The Shapes in the Clouds

Can a computer see the shapes we see in clouds?

Can a computer model be trained to identify complex organizations of clouds in satellite images?

Problem Statement

It all started with...

Fire Damage Assessment using Neural Networks

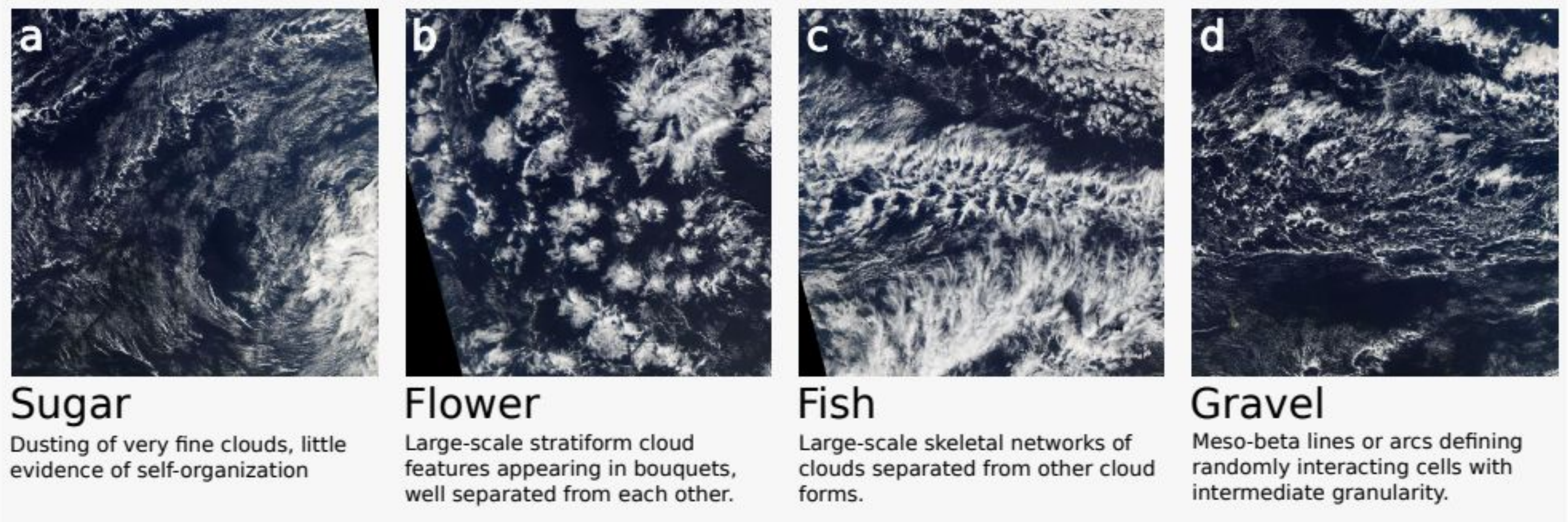
Let's start with something easier.

Like CLOUDS!?

KAGGLE COMPETITION:

Understanding Clouds from Satellite Images

Can you classify cloud structures from satellites?



Competition Based off of:
https://arxiv.org/pdf/1906.01906.pdf

Data Source:

Four subjective patterns of organization were defined:

