First, each methods used is described in section 1, and the results of experiments with each methods, as well as the results of experiments using combinations of these methods, are discussed in section 2.

1 Methods

1.1 Image processing

1.1.1 Edge detection

Edge detection could be used to detect ice leads.

Canny edge detection [1] is a popular edge detection algorithm. It consists of 4 steps:

1. Noise reduction
2. Calculate image gradient
3. Detect edges
4. Sanitize edges

Noise reduction First, the image is smoothed using a Gaussian filter. This removes the noise from the image, which may affect the noise detection.

Calculate image gradient Next, the image is filetered in both horizontal and vertical direction with a Sobel kernel to obtain its first derivative. Then, gradient and its direction can be calculated as follows where G_x and G_y are the first derivative of horizontal and vertical direction respectively:

$$\begin{aligned} \text{Gradient} &= \sqrt{G_x^2 + G_y^2} \\ \text{Direction} &= \arctan \frac{G_y}{G_x} \end{aligned}$$

The direction is rounded to one of the four directions we can handle on an image for the next step.

Detect edges An "edge" can be defined as a collection of pixels whose gradient is local maximum. Thus, given a pixel, compare its gradient with two adjacent pixels in the gradient direction, and if the pixel has the largest gradient, it can be said that the pixel constitutes an edge. By applying this calculation to all pixels, we obtain a binary image with "edges".

Sanitize edges Given two thresholds $t_{\max}$ and $t_{\min}$ ($t_{\max} > t_{\min}$), an edge can be judged if it is a "sure" edge or not. At a pixel constituting an edge, if the gradient is above $t_{\max}$, then the pixel is determined as a "sure" edge. Conversely, if the gradient is below $t_{\min}$, the pixel will be discarded. For pixels whose its gradient is between two thresholds, the pixel can be said as a "sure" edge if it is connected to a "sure" edge, and if it's not, it will be also discarded.

1.1.2 Blur detection

Since satellite images contain a certain amount of clouds and its shadow, blur detection could be used to remove them as a preprocessing.

Using Laplacian kernel and calculating its variance, an image can be determined if it is blurred or not [2]. Such Laplacian kernel is expressed as follow:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

The kernel highlights rapid intensity changes, thus if its variance is small than given threshold, it means the image contains less intensity changes, which means it is "blurry".

This calculation is applied to subdivided regions of the original image and if the region is "blurry", the region is determined to contain cloud (or shadow).

1.2 Unsupervised methods

1.2.1 K-means clustering

K-means clustering [3] is a popular clustering algorithm. The goal is to classify all given data into K clusters. The algorithm can be written as follows:

1. Assign each data to random cluster.
2. Calculate center of each cluster.
3. Reassign each data to the cluster with the closest center.
4. Repeat 2 and 3 until the convergence.

1.2.2 Random forest classifier

Random forest classifier [4] is a ensemble classifying algorithm. It consists of decision trees each trained on sub-divided data. The idea here is to take average of a certain number of decision trees over-trained in diferent data respectively.

1.2.3 Autoencoder

An important aspect of data classification is the extraction of their features. An autoencoder [5] is often used to extract features. It consists of an encoder and a decoder, which the encoder is trained to extract features as its output, and the decoder is trained to reconstruct the input from the output of the encoder.

By training the autoencoder to reduce the difference between the input and output, it can be said that the output of the encoder is the features necessary to reconstruct the original input.

1.3 Semi-supervised methods

1.3.1 Label spreading

Label spreading [6] is semi-supervised classifying algorithm. Given data with labels and without labels, the goal is to label all unlabelled data based on the labelled data.

For a given data, the algorithm constructs a fully connected graph with each datum as nodes, and the similarity between data as edges. Then, the labels are spread to all unlabelled nodes from labelled nodes. Additionally, the algorithm has a parameter $\alpha \in (0, 1)$, which indicates the uncertainty of given labels. The algorithm replaces the label of labelled data with probability α while spreading the label.

