**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, J.T. Wilgus1,2

\*Corresponding author

1Department of Earth and Planetary Science, The University of New Mexico

2United States Geological Survey-Albuquerque Seismic Laboratory

# Abstract

Insert abstract here

# 1. Introduction

## 1.X The WWSSN archive

Insert a paragraph outlining the WWSSN

Over the 16 years between 1962 and 1978, the WWSSN generated approximately 3.7 million single-day record film chips [1]. Over two different efforts, film chips were digitized via high-resolution scanning at ~3200dpi and the resulting 8-bit images span ~8,000x3,500 pixels (28 megapixels) [1]. At the time of writing, the USGS has 189,180 scanned film chips, comprising some 6TB of data. Of these, ~27,000 are available online: <ds.iris.edu/spud/filmchip> [1]. The currently digitized chips comprise only 5.1% of the entire WWSSN archive, but at a cost of ~$4 per scan [1], significant funding will be required to digitize the remaining 95% unscanned film chips.

That said, the archive is already too large to manually explore at present. Even with the help of low-cost intern work, systematically going through this archive to identify interesting film chips would be daunting and labor-intensive. As such, we identified a need to develop an automated method for rapidly parsing film chips and identifying chips with interesting events. To that end, in this study we develop and present a novel approach to classifying the WWSSN records using a simple CNN we call “SeisNet”.

Insert paragraph here about the usefulness

## 1.X CNNs and image classification

Introduction to CNNs?

Convolutional neural networks (CNNs) have been widely applied to image classification and computer vision problems [2–4]. Research has found that a simple CNN can have the ability to perform many different complex tasks depending on its training data, including object recognition and classification, attribute detection, and object/image retrieval [5]. CNNs have been highly effective in problems such as handwriting recognition [6,7], cancer detection in medical images [8–12], and classifying features in remotely sensed data [13–15]. As CNNs have grown in popularity, their accessibility has also increased. There are many excellent libraries and resources for developing custom CNNs [16–24], and more complex off-the-shelf models have become extremely prevalent for a range of applications as transfer learning has improved [25–28].

# 2. Methods and data

## 2.X Data pipeline

We developed a simple nine-step data processing methodology for generating labels for WWSSN film chips (Figure 1). Once a WWSSN film chip has been loaded into memory, the edges of the chip are masked out to eliminate any irrelevant scan errors or annotations that are not seismic data (Figure 2). The image is converted from 8-bit greyscale to 1-bit binary. A random 200x200 ‘tile’ is cropped out from the scan, and its brightness level is checked. Some of the film chips have dark smudges which obscure the seismic data (Figure 3), and the brightness check is able to eliminate tiles which contain those smudges. The pipeline will only accept tiles that are between 50% and 95% white. If the tile fails this check, the tile is. If the tile does pass the brightness check, it is classified by SeisNet into one of three classes: ‘no events’, ‘minor events’, and ‘major events’. This crop-check-classify process is repeated until a desired number of sample tiles have been generated. Once the minimum number of samples has been reached, the tiles and their classifications are evaluated in ensemble to generate one of four labels for the entire film chip: ‘no interest’, ‘little interest’, ‘some interest’, and ‘high interest’.

Diagram

Description automatically generated

Figure 1. 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 (See Figure 2 below). 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.

