Scale-invariant Image Recognition using Convolutional Neural Networks and Wavelet Analysis

Heather K. Hardeman[1], Matt McDonald[2], Michael P. Lamoureux[1]

1. CREWES, University of Calgary, Department of Mathematics and Statistics, 2. Fotech Solutions

Summary

We introduce a new wavelet transform called the inverted tree-structured wavelet transform which renders scale-invariance for image recognition. We introduce a training set extracted from real data. We test the trained convolutional neural network on a portion of the training set to determine accuracy. Afterwards, we employ this trained convolutional neural network to identify events in microseismic data.

Introduction

Seismic imaging and interpretation depends on finding events in seismic data. Geophysical scientists employ computers to process data in order to find events. Teaching computers to locate events in seismic data through image recognition seems like a natural next step. One method for teaching image recognition to computers is machine learning.

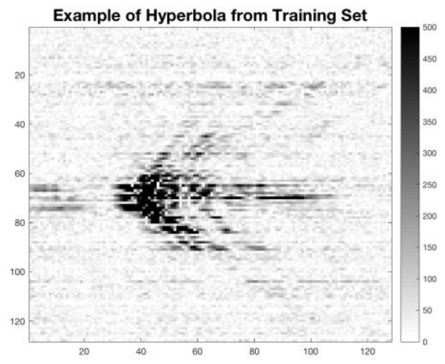
Inverted Tree-Structured Wavelet Transform

An inverted tree-structured wavelet transform has the same basic principle as the tree-structured wavelet transform [1]. The top level contains the original data and successive levels contain the results of applying a specific wavelet from the wavelet family to the data in the node. The main difference is that our top level has 2^{M-1} nodes, where M is the max number of levels in the tree. Each of the nodes in the top level holds a $N\times D$ strip of data where N is some power of two that divides the x-domain of the data at least twice and D is the size of the data in the t-domain. The next level of the data is produced by pairing the first node with the second and then pairing each subsequent node to its successor if it is not already paired with the node that directly precedes it and applying the wavelet transform. The new node is the result of the wavelet transform applied to the concatenation of the data in the two nodes in the previous level. The next level of the tree contains $2^{M\text{-}m}$ nodes where m is the level of the tree the node resides.

Architecture, Training, and Testing of the Convolutional Network

We built a convolutional neural network using the UFLDL Tutorial from Stanford University [2] and [3]. Due to time constraints, we used code for the cost function and stochastic gradient descent based on [5]. The CNN has a convolutional layer, a pooling layer, and a densely-connected output layer which feeds into a softmax regression with cost entropy. We applied the stochastic gradient method to a cost function which was the average of the least squares difference between the hypotheses of each sample and the label for that sample with a regularization term.

We built a training set from a DAS-acquired data set of someone walking parallel to a fibre-optic cable. Each step is represented by a hyperbola. We extracted these hyperbolas from the data set creating 528 training images of size 128x128. The set contains 371 hyperbolas and 157 non-hyperbolas, all manually labeled. Fig. 1 shows examples of hyperbolas and non-hyperbolas from the training set.



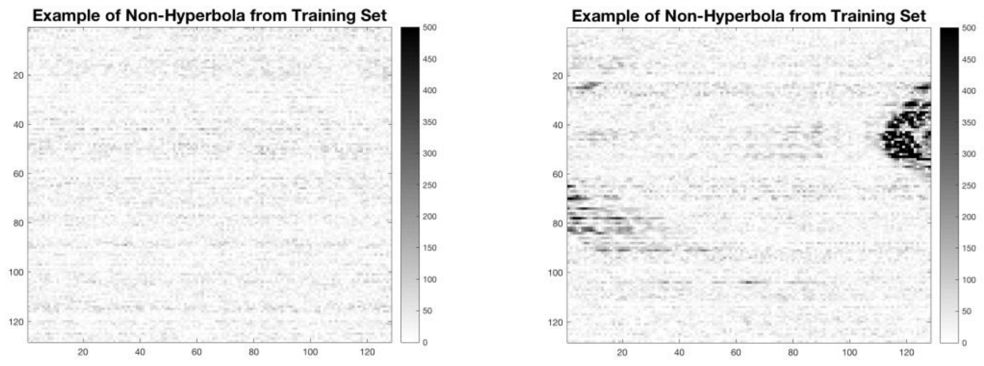


Figure 1: (Top) An image of a hyperbola from the training set. (Bottom) Examples of non-hyperbolas from the training set.

We trained the CNN on 400 training images and tested the CNN on the remaining 128 training images. The CNN predicted the events with a 85.16% accuracy.