2 Experiments

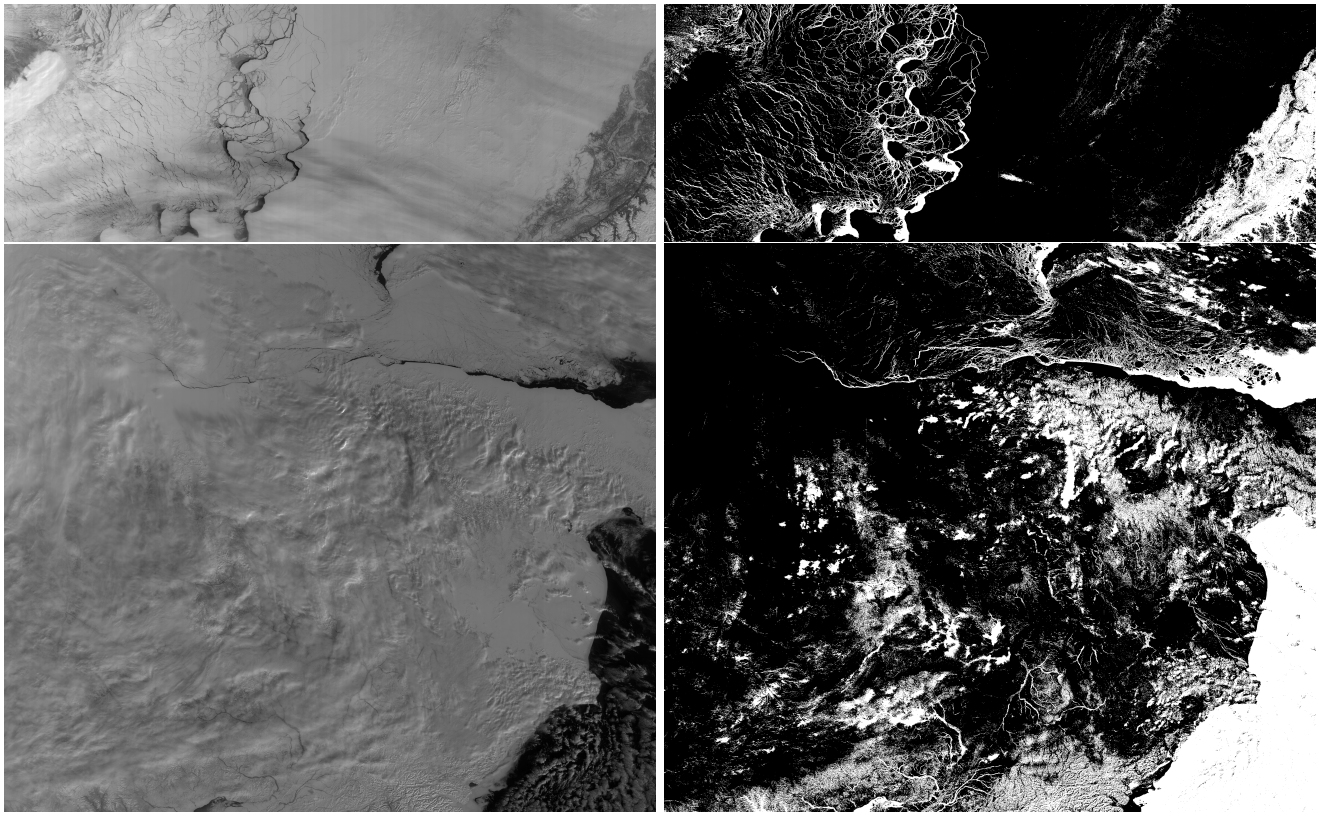
2.1 Dataset

In the experiments described later, 1×1 and 3×3 pixels cropped from the original image are used as data. For some methods which use patched data, 128×128 patches extracted from the images are used. Since the original image has 21 channels, the data shape is $1 \times 1 \times 21$ or $3 \times 3 \times 21$. As training and validating data, 1×1 and 3×3 data labelled as whether the pixel (for 3×3 datum, its center pixel) is a lead or not, are used.

As an important fact, as shown later, the accuracy of a model for validating data does not directly represent its usefulness. The reason for this is thought to be the difference of their probability space between labelled data and (unlabelled) image data, which the former is not collected from an whole labelled image, but the labels are assigned to the pixels extracted from unlabelled image.

