# Deep Tiered Image Segmentation for Detecting Internal Ice Layers in Radar Imagery

Yuchen Wang[1] Mingze Xu[1] John D. Paden[2] Lora S. Koenig[3] Geoffrey C. Fox[1] David J. Crandall[1]
[1] Indiana University [2] University of Kansas [3] University of Colorado
{wang617,mx6,gcf,djcran}@indiana.edu, paden@ku.edu, lkoenig@nsidc.org

## Abstract

*Understanding the structure of the ice at the Earth's poles is important for modeling how global warming will impact polar ice and, in turn, the Earth's climate. Ground-penetrating radar is able to collect observations of the internal structure of snow and ice, but the process of manually labeling these observations with layer boundaries is slow and laborious. Recent work has developed automatic techniques for finding ice-bed boundaries, but finding internal boundaries is much more challenging because the number of layers is unknown and the layers can disappear, reappear, merge, and split. In this paper, we propose a novel deep neural network-based model for solving a general class of tiered segmentation problems. We then apply it to detecting internal layers in polar ice, and evaluate on a large-scale dataset of polar ice radar data with human-labeled annotations as ground truth.*

## 1. Introduction

The Earth's climate is changing [13]. As temperatures rise, the ice at the north and south poles is melting, creating a cascade of potentially catastrophic consequences: ice sheets break apart, glaciers melt, sea levels rise, and exposed land masses and sea absorb more solar energy, causing global temperatures to rise even faster.

To predict and potentially mitigate these changes, glaciologists have developed models of how polar ice and snow will react to changing climates. But these models require detailed information about the current state of the ice. While we may think of polar ice sheets as simply vast quantities of frozen water, in reality they have important structure that influences how they will react to rising temperatures. For example, deep below ice on land is the bedrock, which has all the same diversity as the rest of the Earth's surface — mountains, valleys, ridges, etc., obscured by deep ice and snow — that affect how melting ice will behave. The ice sheets themselves also have structure: snow and ice accumulate in annual layers year after year, and these layers record important information about past climatological events that can help predict the future.

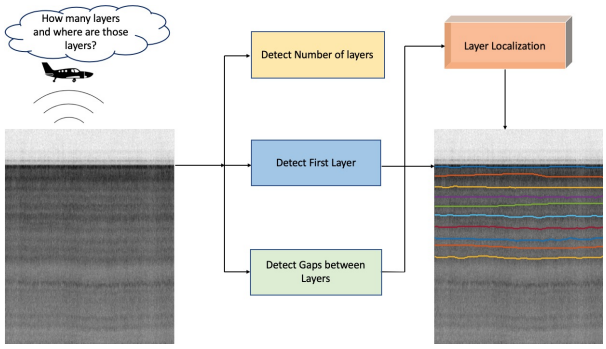


Figure 1. Given an echogram produced by ground-penetrating radar flying over polar ice, our task is to estimate the number of annual layers and localize them automatically.

To directly collect data about the structure of ice requires drilling ice cores — a slow, expensive, and extremely laborious process. Ground-penetrating radar systems have been developed that can fly above the ice sheet and collect information about the material (*e.g.*, air, ice, and terrain) boundaries deep under the ice. This process generates radar echograms (such as those in Fig. 1), where the vertical axis represents the depth of the return, the horizontal axis corresponds to distance along the flight path, and the pixel brightness indicates the amount of energy scattered from this subsurface structure. However, the echograms can be very noisy, typically requiring laborious and expensive manual annotation [22].

Some recent work [9, 14, 19, 28, 29] has developed automatic techniques for finding layers in these images, but has only considered the two most prominent layers: ice-air and ice-bedrock. A much more challenging problem is to identify the "internal" layers of the ice and snow caused by annual accumulation: not only are these layer boundaries much more subtle, but the number of layers that are visible

varies dramatically across different areas of the ice. Unlike traditional segmentation work in computer vision, the regions of interest in this problem do not correspond to "objects" with distinctive edges or appearance.

More generally, the problem we seek to solve here can be viewed as a generalization of the tiered scene segmentation problem [11]. Tiered image segmentation partitions an image into a set of regions $\{r_1, r_2, ..., r_n\}$ such that in each image column, all pixels belonging to $r_i$ are above (have lower row index than) all pixels corresponding to $r_j$ for $i < j$. Felzenswalb and Veksler [11] solved this problem using energy minimization with dynamic programming, but they assumed no more than three distinct labels per column because their inference time was exponential in the number of labels.

Here we revisit this tiered labeling problem using deep learning, but we consider the much more challenging problems in which the number of labels may be unknown ahead of time, and much greater than 3. We propose a novel deep neural network which performs the tiered segmentation in two stages. We first use a 2D convolutional network (CNN) to simultaneously solve three problems: detect the position of the top layer, roughly estimate the average distance between adjacent layers, and estimate the number of layers present in the image. Propagating the first layer downward with this rough gap gives a rough estimate of the tiered segmentation. Then we refine the pixel-level layer positions using a recurrent neural network (RNN) to account for differences across different layers.