Surgar, Flowers, Fish, and Gravel

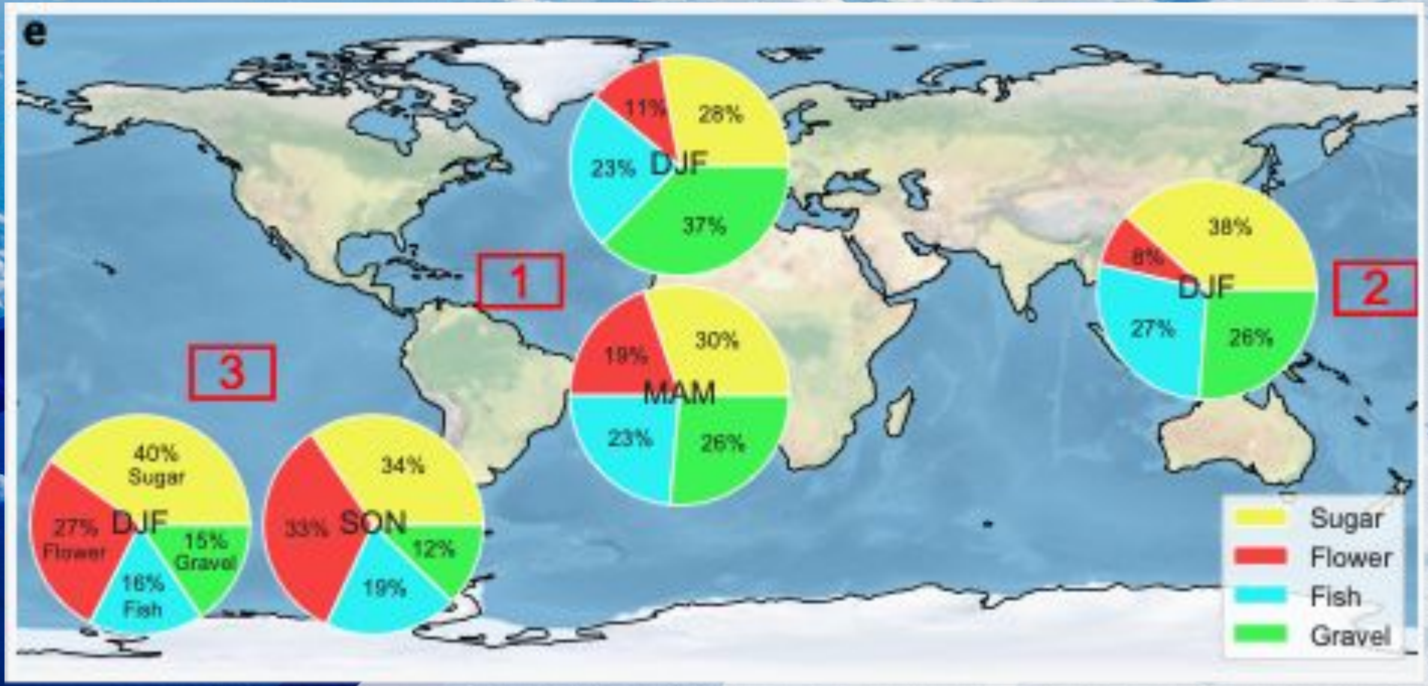
Cloud labeling took place with volunteers at two institutions.

- **Technical University of Munich, Germany**
- **Max Planck Institute for Meteorology**

67 scientists screened 10,000 satellite images on a crowd-sourcing platform

After 250 man hours they classified almost 50,000 mesoscale cloud clusters.

