Accelerated Change Detection in Synthetic Aperture Radar Images based on Deep Neural Networks

Lizzie Koshelev*, Malcolm Milton*, Frank Liao*, Yuanwei Jin**, Enyue Lu†

*Department of Mathematics & Computer Science, Salisbury University, Salisbury, MD; **Department of Engineering and Aviation Sciences, University of Maryland Eastern Shore, Princess Anne, MD; †Department of Mathematics & Computer Science, Salisbury University, Salisbury, MD

Introduction

Image change detection involves identifying the changes that have occurred between two images of a specific area over different time periods. It is an important problem for both civil and military applications. Synthetic Aperture Radar (SAR) images are especially difficult to analyze, since these satellite images produce an abundance of speckle noise. Current methods involve generating a Difference Image (DI) and analyzing the DI. This project attempts to apply the concept of neural networks to detect changes between two images, avoiding the process of analyzing a DI and/or proactively reducing noise. The process took form in 3 steps: preclassifying before and after SAR images to obtain good samples to train the network with, creating and training networks, and analyzing results of the network. Parts of the process are also accelerated through principles of parallelization.

Preclassification

1. Calculate a similarity matrix of the original two images based on:

$$S_{ij} = \frac{|I_{ij}^1 - I_{ij}^2|}{I_{ij}^1 + I_{ij}^2}$$

I_{ij}^n represents the gray level of the n^th image at the position (i, j)

2. Calculate a variance matrix for each of the original two images based on:

$$\delta_{ij}^2 = I_{ij}^1 \frac{I_{ij}^1 I_{ij}^2}{I_{ij}^1 + I_{ij}^2} [S_{ij}]^2$$

1. Iterate over all I_{ij}, if $S_{ij} > T$, where T represents an iterative threshold, then jointly label I_{ij}^1 and I_{ij}^2 by FCM based on the principle of minimum variance. Otherwise label I_{ij}^1 and I_{ij}^2 separately.
2. Pick good samples to feed to the neural network. We pick the "good" pixels based on a comparison between its label and the labels of the pixels in the neighborhood around it.

$$\frac{Q(p_{\xi n} \epsilon N_{ij} \land \Omega_{\xi n} = \Omega_{ij})}{n \times n} > a$$

den. represents similar pixels
num. represents neighborhood size
α represents a chosen threshold

FCM Clustering

FCM is a popular image segmentation technique that segments an image by discovering cluster centers.

Main objective of fuzzy c-means algorithm is to minimize:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{c} (\mu_{ij})^m \, {d_{ij}}^2$$

x_i is the i^th data element
v_j is the j^th center
n is the number of data points.
m is the fuzziness index, $m \in [1, \infty]$.
c is the number of cluster center.
μ_{ij} is the strength of x_i belonging to v_j
d_{ij} is the Euclidean distance between x_i and v_j

1) Randomly select c cluster centers.

2) Calculate the fuzzy memberships μ_{ij} using:

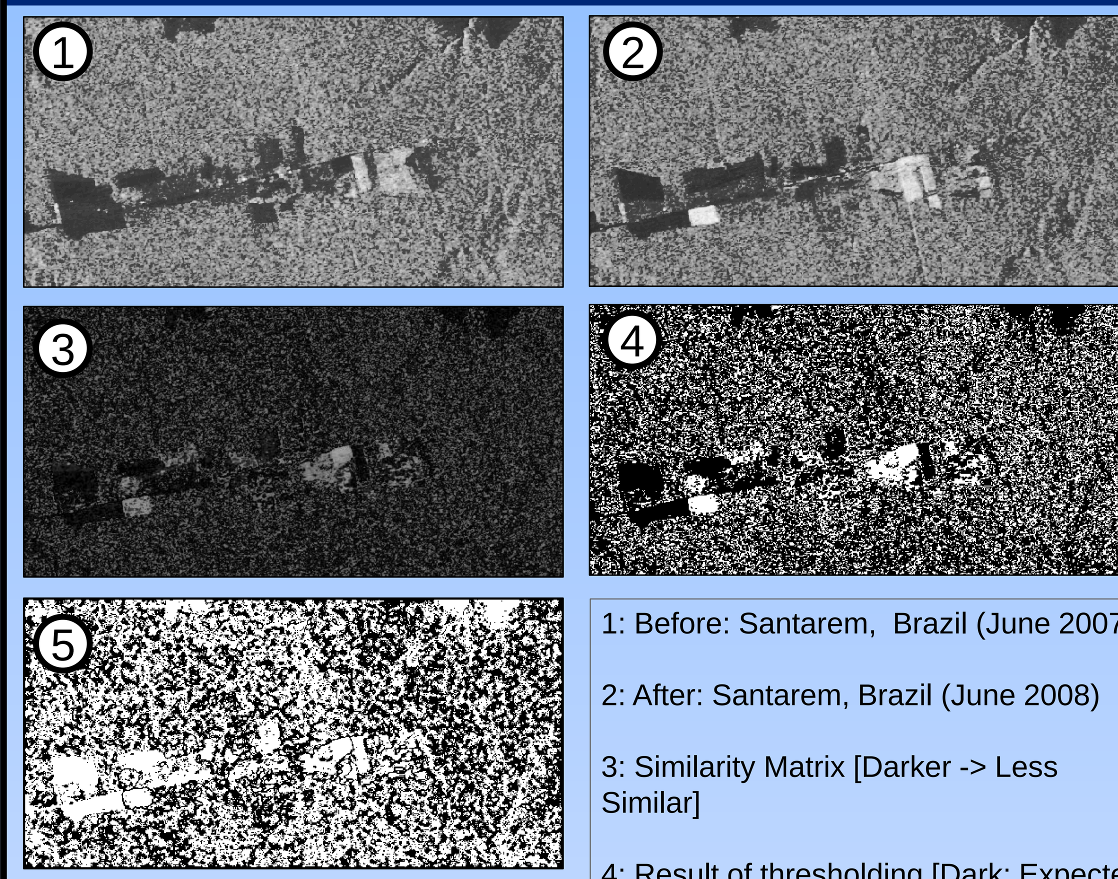
$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{m-1}}}$$

3) Compute the fuzzy centers v_j using:

$$v_j = \frac{\sum_{i=1}^{n} \mu_{ij} x_i}{\sum_{i=1}^{n} \mu_{ij}}$$

4) Repeat steps 2) and 3) until the minimum J is achieved or until the update change of membership values is deemed negligible.

Preclassification Results



1: Before: Santarem, Brazil (June 2007)

2: After: Santarem, Brazil (June 2008)

3: Similarity Matrix [Darker -> Less Similar]

4: Result of thresholding [Dark; Expected Unchanged || Light; Expected Change]

5: Selected Good Samples [Represented by light pixels]

Deep Neural Networks

A DNN is a mathematical model to represent feature recognition. The neural network consists of a network of nodes in layers, where certain nodes are connected. These connections have different weights and these nodes have biases. An activation of a node can, in turn, activate a connected node based on the following function:

$$\sigma\left(\sum W_i v_i + c_i\right)$$

σ represents the logistic function, $\frac{1}{1+e^{-x}}$
W_i represents the weight of the connection
v_i represents the state of the input node
c_i represents the bias of the connection

The weights of the connections are initially set randomly. The input layer of nodes are set as the features of the good sample neighborhoods. After updating the states of all nodes in the network, the neural network reconstructs a set of input states based on the states of the output node. The weights are then updated based on the following function:

$$\varepsilon(\langle v_i h_j \rangle_{initial} - \langle v_i h_j \rangle_{reconstructed})$$

v_i represents input state
h_j represents output state
ε represents a chosen learning rate

We trained a restricted Boltzmann machine network (RBM), which consists of a type of layer-by-layer training that restricts nodes from communicating in their own layer.

