River Ice Segmentation with Deep Learning
CMPUT 617 Project Report

Abhineet Singh
Student ID: 1395723

Abstract

This report deals with the problem of computing surface ice concentration for two different types of ice from river ice images. It presents the results of applying several state of the art semantic segmentations methods utilizing deep CNNs for solving this problem. This task presents two main challenges - very limited availability of labeled training data and the great difficulty of visually distinguishing the two types of ice, even for humans. The results are used to analyze the extent to which some of the best deep learning methods currently in existence can handle these challenges.

1. Introduction

The study of surface ice concentration and variation over time and place is crucial for understanding the river ice formation process. The temporal and spatial ice distributions thus computed can help to validate models of this process. The additional ability to distinguish frazil and the sediment-carrying anchor ice can also help to increase the estimation accuracy of the sediment transportation capacity of the river. Towards this end, a large amount of video data has been captured using UAVs and bridge mounted game cameras from two Alberta rivers during the winters of 2016 and 2017. The objective of this work is to analyze this data and perform dense pixel wise segmentation on these images and videos to be able to automatically compute the concentrations of the two types of ice.

The main challenge in this task is the lack of labeled data since it is extremely difficult and time consuming to manually segment images into the three categories due to the arbitrary shapes that the ice pans can assume. As a result, there are currently only 50 labeled images (Fig. 1) to accompany 564 unlabeled test images and over 100 minutes of unlabeled 4K videos. These labeled images along with 205 additional images with only ice-water labeling have already been used to train an SVM [17, 16, 15] to perform the segmentation. It provided water-ice classification accuracies ranging from 80.1% - 93.5% and surface ice concentration errors of 0.7% - 3.3%. Though it was fairly successful at separating ice from water, it had much greater difficulty in distinguishing between frazil and anchor ice pans, especially in cases where they are not physically separated and are hard to differentiate even for human eyes. This project is mainly concerned with handling these more difficult cases.

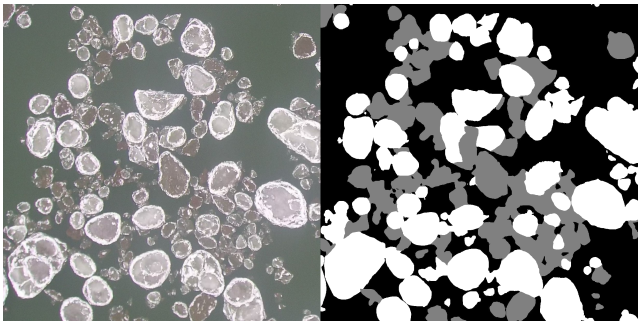

Figure 1. Sample training image with label where white, grey and black pixels respectively denote frazil ice, anchor ice and water

To address the limitations of SVM, this work uses recent deep learning based methods of semantic segmentation. Since these methods need large amounts of training data to work well, the training images have been subjected to several data augmentation techniques (Sec. 3.2) to generate enough data. Another promising approach, though not tested here due to time limitations, is to use this augmented dataset as the starting point for a boot-strapping process. Successively better models can be trained by manually correcting the segmentation results produced on the test images by each stage of the process and adding these corrected images to the labeled set for training the next stage model.

2. Background

The river ice images look very similar to microscopic images of cells in the bloodstream so my initial idea was to try existing cell classification networks in medical imaging after fine tuning them on the training images. I found several promising works employing a range of architectures including ConvNet [30], LeNet [27, 23], Resnet [19] and Inception [20] that might have provided the base network for this

work. However, more detailed examination revealed that medical imaging tasks are mainly concerned with the detection and localization of specific kinds of cells rather than performing pixel wise segmentation that is needed here.

Further, I looked for unsupervised or semi-supervised video segmentation techniques to utilize the large amount of high-quality video data. I found that most of them use optical flow for performing motion segmentation [10], though some appearance based [2] and hybrid [11, 14] methods have also been proposed. A recent work [24] proposes an unsupervised bootstrapping approach, somewhat similar to the one mentioned above. Under the assumption that all the moving pixels belong to the same foreground object, it uses the motion segmented images as training data to learn an implicit representation of this object. The model is used for refining the motion segmentation and the improved results are in turn used to bootstrap further refinements.