https://arxiv.org/pdf/1906.01906.pdf



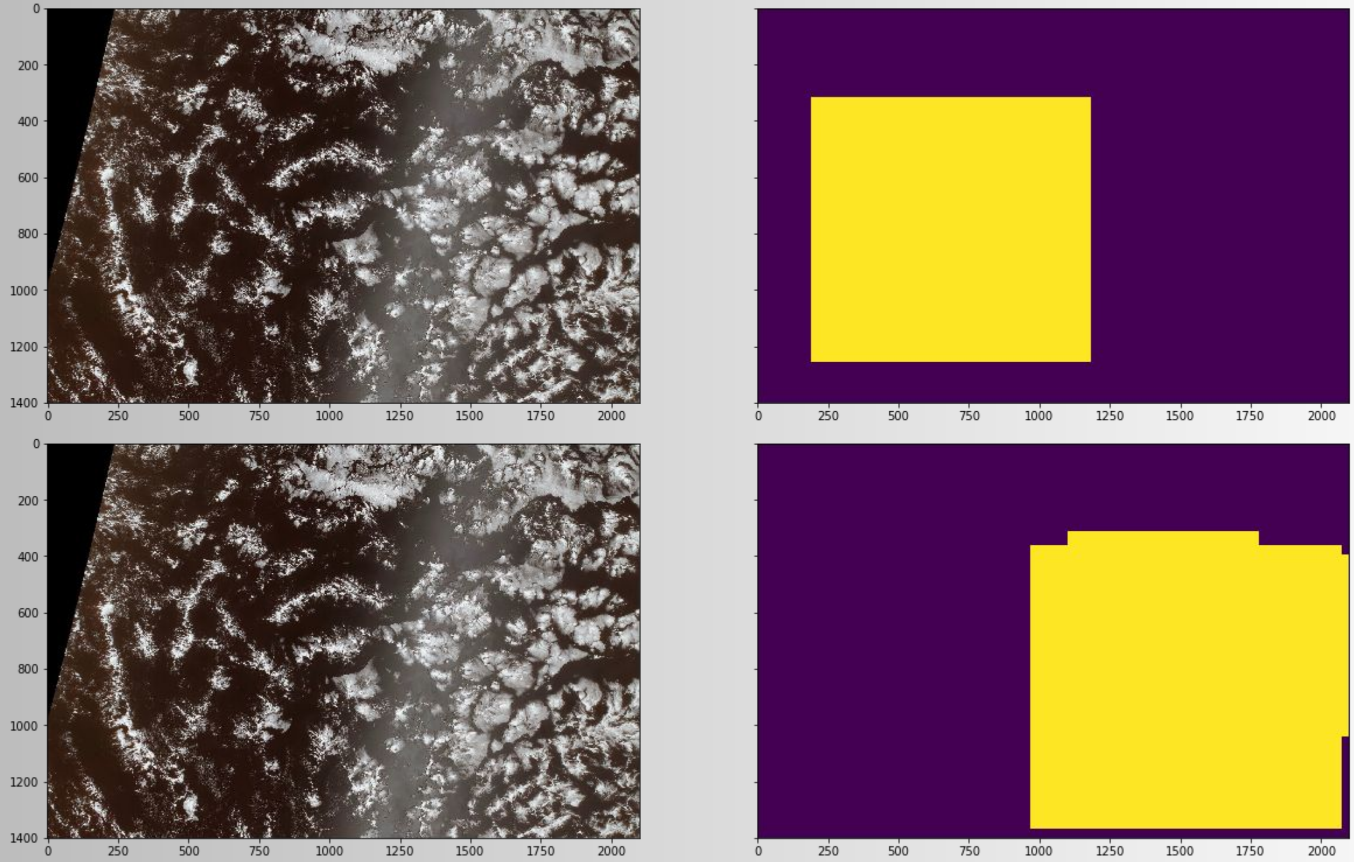
https://arxiv.org/pdf/1906.01906.pdf

Training Data Preprocessing:

- 5546 total images.
- 2100 x 1400 pixels
- 4 possible cloud types in each image (that can occur more than once per image)
- Drop the NaNs

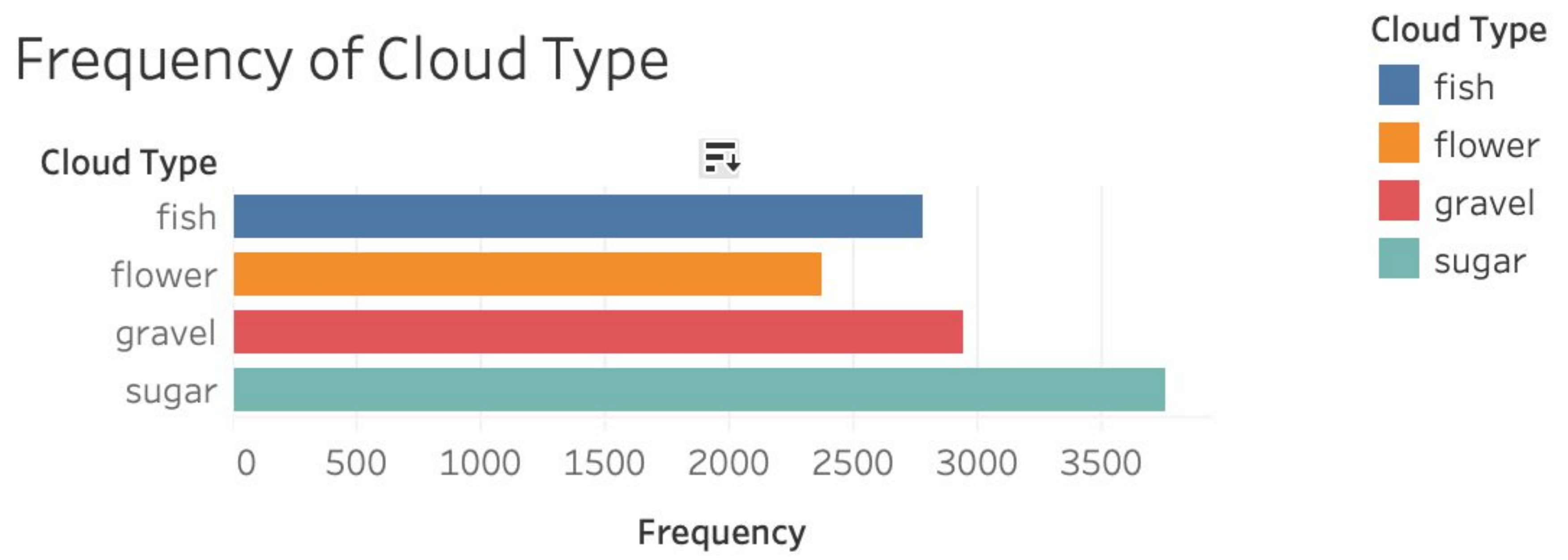
Image_Label	EncodedPixels	ImageId	ClassId	hasMask
0011165.jpg_Fish	264918 937 266318 937 267718 937 269118 937 27...	0011165.jpg	Fish	TRUE
0011165.jpg_Flower	1355565 1002 1356965 1002 1358365 1002 1359765...	0011165.jpg	Flower	TRUE
0011165.jpg_Gravel	NaN	0011165.jpg	Gravel	FALSE
0011165.jpg_Sugar	NaN	0011165.jpg	Sugar	FALSE
002be4f.jpg_Fish	233813 878 235213 878 236613 878 238010 881 23...	002be4f.jpg	Fish	TRUE