We demonstrate the effectiveness of our method by applying it to finding internal ice layers, using a large-scale, publicly available polar echogram dataset. To our knowledge, ours is the first fully-automatic approach for solving this multi-layer ice segmentation problem: all previous techniques can either only find a fixed number of layers (1 or 2), or require extensive human interaction. Experimental results show that our approach significantly outperforms baseline methods, and is especially efficient on multi-layer detection. While we consider ice layer finding here, our technique is general and can be applied to other similar tiered segmentation problems.

## 2. Related Work

**Layer Detection in Radar Images.** A number of semi-automated and automated methods have been proposed for detecting boundaries between material layers in radar images. Crandall *et al*. [9] detected two specific types of layers (*i.e*., the ice-air and the ice-bed layers) in echograms by posing the problem as discrete energy minimization with a pretrained template model and a smoothness prior. Lee *et al*. [19] proposed a more accurate and efficient method by using Gibbs sampling from a joint distribution over all candidate layers, while Carrer and Bruzzone [4] further reduced the computational cost with a divide-and-conquer strategy. Xu *et al*. [28] first extended the work to the 3D domain and estimate 3D ice surfaces using a Markov Random Field (MRF), and Berger *et al*. [1] followed up with better cost functions that incorporate domain-specific priors into the cost computation and provide more specific characterizations of the ice sheets as additional evidence. Kamangir *et al*. [14] detected ice boundaries using wavelet transform with convolutional neural networks. Xu *et al*. [29] proposed a multi-task spatiotemporal neural network to reconstruct 3D ice surfaces from sequences of tomographic images.

However, the above work focuses on detecting a small, known number of layer boundaries (at most two) and cannot be directly applied to radar images that may contain an arbitrary number of layers. The state-of-the-art [23] for estimating surface internal layers is still based on a semi-automated model that requires human input.

**Edge Detection and Semantic Segmentation.** Our work is also related to edge detection and semantic segmentation. Traditional edge detectors [3, 16, 18] try to identify points with sharp changes in brightness, color, texture, etc. These methods are built on top of handcrafted features and parameters are carefully adjusted according to different scenarios. Learning based approaches, especially convolutional neural networks, take advantage of human annotated data and achieve better results. Among them, Dollar *et al*. [10] proposed a more effective and efficient edge detector by learning structure present in local image patches. DeepEdge [2] was the first top-down detection approach using a multi-scale deep network to extract hierarchical features. Xie *et al*. [27] proposed an end-to-end model that leverages hierarchical representations from different intermediate layers with skip-connections. Liu *et al*. [20] further learned richer deep features by utilizing information from all convolutional layers.

Deep learning models have made impressive advances for semantic segmentation. FCN [21] pioneered the idea of replacing all fully-connected layers with convolution operations. UNet [25] addressed the problem of losing information in pooling methods by building connections between shallow and deep layers. Deep Lab [5–7] captured sharper object boundaries using dilated convolution. PSPNet [31] incorporated global prior representations and applied a pyramid parsing module. ICNet [30] was introduced to achieve real-time inference with high-quality results.

However, directly applying edge detection and segmentation models to our problem does not work well, because the snow layers in our radar echograms have very subtle boundaries, and different layers do not have distinct internal characteristics like color or texture. Moreover, our images have structure (caused by physical processes of how ice forms) that is not captured by these models: layer bound-

aries are roughly parallel and do not cross, for example.

Our problem can be thought of as a more general version of the tiered segmentation problem [11] proposed by Felzenswalb and Veksler, who presented an algorithm based on dynamic programming. However, their solution required the number of tiers (labels) to be fixed ahead of time to a small number (3) because inference was *exponential in the number of labels*, and used hand-crafted features. In this paper, we propose a novel approach to a generalized version of the tiered segmentation problem in which the number of labels can be large and not known ahead of time. Our technique combines convolutional and recurrent neural networks for counting and detecting all ice internal layers in radar images.

## 3. Technical Approach

Given a noisy radar echogram $I$, which is a 2D image of size $1 \times H \times W$ pixels, our goal is to localize $N$ internal ice layers and exactly one surface boundary between the ice and the air. The output thus should be $N + 1$ layers. We need to estimate both the number of layers $N$ (which varies from image to image, although our implementation assumes $N < 30$) and all the layers' locations based on noisy and ambiguous data.

Our technique encodes the physical constraints of this tiered segmentation problem. First, since the labeled regions correspond to physical layers, layer boundaries cannot cross; more precisely, we partition the image into regions $\{r_1, r_2, ..., r_n\}$ such that in each image column, all pixels belonging to $r_i$ are above all pixels corresponding to $r_j$ for $i < j$. Second, we assume that adjacent layers are roughly parallel, which is reasonable since the amount of snow or ice that falls in any given year is roughly consistent across spatial location. Finally, we assume that the height of all layers is roughly the same, which is reasonable since the amount of snow or ice is similar across different years. These are all weak, rough assumptions, and our model is able to handle the significant deviations from them that occur in real radar data.

We address this problem using a two-step model. First, we predict $\hat{N}$ (number of ice layers), $\hat{F}$ (the top layer location, encoded as a $W$-d vector indicating the row index for each column of the image), and $\hat{\Delta}$ (the average vertical gap between all layers). Second, we predict $\hat{GapM}$, the gap between each adjacent pair of layers at each column (which is a $N \times W$ matrix) based on the information in the former step. Finally, we combine $\hat{F}$, $\hat{\Delta}$, and $\hat{GapM}$ according to $\hat{N}$ to generate output $\hat{M}$ which is a $(N+1) \times W$ matrix. Each element $\hat{m} \in [0, H]$ in $\hat{M}$ indicates the specific locations for that layer in the input $I$.