However, there are two assumptions underlying this work as well as motion segmentation in general, which renders such methods unsuitable for our task. Firstly, they assume that there is a single moving foreground object while, in our task requires distinguishing between two different types of moving ice, which are both foreground objects. Secondly, they assume a static background while the river, which makes up the background in our case, is itself moving. Preliminary attempts to perform optical flow-based motion segmentation on these videos confirmed their unsuitability for this work. I did find a recent method [29] for performing simultaneous optical flow estimation and segmentation which might be able to address these limitations to some extent. However, I was unable to gets its Matlab code working in time and so deferred its further exploration to future work. Finally, I looked for existing applications of deep learning for surface ice analysis and though I did find one [6], it uses microwave sensor data instead of images.

3. Methodolgy

3.1. Image Segmentation

Since neither cell classification nor video segmentation methods seemed promising, I decided to use supervised image segmentation instead. I found a couple of excellent resources for these methods [22, 9] and selected three of the most widely cited and best performing ones with publicly available code that I was able to get working in time.

The first of these is UNet [25] from the medical imaging community. It was introduced for neuronal structure segmentation in electron microscopic images and won the ISBI challenge 2015. As the name suggests, UNet combines a contracting part with a symmetrix expanding part to yield a U-shaped architecture that can both utilize context information and achieve good localization owing to the two parts respectively. It was shown to be trainable with relatively few training samples while relying heavily on patch based data augmentation which seemed to make it an ideal fit for our requirements.

The second network is called SegNet [1] and was introduced for segmenting natural images of both outdoor and indoor scenes for scene understanding application. It uses a 13-layer VGG16 net [28] as its backbone and features a somewhat similar architecture as UNet. The contracting and expanding parts are here termed encoder and decoder respectively and the upsampling units in the latter are not trainable, instead utilizing the weights learned by the corresponding max-pooling layers in the former. I have used Keras [8] implementations for both UNet and SegNet. available as part of the same repository [12] along with a couple of variants of the FCN architecture [21, 26] which, however, did not perform as well and are thus excluded from this study.

The third method is called Deeplab [3] and is one of the best performing methods in the Tensorflow research models repository [4]. It uses convolutions with upsampled filters - the so called atrous convolutions - to both achieve better control over the feature response resolution and to incorporate larger context without increasing computational cost. It also achieves scale-invariance by using a pyramidal max pooling and improves localization accuracy while maintaining spatial invariance by combining the last layer output with a fully connected conditional random field layer. I used a more recent version called Deeplabv3+ [5] which adds a decoder module to produce sharper object boundaries while also incorporating the Xception model [7] for further performance improvements.

In addition to these methods, I also used an adapted version of an unpublished approach called indicator learning that is based on the DenseNet architecture [13]. The basic idea of DenseNet is to connect each layer of the network to all subsequent layers so that the feature maps output by each layer are used as input in all subsequent layers. This provides for better feature propagation and reuse while also reducing the total number of parameters and mitigating the vanishing gradient problem.

Indicator learning uses a loss function which forces the n channel output probability map (where n is the no. of classes) to learn one channel for each class by forcing the pairwise inner product between the probability values for labelled pixels to be 1 when the pixels belong to the same class and 0 otherwise. This has the disadvantage that, though it separates the pixels into distinct classes, it does not keep track of which channel corresponds to which class and hence class identification is not possible. In addition, it leads to a lot of flip-flopping during training as different channels are successively trained for different classes. To resolve these issues, the standard L2 loss was used instead.

Table 1. Augmented dataset sizes. Last column shows the number of training samples used for DenseNet (Sec. 3.2)

K	All	Training	Validation	DenseNet
256	159336	84075	44013	-
384	44130	29268	15384	348
512	18231	11895	6426	361
640	7347	4860	2358	349
800	2292	1587	831	191
1000	876	609	375	381

3.2. Data augmentation and Training

A simple sliding window approach has been used to extract a large set of sub-images or patches from each training image. The window is moved by a random stride between 10% to 40% of the patch size K. This process is repeated after applying random rotations to the entire image between 15 to 345 degrees divided into four bands of equal width to allow for multiple rotations for each image. Finally, each patch is also subjected to horizontal and vertical flipping to generate two additional patch. All resultant patches are combined together to create the dataset for each K. For testing a model, patches of size K are extracted from the test image using a stride of K, segmentation is performed on each and the results are stitched back to get the final result.

