Cloud Classification Project

Bryce Bodley-Gomes
College of Information and Computer Science
University of Massachusetts at Amherst
bbodleygomes@umass.edu

Kaivankumar
College of Information and Computer Science
University of Massachusetts at Amherst
kaivankumars@umass.edu

ABSTRACT

Proposed as a Kaggle challenge was the idea to automatically classify regions of satellite images in which clouds of type 1 through 4 dominate. These regions were then segmented in an image, to enable scientists to easily analyze developing cloud patterns in regions around the world. Our objective was to build a model and analyze why current results on this task performed so poorly.

KEYWORDS

Machine Learning, Automated Classification,

1. Introduction

Shallow clouds play a huge role in determining the Earth's climate. Classifying different types of cloud organization, will help scientists build better climate models.

There are many ways in which clouds can organize, but identifying the boundaries between different forms of organization is challenging. The human eye is fairly good at detecting features—such as clouds that resemble flowers or fish. In this project we are supposed to do multiclass segmentation: finding the regions of an image dominated by one of the four different clouds. This project is based on the Kaggle challenge: Understanding Clouds from Satellite Images[1], where the scientists proposed a challenge to develop an effective machine learning model they could use to help them with building their own model for weather/climate patterns. The classification model would help them by removing the onus of manually labeling satellite images for cloud structures.

2. Background

2.1 Cloud types

The cloud types that are proposed to be detected by the models in this Kaggle Challenge are:

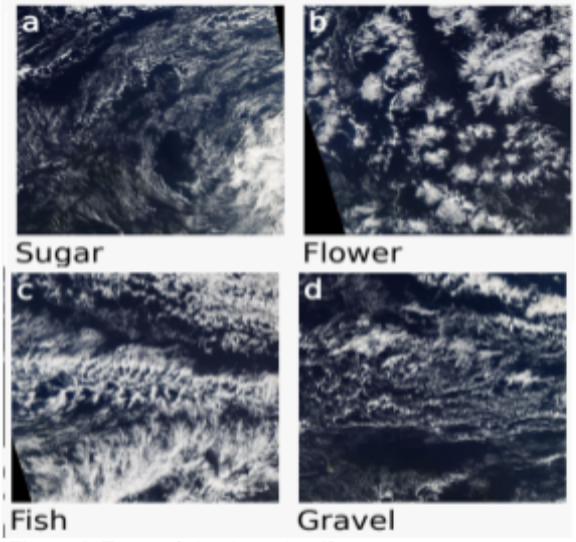

Figure 1: Types of clouds to classify

As defined in the paper [2] these cloud types are distinguishable/defined as:

- "Sugar" describes wide-spread areas of very fine cumulus clouds.
- "Flowers" are areas with circular cloud structures, each ranging from 50 to 200 km in diameter, with similarly wide cloud-free regions in-between.

- "Fish" are elongated, skeletal structures that sometimes span up to 1,000 km, mostly longitudinally.
- "Gravel" describes fields of granular features marked by arcs or rings.

2.3 How we plan to detect cloud types

We planned to phrase this problem as a segmentation/classification problem and then solve the labels for the regions with the same model. To do this we intended to use a pre-trained Mask RCNN model which we would fine tune using the provided dataset.

2.3 Other studies this builds on.

This Kaggle challenge was built from the paper: Combining crowd-sourcing and deep learning to explore the meso-scale organization of shallow convection [2]. The authors of that paper were trying to analyse the clouds that were human labeled to build a climate model for other climatologists to employ. This challenge is a natural extension of that paper. Human labeled data is a sparse commodity in many areas of computer vision. If machine learning models can automatically label the data, then the growth of the pool of labeled data can improve accuracy on the higher order models that are, for example, trying to model climate change.

3.Implementation tools

3.1 Implementation

Language: We implemented our system in Python. This language is fairly common for machine learning problems. It supports many popular machine learning frameworks such as Keras, SKlearn, Tensorflow, and Pytorch, that can be employed to solve practically any machine learning problem.