### 3.1. Triple Task CNN (CNN3B)

Convolutional Neural Networks (CNNs) work well in feature extraction and classification problems. To solve our problem, we design a three-branch CNN to predict a rough result including the surface layer location, the number of layers, and the average gap between all of the internal ice layers. We use VGG16 [26] as the CNN backbone. Fig. 2 shows the details of our CNN architecture, which was inspired by Xu *et al*. [29] but with significant modifications. Our model takes one 2D image $I$ as input. Then we use three shared convolutional blocks, each of which is followed by max pooling operations. The shared convolutional blocks are used to extract low-level features for the next three branches because the first layer, layer number, and average gap share similar low level patterns.

The model then divides into three branches. The first branch estimates the position of the surface layer, and uses six convolutional layers for modeling features specific to the first layer and one fully connected layer to generate outputs $\hat{F} = \hat{f}_1, \hat{f}_2, ..., \hat{f_W}$. Each element represents the row coordinate of the first layer within that column. The ground truth vector $F = f_1, f_2, ..., f_W$ is generated from the top boundary of the human-labeled ground truth $M_N = m_{N,1}, m_{N,2}, ..., m_{N,W}$. We apply L1 Manhattan distance to encourage the model to estimate the correct labeling according to human-labeled ground truth,

$$L_{fl} = \text{mean} \sum_{w=1}^{W} \left| \hat{f}_w - f_w \right|. \tag{1}$$

The second branch predicts the number of ice layers. It includes six convolutional layers and three fully connected layers to produce a vector $v$ which is a 31-class possibility vector. The ground truth is the number of labeled layers $N$ in the human annotations for data $M_N$. Cross-entropy loss is used during training since this part is a classification problem,

$$L_{number} = -\log\left(\frac{\exp(v[\mathrm{N}])}{\sum_j \exp(v[j])}\right). \tag{2}$$

The third branch is the average gap prediction branch. It follows the first branch design but with a single value output from the final fully connected layer. The L1 distance is used to calculate the loss between the output $\hat{\Delta}$ and the ground truth $\Delta$ generated from the human labeled matrix $M_N$,

$$L_{\Delta} = \left| \hat{\Delta} - \Delta \right|. \tag{3}$$

Finally, the three branch losses are added together to train our CNN model,

$$L = L_{fl} + L_{number} + L_{\Delta}. \tag{4}$$

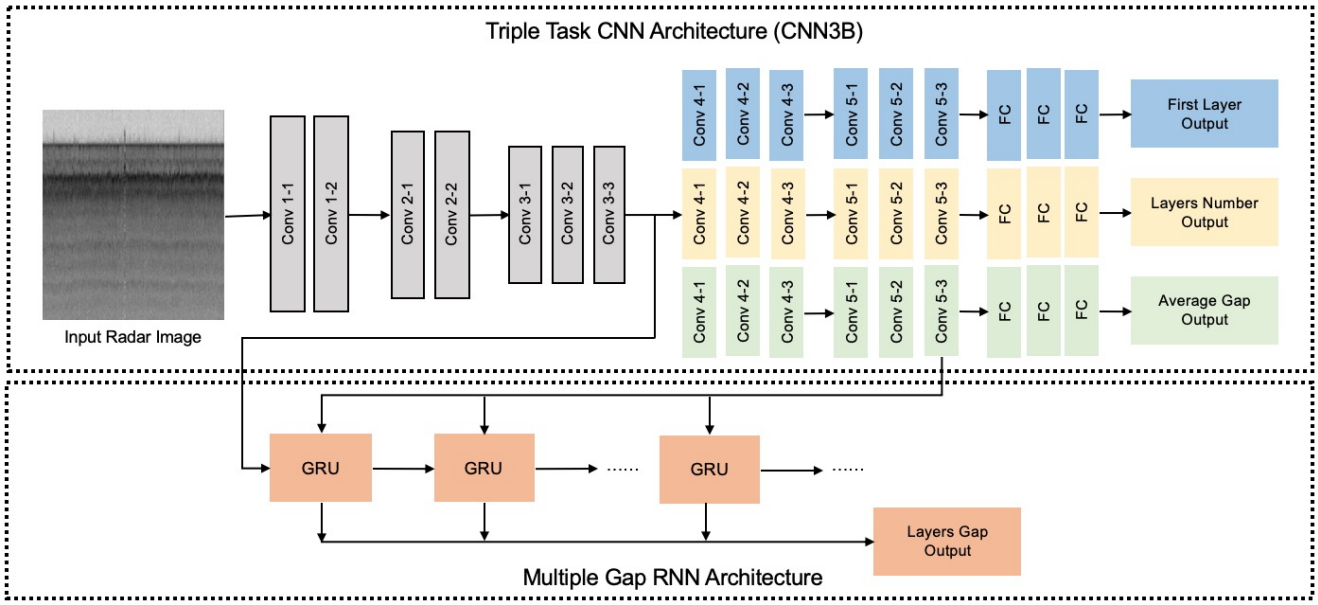


Figure 2. Architecture of our model for detecting internal ice layers in radar echogram images. Through a combination of CNN and RNN networks, we both estimate the number of layers and the boundaries of each layer.