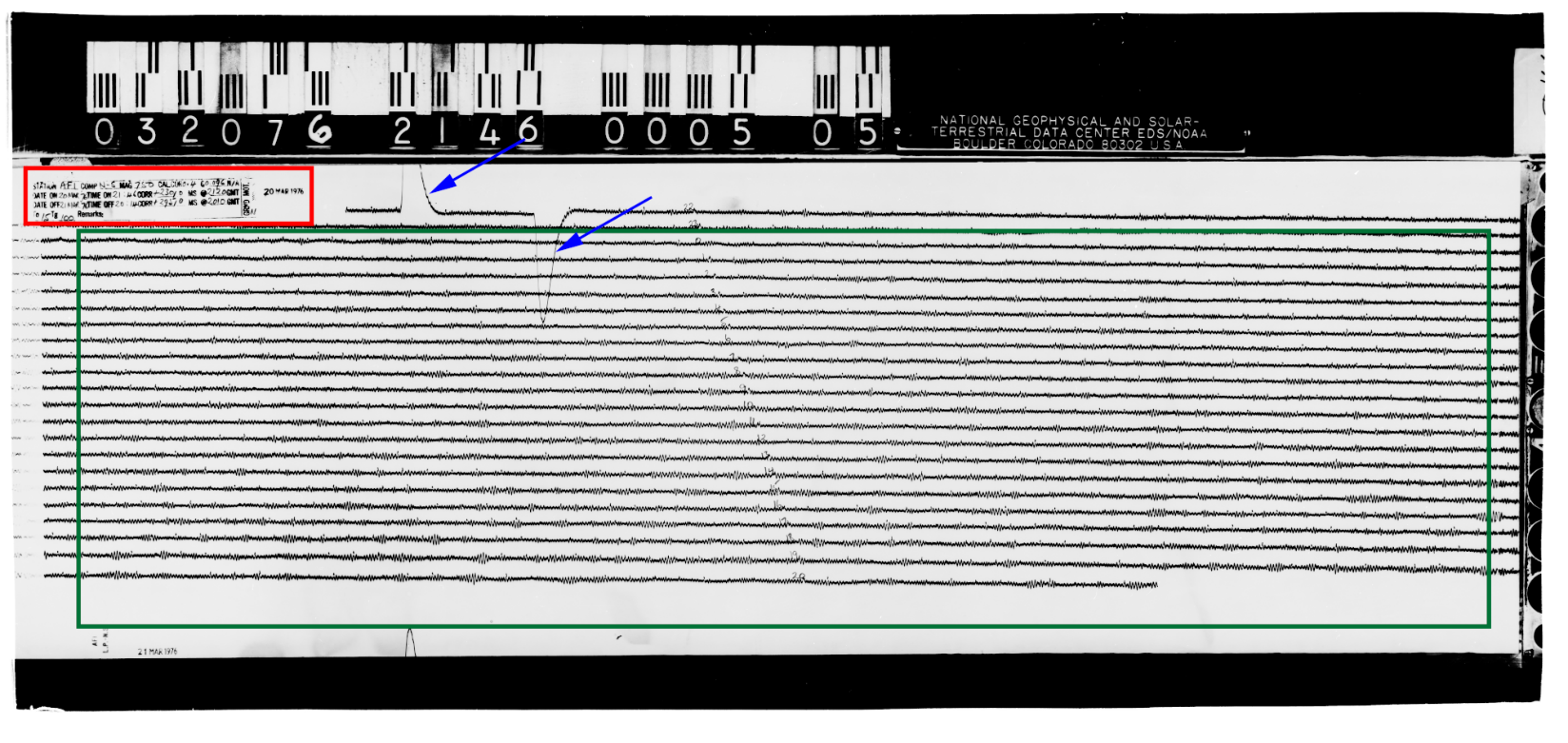
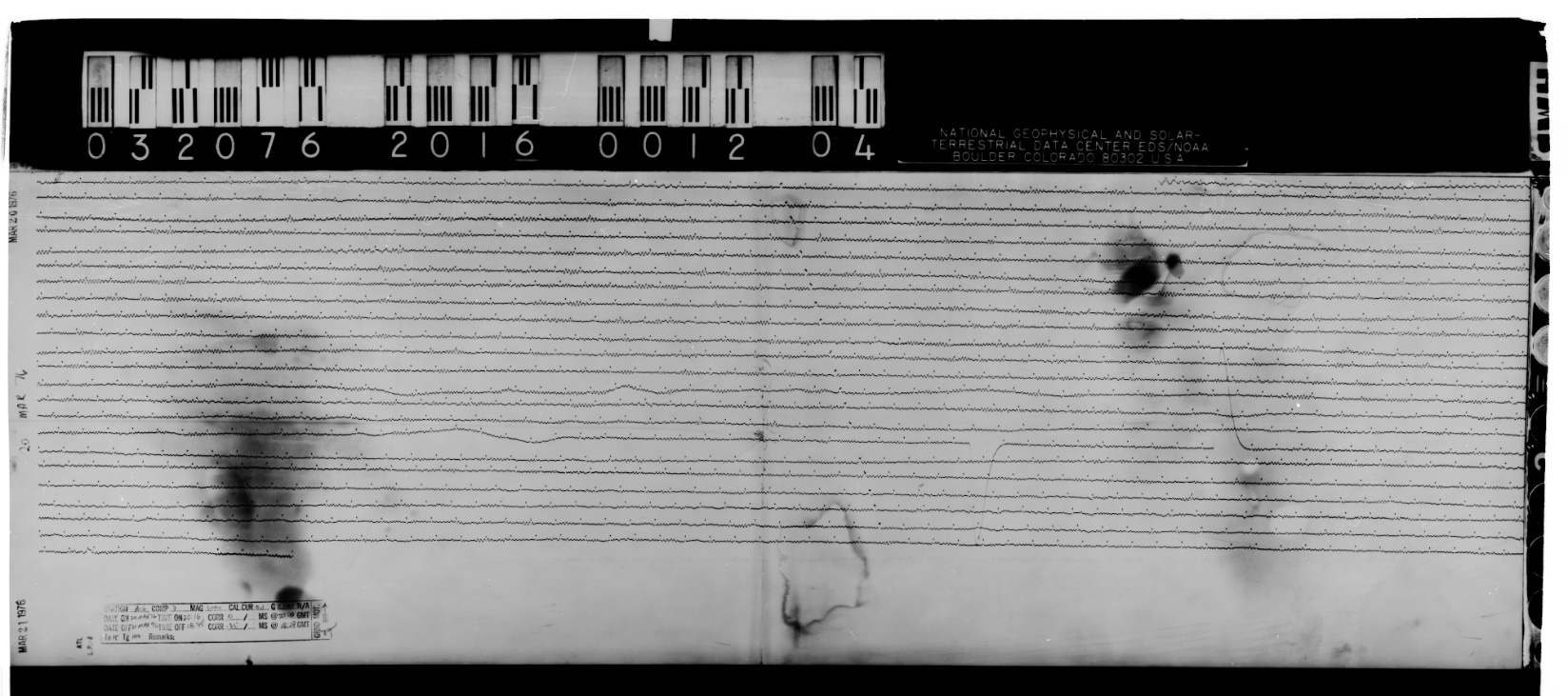