2.2 Prior experiment

Using labelled dataset, supervised training has done to a model based on Vision Transformer [7]. It performed accuracy of 0.8395, and its rollout is shown in Figure 1. Although the accuracy inferior to other methods as shown later, this method is likely the most effective when looking at rollout.



(a) (b)

Figure 1: Prior experiments results. (a) Channel 1 of the original image. (b) Rollout of Vision Transformer model.

2.3 Edge detection

As two thresholds, $(t_{\max}, t_{\min}) = (0, 96)$ is used. The result is shown in Figure 2. We can see that the result is similar to the rollout of prior experiment.

2.4 Blur detection

Images are subdivided into 128×128 pixels, and the variance of the Laplacian is calculated for each region. The original image and result are shown in Figure 3.

We can see that regions with clouds are determined blurry, but especially in the bottom image, region with lead also has a certain value of the variance, which leads the conclusion that the method is severe to be used for cloud removal.

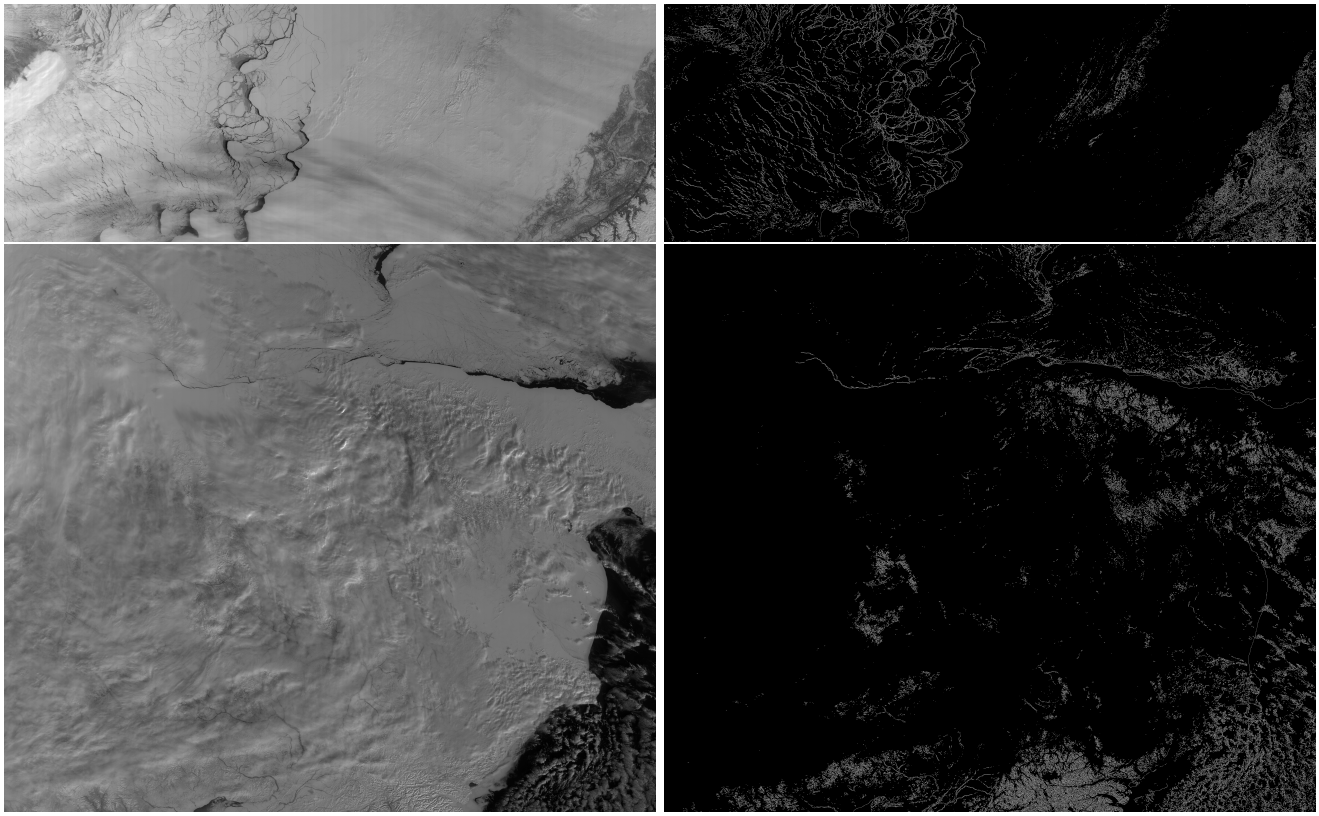
2.5 K-means clustering

K-means clustering with $K = 8$ is done with sampled data both 1×1 and 3×3 size. Considering the 2 "darkest" clusters as the lead, the model performed accuracy of 0.7498 training with 1×1 data, and 0.7893 with 3×3 data. The rollouts of model trained with 3×3 data are shown in Figure 4.

