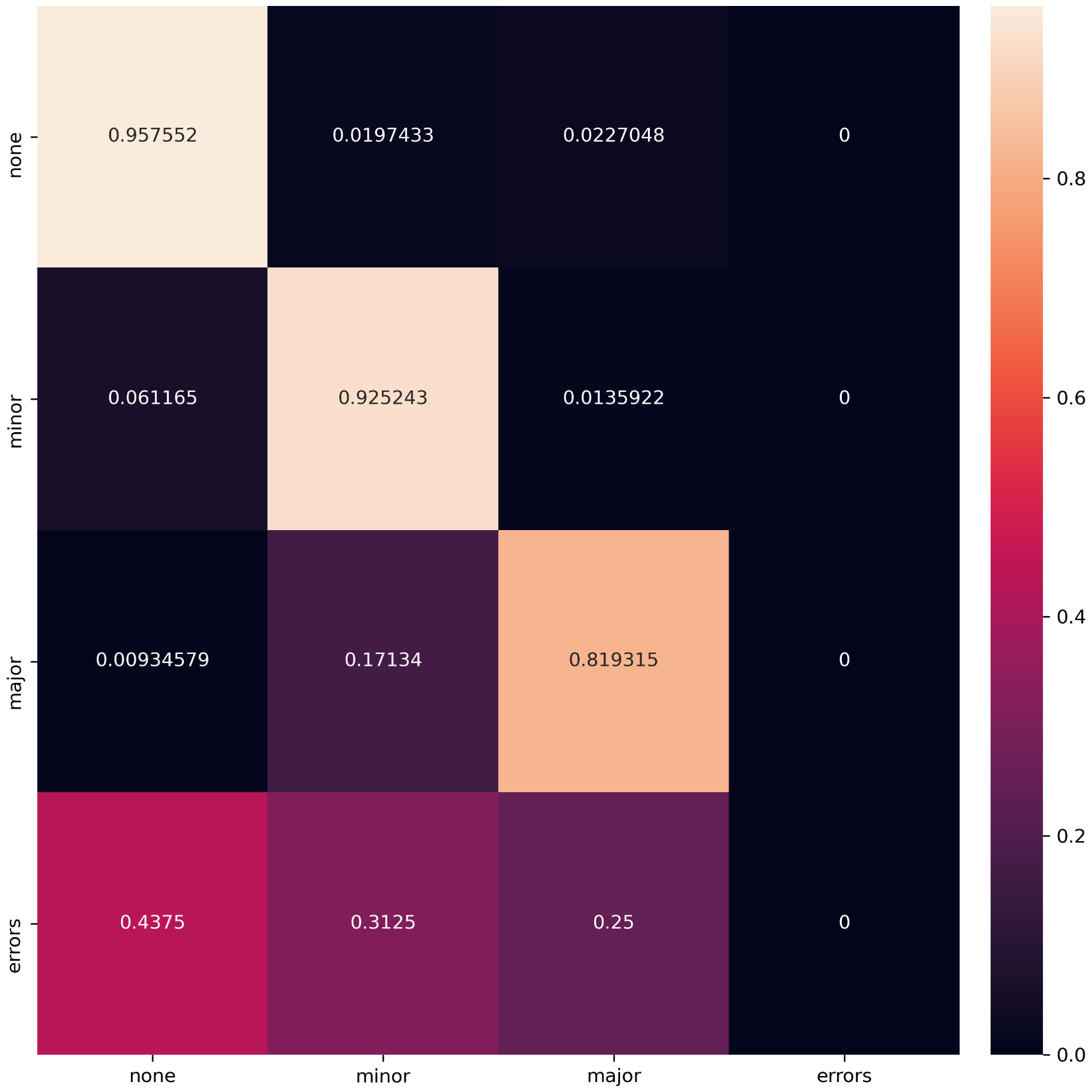


Figure X. The normalized confusion matrix for the validation dataset. The intermediate ‘minor events’ class was sometimes confused for the end-member classes, but the end-members were almost never confused. Note that the model did not classify any tiles as ‘errors’.

## 2.X Determining sample frequency

The data pipeline was designed to randomly sample unique tiles from each film chip until a minimum number of samples had been generated. Sampling a fraction of the chip’s area is a more efficient than simply classifying the entire area, and given the number of film chips in the USGS archive, optimizing this process was crucial. This minimum value was selected by evaluating the variability of each sample size.

To measure this variability, different sample sizes were tested repeatedly across a test set of 50 film chips. The masked area of each film chip is composed of approximately 250 tiles, so sample sizes of 10, 25, 50, 100, and 200 tiles were tested. With a confidence threshold of 85% an average score across five runs for each chip was calculated using values of 0 for ‘no events’, 1 for ‘minor events’, and 2 for ‘major events’. For example, the average scores of the 10 samples tiles in film chip ALQ\_62\_05\_22\_1626\_LHZ were 1.2, 1.3, 1.0, 1, and 1.2 (a range of 0.3). Across all 50 chips samples with 10 samples, the mean score varied by 0.246 on average. Based on an analysis of these metrics (Figure X), a sample size of \_\_ was used.

Figure X. The range of scores for each of the 50 film chips using the different sample sizes. Note the pronounced decrease in variability with increasing sample size.

## 2.X Film chip labelling

Because many sample tiles are taken from each film chip, and the samples are evaluated in ensemble to generate an overall label, an accuracy of 92% was sufficient for our application. The following logic was then applied to the each chip’s database for each film chip to determine an overall label:

1. If the maximum tile score for the chip were 0; label = 'no interest'
2. If the maximum tile score for the chip were 1; label = ‘little interest'
3. If the mode tile score for the chip were 2; label = ‘high interest'
4. If the maximum tile score for the chip were 2 OR the mean score were > 1; label = 'no interest'

## 2.X Data access and availability

All of the film chips used to train the model are publicly available at <http://ds.iris.edu/spud/filmchip>. A list of the film chips used is available at <https://github.com/TimNagle-McNaughton/USGS>. Additionally, the fully trained SeisNet model, the code used to create and train SeisNet, and the pipeline code are available in the same repository.

# 3. Results

# 4. Discussion

# References

1. Shijie, J.; Ping, W.; Peiyi, J.; Siping, H. Research on Data Augmentation for Image Classification Based on Convolution Neural Networks. *Proceedings - 2017 Chinese Automation Congress, CAC 2017* **2017**, *2017-Janua*, 4165–4170, doi:10.1109/CAC.2017.8243510.

2. Perez, L.; Wang, J. The Effectiveness of Data Augmentation in Image Classification Using Deep Learning. **2017**.

3. Hussain, Z.; Gimenez, F.; Yi, D.; Rubin, D. Differential Data Augmentation Techniques for Medical Imaging Classification Tasks. *AMIA ... Annual Symposium proceedings. AMIA Symposium* **2017**, *2017*, 979–984.

4. Kingma, D.P.; Ba, J.L. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* **2015**, 1–15.

5. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. **2012**, 1–18.

6. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The journal of machine learning research* **2014**, *15*, 1929–1958.