All models were trained and tested using patch sizes $K \in \{256, 384, 512, 640\}$ except DenseNet where larger sizes of 800 and 1000 were used instead of 256 as it seemed to perform better with larger images. The 50 labeled images were divided into two sets of 32 and 18 for generating the training and validation images respectively. Table 1 lists the sizes of the datasets thus generated for each patch size. Note that the training sets were used for generating the quantitative performance results on the validation sets while the combined sets generated using all 50 images (first column of Table 1) were used for producing qualitative results on the unlabeled test set.

UNet and SegNet were both trained for 1000 epochs and the training and validation accuracies were evaluated after each. The trained model used for testing was the one with either the maximum validation accuracy or the maximum mean accuracy depending on how well the training and validation accuracies were matched in the two cases. Deeplab was trained for between $50,000$ and $100,000$ steps. Batch size of 10 was used for $K = 256$ and 2 for $K \in \{640, 800\}$ with the latter chosen due to memory limitations. $K = 384$ was tested with batch sizes 6 and 8 while $K = 512$ was tested with 6 and 2. Most tests were conducted using the default stride of 16 with corresponding atrous rates of $[6, 12, 18]$ though one model with $K = 256$ was also trained using Stride 8 with atrous rates of $[12, 24, 36]$.

DenseNet training was a bit trickier. Simply using all the pixels for training caused the network to rapidly converge to labeling all pixels with the class that had the maximum number of pixels, which was water in most cases. To get meaningful results, the number of pixels belonging to each of the classes had to be balanced. Therefore $10,000$ random pixels belonging to each class were selected in each epoch, with different set of pixels selected each time, and only these were used for computing the loss. Training images with less than $10,000$ pixels in any class were discarded. Also, DenseNet took much longer to train than the other networks and so was trained with a much smaller number of training images as shown in the last column of Table 1. Number of epochs were between $1300 - 1600$ for all K except 1000 which could only be trained for 360 epochs due to time limitations. In all cases, the performance metrics in Sec. 1 were computed on the validation set every 10 epochs and training was stopped when these became high enough and remained unchanged long enough.

4. Results

4.1. Evaluation metrics

Following four metrics [26, 18] have been used for evaluating the segmentation results:

1. Pixel accuracy:

$$\frac{\sum_i n_{ii}}{\sum_i t_i} \tag{1}$$

2. Pixel accuracy:

$$\frac{1}{n} \sum_i \frac{n_{ii}}{t_i} \tag{2}$$

3. Mean IOU:

$$\frac{1}{n} \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{3}$$

4. Frequency Weighted IOU:

$$\sum_k (t_k)^{-1} \sum_i \frac{t_i n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{4}$$

where $n = 3$ is the number of classes, n_{ij} is the number of pixels of class i predicted to belong to class j and t_i is the total number of pixels of class i in the ground truth

4.2. Quantitative

Fig. 2 and 3 show the performance results for the four models with all the configurations that they were tested in while Fig. 4 compares the best configuration of each. It can be seen that UNet and SegNet are relatively invariant