2.5.1 Patched K-means clustering

The K-means clustering can be more efficient by not clustering the whole image, but patched image.

In result, the model was susceptible of clouds, Although more accurate for smaller leads. This method should be considered combining with a cloud removal processing. The rollouts of patched K-means clustering are also shown in Figure 4.



(a) (b)

Figure 2: Edge detection results. (a) Channel 1 of the original image. (b) Detected edges.

2.6 Label spreading

2.6.1 Using raw data

Label spreading is done with sampled 1×1 data. The model performed accuracy of 0.7127.

2.6.2 Using autoencoder

Since the data has 21 channels, which may be too large to classify, an autoencoder is trained to extract features, and apply label spreading with the features. The model performed accuracy of 0.8321 with 1×1 data, and 0.8943 with 3×3 data. The rollouts of model trained with 3×3 data are shown in Figure 5.

2.7 Random forest

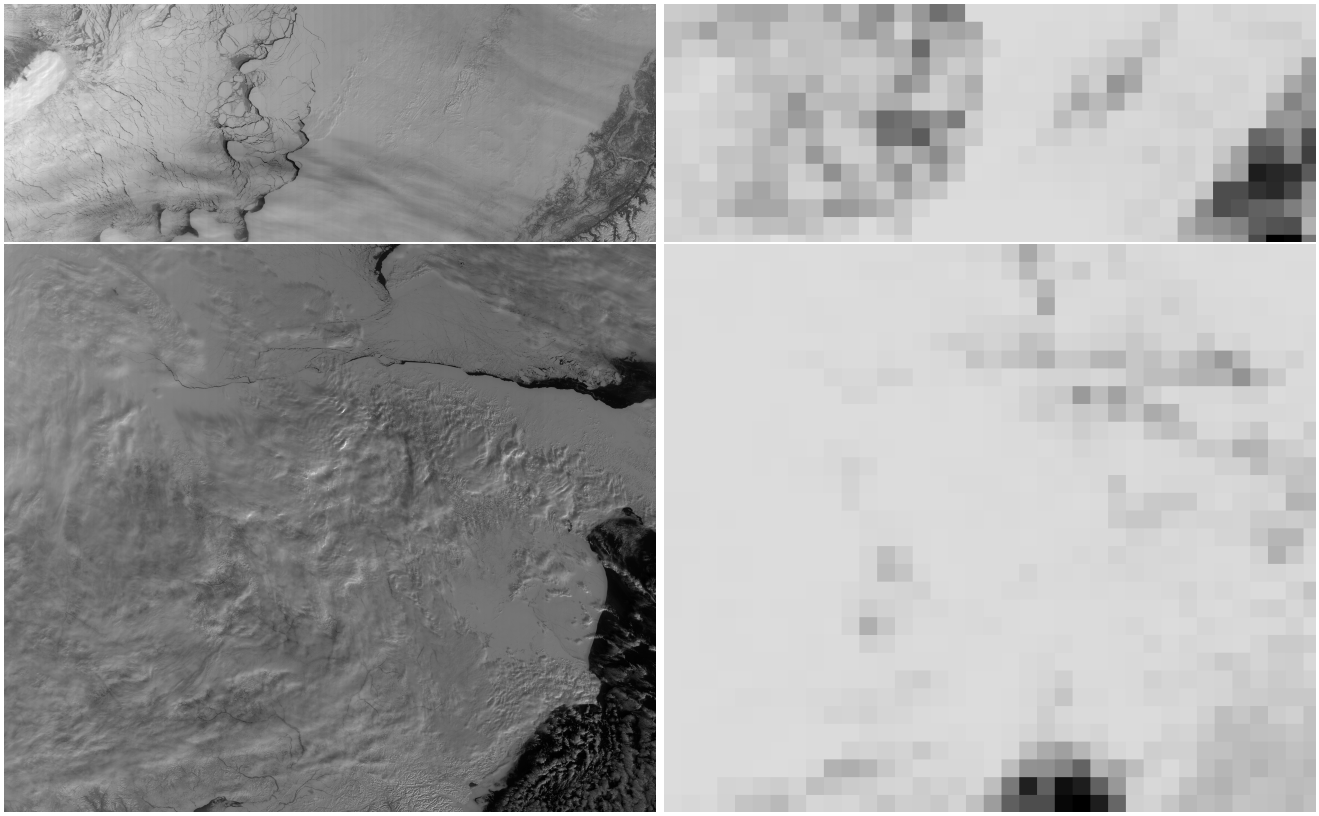
Random forest classifier with 20 ensembles is trained with both 1×1 and 3×3 size data. The model performed accuracy of 0.9244 trained with 1×1 data, and 0.9478 with 3×3 data. We considered using features gained from autoencoder just as we did when using the label spreading method, but no improvement was seen. The rollouts of model trained with 3×3 data are shown in Figure 6