Figure 2. An annotated film chip. The red rectangle marks metadata stamped on the film chip, the blue arrows mark the calibration pulse, and the green rectangle annotates the masked area containing useful information. Note that the masked area excludes the black film chip header and footer, the metadata, and artifacts on the right of the image. This prevents unusual or erroneous data from being included in sampling. In the center of the record the station operator manually labeled each hour.

Figure 3. An example of a film chip with dark smudging inside the masked area. Checking the overall brightness of each tile allows these areas to be skipped during sampling.

## 2.X Model architecture

SeisNet is a relatively simple sequential CNN developed using the Python package Keras [17]. SeisNet is composed of three convolutional blocks, one dense block, a dropout layer, and an output layer (Table 1). 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 1. 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. A list of the source film chips used is available at: <https://github.com/TimNagle-McNaughton/USGS/scan_classification>. 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 2.

Table 2. Training dataset class counts. Note the low count of error tiles.

|  |  |
| --- | --- |
| **Class name** | **Occurrences in training data** |
| No events |  |
| Minor events |  |
| Major events |  |
| Errors |  |

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 [29–31].

## 2.X Model training

During training, the model used the Adam optimizer and a learning rate of 1x10-5 [32]. 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 [33,34]. 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 [33,34]. 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.

## 3.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 an evaluation 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’. The scores were then compared to determine the optimal sampling frequency.

## 3.X Label generation

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% for each chip was sufficient for our application. The following logic was then applied to each chip’s database for each film chip to determine an overall label:

* REWRITE

These labels are intended to assign some gradation of interest to the film chips. Here again the differentiations between adjacent classes (i.e. between no and little interest, or between high and some interest) are likely to be fuzzy and imperfect. However, these labels are not the end-all-be-all of this project. Even with some intrinsic fuzziness, these labels should enable better filtering of the chips, more efficient exploration of the archive, and ultimately the production of further research. To measure the model’s efficacy in generating these labels, manual and automated classifications were compared. The 50 evaluation tiles were independently reviewed and classified by each author. The average classification was then compared to those generated by the model.

## 2.X Applying the model

## 2.X Data access and availability

All of the film chips used to train the model are from the publicly available database at <http://ds.iris.edu/spud/filmchip>. A list of the film chips used is available at <https://github.com/TimNagle-McNaughton/USGS/scan_classification>. 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

## 3.X Model training

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. 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.

## 3.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 3. 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 4).

|  |  |
| --- | --- |
| 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 4. Examples of low-confidence tiles with confidences for each class: (none, minor, major, error). Both A) and B) 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 3. 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 5). 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. Given their extreme rarity in the training and test datasets, these misclassifications of error tiles are acceptable. The network’s inability to classify these features likely is a combination of class imbalance in the training data, and the fact that tiles with errors still contain other information that the network can identify (Figure 6).

