Understanding Clouds from Satellite Images
Deep learning Project

Klemen Škrlj (63180282)

Abstract

In our project we look at a problem of cloud classification and segmentation. Our goal is to extract which type of clouds are present on a satellite image. To do this we compare popular deep learning approaches for segmentation like U-Net, FPN and DeepLabV3+. Additionally we test out some ideas which are normally used on such problems like: test time augmentation, classifiers as false positive detectors, grid based threshold search etc. We compare this methods based on Dice coefficient and come to the conclusion that original U-Net architecture with ResNet-50 as a backbone performs the best with Dice=0.654.

1 Introduction

Cloud is a visible mass of condensed water vapor in the atmosphere typically floating high above the general level of the ground. Clouds play an important role in the field of meteorology such as predicting and analysing cyclone, hurricane, thunderstorm, tornadoes, weather, climate etc. National Aeronautics and Space Administration's ICESat satellite proves that 70% of this world is covered by clouds. Shallow clouds especially play a huge role in determining the Earth's climate. They're also difficult to understand and to represent in climate models. By classifying different types of cloud organization, researchers at Max Planck or other institutes hope to improve our physical understanding of these clouds, which in turn will help us build better climate models. On the other hand, clouds become an obstacle when earth surface study is the objective of satellite images. Clouds on the input image are treated as noise that is why detection and removal of cloud from satellite image are an important preprocessing phase on most of the applications in remote sensing. The large size of high-resolution satellite images, as well as the tedious nature of pixel-scale manual annotations, have been strong motivators for the development of fully automatic algorithms which aim to accurately and reliably detect clouds in satellite imagery.

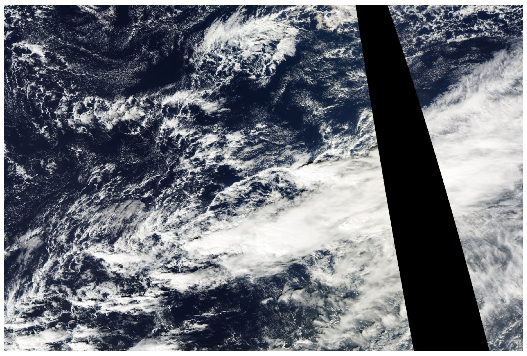
Figure 1: Satellite image of clouds

2 Related word

Similarly to other computer vision tasks, the beginnings in this domain were done with traditional methods. Survey [1] states that spectral and textural features play an important role for could classification. The major challenge of cloud detection approaches is high miss-classification rate of cloud pixels due to lower contrast of cloud edges against the land or sea background. A posteriori probability–Markov random field method shows improvements in that regard. Article [2] suggests a two stage segmentation. In the first stage, cloud candidates are determined by thresholding and k-clustering. Then, using local threshold, they are classified in one of three classes: thick clouds, thin clouds or ground. Researchers in paper [3] also separate problem into those three classes. Instead of thresholding, they divide a satellite image into superpixels and closely evaluate some of the brightest ones. They extract Gabor features from each on of those selected superpixels and feed them into SVM model which predicts if patch is a cloud or snow/building.

With more and more image data available from many different satellites the task of cloud segmentation and classification is mainly being solved with deep learning models nowadays. Article [4] shows that a simple CNN with 6 layers and a classification head already greatly outperforms traditional models like SVM. Predictions of this network show superior stability and robustness to different could systems. After that researchers in paper [5] used U-Net architecture for cloud segmentation task. The results show high accuracy of such method and also robustness to noisy data in a wide range of terrains. Similar work is also done in article [6] where a U-net arhitecture is trained. In addition they also show that adding an infrared channel to RGB image can be usefull for better segmentation results.

3 Methods

In this section we present our dataset and state some key features about it. We describe different deep-learning architectures that we use to solve the problem of cloud segmentation on satellite images. Additionally some approaches to make better predictions are listed and the background behind those ideas is described.

3.1 Data

The dataset consists out of 9244 satellite images where 5546 are used for training and 3698 are used for testing purposes. Each one of them contains at least one of the annotated cloud structures: Fish, Flower, Sugar and Gravel. True color image was generated from satellite images from two polar-orbiting satellites. Due to the small footprint of the imager on board these satellites, an image might be stitched together from two orbits. The remaining area, which has not been covered by two succeeding orbits, is marked black. Original size for the images is 1400 by 2100 pixels.