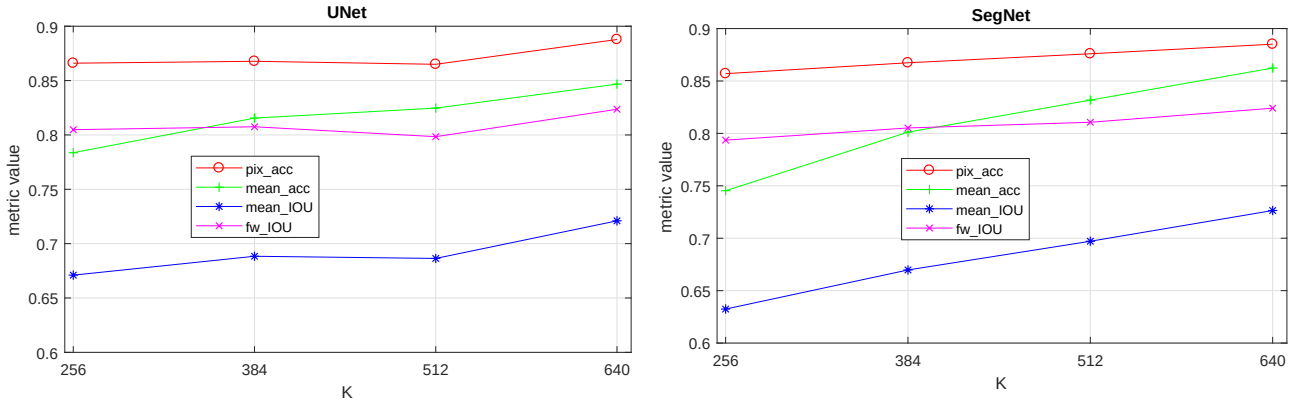

Figure 2. Performance of UNet and SegNet on the validation set

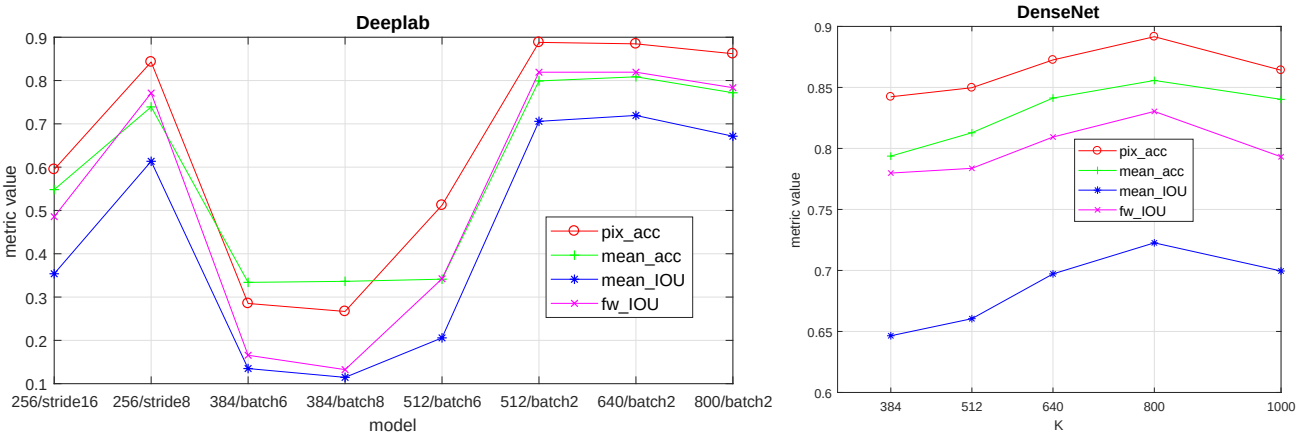

Figure 3. Performance of Deeplab and DenseNet on the validation set

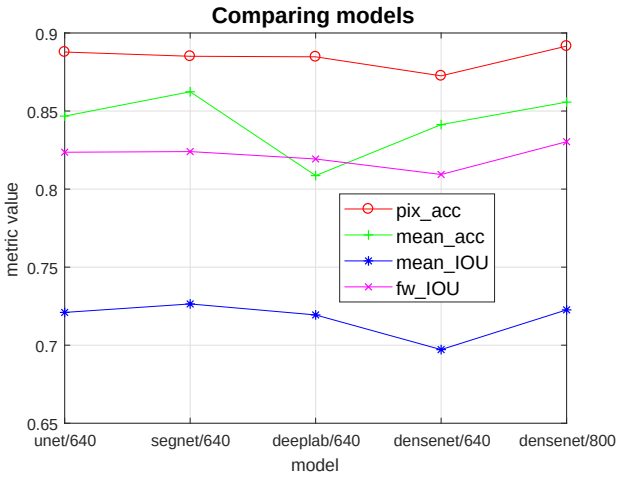

Figure 4. Performance of the best configurations of the models

to variations in K though they do show a slight improvement with $K = 640$. They also perform quite similarly to each other which is to be expected as they share the same VGGNet16 backbone network. All of the models show general improvement with increase in K except Deeplab with 384 and 512 and DenseNet with 1000. The former is probably due to incorrect batch size as 512 did manage to recover with a batch size of 2 instead of 6 whic is remarkable as the guidelines in the repository [4] mention that larger batch sizes of 12 or more are needed for the best performance. The latter is probably caused by the limited number of epochs (360) for which it could be trained as the validation accuracies were still increasing when training had to be terminated. DenseNet with $K = 800$ is the best performing model overall which is impressive as it uses by far the least number of trainable parameters - only around 90,000 - while UNet and Segnet each have around 12 million and Deeplab has by far the maximum of 140 million. DenseNet was also trained using the least number of images which were as little as a hundredth of the other models to as much as a tenth which makes its superiority even more remarkable. Finally, Deeplab shows significantly better performance with a stride of 8 in the sole case where this was tested which might indicate possibilities for future improvements.