2.8 Autoencoder with supervised training

2.9 Unsupervised segmentation

References

[1] J. Canny, "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986



(a) (b)

Figure 3: Blur detection results. (a) Channel 1 of the original image. (b) Variance of the Laplacian by region. The brighter the region, the more blurred it is.

Table 1: Accuracy of models

	1×1 data	3×3 data
Vision Transformer	-	0.8395
K-means	0.7498	0.7893
Label spreading	0.8321	0.8943
Random forest	0.9244	0.9478

[2] R. Bansal, G. Raj and T. Choudhury, "Blur image detection using Laplacian operator and Open-CV", International Conference System Modeling & Advancement in Research Trends, 2016

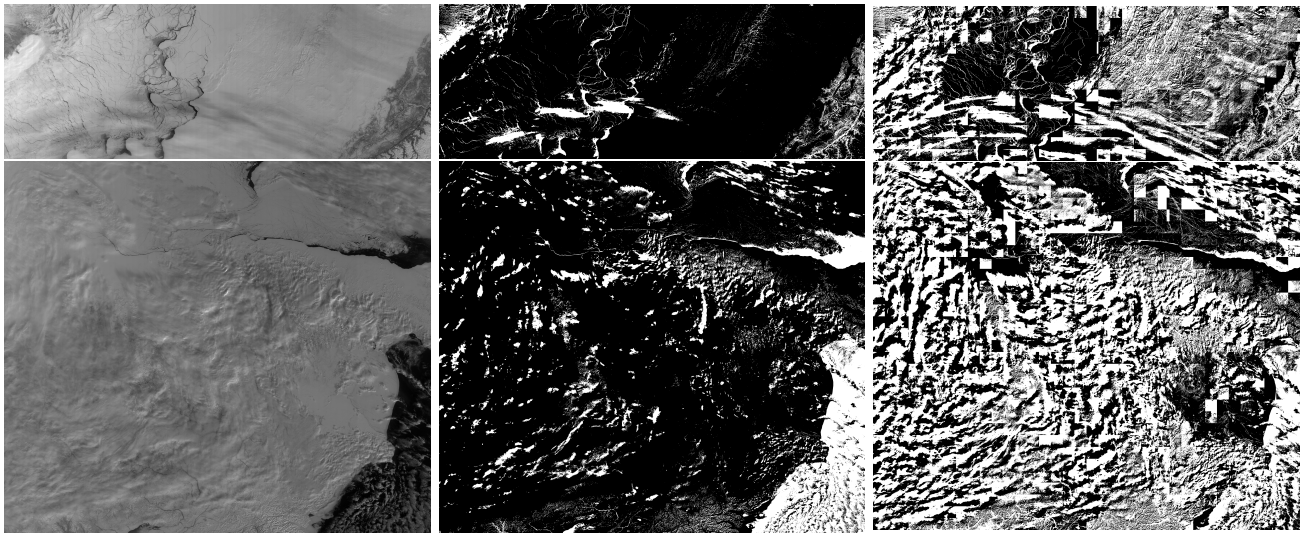
[3] G. Ball and D. Hall, "ISODATA, a Novel Method of Data Analysis and Pattern Classification", Stanford Research Institute, 1965