### 3.2. Multiple Gap RNN

Since our CNN architecture only generates a general average gap between all the internal layers, we include a Recurrent Neural Network (RNN) which generates a more accurate gap value for every pair of adjacent layers. In particular, we use Gated Recurrent Units (GRUs) [8], which require less computational cost and are easier to train than other RNNs such as LSTMs [12].

As shown in Fig. 2, our model has one hidden layer and in that hidden layer, each GRU cell takes feature map $Avg_F$ generated before the fully connected layer of our Triple Task CNN model's third branch and the output of the previous GRU cell as inputs, and produces $W$ real-valued numbers indicating the predicted gap between layers within each column of the data. $Avg_F$ is projected to the size of the GRU hidden state with a fully connected layer before GRU takes it as input. During training time, the GRU cell is operated for $N$ iterations, where each iteration $n$ predicts the gap value between layer $n+1$ and layer $n$. In a given iteration $n$, the GRU cell takes the projected $Avg_F$ as input. The GRU cell outputs a sequence of hidden states $h_1, h_1, ..., h_n$ with iteration $n \in [1, N]$, and each hidden state $h_n$ is followed by a fully-connected layer to predict gap value $\hat{GapM}_n$. We use L1 Manhattan distance to supervise the model to predict $\hat{GapM}$ according to human-labeled ground truth $GapM$,

$$L_{GapM} = \left| \hat{GapM} - GapM \right|. \tag{5}$$

The GRU computes as follow,

$$\begin{aligned} z_n &= \text{sigmoid}\left(U_{iz}\mathcal{F}\left(Avg_F\right) + U_{hz}h_{n-1} + b_z\right), \\ r_n &= \text{sigmoid}\left(U_{iz}\mathcal{F}\left(Avg_F\right) + U_{hz}h_{n-1} + b_r\right), \\ g_n &= \tanh\left(U_{in}\mathcal{F}\left(Avg_F\right) + U_{hn}\left(r_n \circ h_{n-1}\right) + b_g\right), \\ h_n &= z_n \circ h_{n-1} + (1 - z_n) \circ g_n, \text{ and} \\ s_n &= U_y h_n + b_y, \end{aligned} \tag{6}$$

where $\circ$ is the Hadamard product, and $z_n$, $r_n$, $g_n$, $h_n$, and $s_n$ are the reset, input, new gate, hidden state, and output gap values, respectively, between layer $n$ and $n+1$. We use 512 neurons in the hidden layer of the GRU with L1 loss between $\hat{GapM}$ and $GapM$ to train the network.

### 3.3. Combination

We combine our Triple Task CNN and Multiple Gap RNN to predict the number of internal ice layers and their positions in the input image $I$. The RNN uses general features as shown in Fig. 2 to initialize the GRU's hidden state and takes an average feature map $Avg_F$ as input. Based on the first layer output and the number of layers from the Triple Task CNN, our model generates the first layer $M_0$ in our result $\hat{M}$. We then apply the layer gap output $GapM$ predicted by our multiple Gap RNN according to the first layer result,

$$M_i = M_{i-1} + GapM_i, i \in (0, N), \tag{7}$$

where N is number of layers minus 1 since it indicates the number of gaps between layers. To complete processing

an image, we compute all $M_i$'s to create $\hat{M}$ which is a $(N+1, W)$ matrix, and compare it with ground truth $M$ to evaluate our model.

# 4. Experiments

## 4.1. Data

We use a publicly-available annual ice layer dataset collected by the Center for Remote Sensing of Ice Sheets (CReSIS) at the University of Kansas and the National Snow and Ice Data Center at the University of Colorado [17]. The data is collected by ultra-wideband snow radar operated over a frequency range from 2.0 to 6.5 GHZ, and consists of 17,529 radar images with human-labeled annotations that identify the positions of internal ice layers. Formally, our task is to detect all internal ice layers $\hat{M}$ in a given single-channel image $I$. Each element in $\hat{M}$ indicates the row coordinate (in the range $[1, H]$, where $H$ is image height) of an ice layer for a given column.

**Preprocessing.** We resize all input images to $300 \times 256$ by using bi-cubic interpolation. We normalize the grayscale pixel values by subtracting the mean and dividing by the standard deviation (both of which are calculated from the training data). Following [29], we also normalize the ground truth row labels to a coordinate system spanning $[-1, 1]$ in each image. We also remove input images that have missing data.

## 4.2. Evaluation metrics

Past work has used mean absolute error in pixels between predicted and ground truth layers [9, 19], a familiar evaluation metric in signal processing applications. However, in our problem of internal ice detection, the number of layers is unknown, which means the evaluation metric must capture both the accuracy of estimated layer count and the localization accuracy of the layers. Past work considering the internal layer problem has evaluated qualitatively [23].

We thus introduce two quantitative, objective evaluation approaches for the tiered segmentation problem. Our first evaluation protocol assumes that the correct number of layers is known via an oracle, and then measures mean absolute error in pixels. This metric assumes that the correct number of layers is specified to the model, which is useful both for isolating the accuracy of layer localization, and for allowing comparison with models that are not able to estimate layer counts.