4.3. Qualitative

Fig. 5, 6 and 7 show the results of applying the best configurations of the four models to segment several images from the test set. UNet, SegNet and Deeplab were tested using $K = 640$ while Densenet used $K = 800$. Several interesting observations can be made. Firstly, both UNet and SegNet misclassify water as frazil ice in several cases where most of the image contains water, e.g. in images 2 and 3 respectively and of Fig. 5 and 6. DenseNet too seems to be somewhat susceptible to this issue though to a much lesser extent. Secondly, Deeplab results show the largest degree of discontinuity between adjacent patches due to its tendency to occasionally produce completely meaningless segmentations on some individual patches. Examples include image 6 in Fig. 5, image 5 in Fig. 6 and images 1 and 4 in Fig. 7. Thirdly and consistently with the quantitative results of the previous section, DenseNet is overall the best performing model even though its results are slightly more fragmented than the others. This is particularly noticeable in the more difficult cases of distinguishing between frazil and anchor ice when they both form part of the same ice pan. Examples include images 1 and 7 in Fig. 6 and image 1 in Fig. 7.

5. Conclusions

This report presented the results of using four state of the art deep learning based semantic segmentation methods for segmenting river ice images into water and two types of ice. Three of these - UNet, SegNet and Deeplab - are previously published and well studied methods while the fourth one - DenseNet - is a new method, though based on a recent architecture. All of the models provided fairly good results, both quantitatively on the labeled validation images as well as qualitatively on the unlabeled test images. These were significant improvements over the previous attempts using SVM especially in distinguishing between the two types of ice. Of the 4 models, DenseNet performed the est even though it uses the fewest parameters by far and was also trained using the least number of images. This provides a promising avenue for future exploration that might be able to yield much better performance with more layers and training images.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(12):2481–2495, 2017. 2

[2] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2018. 2

[4] L.-C. Chen, Y. Zhu, and G. Papandreou. DeepLab: Deep Labelling for Semantic Image Segmentation. Github, 2017. hhttps://github.com/tensorflow/models/tree/master/research/deeplab. 2, 4

[5] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv:1802.02611*, 2018. 2

[6] J. Chi and H.-c. Kim. Prediction of arctic sea ice concentration using a fully data driven deep neural network. *Remote Sensing*, 9(12), 2017. 2

[7] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pages 1800–1807. IEEE Computer Society, 2017. 2

[8] F. Chollet et al. Keras. https://keras.io, 2015. 2

[9] Dr.Tang. Semantic-Segmentation. Github, 2017. https://github.com/tangzhenyu/SemanticSegmentation_DL. 2

[10] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *BMVC*, 2014. 2

[11] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition (CVPR 2010)*, 2010. 2

[12] D. Gupta. Image Segmentation Keras : Implementation of Segnet, FCN, UNet and other models in Keras. Github, 2017. https://github.com/divamgupta/image-segmentation-keras. 2

[13] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269. IEEE Computer Society, 2017. 2

[14] S. Jain, B. Xiong, and K. Grauman. FusionSeg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. *CVPR*, 2017. 2

[15] H. Kalke. Sediment Transport by Released Anchor Ice: Field Measurements and Digital Image Processing . Master's thesis, University of Alberta, 2017. 1

[16] H. Kalke and M. Loewen. Support Vector Machine Learning Applied to Digital Images of River Ice Conditions. Submitted for review to Cold Regions Science and Technology, September 2017. 1

[17] H. Kalke and M. Loewen. Predicting Surface Ice Concentration using Machine Learning . In *19th Workshop on the Hydraulics of Ice Covered Rivers*, Whitehorse, Yukon, Canada,, July 9-12 2017. CGU HS Committee on River Ice Processes and the Environment. 1

[18] M. Kerner. Image Segmentation Evaluation. Github, 2017. https://github.com/martinkersner/py_img_seg_eval. 3

[19] H. Lei, T. Han, W. Huang, J. Y. Kuo, Z. Yu, X. He, and B. Lei. Cross-modal transfer learning for hep-2 cell classification based on deep residual network. In *2017 IEEE International Symposium on Multimedia (ISM)*, pages 465–468, Dec 2017. 1