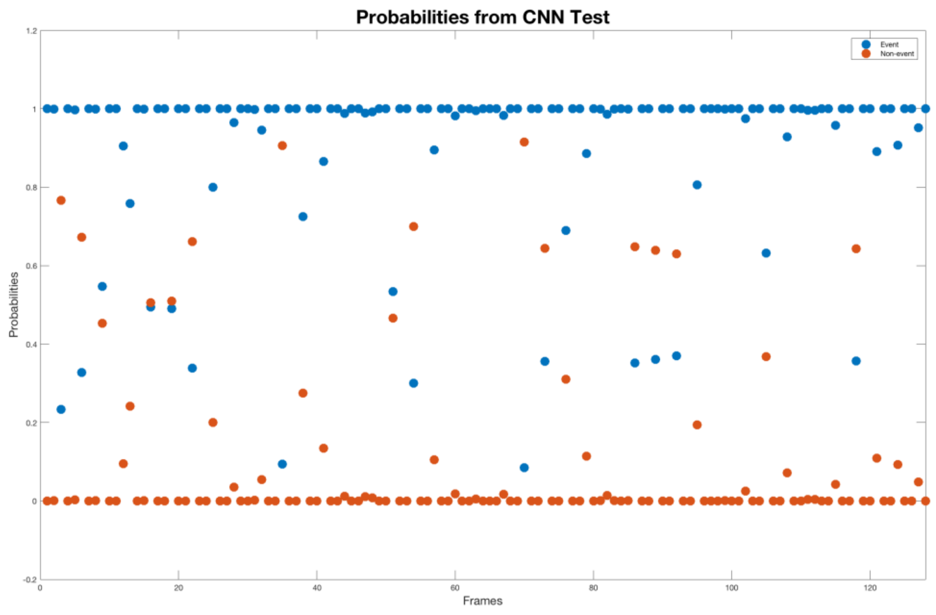


Figure 2: The probability that each image in the remaining 128 images of the training set is a hyperbola (described by the blue dot) or a non-hyperbola (described by the red dot).

Testing on Outside Sample

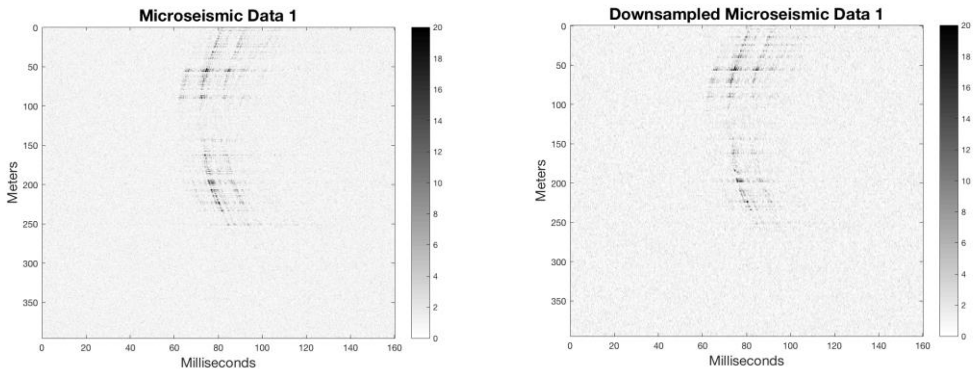


Figure 3: (Left) The microseismic data acquired by by Fotech Solutions using distributed acoustic sensing and fibre-optic cables. (Right) Downsampled version of the microseismic data.

We used microseismic data as the outside sample. We downsampled the data by 75% as it would be computationally expensive to create an inverted wavelet tree big enough to scale the events to the size of the training images.

We built a 3-level inverted wavelet tree using the approximation coefficients from the Haar transform for each node to address scaling and to window the CNN throughout the dataset. The CNN predicted events in Levels 1 and 2; however, it did not detect any events in Level 3 of the inverted wavelet tree.