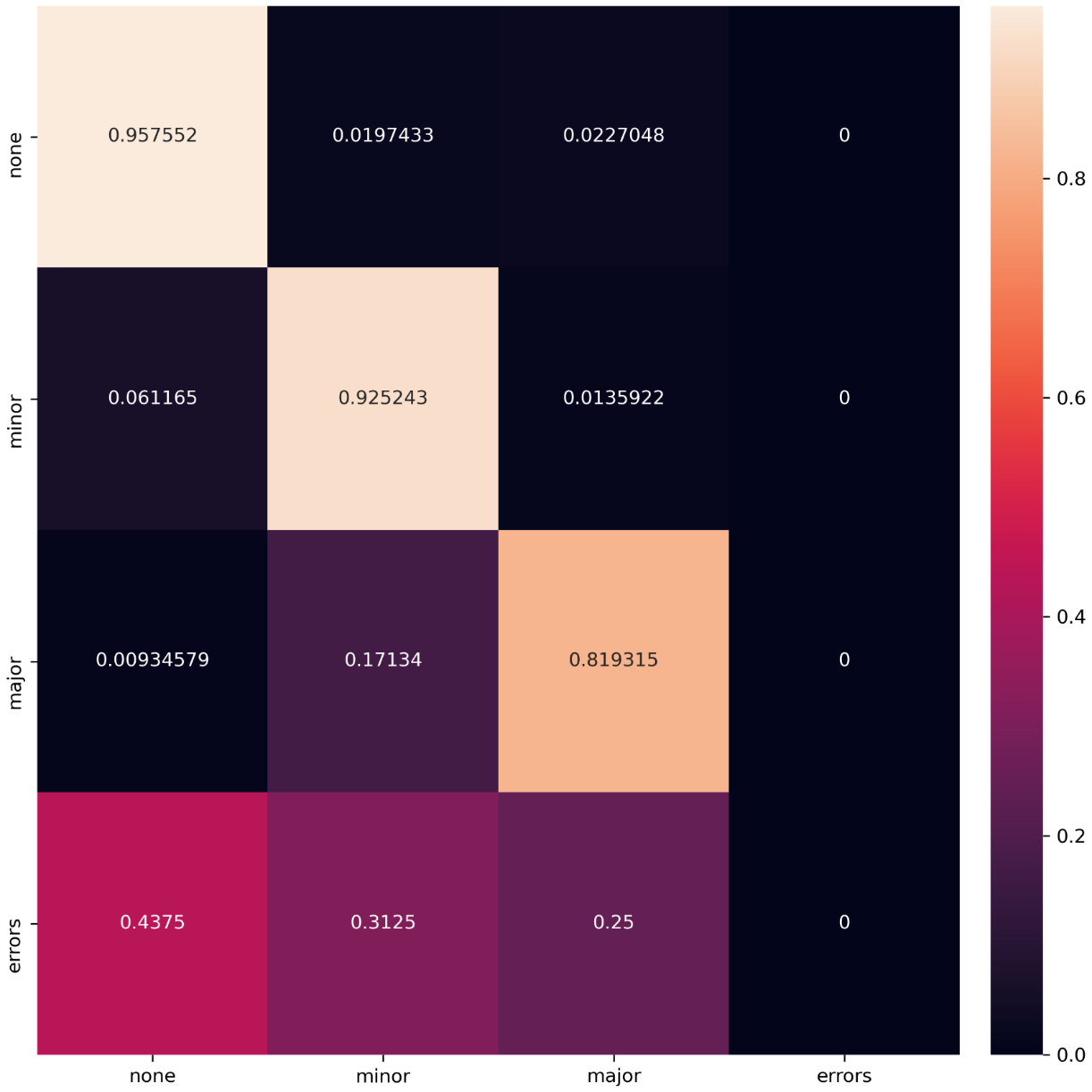


Figure 5. The normalized confusion matrix for the validation dataset. The intermediate ‘minor events’ class was sometimes confused for the end-member classes, but the end-members were almost never confused. The model did not classify any tiles as ‘errors’.

|  |  |
| --- | --- |
| A. (41%, 21%, 29%, 8%) | B. (90%, 4%, 1%, 5%) |
| C. (5%, 79%, 13%, 3%) | D. (18%, 24%, 45%, 13%) |

Figure 6. Tiles containing errors with confidences for each class: (none, minor, major, error). Both A) and B) were correctly classified as ‘No events’ despite the calibration pulse cutting across the tile. C) contains noise some light smudging which got passed the brightness check and was misclassified as ‘minor events’ when no microseisms are present. In D) the calibration pulse seems to have confused the network and it was misclassified as having ‘major events’. Note that only B) surpasses the 85% confidence threshold.