Translate “Encoded Pixels” to masks

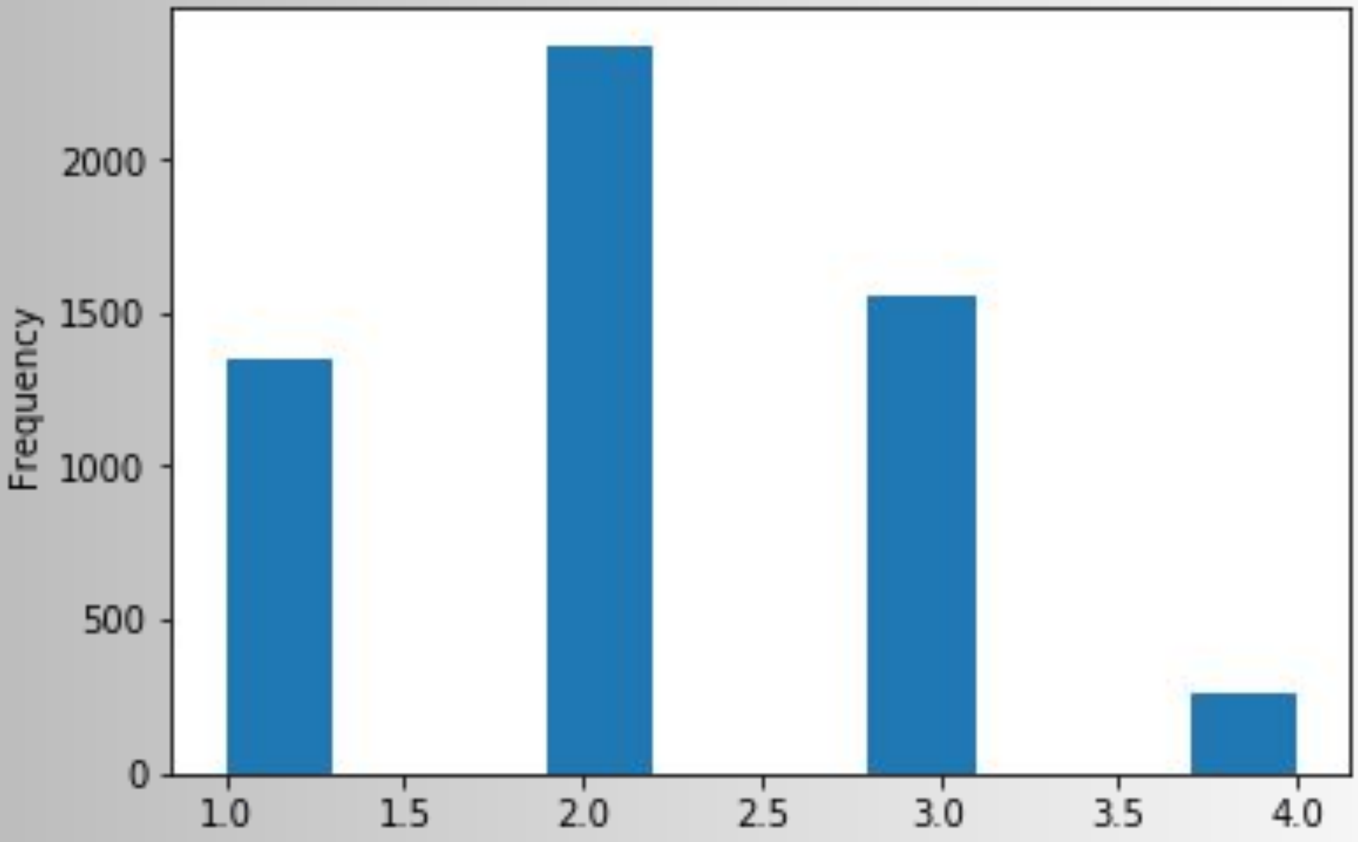


https://www.kaggle.com/paulorzp/rle-functions-run-lenght-encode-decode

Cloud Type Frequency from the 5,546 Images



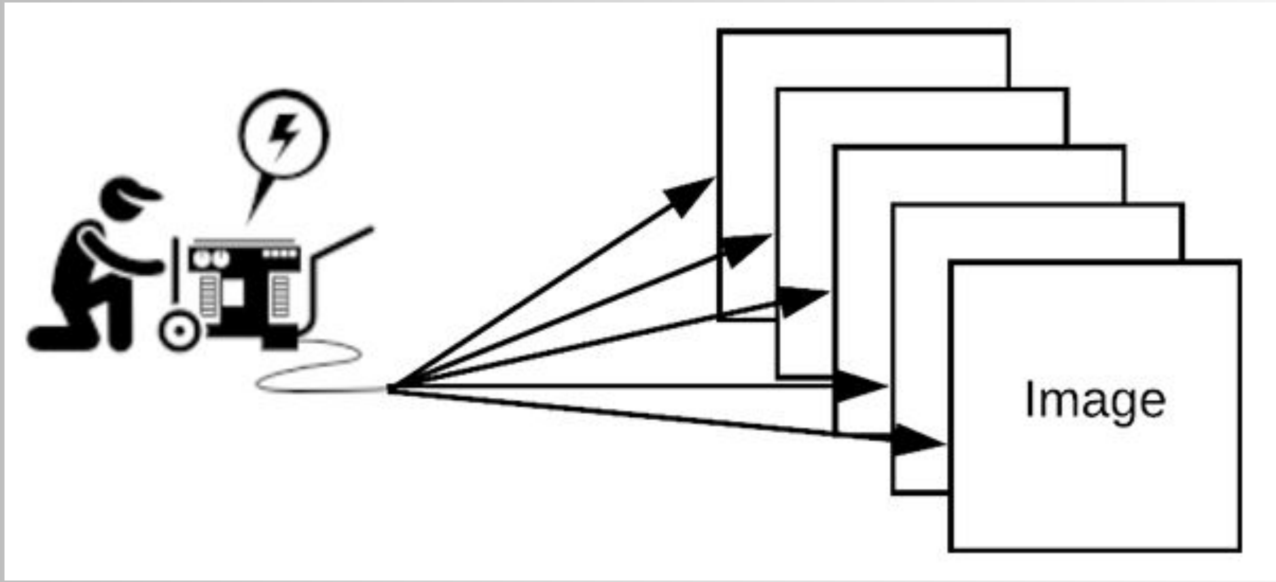
Cloud Occurences per Image



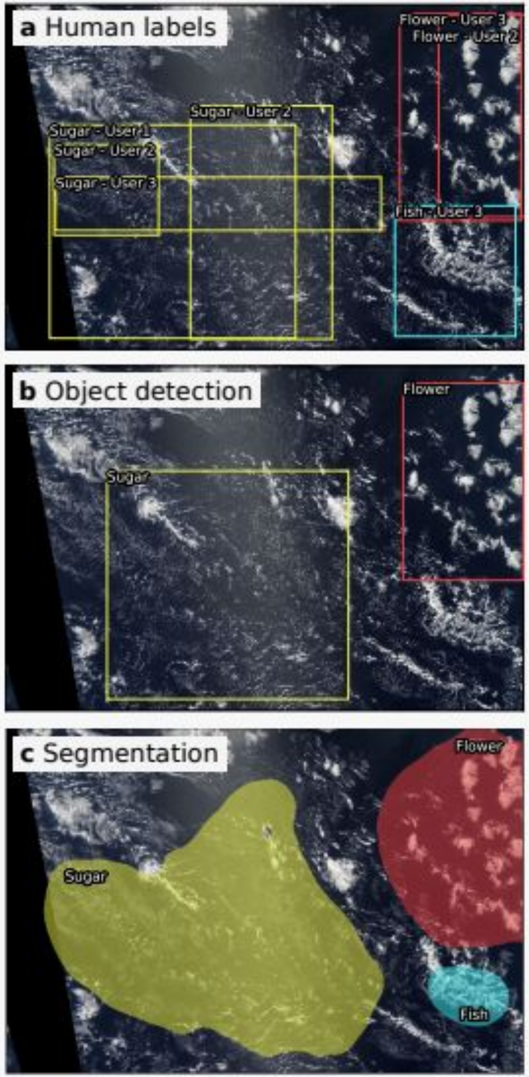
Tensorflow Image Generator

With so many images of such a large size. We need to break them up into smaller segments that can be passed for analysis and patch processing.

Enter the ImageGenerator!



https://towardsdatascience.com/keras-data-generators-and-how-to-use-them-b69129ed779c



The pattern recognition task can be framed as two machine learning problems:

- **object detection**