To evaluate both the accuracy of both the estimated layer count and layer boundaries, we propose *layer-AP* based on widely-used average precision. For each estimated layer, we search through the ground truth layers to find the closest match according to mean absolute error. Each ground truth layer is allowed to match at most one estimated layer. Then we define a set of threshold values $t_l$. For each threshold, we count the number of estimated layers which have a mean absolute error in pixels under the threshold, and call this the number of matches $m_i$ for that threshold. The layer average precision is then computed as,

$$\text{layer-AP} = \frac{\sum_{i=1}^{l} \frac{m_i}{N+1}}{l}, \tag{8}$$

where $N+1$ is the true number of layers ($N$ is the number of gaps between layers), and $l$ indicates the number of thresholds (we use 10, $t_l = [1, 4, 7, 10, 13, 16, 19, 22, 25, 27]$, which assume $300 \times 256$ input images).

## 4.3. Baselines

We are not aware of any existing work that solves the problem we consider here: existing fully-automatic approaches to the tiered segmentation problem assume the number of layers is no greater than 2 and is known ahead of time. We thus develop several baseline models to compare our results against.

- Crandall *et al.* [9] proposed a technique based on graphical models to find layer boundaries. However, they assume exactly two layer boundaries because the running time is exponential in the number of layers. Here, we adapted it to our problem by using an oracle to determine the number of layers (by looking at ground truth), and then running this technique sequentially to find each layer one-by-one. We call this *Sequential [9]*.

- *Naive CNN* is a traditional VGG16 [26] network which directly predicts a fixed number of internal layers by producing a label matrix in one-shot.

- *CNN2B* is a simpler version of our model that uses only two branches, one to predict the top layer (*i.e.*, the boundary between air and ice), and one to predict the average gap between each two layers. We use the first layer location and average gap to estimate succeeding internal layer positions.

- *CNN3B* is a version of our model with all three branches, but without the RNN refinement. This baseline is the same as *CNN2B* but with a third branch that also estimates the number of layers automatically.

- *RNN256* is a baseline that uses a recurrent neural network (RNN) to model sequential dependencies across columns. This baseline takes data from a given column and previous column information and predicts the internal layers in the next column, assuming a fixed number of layers. Previous information for the first column data is initialized randomly.

| | Mean Error (in pixels) ↓ | |
|---|---|---|
| | # layers from oracle | # layers estimated |
| Sequential [9] | 88.98 | - |
| Naive CNN | 24.32 | - |
| RNN30 | 21.79 | - |
| RNN256 | 20.20 | - |
| CNN2B | 11.94 | - |
| CNN3B | 7.91 | 9.27 |
| CNN3B + RNN | 6.96 | 8.73 |

Table 1. Experimental Results. Error in terms of the mean absolute column-wise difference compared to ground truth, in pixels.

| | layer-AP ↑ | |
|---|---|---|
| | # layers from oracle | # layers estimated |
| Sequential [9] | 0.059 | - |
| Naive CNN | 0.183 | - |
| RNN30 | 0.218 | - |
| RNN256 | 0.254 | - |
| CNN2B | 0.635 | - |
| CNN3B | 0.843 | 0.822 |
| CNN3B + RNN | 0.882 | 0.853 |

Table 2. Experimental Results. Layer average precision with thresholds compared to ground truth.

- *RNN30* models the dependencies in the vertical direction: given the estimated boundary for a given layer and previous layers, it predicts the boundary for the subsequent (next-oldest) annual layer.

Finally, our full model, *CNN3B+RNN*, combines the three-stream CNN model with the RNN. This full model uses a convolutional neural network to predict both the number of internal layers and a rough internal layer matrix, then feeding these estimates into the RNN for refinement.

### 4.4. Implementation Details

We use PyTorch [24] to implement our model and do the training and all experiments on a system with Pascal Nvidia Titan X graphics card. We randomly choose $80\%$ of data as training and remaining $20\%$ as testing. Adam [15] optimizer is applied on the CNN to learn the network parameters with batch size of 16. The training process is stopped after 30 epochs, starting with a learning rate of $10^{-4}$ and reducing it in half every 10 epochs. The RNN training uses the same optimizer, update rule, and batch size, but initial learning rate is $10^{-3}$.

### 4.5. Results

Tables 1 and 2 present the results of our quantitative evaluation, in terms of mean absolute error and layer-AP, respectively. In each table, we present two sets of results: one in which the number of layers is known ahead of time by an oracle (*i.e.*, by consulting the ground truth), and one in which it is predicted automatically. Note that only the techniques that use *CNN3B* are able to estimate the number of layers automatically, which is why the other results are listed as missing in the table. For calculating mean absolute error when models incorrectly estimate the number of layers, we pad either the ground truth or estimate with extra layers consisting of zero vectors to penalize these incorrect estimations

Comparing with other models in Tables 1 and 2, we see that our combination of three-branch CNN and RNN models significantly outperforms all baselines in terms of both mean average error and layer-AP. Our two models *CNN3B* and *CNN3B+RNN* have the ability to estimate the number of internal ice layers, and reach $85.2\%$ accuracy on this layer counting task, which is why their accuracy decreases only slightly when the number of layers is not provided by the oracle. Our model *CNN3B+RNN* shows the best results of all other baselines, even when our model must estimate the number of layers and the the baselines know it from the oracle.