Kaggle: Fine-tuning our model is slow even on GPU's. Since Kaggle is a cloud based platform (like AWS), it was an appropriate platform choice for our work. Kaggle, offers (limited) free GPU time and unlimited CPU time to allow for training and evaluation of systems related to their tasks. We believed this would make training our model easier since we lacked local computers that were powerful enough to train the model ourselves. This platform also supports distributed notebook sharing so that as a team, we could both be working on the same code and Kaggle automatically imports the dataset related to the Kaggle challenge without requiring a more complex setup like Google Colab or Amazon Web Services (AWS). Thus, we assumed this platform would be the best for building our model in a team environment (though we encountered many annoying idiosyncrasies with the system which later disproved this assumption).

Libraries:

Keras: Because model training is time consuming and keeping the Kaggle notebook open 24/7 was not an option, we clearly needed ways to easily save and load our fine-tuned model once it was trained. Keras is designed to work with Tensorflow, and it incorporates this additional functionality. Thus, we decided to use Keras to enable easy saving and loading of our trained model.

Tensorflow: This framework is commonly used in machine learning and supports most of the functionality we needed for our model. We used this framework for our machine learning system so that we could perform backpropagation and gradient computations on our model without having to perform the operations ourselves.

3.2 Dataset

The database provided with the challenge consists of satellite images that contain certain cloud formations, with label names: Fish, Flower, Gravel, or Sugar. For each image in the test set, we needed to segment the image into regions representing each cloud formation label. Each image has at least one cloud formation, and can possibly contain all four label types.

The dataset size is approximately 6GB and consists of about 10,000 images. Kaggle pre-splits the data into training and testing data with the approximate distribution of: 60% training and 40% test data.

For the 5546 training images, we further split this data into 90% train and 10% validation (fairly standard).

Testing Data: With the 3697 test images, Kaggle withholds the labels for testing purposes. Presumably this is to stop contestants from being able to train their

models on both test AND train, which would give the contestent's model(s) inappropriately high accuracy by overfitting. Inevitably this would lead to poor performance when the model was challenged with real-world data. When submitting a model to the Kaggle challenge, the model predicts labeling for the test images. This is followed by Kaggle automatically checking your accuracy on these predictions to score the model(s)'s effectiveness. However, because the labels for this part of the data are withheld from developers, we could not use this data to analyze why our model was performing poorly. Instead we used our validation data (for which we have the labels for masked regions) to analyze results.

The data was compiled using crowdsourcing to provide labels. However, it was found that humans tend to label the regions where cloud dominates quite differently (see figure 2). As a result, the dataset creators decided to label the regions in the image that are masked to be the pixel regions where the majority of the users, voting on the image have masked a region (example in figure 2). In other words, at each pixel a vote of the majority assigns the pixel whether or not it is included in the labeled region.

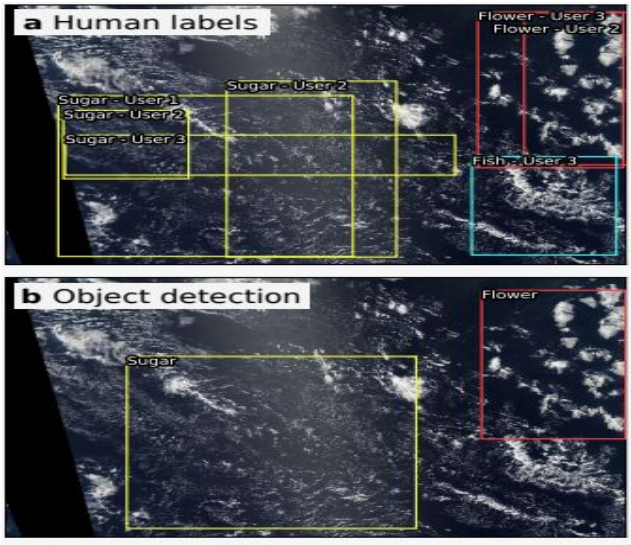

Figure 2: Human labels for an image and final result

3.2.1 Database Schema

For each image in the training set, Kaggle provides the masked regions in which a cloud type dominates in that region in run-length encoding format. Thus each image has its segmented regions in the image represented as 4 lines (one for each cloud type) in the dataset.