Labeling was performed by a team of 68 scientists from Max-Planck-Institite for Meteorology in Hamburg, Germany and Laboratoire de météorologie dynamique in Paris, France. Since every image was masked by at least 3 scientist, the final ground truth was generated by a union of all masks. An example of input image and annotated masks is show in Figure 2.

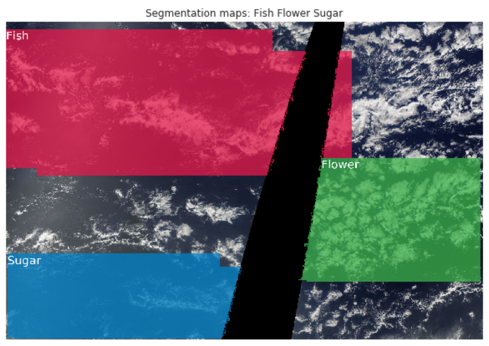

Figure 2: Train example with annotated masks

When looking at a distribution of classes though the train dataset, we see that we are working with a pretty balanced dataset. Sugar cloud formation is the most frequent (32%) while flower formation occurs the lest amount of times (in 20% of train images). A bigger discrepancy however comes from number of formations that are present on each image. Looking at Figure 3 we see that there isn't a lot of train examples that have all 4 classes. The most common combinations are Sugar-Gravel and Sugar-Fish pairs.

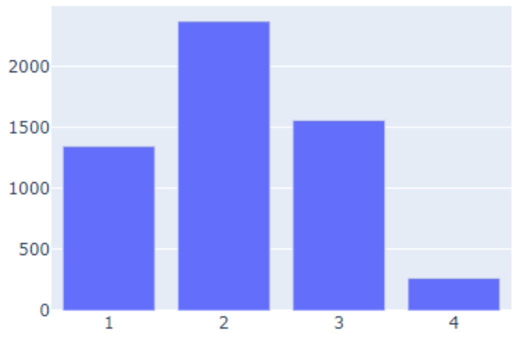

Figure 3: Distribution of number of cloud formations per image

3.2 Approaches

Arhitectures

We define our problem as semantic segmentation problem where we have an RGB image as an input and segmentation mask as an output. Since there can be overlapping masks in ground truth (one pixel can be part of several classes), we have to generate 4 separate masks.

Similarly to newer articles in related work, we too use U-Net[7] architecture at first. This model consists of encoder and decoder part. The encoder block is a classical contracting CNN which is responsible for gradual extraction of deep features from input image. This features are then inputed into a decoder which performs up-sampling with several transposed convolution operations. Since the up-sampling is done from semantically rich but spatially small features we can't generate back original image in as high resolution. To combat this, the network has skip connections. At every decoder block we merge together volume from previous decoder block and volume from encoder block at same level. Such architecture helps with sharper edges in a final segmentation mask.

As our next approach we use Feature Pyramid Network architecture (FPN)[8]. Originally the authors proposed this model for object detection task, especially to combat poor performance with detections of small objects. Instead of performing predictions on same image at different scales, the idea is to create a pyramid of features and use them as a help for better detection. FPN is composed of bottom-up and top-down pathways. The idea is pretty similar to U-Net where a bottom-up part is trained to extract deep features from the input and the top-down up-samples spatially coarser, but semantically stronger features maps from higher pyramid levels. Key difference between this and U-Net architecture is that here the model generates segmented mask at each pyramid level going down. The smaller masks are then resized and merged into final predictions. Unlike in U-Net where we merge volumes from encoder and decoder, we here perform 1x1 convolution on encoder volumes and then add them as a new channel to decoder part.

Our final segmentation arhitecutre is DeepLabV3+[9]. Main key-points of the model are Atrous Spatial Pyramid Pooling (ASPP) and encoder-decoder architecture. Atrous convolution means that instead of performing convolution over immediate neighbours, we can take neighbours of neighbours (depending on the atrous rate). To capture bigger area they propose pyramid pooling where various atrous rates are computed at same time and then the results are merged together. To make computation quicker they use depth-wise separable convolution. The decoder takes low level features and up-sampled output of the encoder as an input to perform final convolution and produce output mask. The importance of the decoder is seen especially on the edges of the masks.

