IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium | 978-1-7281-6374-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/IGARSS39084.2020.9323957

SNOW RADAR LAYER TRACKING USING ITERATIVE NEURAL NETWORK APPROACH

Oluwanisola Ibikunle [1], John Paden [1], Maryam Rahnemoonfar [2], David Crandall [3], Masoud Yari [4]

1. Center for Remote Sensing of Ice Sheets, University of Kansas, Kansas, USA.
2. Department of Information Systems, University of Maryland, Baltimore County, Maryland, USA
3. Luddy School of Informatics, Computing, and Engineering, Indiana University, Indiana, USA.
4. Department of Computer Science, Texas A&M University-Corpus Christi, Texas, USA.

ABSTRACT

This paper presents preliminary results using a fully connected neural network (NN) to automatically track the internal layers of snow radar echograms using an iterative ”row-block-column” approach. Snow radar images, when accurately tracked, provide relevant information for estimating snow accumulation rates in polar regions which is a key measurement needed to understand and predict the impact of climate warming in Greenland and Antarctica. A multiclass NN was designed and trained with a training set of 121,408 columns of simulated snow radar data and learns to automatically track the internal layers with an accuracy of 92.8%, a RMSE of 0.24 pixels, and with 98% of pixel errors less than or equal to 1 pixel.

***Index Terms*—**
neural network, radar images, automatic tracking, machine learning, multiclass classification

1. INTRODUCTION

The snow radar [1], developed by the Center for the Remote Sensing of Ice Sheets (CReSIS), has a vertical resolution of <4 cm in snow and is used to measure annual snow fall by tracing annual layers in echograms [2]. Fig. 1 shows an example radar echogram from the middle of the Greenland ice sheet where the horizontal layers are clearly visible underneath the ice surface. In this work we propose a method to automate the tracking of these layers.

We propose a simple iterative approach that breaks an input echogram sequence into row blocks each containing at least one layer and using multiple columns of this as input to a NN. The NN iteratively detects the layers in an echogram one layer at a time until all the layers have been detected. We show preliminary results of this algorithm on simulated data. Although the simulated data we use do not contain some of the complicating image features of real snow radar data, these initial tests indicate this as a promising approach to pursue at least in part because of the simplicity of the algorithm and to use as a benchmark to compare against more capable NN [3] [4].

Work supported by NSF IIS-1838236 and ACI-1443054.

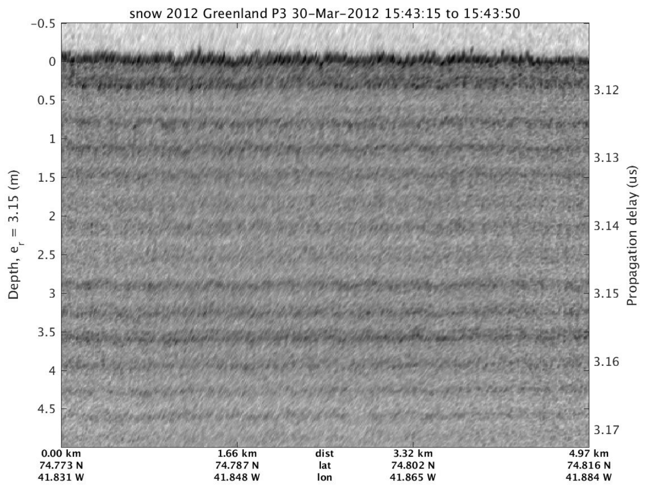

Fig. 1. Snow radar echogram with annual layers. Air-ice surface is at 0 m on the depth axis.

2. METHODOLOGY

2.0.1. *Iterative Column Layers Approach*

An echogram image consists of columns which always contain the air-ice surface and may contain internal snow layers underneath the surface. The first layer is always the ”surface” and it is the signal return from the interface between the air and the snow while deeper layers beneath this are the internal layers. The surface layer is easy to track using a simple threshold method since it is the first scattering return in each column and can be aided with existing digital elevation models of the surface. We use an iterative approach for detecting the internal layers of the snow radar echogram starting with the known surface layer. This approach uses the tracking information of the preceding layer (starting with the surface) to detect the next layer using only the next few rows of the echogram.