These run-length encodings are supposed to be the segmented regions of the image where a specific cloud type dominates (though this isn't necessarily true as discussed in our future works section below).

	Image_Label	EncodedPixels
0	0011165.jpg_Fish	264918 937 266318 937 267718 937 269118 937 27...
1	0011165.jpg_Flower	1355565 1002 1356965 1002 1358365 1002 1359765...
4	002be4f.jpg_Fish	233813 878 235213 878 236613 878 238010 881 23...
5	002be4f.jpg_Flower	1339279 519 1340679 519 1342079 519 1343479 51...
7	002be4f.jpg_Sugar	67495 350 68895 350 70295 350 71695 350 73095 ...

Figure 3: Example of run-length encoding provided with the dataset

4. Design choices

In this challenge, we employed an existing model that classifies the cloud organization patterns and returns the bounding box (encoded pixels). For this purpose we figured, that Mask RCNN[3] would be a perfect architecture, as it outputs the class label, bounding box coordinates for each object, and object mask(s). It's based on Feature Pyramid Network (FPN) and is further founded on a ResNet101 backbone.

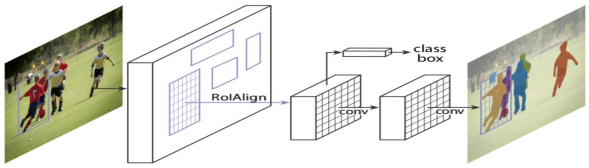

Figure 4: Mask RCNN

We used a Mask RCNN which has pretrained weights for the MS-Coco dataset. This was then fine-tuned on the cloud satellite image dataset provided in this Kaggle challenge. The model generates bounding boxes and segmentation masks for each instance of an object in the image. Due to the alleged rectangular shape of the "ground-truth", we initially decided to use the ROIs and not the masks given by the MaskRCNN.

4.1 Data Augmentation:

Due to the small size of the dataset, we decided to do some augmentation. This increases the diversity of training dataset available to tune the

pre-trained model. Thus we performed cropping, padding, and horizontal and vertical flipping on the 5546 training images. We considered and rejected the idea of doing rotation as well for the augmentation. Many of the other teams did perform this type of augmentation. However, we excluded this as we didn't want to add unrealistic examples into the dataset. In their description of cloud types they indicate that some of the cloud types are (usually) rotation specific "Fish: are elongated, skeletal structures that sometimes span up to 1,000 km, mostly longitudinally." [2]

5. Experiment and Results

5.1 Evaluation Metrics

This competition is evaluated on the mean Dice coefficient. The Dice coefficient can be used to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth. The formula is given by:

$$\text{Mean Dice Coefficient} = \frac{2*|X \cap Y|}{|X| + |Y|}$$

5.2 Experiment

We trained our model for 9 epochs, which took approximately 8-9 hours. Then to amortize overfitting issues, we picked the model version with the minimal validation loss. As you can see in figure 5, epoch 9 gave us the best model, which we used to run on the validation and test dataset as well as our model analysis.

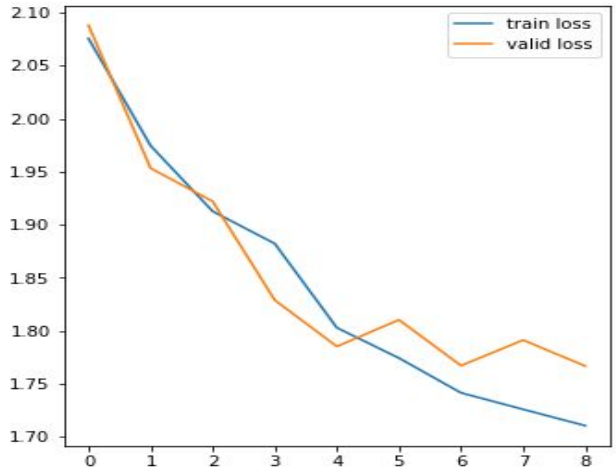

Figure 5: 9 epochs of training the model

We did some hyperparameter tuning and found that set a (Table 1) hyperparameters worked the best with an accuracy of 58.89%.