- **semantic segmentation.**

Object detection algorithms draw boxes around features of interest,
“Replicating” what the human labelers were doing.

Segmentation algorithms classify every pixel of the image at a small scale.

https://arxiv.org/pdf/1906.01906.pdf

https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/

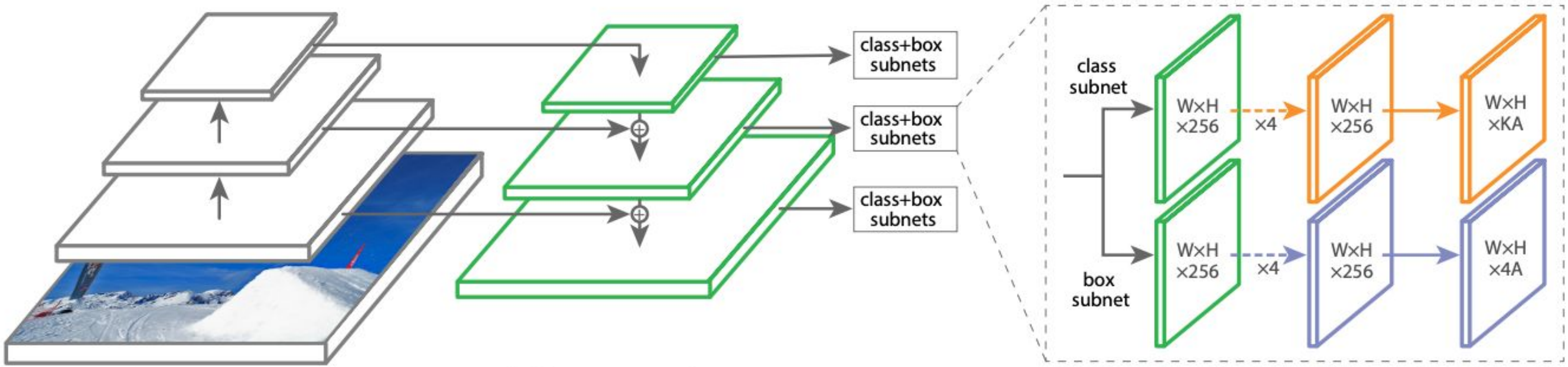
RetinaNet Object Detection Algorithm

Feature pyramid is used to compare possible objects at different scales.

Two Stage Approach:

Stage 1 finds possible candidates for cloud formations.

Stage 2 classifies candidates as either “foreground” or “background” class.



(a) ResNet (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)