In all network architectures that we use, we have an encoder part which is also called backbone. Since this part is independent of the decoder, we can use different architectures for feature extraction. In our project we first use ResNet-50[10]. This is a popular backbone in computer vision applications. The main features of this network are residul blocks. Instead of having connections only between pair of subsequent blocks we add additional skip connections. This way we create two paths where one puts the input vector through convolutions and activation functions, while the other just skips this computation. The authors show that the use of such blocks greatly improve the problem of vanishing gradients and thus allow deeper networks to be trained. Our variant has 50 residual block which also incorporate bottleneck. Instead of performing convolutions on full volumes, we first perform 1x1 convolution to reduce dimensionality, followed by original convolution and again 1x1 convolution to increase the dimensions back to original. With this the training is less computationally intensive.

We also use EfficientNet-B1[11] as our backbone. Here authors systematically show that increase in depth, width and resolution in compound manner yields better results. They also use mobile inverted bottleneck as their main building block, which were first presented in MobileNetV2[12], and squeeze-and-excitation optimization from [13]. With this improvements we get a model that has almost five times less parameters and higher accuracy than ResNet-50.

Lastly we use Dense-Net-121[14] as our classifier. This network has a similar idea as ResNet, but instead of connecting skip connections only between 2-3 convolutional layers we make connections from any layer to all subsequent layers inside dense block. This allows even better gradient flow and implicit deep supervision.

Loss function
During training of U-Net and FPN model we use combination of Binary cross-entropy and Dice loss. Binary cross-entropy calculates the score that penalizes the probabilities based on the distance from the expected value. Dice coefficient on the other hand is a popular metric for evaluating segmentation in computer vision. It is calculated by Equation 1.

$$Dice = \frac{2 \cdot |X \cap Y|}{|X| + |Y|} \tag{1}$$

Since the coefficient measures relative overlap between ground truth and predictions it is good for handling imbalance in data. This is present in our case because we have many pixels in background and not many classified as clouds. To convert Dice coefficient into loss we take 1-Dice. Our final loss function is stated in Equation 2.

$$Loss = 0.75 \cdot BCE + 0.25 \cdot (1 - Dice) \tag{2}$$

We also tried training with Focal loss, which is good with imbalanced data and in addition optimizes the learning for hard examples, but the results were not as good.

Classifier
From our data exploration we see that rarely there are all 4 cloud structures present in the same image. This means that out of 4 masks that the segmentation model generates there should be non trivial amount of those that don't contain anything. But segmentation network is learned for both empty and non-empty predictions so it is hard for it to produce completely black masks. To try to help with final prediction we can have another network that is specialized only in classification. The input to such model is a satellite image and the output is a vector of 4 probabilities, one for each cloud structure. We separately train three different model architectures for this task: ResNet-34, Efficient-Net-B0 and DenseNet-121. We use Binary cross-entropy loss for all models during training. We hypothesise that the use of different ideas would benefit the accuracy of final ensemble model, where we employ majority voting scheme.

4 Results

We split our training data into train and validation with 9:1 ratio for segmentation models and train them for 20 epochs. For ResNet and EfficientNet we use batch size of 16 while for DeepLabV3+ batch size of 8 is used due to computational intensity. All of the backbones are pre-trained on ImageNet, which helps with lower training time for our specific task. Because of this we use 1e-4 learning rate for encoder part and 1e-3 for decoder part of the network. As our optimizer we use Adam and ReduceLRonPlateau as our learning rate scheduler.

As for classifier we train with 8:2 test/validation split for 20 epochs with batch size of 16. Here learning rate of 1e-4 is used for the whole model, while other parameters (optimizer, scheduler) are the same.

To make our models more general and generate more train data we perform data augmentation on the train dataset. For this we use horizontal and vertical flips, random shifts, scales and rotations and grid distortion all with probability of 50%. This are all augmentations that happen also in real world so the train distribution is not skewed.