Neural Network Results

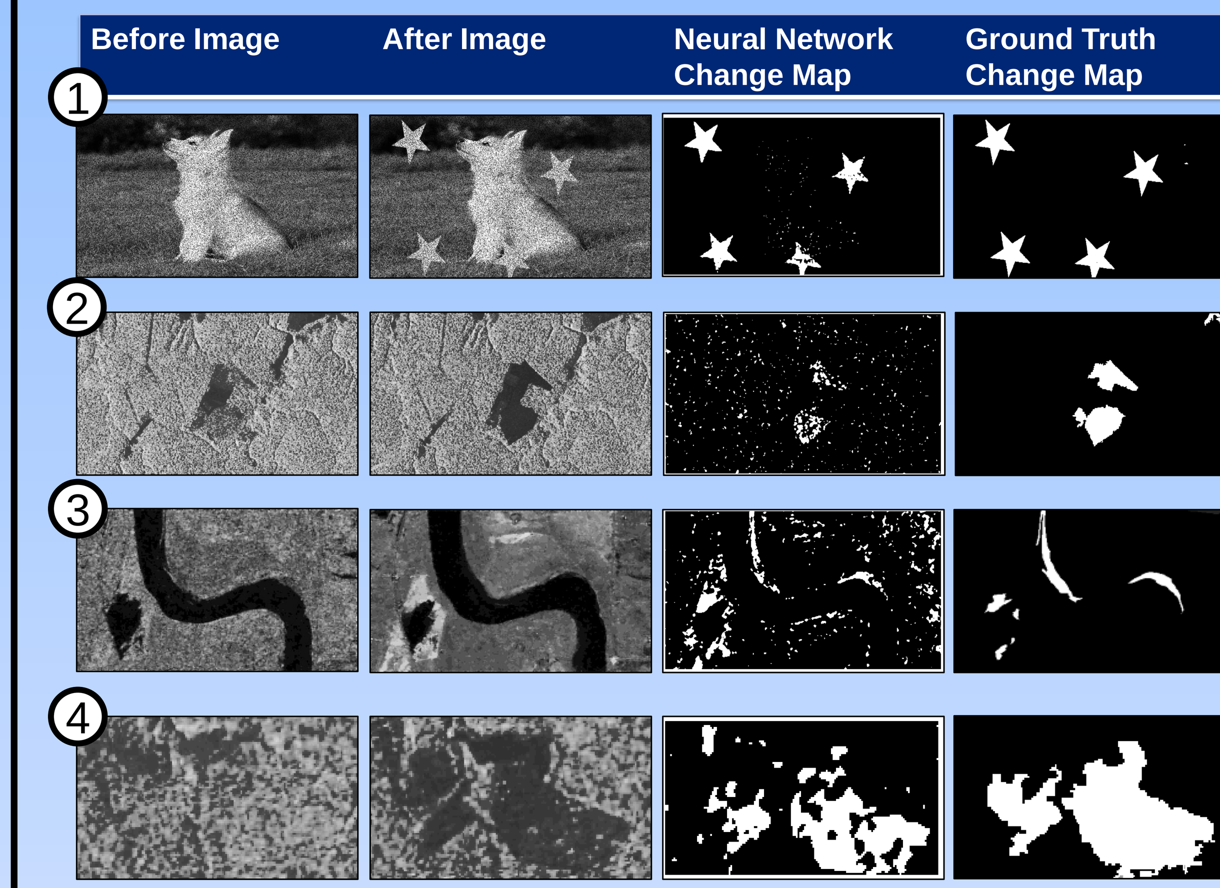
	PCC	TP (%)	TN (%)	FP (%)	FN (%)
Puppy (Artificial)	98.21	3.80	94.41	1.78	0.01
Santarem	96.00	1.15	94.86	3.39	0.61
River	97.11	1.10	96.00	2.25	0.64
Santarem 2	87.68	27.79	59.90	6.16	6.16

$$PCC = \frac{TP + TN}{TP + TN + FP + FN}$$

TP – correctly classified as changed
TN – correctly classified as unchanged
FP – incorrectly classified as changed
FN – incorrectly classified as unchanged