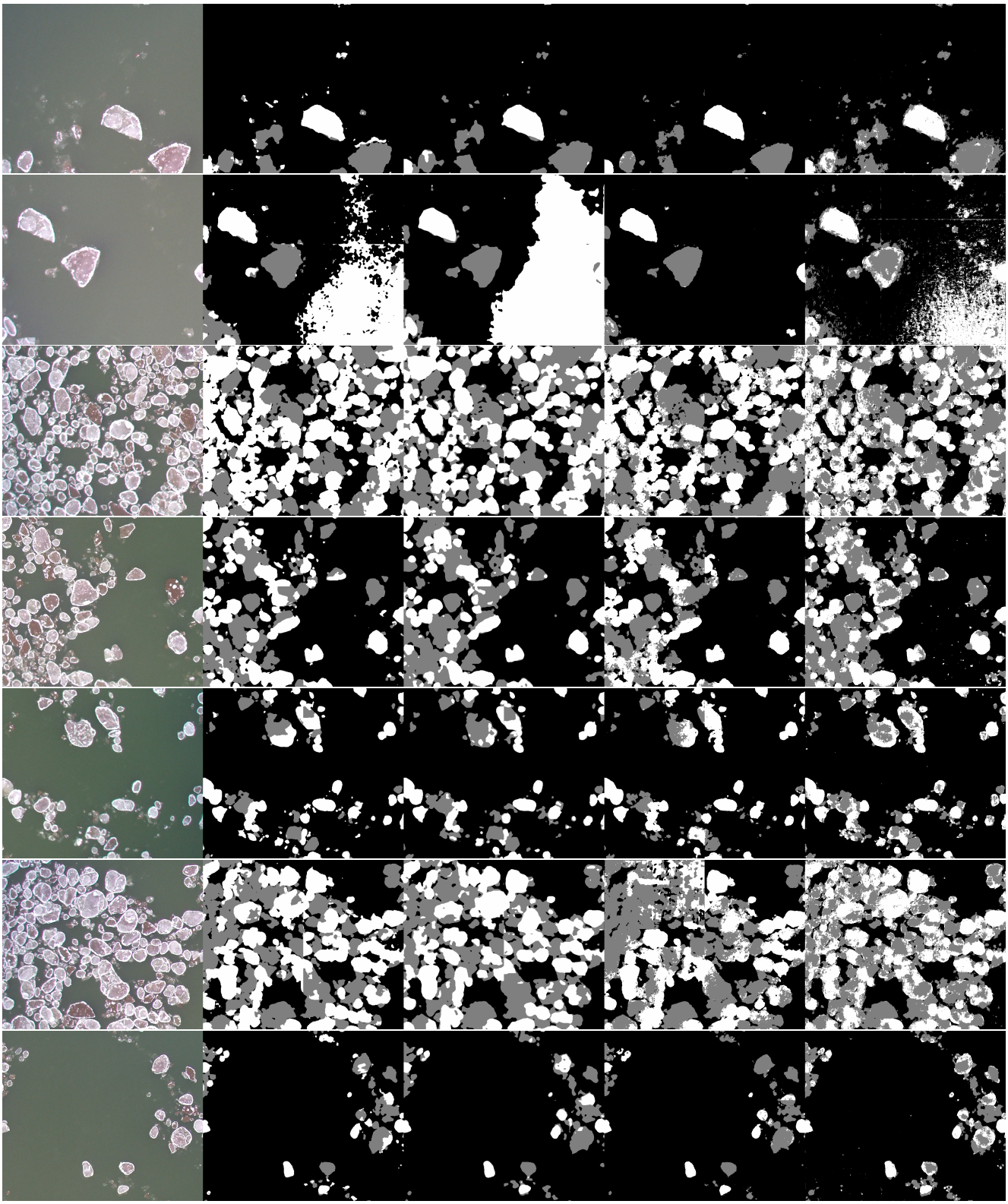

Figure 5. Results of applying the best configurations of the four models on test images: left to right: UNet, SegNet, Deeplab, DenseNet