4.1 Analysis

The output of segmentation model is non-binarized and to generate a final mask we have to decide on a threshold. A naive approach is to set it to globally to some number (normally 0.5). But since we have a validation set we can use it to perform grid search on threshold for binarization and threshold for smallest group of pixels, that are still considered valid object. In this fashion we compute pairs of tresholds over whole validation set for each type of cloud formation separately. In Table1 we see comparison of results from U-Net model with ResNet-50 backbone with global threshold and adaptive one. The adaptive threshold yields much better results which is expected since every cloud structure has its own characteristics in terms of size and transparency which affects segmentation.

Model	Dice
U-Net$_{t=0.5}$	0.544
U-Net$_{t=adapt}$	**0.654**

Table 1: Comparison between global and adaptive threshold

Table 2 show comparison between different model architectures and backbones. Here we do not test all configurations since the training time for such models is quite large (approximately 6 to 7 hours per model). U-Net with ResNet-50 backbone performs the best on a hidden test dataset. Even though we use the same architecture with EfficientNet backbone this experiment didn't perform as good. From this we can deduce that residual blocks help in our domain for better feature extraction. We also see that DeepLabV3+ performs better than FPN model. Since the clouds are usually quite large structures on a satellite images we don't benefit from FPN architecture, which was designed to help with small object detection/segmentation.

Arhitecture	Backbone	Dice
U-Net	**ResNet-50**	**0.654**
U-Net	EfficientNet-B1	0.634
FPN	ResNet-50	0.639
DeepLabV3+	ResNet-50	0.649

Table 2: Results of different architectures and backbones

The original images in the dataset are 1400 by 2100 pixels in size but the input to our models (U-Net, FPN and DeepLabv3+) is much smaller (350 by 525 in our case). We explore if the segmentation output is better if we try to extract more features from original image size. To do this we add additional convolutional blocks before the input gets to the backbone (ResNet-50) of U-Net as seen in Figure 4. By doing this the image gets gradually spatially smaller but has richer visual features.

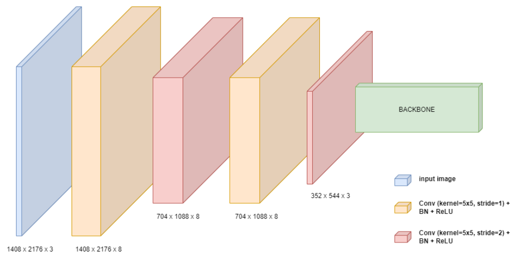

Figure 4: Additional convolutional blocks for feature extraction

From results in Table 3 we see that the overall result on test dataset didn't get any better. One of the reasons for this could be that we are using pretrained backbone. The weights were trained for extracting features from real world images but our new input is already composed of richer, unusual structures. Because of this the feature extraction might not work as well.

Model version	Dice
Custom resize head	0.638
No resize head	**0.654**

Table 3: Effect of custom resize head

As mentioned before we use data augmentation techniques during our training which helps to generate more training examples virtually for free while also boosting model robustness. But papers like [15] and [16] also use some data augmentation (horizontal, vertical flips and cropping) in test time (also called Test Time Augmentation-TTA). We essentially put the same test input many times through the model and use different augmentation each time. Then an average prediction is taken and binarized into final prediction. In our case we do vertical and horizontal flips. Our comparison also includes usage of TTA on validation data, since there we search for optimal size and value threshold. U-Net with ResNet-50 backbone is used as our baseline model. As seen in Table 4, it is best to use TTA both on validation and test set in our case. Comparing original model with the one that performs TTA we get only slight improvement on lower decimal places. This means that our original model is by itself robust enough and doesn't benefit from merging multiple predictions from augmentations.

Model version	Dice
TTA on validation+test	**0.654**
TTA on test	0.643
No TTA	0.654

Table 4: Effect of TTA

In Table 5 we see F1 scores of our classifiers for each class separately. We perform a threshold search by calculating F1 on all steps of 0.05 between 0 and 1 while optimizing for final score. Results show that no classifier is a clear winner in every category.