Given an echogram image (Fig. 2) and the surface information, the second or next layer can be extracted by grabbing the pixels in the rows directly beneath the surface. We term these extracted pixels as a *row block* since it is roughly parallel with the rows. It is not perfectly parallel, since it follows the layer above that defines it. The third or subsequent layer is in the next few rows after the second layer and can be extracted by grabbing the pixels in the rows beneath the second layer. We pose the tracking problem as an iterative detection problem which is solved one layer at a time using “row blocks”. An example of forming the first two row blocks of an image is shown in Fig. 2. The number of rows in each row block (N_{rb}) is chosen to be large enough so that the next layer occurs within the row block otherwise the algorithm will not be able to track the next layer. It may be that part of an even deeper layer (e.g. the next layer after the next layer) could be included in the row block and the NN will need to learn how to ignore these deeper layers and only track the next layer. N_{rb} was manually chosen by us based on the typical layer spacing. The example in Fig. 2 only shows row blocks with a single layer in each row block; if the number of rows in each row block had been increased, then part of layer 2 would have shown up in row block 1. The reason for restricting the number of rows in each row block (e.g. to $N_{rb} = 5$ rows in Fig. 2) is to reduce the size of the NN and therefore the learning time of the NN. However, this must be balanced with the need for N_{rb} to be large enough to always ensure that the next layer will be completely contained in the row block.

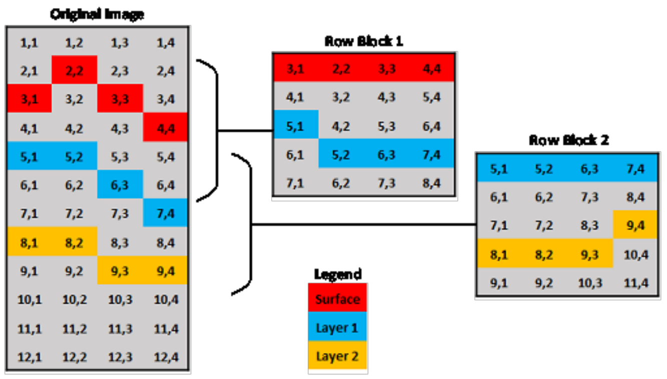

Fig. 2. Echogram matrix divided into row blocks

Using the information of the first layer traced out, the next row block is deduced. This is then used to trace out the next layer and this continues until the algorithm no longer finds a layer in the echogram. To trace out a layer from a selected row block, we further break the row block into columns and solve for the layer in each column independently of the solution for all the other columns. Each individual column solution is trained to depend on a fixed number of neighboring columns.

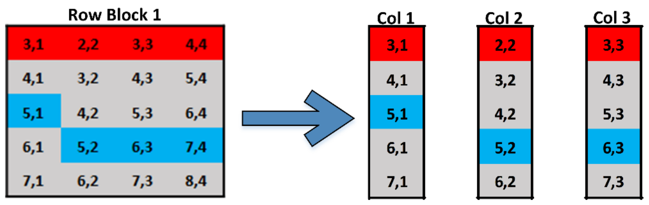

Fig. 3. Each row block is processed so that the solution of each column is determined independently of the solution to any of the other columns

2.1. Neural Network

To identify the location of a layer in each column, we designed a multiclass classification NN where the classes are the rows of the selected column and only one row of the selected column is allowed to contain the layer. This is a reasonable assumption for snow accumulation layers since they never fold in on themselves. For each column, we train the NN to output the most probable row containing the layer. We also include a *no-layer* class so that the NN can learn to recognize when there is no layer.

For training, the ground truth and number of layers in each echogram were provided and the network was trained to identify the last or deepest layer of an echogram by classifying the last row block after all the layers have been traced as belonging to the *no-layer* class. However, during testing, only the surface information is provided. Using the surface information, the first layer of each echogram is traced and this result is used to form the row block of the next/second layer and this process continues until no further layers are found by the NN. Errors in one layer may therefore cause deeper layers to not track as well because the row block will have errors in it.

To handle the termination condition, the number of columns in the current row block predicted by the NN as *no-layer* is checked against a threshold such that if the number of columns classified as *no-layer* exceeds this threshold, it is assumed that all the layers have been traced; consequently, the search for layers in the current echogram is then halted. In the instance where there are *no-layer* columns but less than this threshold, the iteration process for the next row block is no longer well defined since it is not clear which pixels are needed to form the columns of the next row segment. Linear interpolation was used to fill in columns which have missing layers for generating the next row block and the iterative search for layers continues until the threshold is exceeded. If extrapolation was required (e.g. if an edge column had a missing layer), then nearest neighbor extrapolation was used rather than linear interpolation.

2.2. Neural Network Input

To solve for each column, the input to the multiclass NN is the column and the neighboring N_{cols} columns to the left and

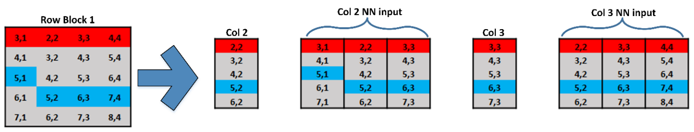

Fig. 4. NN input for columns 2 and 3 of row block 1 using neighboring columns (1-3) and columns (2-4) as inputs respectively.

right. Fig. 4 illustrates this with $N_{cols} = 1$ column to the left and right but for our experiment, $N_{cols} = 7$.