Fig. 3 shows three examples with different numbers of internal layers. In each example, the first column shows the human annotated layers with input data as background. The second column shows the estimated result generated by one of our baselines, *CNN3B*. The results of this baseline roughly agree with the ground truth, but have some clear defects. For example, in the first row, the second orange layer in the *CNN3B* result fails to match the ground truth well. In the second row, the third yellow layer and fourth purple layer show clear mismatches. In the third row, all layers except the first show different degrees of dislocation compared with the human annotations in the first column.

The third and fourth columns show the prediction result with and without the layer number oracle. Since our *CNN3B+RNN* model successfully predicts the number of layers in these three cases, the output with and without the oracle are nearly the same. Both our *CNN3B+RNN* results show improvements in all three cases. In the first row, the second orange layer matches the human annotation in the first column better than *CNN3B* in the second column. In the second row, the third yellow layer and fourth purple layer are clearly closer to the ground truth than the result, while in the third row, all other layers show better results than the second column.

There are two failure cases showed in Fig. 4. In the fist failure case, there are 6 internal layers in the ground truth, but *CNN3B+RNN* without the oracle fails to predict the number of layers correctly. Our *CNN3B* only predicts the first layer well and shows a clear mismatch in all the other layers, but our *CNN3B+RNN* result in the fourth column predicts the first 5 layers reasonably well. Our *CNN3B+RNN* with the oracle shows the best results for this case, but there are still many ripples in the result showing that our model failed to predict it perfectly. In the second

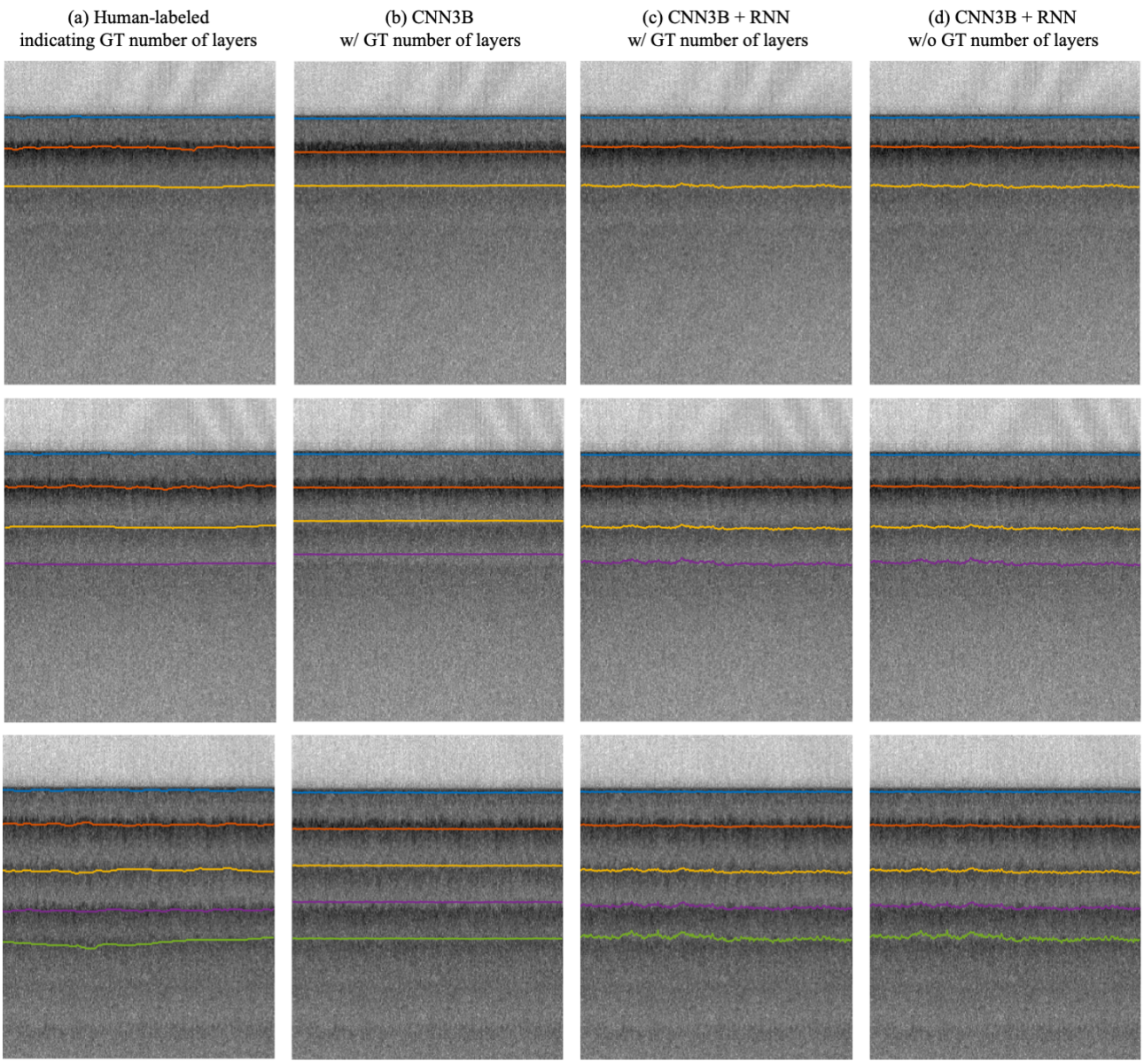


Figure 3. Sample results. First column is input data with ground truth. Second column is one of our baselines, *CNN3B*, with ground truth number of layers. Third column is our best model, *CNN3B+RNN*, with ground truth number of layers. Fourth column is our best model, *CNN3B+RNN*, with estimated number of layers. Four rows represent four different input data with annotations and prediction results.