Classifier	F1			
	Fish	Flower	Gravel	Sugar
ResNet-34	0.718	0.810	0.728	**0.846**
EfficientNet-B0	0.708	**0.829**	0.728	0.840
DenseNet-121	**0.719**	0.803	**0.734**	0.844

Table 5: Results of different classifiers

We try using every one of those classifiers alone and compare their results. Based on results in Table 5 we also use all classifiers as an ensemble in a majority vote fashion in order to eliminate false positive segmentation masks. Our baseline model is again U-Net with ResNet-50. As seen from the results in Table 6 this method is not beneficial. The classifiers aren't good enough to correctly predict if a cloud structure is or isn't present on the image. For every false negative prediction of the classifier we are greatly punished because we remove the whole mask, even though it might contain true positive predictions. We also try ensemble of different classifiers and see that the results do not improve.

Model version	Dice
ResNet-34 cls[1]	0.510
EfficientNet-B0 cls	0.495
DenseNet-121 cls	0.510
Classifier ensemble	0.510
No classifiers	**0.654**

Table 6: Effect of classifiers

We further explore the idea of using multiple models in ensemble fashion with segmentation models. We take our three best performing models (U-Net ResNet-50, FPN ResNet-50 and DeepLabV3+ ResNet-50), generate binary mask from each one of their predictions and take per pixel majority vote for final output. We also add a classifier ensemble to potentially improve the results. Table 7 shows results of our tests. We again see the problem of classifier false negative predictions which limit segmentation performance. With ensemble of all three segmentation models we don't see an improvement. The results are comparable with our best model. From this we can deduce that all three models perform relatively similar. If a pair of models would be better in some cases and another pair better in other cases, then this pairs would show up in ensemble method and produce better results. But this isn't true in our case.

Model version	Dice
Segmentation ensemble	0.652
Segmentation ensemble + Classifier ensemble	0.510
No ensemble	**0.654**

Table 7: Effect of segmentation ensemble

5 Conclusion

Cloud formation segmentation and classification from satellite images is an important task for improving weather prediction models and keeping track of the environment. We look at some of the popular deep learning approaches for solving this task like U-Net, FPN and DeepLabV3+ models. We compare the effect of adaptive threshold searching, model's backbones, test time augmentations, classifiers as a false positive filter and custom resizing modules. We evaluate them in terms of Dice coefficient, which is computed on a hidden test set. Our findings show that the original U-Net architecture with ResNet-50 backbone outperforms all other approaches.

References

[1] G. C. A. J and C. Jojy, "A survey of cloud detection techniques for satellite images," *IRJET*, vol. 2, no. 9, Dec 2015.

[2] I.-H. Lee and M. T. Mahmood, "Robust registration of cloudy satellite images using two-step segmentation," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 5, pp. 1121–1125, 2015.

[3] C. Deng, Z. Li, W. Wang, S. Wang, L. Tang, and A. C. Bovik, "Cloud detection in satellite images based on natural scene statistics and gabor features," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 4, pp. 608–612, 2019.

[4] K. Cai and H. Wang, "Cloud classification of satellite image based on convolutional neural networks," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2017, pp. 874–877.

[5] A. Francis, P. Sidiropoulos, and J.-P. Muller, "CloudFCN: Accurate and robust cloud detection for satellite imagery with deep learning," *Remote Sensing*, vol. 11, no. 19, p. 2312, Oct. 2019. [Online]. Available: https://doi.org/10.3390/rs11192312

[6] J. H. Jeppesen, R. H. Jacobsen, F. Inceoglu, and T. S. Toftegaard, "A cloud detection algorithm for satellite imagery based on deep learning," *Remote Sensing of Environment*, vol. 229, pp. 247–259, Aug. 2019. [Online]. Available: https://doi.org/10.1016/j.rse.2019.03.039

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[8] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: http://arxiv.org/abs/1612.03144

[9] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. [Online]. Available: http://arxiv.org/abs/1802.02611

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[11] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019. [Online]. Available: http://arxiv.org/abs/1905.11946

[12] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[13] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *CoRR*, vol. abs/1709.01507, 2017. [Online]. Available: http://arxiv.org/abs/1709.01507

[1]Abbreviation for classifier

[14] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567