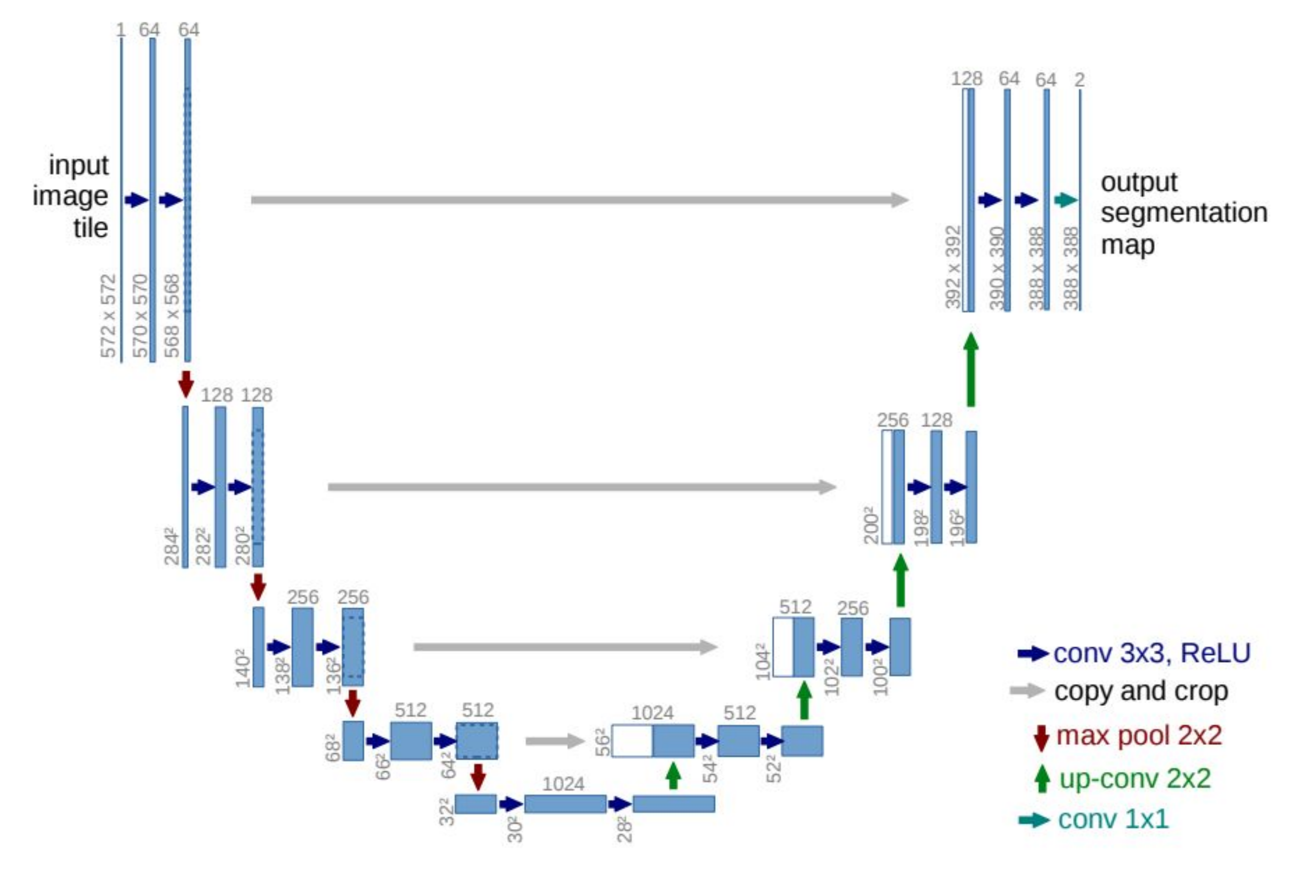
https://arxiv.org/pdf/1708.02002.pdf

UNet for Segmentation

Looking at smaller and smaller boxes of pixels

Deciding what is our object of interest and what isnt.

Then merging them back together in larger and larger boxes of pixels to verify that it still makes sense.

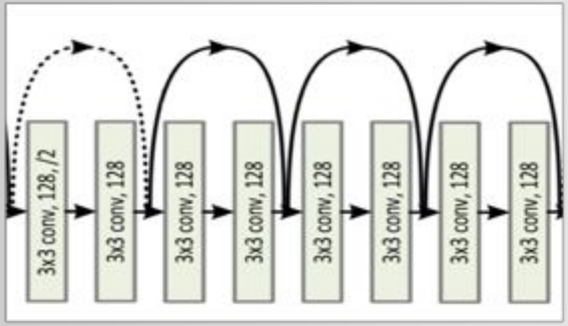


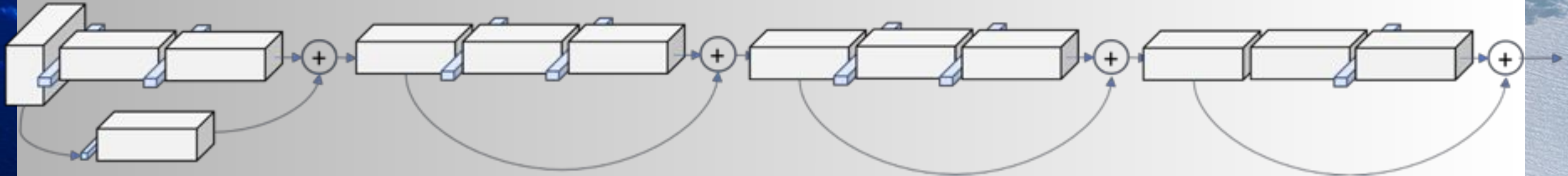
https://arxiv.org/pdf/1505.04597.pdf

With so many layers, will our model keep learning?