failure case, our *CNN3B* model fails to predict almost all the layers, while *CNN3B+RNN* both fails to estimate the number of layers or match most of the internal layers. However, *CNN3B+RNN* still works well for the first two layers.

There are two causes for these failure cases. First, the input data is noisy and complex, and difficult to be annotated even by an experienced human annotator. Not only does this mean that our model must learn to process the complex and noisy input data, but it also means that the "ground truth" from human annotators also has much noise and inconsistency. This annotation noise not only adds noise during training, but also means that we are measuring error against ground truth that in itself has many errors. Second, we face a significant unbalanced dataset issue: there only 1.35% of the images have more than 5 internal ice layers, for example, which may affect our model's learning capability.

In summary, it is time consuming for humans to label the number and positions of layers in an image. As shown in the examples, our model only needs the input image to generate results which are very close to human annotations in most

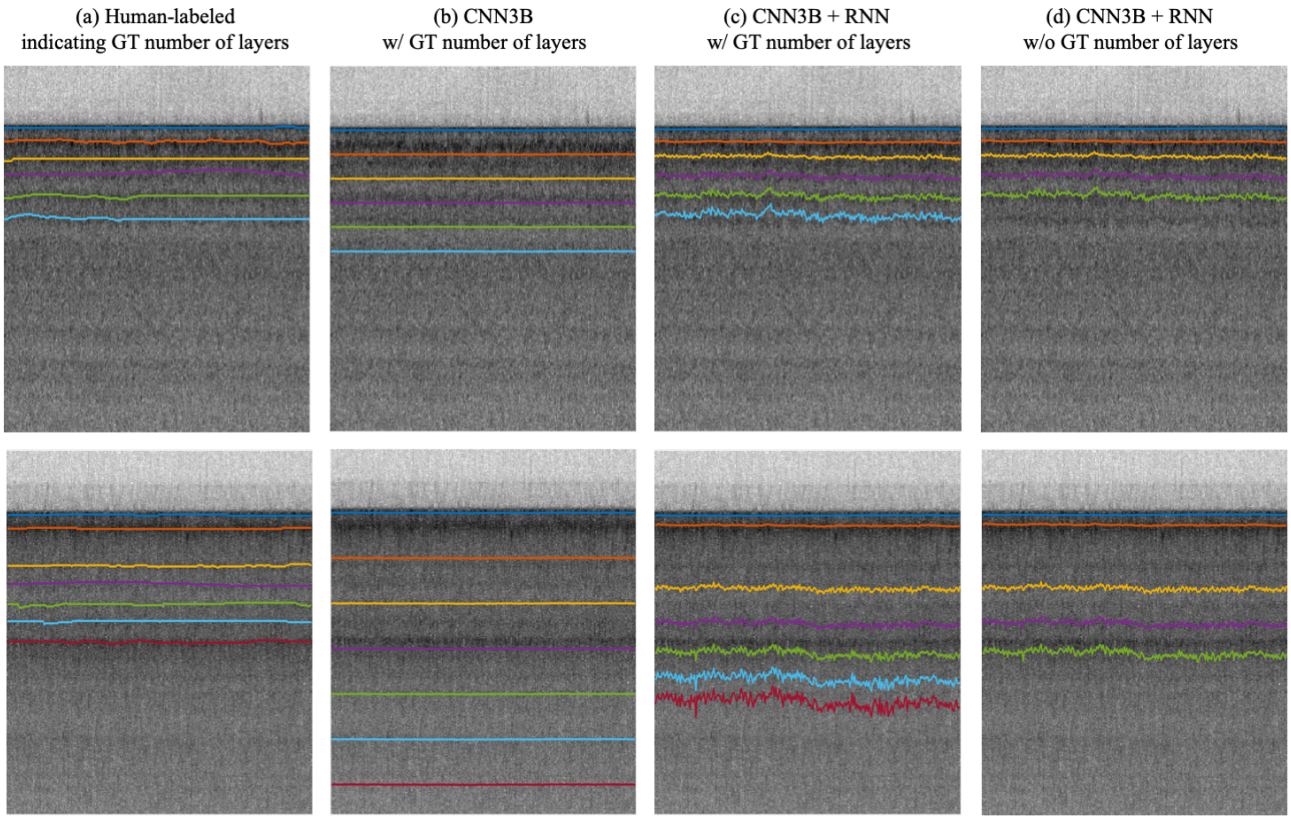


Figure 4. Failure cases. First column is input data with ground truth. Second column is one of our baseline *CNN3B* result with ground truth number of layers. Third column is our best model *CNN3B+RNN* result with ground truth number of layers. Fourth column is our best model *CNN3B+RNN* result with prediction number of layers. Three rows represent three different input data with annotations and prediction results.

of the cases. Our best model outperforms all the baselines, and the improvement between *CNN3B* and *CNN3B+RNN* indicates that the RNN contributes to our final result even though the *RNN30* and *RNN256* baselines fail to work well. The results show that both steps of our model are important to achieve the final performance.