Neural Network Results

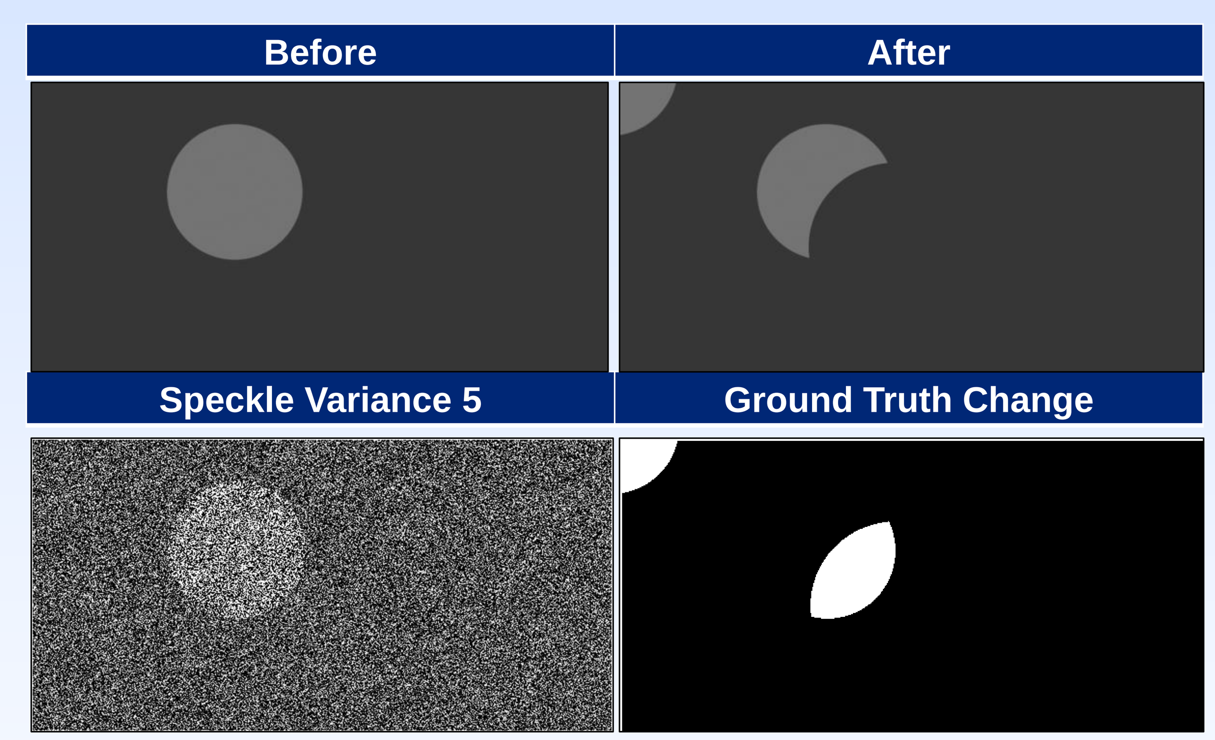
Performance of Neural Network utilizing SAR Data Sets and Artificial Images:

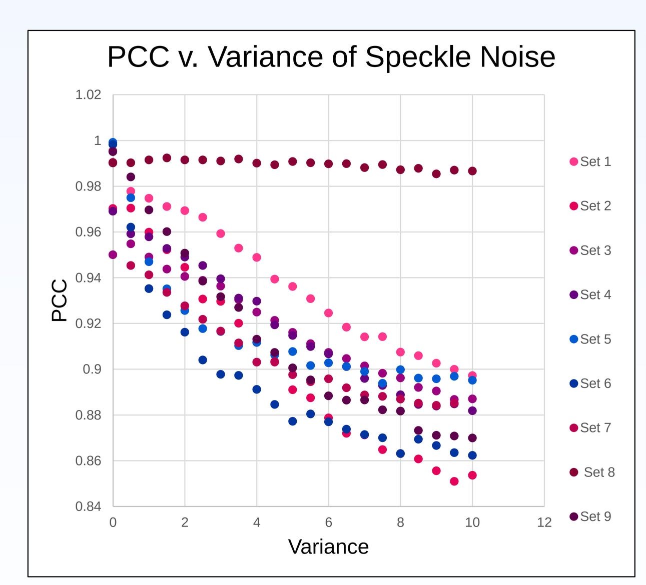


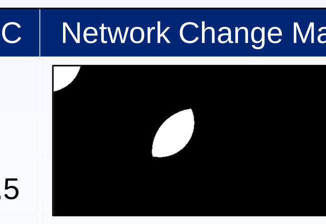
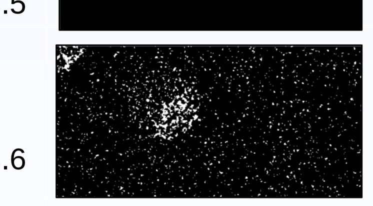
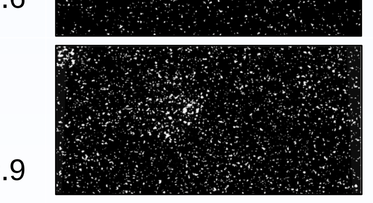
1: Puppy Image w/ Artificial Noise and Change
2: Santarem, Brazil (June 2007 & May 2008)*
3: Yellow River Estuary (June 2008 & June 2009)
4: Santarem, Brazil (June 2007 & May 2008)*

*SAR Images courtesy of NASA Spatial Data Access Tool

Performance of Neural Network against Images with Varying Levels of Artificial Noise:





Variance	PCC	Network Change Map
0	99.5	
5	93.6	
10	89.9	

Parallelization of Code:

We parallelized our code with Matlab's Parallel Processing Toolbox to speed the process of sorting data to train and test the network.