Hyperparameters	Set a	Set b
Learning Rate	0.001	0.005
Batch size	4	4
epochs	9	9
Validation steps	500	1000
Weight decay	0.0001	0.0005

Table1: Model Hyperparameters

5.3 Result

After we trained our model, we intended to try to analyze why current models' accuracies were so low (we predicted around 50-65% based on other Kaggle teams results). Our actual accuracy was 58.89% for the first run, and after minor modifications to our model, (discussed below) we obtained 59.2% accuracy. In comparison, the best team obtained an accuracy of 67.175% using an ensemble of FPN based models for more precise segmentation and a UNET architecture with a classification head for the classification. [4] Thus, while our resulting accuracy is lower, we can consider it somewhat comparable to the leaderboards' UNET based architectures. In the images below, our Mask RCNN finds the label(s), mask(s) and box(es) for the images. In Figure 6, our model can differentiate between 3 different types of clouds while in Figure 7 it avoids the noise and finds the cloud in the separate region.

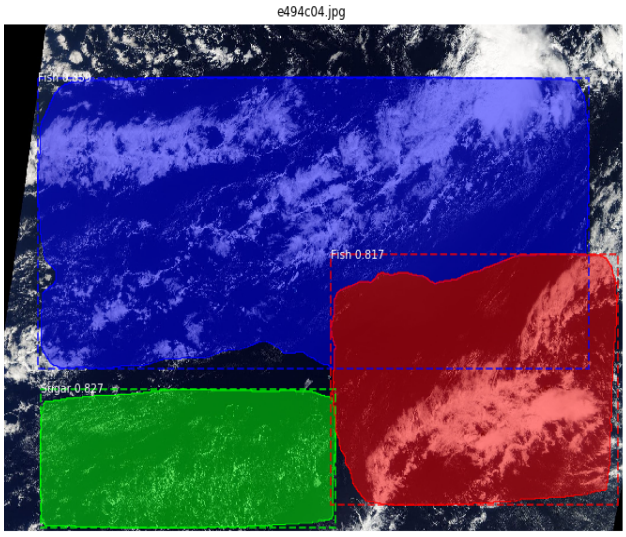

Figure 6 Our predicted regions for image e49c04

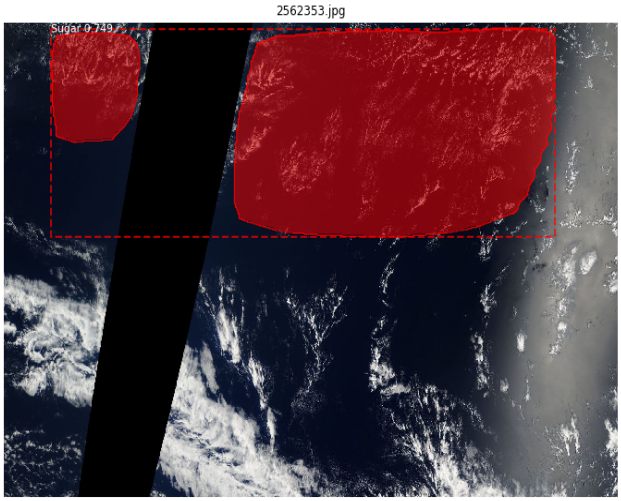

Figure 7: Results of Mask RCNN on the dataset

6. Discussion

Analysis we performed on our model's results indicate that our moderate accuracy is primarily due to:

High false negatives: Our model's primary error with accuracy comes from its very high false negative rate. We miss bounding areas in the image where we should have a mask. As shown in Table 2, many of the images' masks are completely omitted by our model.