## 5. Conclusion

We have considered a generalization of the tiered segmentation problem. Unlike existing work [11] which assumed that the number of labels was fixed and small (no more than 3), our approach automatically estimates the number of layers at inference time. Our approach also uses deep learning in place of the hand-crafted features of [11]. We then applied this technique to a problem of great societal consequence: automatically understanding the internal layer structure of the polar ice sheets from ground-penetrating radar echograms. To the best of our knowledge, this paper is the first to propose an automated approach to detect ice internal layers using deep neural networks. We showed that our approach can effectively estimate arbitrary numbers of snow or ice layers from noisy radar images. Experimental results on a challenging, publicly-available dataset demonstrate significant improvements over existing methods.

Our work also raises several opportunities for future work. Our current approach requires two steps, a CNN model to generate features and an RNN model to generate the final result. Future work may explore more elegant, unified models. Second, since we use real-world training data, the data distribution is unbalanced and biased towards easier cases. Future work could address this in several ways, such as creating synthetic data from a simulator to enrich the training data, which may improve our model's capability to detect the internal ice layers. Finally, we plan to apply our solution to other tiered segmentation and structured segmentation problems, such as internal wave detection, geological formation characterization, tree ring detection, etc.

## 6. Acknowledgments

This work was supported in part by the National Science Foundation (DIBBs 1443054, CAREER IIS-1253549), and by the IU Office of the Vice Provost for Research, the IU College of Arts and Sciences, and the IU Luddy School of Informatics, Computing, and Engineering through the Emerging Areas of Research Project "Learning: Brains, Machines, and Children." We acknowledge the use of data from CReSIS with support from the University of Kansas and Operation IceBridge (NNX16AH54G).

## References

[1] Victor Berger, Mingze Xu, Shane Chu, David Crandall, John Paden, and Geoffrey C Fox. Automated tracking of 2d and 3d ice radar imagery using viterbi and trw-s. In *IGARSS*, 2018.

[2] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, 2015.

[3] John Canny. A computational approach to edge detection. *TPAMI*, 1986.

[4] Leonardo Carrer and Lorenzo Bruzzone. Automatic enhancement and detection of layering in radar sounder data based on a local scale hidden markov model and the viterbi algorithm. *IGRASS*, 2017.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017.

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017.

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[8] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *ICML*, 2015.

[9] David J Crandall, Geoffrey C Fox, and John D Paden. Layer-finding in radar echograms using probabilistic graphical models. In *ICPR*, 2012.

[10] Piotr Dollár and C Lawrence Zitnick. Fast edge detection using structured forests. *TPAMI*, 2014.

[11] Pedro F. Felzenszwalb and Olga Veksler. Tiered scene labeling with dynamic programming. In *CVPR*, 2010.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[13] John Houghton. *Global warming: the complete briefing*. Cambridge university press, 2009.

[14] Hamid Kamangir, Maryam Rahnemoonfar, Dugan Dobbs, John Paden, and Geoffrey Fox. Deep hybrid wavelet network for ice boundary detection in radar imagery. In *IGARSS*, 2018.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[16] Josef Kittler. On the accuracy of the sobel edge detector. *Image and Vision Computing*, 1983.

[17] Lora S Koenig, Alvaro Ivanoff, Patrick M Alexander, Joseph A MacGregor, Xavier Fettweis, Ben Panzer, Richard R Forster, Indrani Das, Joseph R McConnell, Marco Tedesco, et al. Annual greenland accumulation rates (2009–2012) from airborne snow radar. *The Cryosphere*, 2016.

[18] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[19] Stefan Lee, Jerome Mitchell, David J Crandall, and Geoffrey C Fox. Estimating bedrock and surface layer boundaries and confidence intervals in ice sheet radar imagery using mcmc. In *ICIP*, 2014.

[20] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *CVPR*, 2017.

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[22] Joseph A MacGregor, Mark A Fahnestock, Ginny A Catania, John D Paden, S Prasad Gogineni, S Keith Young, Susan C Rybarski, Alexandria N Mabrey, Benjamin M Wagman, and Mathieu Morlighem. Radiostratigraphy and age structure of the greenland ice sheet. *Journal of Geophysical Research: Earth Surface*, 2015.

[23] Jerome E Mitchell, David J Crandall, Geoffrey C Fox, and John D Paden. A semi-automatic approach for estimating near surface internal layers from snow radar imagery. In *IGARSS*, 2013.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPS 2017 Workshop on Autodiff*, 2017.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[27] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[28] Mingze Xu, David J Crandall, Geoffrey C Fox, and John D Paden. Automatic estimation of ice bottom surfaces from radar imagery. In *ICIP*, 2017.

[29] Mingze Xu, Chenyou Fan, John D Paden, Geoffrey C Fox, and David J Crandall. Multi-task spatiotemporal neural networks for structured surface reconstruction. In *WACV*, 2018.

[30] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *ECCV*, 2018.

[31] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.