[4] L. Breiman, "Random Forests", Machine Learning, 2001

[5] M. Kramer, "Nonlinear principal component analysis using autoassociative neural networks", AIChE Journal, 1991

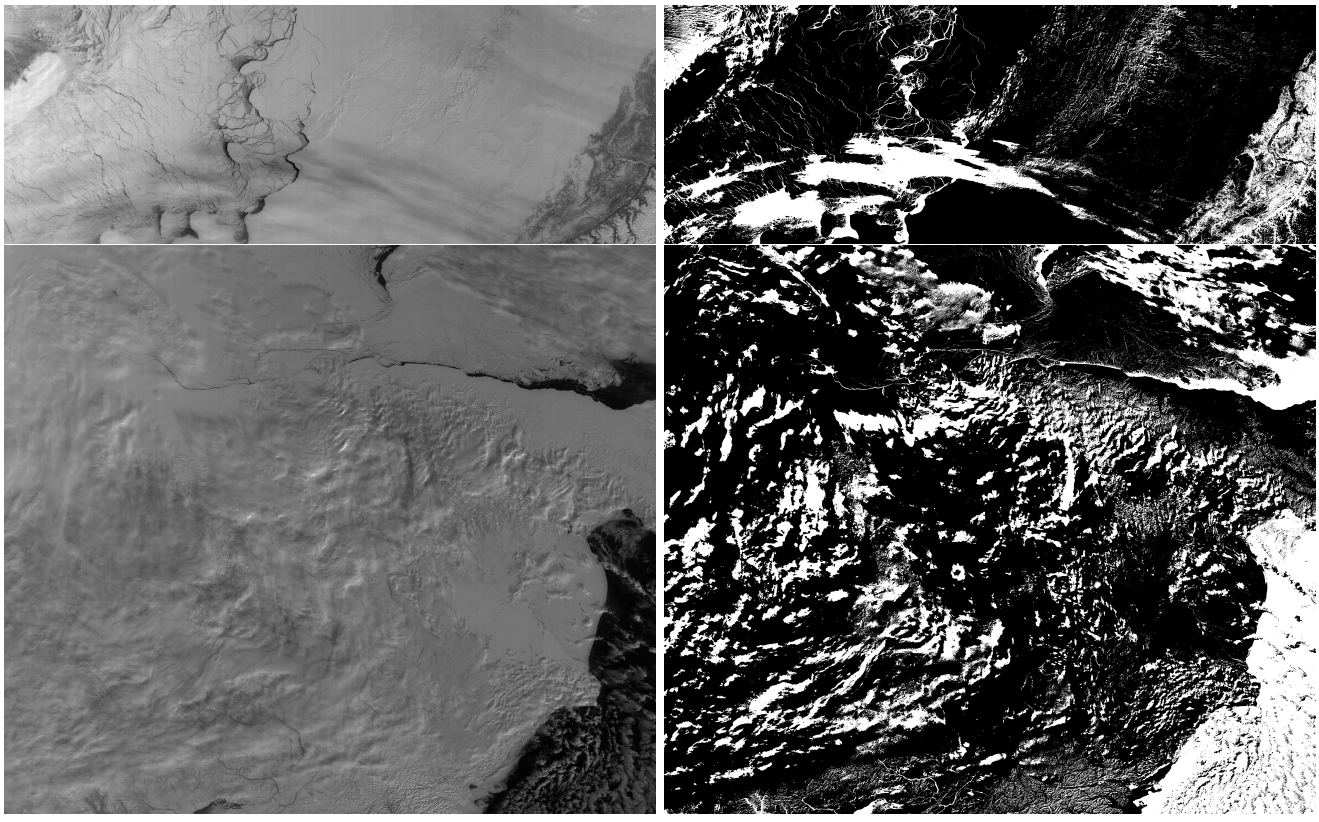
[6] D. Zhou, O. Bousquet, T. Lal, J. Weston and B. Schoelkopf, "Learning with local and global consistency", Advances in Neural Information Processing Systems, 2004