	Sugar	Gravel	Fish	Flower
True Positives	1542	338	147	502
False Negatives	1864	1773	391	814

Table 2: False Positives for mask counts

Clearly our model is failing to learn to accurately bound many of the regions in the image. This is possibly due to the small amount of images we used to fine-tune the model. Even though we did augment the images (through vertical and horizontal flipping as well as image cropping and scaling segments), the number of inputs is still fairly small. This can be seen by the fact that even after as few as 9 epochs, with small learning rates, our model is already beginning to overfit the data (see figure 5). The winning team added additional augmentations such as, "rotate, grid distortion, channel shuffle, invert, to gray" [4]. These additional augmentations could definitely be used to improve our models performance, as our model clearly is not accurately distinguishing cloud types with the data we are using to train it.

Another source of error is that we were using bounding boxes when the database creators are using segmentation for the proposed regions. For example see Figure 8 below.

Figure 8: ground truth labels for sample image.

However, for some unknown reason (probably some setup issue) even though we intended to use our bounding boxes as a rough approximation of segmented regions, our model instead produces a single bounding box for each cloud category in an image. We had initially intended to use the bounding boxes to make our submission. We made this choice because the Kaggle paper [2] seems to imply that the task is looking for classification (masked region) and not segmentation. However, after we implemented

our function to use the Mask RCNN's mask to define cloud regions in an image, we noticed that the data was actually providing segmented regions (presumably this was a later modification to the dataset that wasn't included in the initial paper submission for the dataset documentation). This means that we incur minor errors due to having less precise segmented regions. However, since Mask RCNN produces both the mask (segmented areas) AND Regions of Interest (bounding boxes), this was correctable. Thus we were able to get rid of this error by using the segmentation masks instead of the bounding boxes to create our RLE submission for accuracy scoring (this brought our accuracy up to 59.2%)..

Smaller masked regions: Initially when we analyzed the predictions of our model we were getting another source of error. For some reason, as discussed above, instead of producing separate regions of interest (ROIs) for the various disjoint "objects" in our image, our model only produced 1 bounding box per cloud type in an image. Thus small regions that should be a second masked region were essentially merged into the main mask for that type. This is shown in Figure 9 (below). There are clearly two regions where the cloud type "sugar" dominates. However, our Mask RCNN merges these two ROIs into 1 big masked region. Thus our model inaccurately classes the area in which the cloud dominates, and it fails to exclude the noisy regions (the black obstructed region). As the dataset developers indicated when they initially created their database, this noisy region should have been excluded from our prediction, "after removing any black band area from the areas" [2] . As a result we suffered some accuracy loss according to the dice scoring method. To fix this, we used the specified segmented regions instead of the masked area for the prediction. This should have and did improve the accuracy as the mask is much more accurate in comparison with the ROI our model generates.

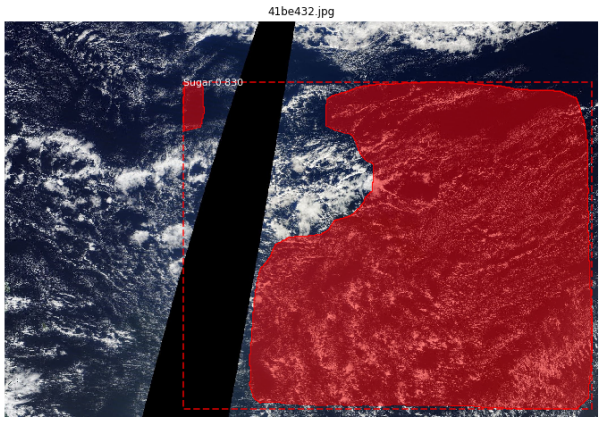

Figure 9: bounding box produced by the Mask RCNN

7. Related works

Combining crowd-sourcing and deep learning to explore the meso-scale organization of shallow convection. This paper is the basis for this Kaggle challenge. The dataset was compiled by the authors of the paper, and the entire reason for the Kaggle challenge is explained in the paper. However, the paper does not effectively employ machine learning to get results for climate analysis. Thus, the authors proposed this Kaggle challenge to see if there were possible solutions they could employ.