For columns near the edge, we mirrored neighboring available columns (see Fig. 5) to form the complete set of columns which are used as the input to the NN.

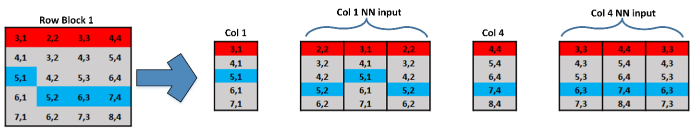

Fig. 5. Columns at the edge (Col 1 and 4) mirror their neighbors (Col 2 and Col 3 respectively) to form the NN input.

2.3. Architecture of the Neural Network

A fully connected NN with three layers: one input, single hidden layer and an output layer was designed.

As described, the input, x, consists of $N_{rb} \times N_{cols}$ pixels. The training set, $X_m = (x_1, x_2, .., x_M)$, consists of the inputs for M columns. The associated outputs, $Y_m = (y_1, y_2, .., y_M)$, are the known ground truth labels of the row containing the layer for each input column of the training set. Each output represents a multiclass where $y_m \epsilon \{no\text{-}layer, 1, .., N_{rb}\}$.

We define $j = (j_1, j_2, .., j_L)$ as the $L = 3$ layers of the network from input layer j_1 to output layer j_L. Each layer contains nodes $i = (i_1, i_2, .., i_N)$ and the network learns the optimum weights $\Theta^{(j)}$ that maps from layer j to layer $j + 1$.

The sigmoid activation function, given by

$$g(z) = 1/(1 + exp^{-z}), \tag{1}$$

is used in layer 2 to compute the activation of the units i in layer j as

$$a_i^{(j)} = g(\Theta_{i0}^{(j-1)} x_0 + \Theta_{i1}^{(j-1)} x_2 + ... + \Theta_{iN}^{(j-1)} x_N). \tag{2}$$

The output layer activation is similarly computed as

$$h_\theta^{(n)}(x_m) = a_i^{(L)} = P(y = n|x; \theta) = g(\Theta_{i0}^{(L-1)} x_0 + ... + \Theta_{iN}^{(L-1)} x_N). \tag{3}$$

The network prediction for each column is $softmax_i(h_\theta^{(i)}(x))$.

2.4. Regularized Cost Function and Backpropagation

A regularized logistic regression cost function is computed for the entire training set as

$$\begin{aligned} J(\theta) = (1/M) \sum_{m=1}^{M} \sum_{k=1}^{N_{rb}} \Big[-y_{mk}^{(i)} log\left(h_\theta^{(k)}\left(x_m^{(i)}\right)\right) \\ -(1 - y_{mk}^{(i)}) log\left(1 - h_\theta^{(k)}\left(x_m^{(i)}\right)\right)\Big] \\ + \lambda/2M \left[\sum_{i=1}^{N} \sum_{m=1}^{M} \left(\Theta_{i,m}^{(L-1)}\right)^2 \right. \\ \left. + \sum_{i=1}^{N} \sum_{k=1}^{N_{rb}} \left(\Theta_{i,k}^{(L)}\right)^2 \right]. \end{aligned} \tag{4}$$

where M = number of training examples, N_{rb} = number of rows in each row block which is equal to the number of neurons in the output layer, N = number of nodes in a layer, and L = number of layers in the NN. Backpropagation is used in the optimization and a regularization term controlled by λ in (4). The optimization is done using the fmincg conjugate gradient descent algorithm from Matlab. The optimization goal is to find the weights which minimize the overall cost function.

3. EXPERIMENTAL RESULTS

The training set contains 800 simulated snow radar echograms which corresponds to 121,408 columns from all the row blocks. The NN was trained on a 128 GB, 3.3 GHz, 8-core Red Hat Enterprise Linux server using Matlab. The simulated 1000 by 256 echogram matrices were decimated to 125 by 64 to reduce the number of inputs and outputs of the NN to keep the training tractable.

For testing, 200 simulated echograms with known surface were created. Using the surface information, the first layer of each echogram was traced and this was used to form the row block for the next/second layer and this process continued until the termination condition was met.

The following a priori information and hyper-parameters were used: the number of rows in a row block $N_{rb} = 16$, 15 neighboring columns ($N_{cols} = 7$ to the left and right) for each column of the row block, number of NN layers $L = 3$, number of nodes in hidden layer $N = 50$, *no-layer* termination threshold $\gamma = 0.5$, and a regularization term $\lambda = 50$.