Time Before (hours)	Time After (hours)	Percent of Original Time
3.27	1.07	32.7

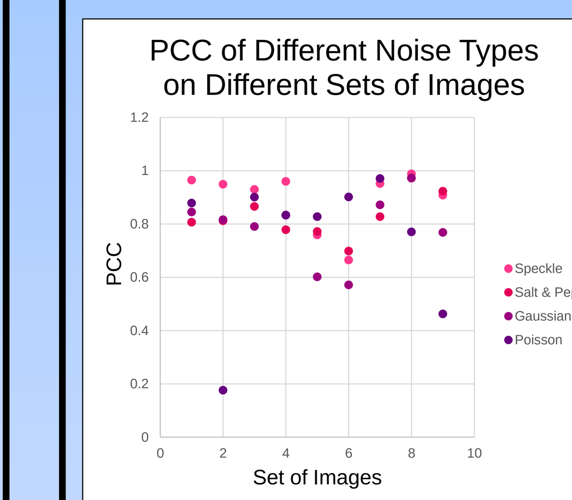
Neural Network Results

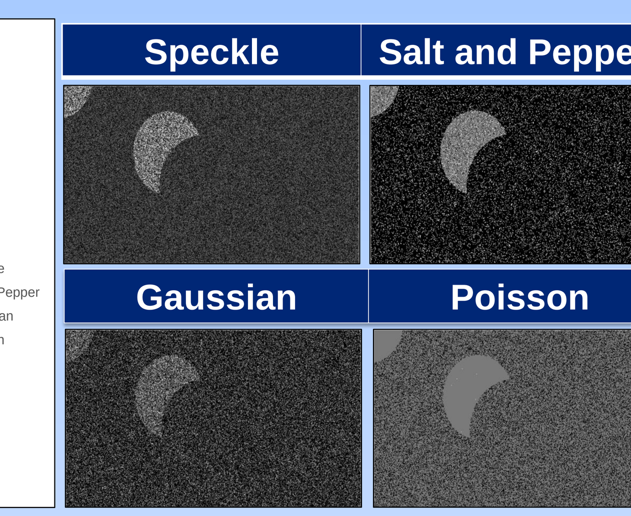
Performance of Neural Network against Different Types of Noise:

Method:
1) Normalize the images using the Frobenious norm,

$$\| A \|_F \equiv \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

2) Divide each image by its Frobenious norm and multiply that of the speckled image.





Conclusion and Future Work

Compared to traditional image segmentation methods, the neural network performed quite well. The network's performance increases when the amount of noise decreases. We did not discover consistent performance with respect to different types of noise. Based on the results, the network can interpret images with Gaussian noise, speckle noise, Poisson noise, and salt and pepper noise.

A few image segmentation techniques were tested for preclassification. FCM was concluded to be the most accurate. However, clustering and thresholding fails to take into account spatial features on an image. Other segmentation techniques that do account for spatial features include edge detection and region growth. Some of these require long processing times and/or human intervention, but can be tested in the future.

Another area of future work lies in accelerating the training of the neural network. Currently, parallelization affects only iterative image processing. Future work could be put towards discovering a parallel structure for the network training.

References

M. Gong, J. Zhao, J. Liu, Q. Miao and L. Jiao, "Change detection in synthetic aperture radar images based on deep neural networks", *IEEE Trans. Neural Netw. Learn. Syst.*

Masayuki Tanaka and Masatoshi Okutomi, *A Novel Inference of a Restricted Boltzmann Machine*, International Conference on Pattern Recognition (ICPR2014), August, 2014.

P. L. Rosin and E. Ioannidis, "Evaluation of global image thresholding for change detection", *Pattern Recognit. Lett.*, vol. 24, no. 14, pp. 2345-2356, 2003

J. C. Dunn (1973): "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters", Journal of Cybernetics 3: 32-57

Acknowledgements

Supported by NSF Grant to the REU EXERCISE - Explore Emerging Computing in Science and Engineering Program