Kaggle Winning Team [4]: Because the Kaggle challenge ended prior to our submission, we were able to see the winning team's solution. They used U-Net architecture, but then also used multiple classification models to improve segmentation accuracy of their system. Since they used multiple models, a voting classifier to predict regions of the image, and a U-Net architecture, they were apparently able to get rid of some of the bad segmentation masks that hamper our model.

8. Future Work

Even with our improvement of classified regions, our model's (and theirs) main flaw is still that in a major portion of the images (see Table 2 above), there is a failure to predict ANY mask for the image when there should definitely be one. For example, see Figure 8. The ground truth says image 4f98f81.jpg should contain: Gravel, and Sugar. However, our model shows no maskable regions at all. Clearly, current models are not accurately able to bound and identify cloud types in images. This could

be due to the small dataset sample size or, as discussed below, bad data used in training our models.

Dataset Analysis: According to the authors who created this dataset, “users [were asked] to draw rectangles around regions where they judged one of the four cloud patterns to dominate”. Then they assigned the masked region based on the label given by the majority of the users. [2] However, as shown in Figure 10 below, the ground truth areas are overlapping and incorrectly assigned.

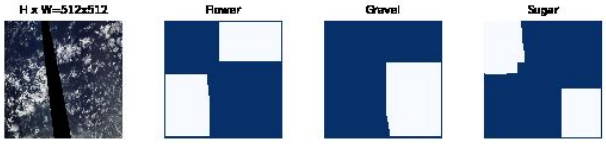

Figure 10: Ground truth labeled regions for an image.

View the bottom region labeled as sugar dominant in this image. That area is entirely contained in the region where supposedly Gravel dominates. It is not possible to have two cloud types “dominate” the same region of an image. Therefore, clearly the data is not 100% accurate. As a result, our model is quite probably incorporating this flawed data when learning. This will definitely cause errors as we use a mask to ensure that any overlapping regions in our predictions are cut out. Thus even if these bad labels in the data didn’t impair our models learning, the Mask RCNN wants to define a segment area where the area is occupied by a single label. This makes sense, as there cannot be two objects in a standard image occupying the same space. As a result, our model, when compared with the “ground truth”, will either not label the gravel region in Figure 10 and be penalized, or it will skip the sugar area in the image and be penalized for that. Therefore, our accuracy will drop through no fault of our model. Better filtering/categorizing the “ground truth” is definitely needed on this dataset, to ensure consistent definition of where the actual masked regions are labeled.

Another area for improvement would be to fine tune the hyperparameters for the Mask RCNN. In comparison with the other challengers who used U-Net architectures, the range of accuracy was about 55%-60%. Due to the fact that we are allocated 30 hours free per month, we were unable to perform extensive hyperparameter tuning for our model. However, based on these other teams’ performance and analysis of their models, this could probably grant our model a small accuracy boost of a few percent.

9. Conclusion

Both our Mask RCNN model and the other challengers’ U-Net models perform rather poorly in extracting the regions where one cloud class “dominates”. However, we found many reasons for the cause of these inconsistencies. Our model is clearly failing (quite often) to detect these regions in images. This could be because either our model needs more training or the Mask RCNN simply is not a model that is well suited to this particular task. Interestingly, we also found that the database is not particularly accurately labeled. Thus for better assurance of a correct model there is clearly a need for the dataset providers to refine their collected labels for the images they provide for Kaggle challengers’ training set.

Our code repo. Llots of our code we wrote, and then rewrote as we made changes and then commited the final version of the notebook to our Github repository. Thus this final notebook version may be slightly different than described in various stages throughout the paper.
https://github.com/Bryce-BG/CMPSCI-670_Cloud-Classification-Project

References

[1] Kaggle challenge: Understanding Clouds from Satellite Images

[2] Stephan Rasp, Hauke Schulz, Sandrine Bony, and Bjorn Stevens. Combining crowd-sourcing and deep learning to understand meso-scale organization of shallow convection. arXiv preprint arXiv:1906.01906, 2019

[3] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. *arXiv:1703.06870*, 2017

[4] Purdue, 1st placed solution with code. (n.d.). Retrieved from https://www.kaggle.com/c/understanding_cloud_organization/discussion/118080.