An example input image is shown in Fig. 6 along with ground truth labels. The NN layer tracks for this image are shown in Fig. 7. Overall, an accuracy of 92.8 % was achieved with an RMSE of 0.24 pixels. Only about 2% of the pixel errors were greater than 1 pixel. Accuracy here is defined as the percent of the NN predictions that exactly match the ground truth. In other words, the percentage of columns in the test data that the NN predicted the correct row/*no-layer* state. Several output situations may occur that might affect

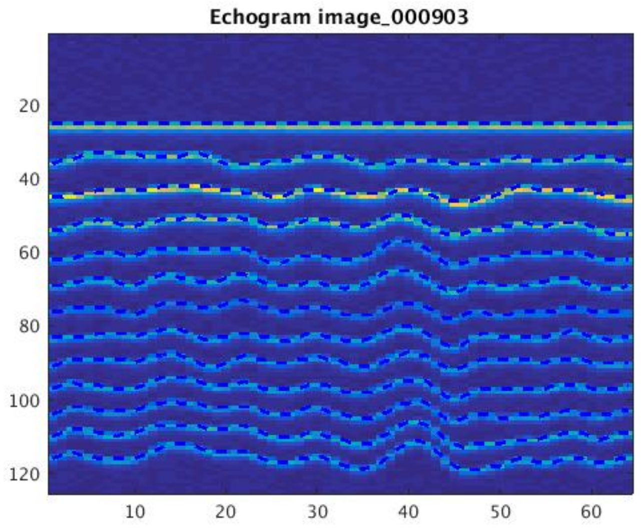

Fig. 6. Simulated snow radar echogram with labeled internal layers

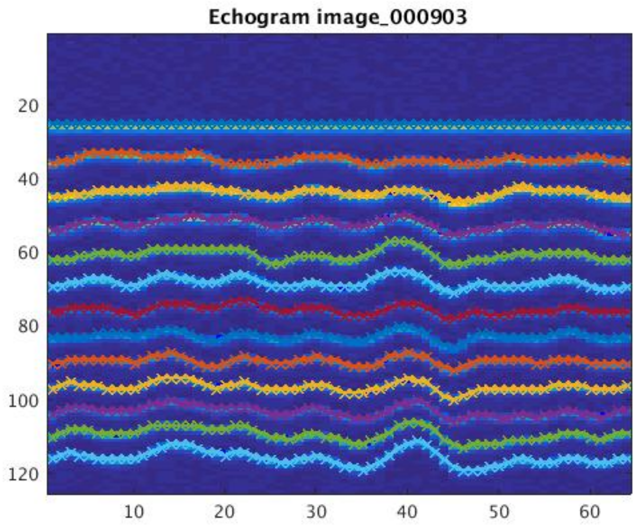

Fig. 7. Same image showing automatically tracked internal layers of the echogram

accuracy. First, if a layer is missed during prediction, but the following layer is tracked, then the "wrong" layer would be compared and the errors would be large for that layer and probably for deeper layers as well since the iterative process would likely be off one layer for every layer after this. Second, if an extra layer is tracked/hallucinated, then again the "wrong" layer would be compared so that layer and deeper layers would have large errors. Third, the tracker could terminate early (i.e. estimate that there is no layer when there is a layer). In this case the deeper layers would not be compared. In this experiment, none of these conditions occur, but we envisage these behaviors when applied to real data, layers which are sufficiently thin or thick, and low signal to noise ratio (SNR) data.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we present preliminary results of a novel iterative NN approach to automatically track the internal layers of a snow radar echogram one layer at a time. Although these results are on simulated data, processing of echograms collected during the NASA Operation IceBridge Mission is underway. We also plan to add convolutional layers to the NN and add memory into the network since layer tracking lends itself to a recurrent CNN framework.

5. REFERENCES

[1] F. Rodriguez-Morales, D. Gomez-Garcia Alvestegui, E. J. Arnold, R. D. Hale, S. Keshmiri, C. J. Leuschen, J. Li, J. D. Paden, and C. Cardenas, "Radar systems for ice and snow measurements onboard manned and unmanned aircraft," *IEEE Latin America Transactions*, vol. 16, no. 9, pp. 2473–2480, 2018.

[2] L. S. Koenig, A. Ivanoff, P. M. Alexander, J. A. MacGregor, X. Fettweis, B. Panzer, J. D. Paden, R. R. Forster, I. Das, J. R. McConnell, M. Tedesco, C. Leuschen, and P. Gogineni, "Annual Greenland accumulation rates (2009-2012) from airborne snow radar," *Cryosphere*, vol. 10, no. 4, pp. 1739–1752, 2016.

[3] I. O. L. K. Maryam Rahnemoonfar, John Paden and L. Montgomery, " Smart Tracking of Internal Layers of Ice in RadarData via Multi-Scale Learning," *IEEE International Conference on Big Data*, 2019.

[4] I. O. L. M. L. K. Maryam Rahnemoonfar, John Paden, " Deep Multi-Scale Learning for Automatic Tracking of Internal Layers of Ice in Radar Data," *submitted to Annals of Glaciology for International Glaciological Society International Symposium on Five Decades of Radioglaciology*, 2019.