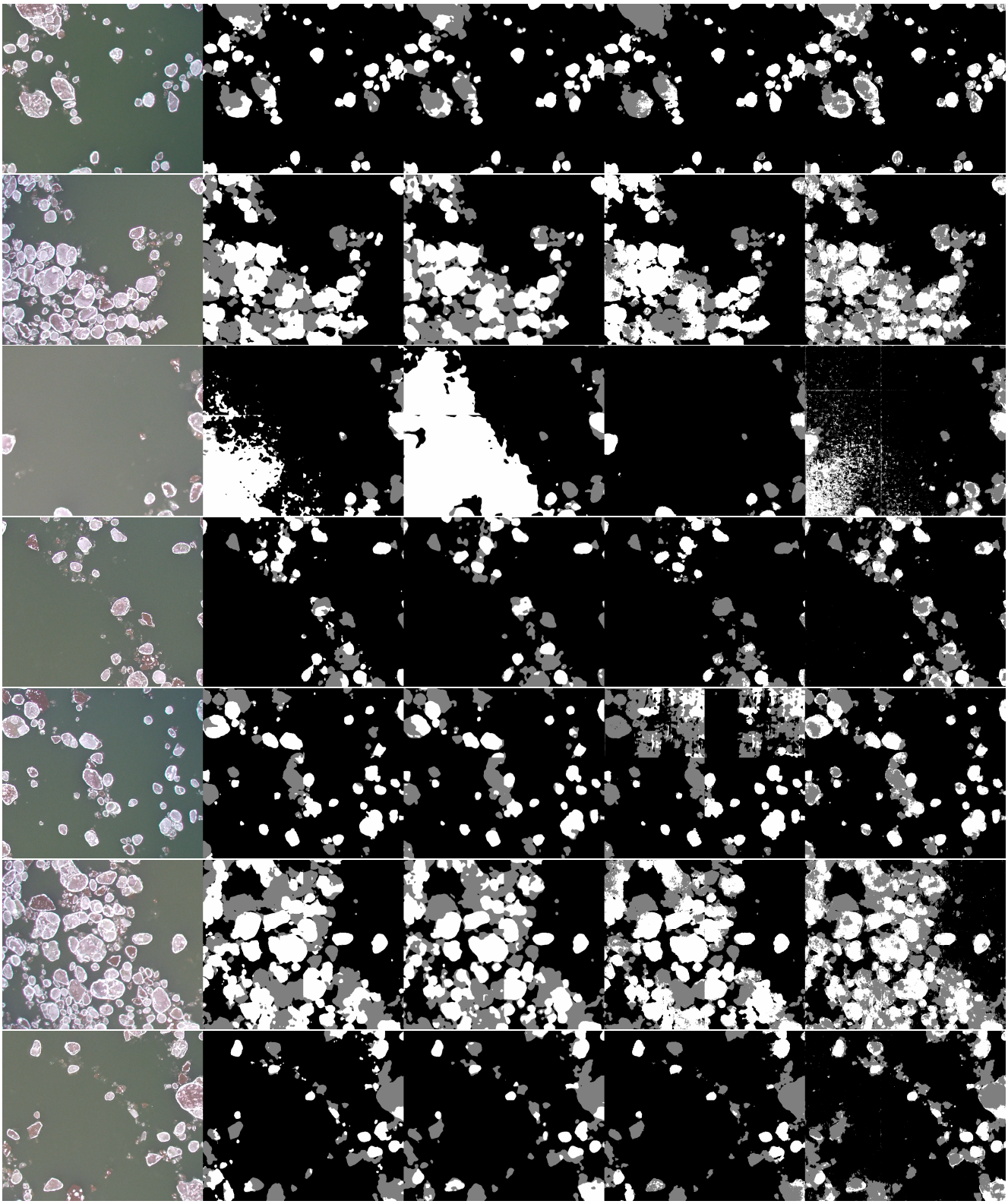

Figure 6. Results of applying the best configurations of the four models on test images: left to right: UNet, SegNet, Deeplab, DenseNet