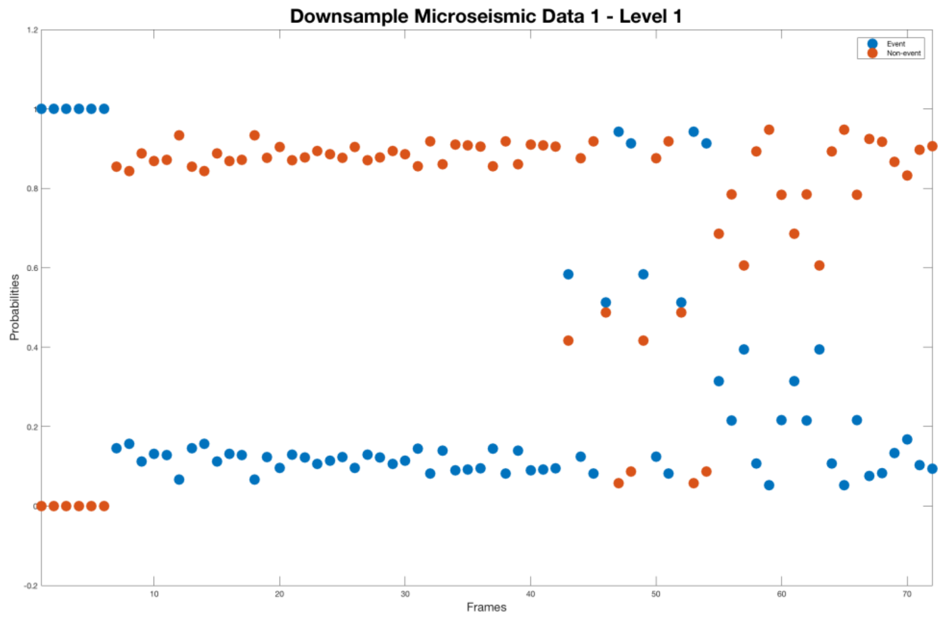


Figure 4: The probabilities in each frame of a hyperbola (blue) being present and a non-hyperbola (red) being present in level 1.

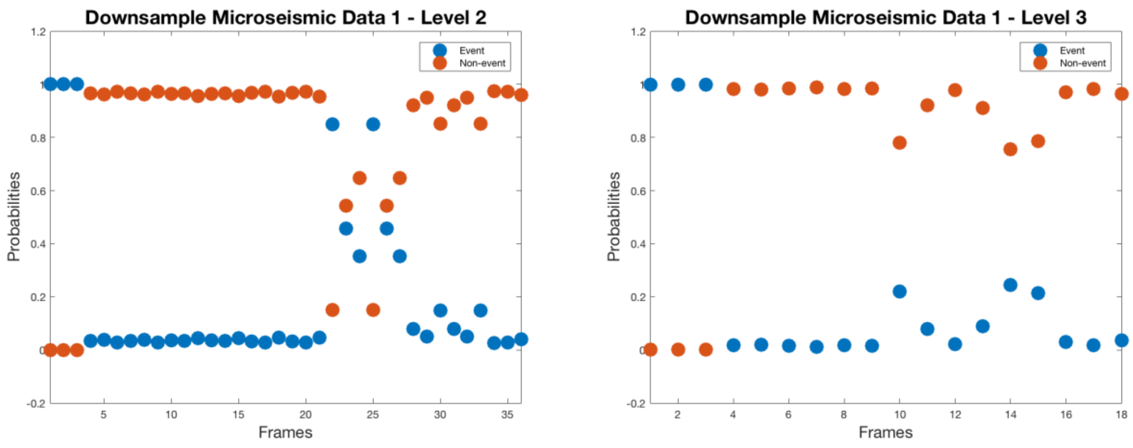


Figure 5: The probabilities in each frame of a hyperbola (blue) being present and a non-hyperbola (red) being present in level 2 (left) and level 3 (right).

We introduced the inverted tree-structured wavelet transform. We also discussed the application of convolutional neural networks to inverted wavelet trees. We considered the architecture of the convolutional neural network we used for experiments on microseismic data. We introduced the training set for the neural network and the parameters we chose to train the CNN. We saw that the CNN had approximately 85% accuracy for detecting hyperbolas in the testing set. Then, we applied the CNN and the inverted wavelet to microseismic data. We were able to detect events in two levels of the inverted wavelet tree.

Acknowledgements

We thank the sponsors of CREWES for their support. We also thank our colleagues at Fotech as well as CREWES for their support. We also gratefully acknowledge support from NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 461179-13 and through a Discovery Grant for the third author. Finally, we gratefully acknowledge the support from NSERC (National Science and Engineering Research Council of Canada) through the Engage Grant - EGP 494101 - 16.

References

1. Chang, T., and Kuo, C.-C. J., 1993, Texture analysis and classification with tree-structured wavelet transforms:IEEE Transactions on Image Processing, 2 , 429–441.

2. Ng, A., Ngiam, J., Foo, C. Y., Mai, Y., Suen, C., Coates, A., Maas, A., Hannun, A., Huval, B., Wang, T., and Tandon, S., 2013, Ufldl: Welcome to the deep learning tutorial. URL http://ufldl.stanford.edu/tutorial/

3. Schmidt, M., 2005, minfunc. URL https://www.cs.ubc.ca/~schmidtm/Software/minFunc.html

4. Yang, B., 2014, Bin Yang. URL https://github.com/byangderek