## 3.X Sample frequency

Figure 7 illustrates the variation in score ranges across the 50 evaluation images. For example, testing 10 samples tiles per film chip on ALQ\_62\_05\_22\_1626\_LHZ five times produced scores of [1.2, 1.3, 1.0, 1, 1.2], a range of 0.3. This process was repeated for the other 49 evaluation chips to generate the 10-sample plot. The optimal sampling frequency was determined to be 100 samples per chip. This frequency balances minimizing the variability in score across tests and processing time. Using 200 samples more than doubled the processing time and only slightly decreased the average spread of the scores.

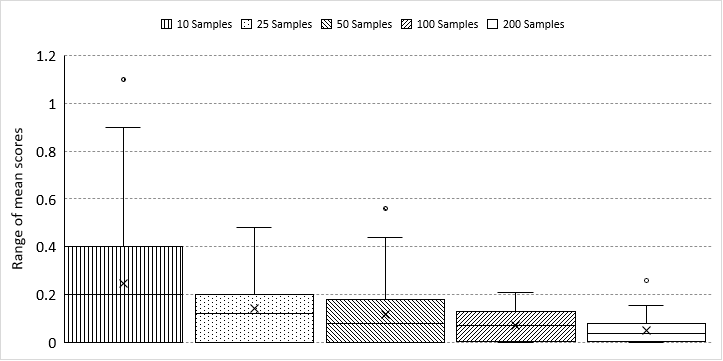


Figure 7. Box-and-whisker plots of the variability in mean scores across five test runs for each of the 50 evaluation film chips using five different sampling frequencies. Note the pronounced decrease in variability with increasing sample size.

## 3.X Label generation

There are potential problems introduced by inconsistent gain settings applied at different stations at different times in the data. Add more detail here.

|  |  |
| --- | --- |
| *A)* | *B)* |

Figure 8. A comparison of gain settings in the archive. Both crops are 600x600 pixels. A) Has normal gain with minute markers clearly visible and flat straight lines (GEO\_62\_05\_31\_1006\_LHZ). This film chip was classified as ‘little interest’. B) Has a hand-written annotation noting a new gain setting that proved to be too high (ARE\_62\_05\_22\_1340\_SHN). This film chip was classified as ‘little interest’. High gain settings make interpretation and classification of the film chips more difficult. The increased amplitude can both cause small events to appear much larger, and also hide detail in the traces, potentially confusing the network since the data are not perfectly comparable across stations.

## 3.X SeisNet results from the WWSSN archive

# 4. Discussion

## 4.X Better balancing training data

# 5. Acknowledgements

# 6. Funding

# References

1. Alejandro, A.C.B.; Hutt, C.R.; Ringler, A.T.; Moore, S. v.; Anthony, R.E.; Wilson, D.C. The Albuquerque Seismological Lab WWSSN Film Chip Preservation Project. *Seismological Research Letters* **2019**, *90*, 401–408, doi:10.1785/0220180275.

2. Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation* **2017**, *29*, doi:10.1162/NECO.

3. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience* **2018**, *2018*, doi:10.1155/2018/7068349.

4. Khan, S.; Rahmani, H.; Shah, S.A.A.; Bennamoun, M. A Guide to Convolutional Neural Networks for Computer Vision. *Synthesis Lectures on Computer Vision* **2018**, *8*, 1–207, doi:10.2200/s00822ed1v01y201712cov015.

5. Razavian, A.; Azizpour, H.; Sullivan, J.; Carlsson, S. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition; Columbus, OH, 2014; pp. 806–813.

6. Ghosh, M.M.A.; Maghari, A.Y. A Comparative Study on Handwriting Digit Recognition Using Neural Networks. *Proceedings - 2017 International Conference on Promising Electronic Technologies, ICPET 2017* **2017**, 77–81, doi:10.1109/ICPET.2017.20.

7. An, S.; Lee, M.; Park, S.; Yang, H.; So, J. An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition. **2020**.

8. Alakwaa, W.; Nassef, M.; Badr, A. Lung Cancer Detection and Classification with 3D Convolutional Neural Network (3D-CNN). *International Journal of Biology and Biomedical Engineering* **2017**, *11*, 66–73, doi:10.14569/ijacsa.2017.080853.