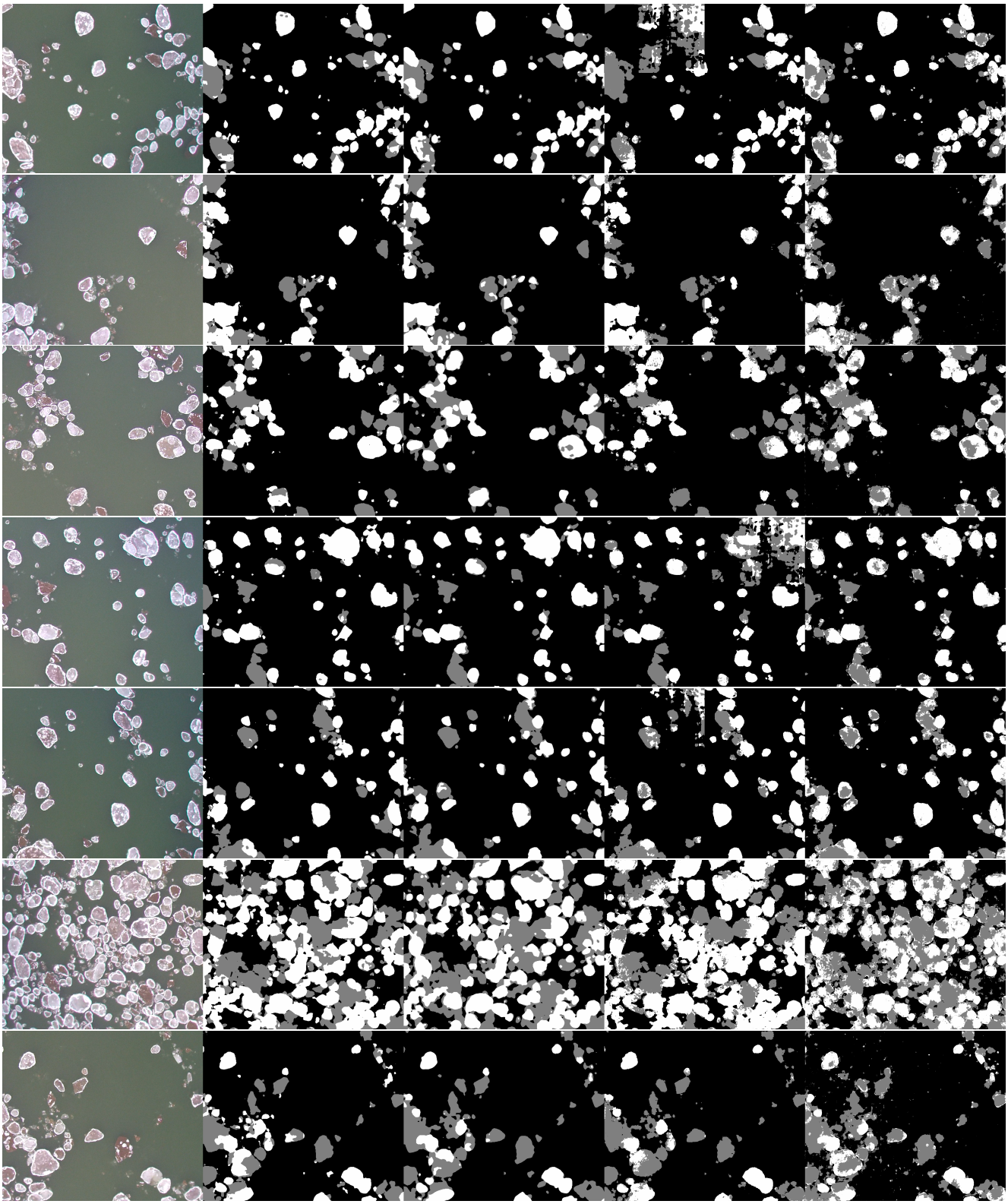

Figure 7. Results of applying the best configurations of the four models on test images: left to right: UNet, SegNet, Deeplab, DenseNet

[20] Y. Li and L. Shen. A deep residual inception network for hep-2 cell classification. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 12–20. Springer International Publishing, 2017. 1

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440. IEEE Computer Society, 2015. 2

[22] mrgloom. Awesome Semantic Segmentation. Github, 2017. https://github.com/mrgloom/awesome-semantic-segmentation. 2

[23] D. Parthasarathy. Classifying white blood cells with deep learning. online: https://blog.athelas.com/classifying-white-blood-cells-with-convolutional-neural-networks-2ca6da239331, March 2017. 1

[24] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *CVPR*, 2017. 2

[25] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI (3)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. 2

[26] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017. 2, 3

[27] A. Shpilman, D. Boikiy, M. Polyakova, D. Kudenko, A. Burakov, and E. Nadezhdina. Deep learning of cell classification using microscope images of intracellular microtubule networks. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1–6, Dec 2017. 1

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2

[29] Y.-H. Tsai, M.-H. Yang, and M. J. Black. Video segmentation via object flow. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3899–3908, 2016. 2

[30] L. Zhang, L. Lu, I. Nogues, R. M. Summers, S. Liu, and J. Yao. Deeppap: Deep convolutional networks for cervical cell classification. *IEEE Journal of Biomedical and Health Informatics*, 21(6):1633–1643, Nov 2017. 1