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", International Conference on Learning Representations, 2021



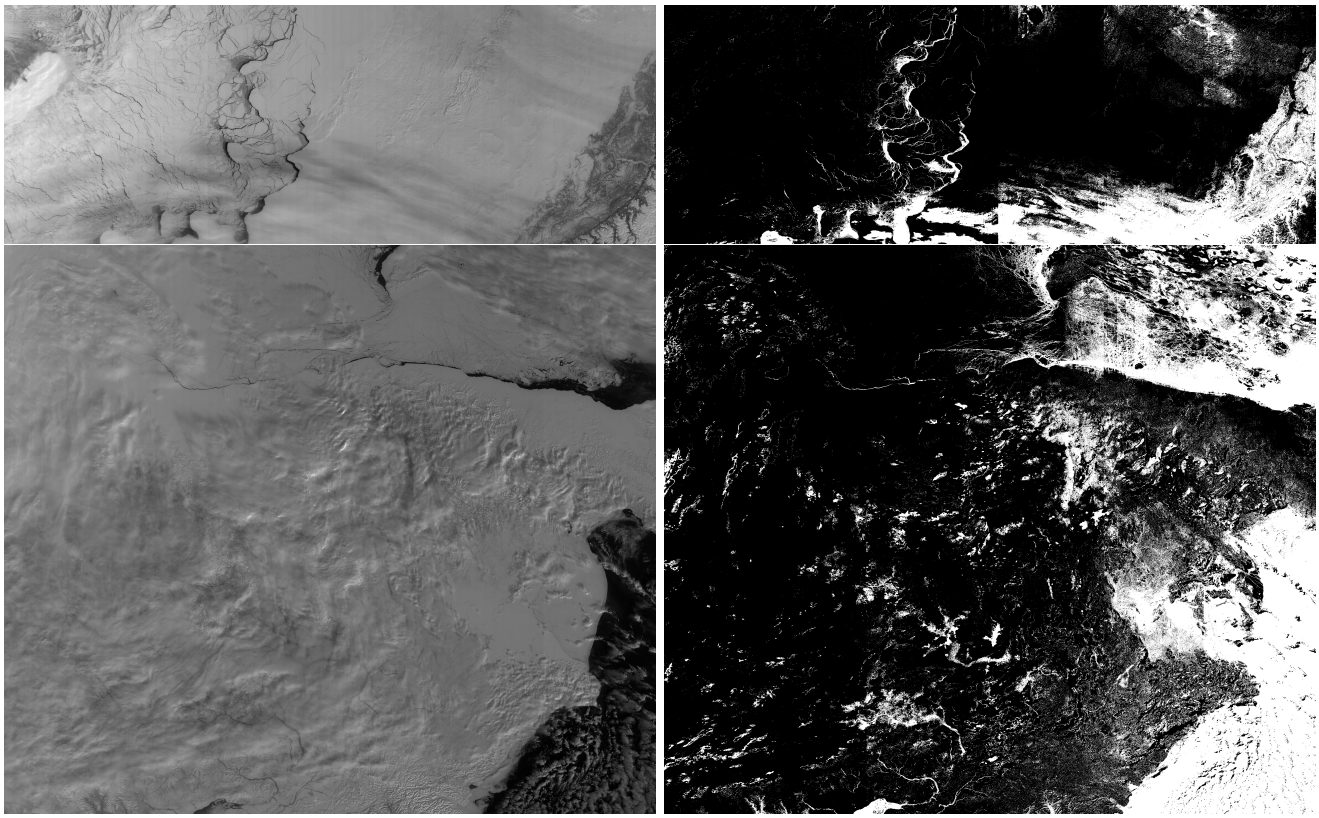
(a) (b) (c)

Figure 4: K-means clustering results. (a) Channel 1 of the original image. (b) Rollout of K-means clustering classifier. (c) Rollout of patched K-means clustering.



(a) (b)

Figure 5: Label spreading results. (a) Channel 1 of the original image. (b) Rollout of label spreading.



(a) (b)

Figure 6: Random Forest clustering results. (a) Channel 1 of the original image. (b) Rollout of Random Forest classifier.