9. Wang, Z.; Li, M.; Wang, H.; Jiang, H.; Yao, Y.; Zhang, H.; Xin, J. Breast Cancer Detection Using Extreme Learning Machine Based on Feature Fusion with CNN Deep Features. *IEEE Access* **2019**, *7*, 105146–105158, doi:10.1109/ACCESS.2019.2892795.

10. Gao, F.; Wu, T.; Li, J.; Zheng, B.; Ruan, L.; Shang, D.; Patel, B. SD-CNN: A Shallow-Deep CNN for Improved Breast Cancer Diagnosis. *Computerized Medical Imaging and Graphics* **2018**, *70*, 53–62, doi:10.1016/j.compmedimag.2018.09.004.

11. Fu, Y.; Lei, Y.; Wang, T.; Curran, W.J.; Liu, T.; Yang, X. Deep Learning in Medical Image Registration: A Review. *Physics in Medicine and Biology* **2020**, *65*, doi:10.1088/1361-6560/ab843e.

12. Muhammad, S.; Muhammad, A.; Adnan, M.; Muhammad, Q.; Majdi, A.; Khan, M.K. Medical Image Analysis Using Convolutional Neural Networks A Review. *Journal of Medical Systems* **2018**, 1–13.

13. Kattenborn, T.; Leitloff, J.; Schiefer, F.; Hinz, S. Review on Convolutional Neural Networks (CNN) in Vegetation Remote Sensing. *ISPRS Journal of Photogrammetry and Remote Sensing* **2021**, *173*, 24–49, doi:10.1016/j.isprsjprs.2020.12.010.

14. Zhu, X.X.; Tuia, D.; Mou, L.; Xia, G.S.; Zhang, L.; Xu, F.; Fraundorfer, F. Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. *IEEE Geoscience and Remote Sensing Magazine* **2017**, *5*, 8–36, doi:10.1109/MGRS.2017.2762307.

15. Ma, L.; Liu, Y.; Zhang, X.; Ye, Y.; Yin, G.; Johnson, B.A. Deep Learning in Remote Sensing Applications: A Meta-Analysis and Review. *ISPRS Journal of Photogrammetry and Remote Sensing* **2019**, *152*, 166–177, doi:10.1016/j.isprsjprs.2019.04.015.

16. Sewak, M.; Karim, M.R.; Pujari, P. *Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python*; Packt Publishing Ltd, 2018; ISBN 1788394143.

17. Chollet, F. Keras: The Python Deep Learning Library. *Astrophysics Source Code Library* **2018**, ascl-1806.

18. Vasilev, I.; Slater, D.; Spacagna, G.; Roelants, P.; Zocca, V. *Python Deep Learning: Exploring Deep Learning Techniques and Neural Network Architectures with Pytorch, Keras, and TensorFlow*; Packt Publishing Ltd, 2019; ISBN 1789349702.

19. Gulli, A.; Pal, S. *Deep Learning with Keras*; Packt Publishing Ltd, 2017; ISBN 1787129039.

20. Nandy, A.; Biswas, M. *Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python*; Apress, 2017; ISBN 1484232852.

21. Raschka, S.; Mirjalili, V. Python Machine Learning: Machine Learning and Deep Learning with Python. *Scikit-Learn, and TensorFlow. Second edition ed* **2017**.

22. Pang, B.; Nijkamp, E.; Wu, Y.N. Deep Learning with Tensorflow: A Review. *Journal of Educational and Behavioral Statistics* **2020**, *45*, 227–248.

23. Srinivasa-Desikan, B. *Natural Language Processing and Computational Linguistics: A Practical Guide to Text Analysis with Python, Gensim, SpaCy, and Keras*; Packt Publishing Ltd, 2018; ISBN 1788837037.

24. Hao, J.; Ho, T.K. Machine Learning Made Easy: A Review of Scikit-Learn Package in Python Programming Language. *Journal of Educational and Behavioral Statistics* **2019**, *44*, 348–361, doi:10.3102/1076998619832248.

25. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. **2018**.

26. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; van Esesn, B.C.; Awwal, A.A.S.; Asari, V.K. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. **2018**.

27. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5MB Model Size. **2016**, 1–13.

28. Li, Y.; Ren, F. Light-Weight RetinaNet for Object Detection. **2019**.

29. 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.

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

31. 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.

32. 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.

33. 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.

34. 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.