Learning comes from using a “loss function” to figure out how to improve.
If the result of the loss function is too close to zero, learning stops and the model stalls.
This is a common issue with extra deep/complex models called “vanishing gradient”.

This is where ResNets come in!
Adding in data from previous
Layers to later layers avoids
Degradation of performance
And vanishing gradient.





https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8

Total parameters: 24,456,589

So many parameters, but the model process is the same:

```
model = sm.Unet(
    'resnet34',
    classes=4,
    input_shape=(320, 480, 3),
    activation='sigmoid'
)
model.compile(optimizer=adam, loss=dice_loss, metrics=[dice_coef])

history = model.fit_generator(
    train_generator,
    validation_data=val_generator,
    epochs=30
)
```

So many parameters, not enough time!

Used GPU enabled TensorFlow to massively speed up computing time.

3x GTX 1080ti @ 3,584 Cuda Cores per card.

Est. Time to Fit on Macbook:

~227 hours.

GPU Enabled time to Fit:

~6 hours.

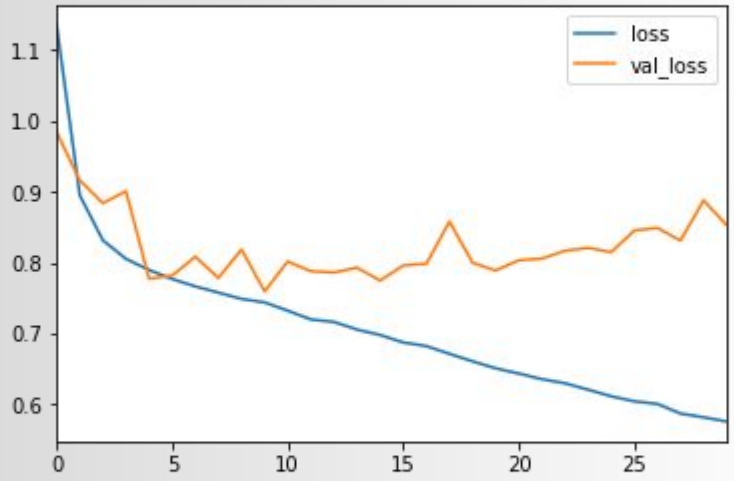
But How Did it Do???

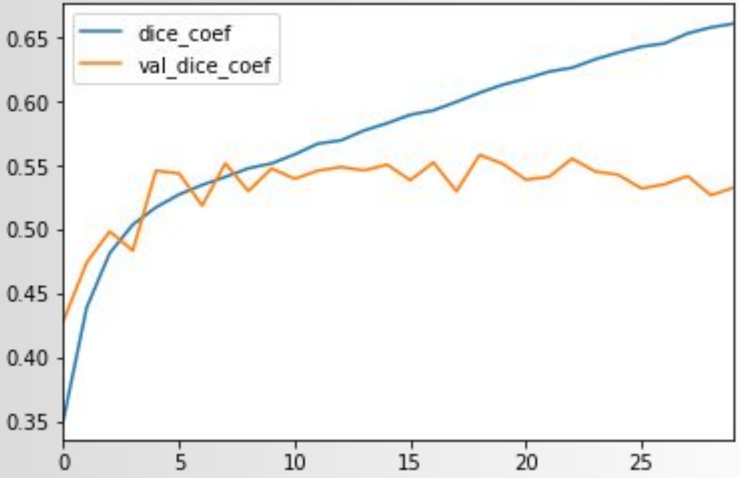
On a scale from 0-1:
First working attempt:
-Basic CNN with "bare bones" methods to complete the task:

Score: 0.317

Revised model
-implementing RetinaNet, UNet, and ResNet structuring:

Score: 0.59254 (~1,200th place!)






WELL...
THAT'S AWESOME!
makeameme.org

What now?

From thinking classifying imagery was cool but way too complicated

To

Now I have a foundation and can start exploring how to apply these techniques to my next project.

If that isnt the core of this course…

I dont know what is.

QUESTIONS!?

Yeah, I have lots too!

Huge Thank You to:

Kaggle Community:

-Xing Han Lu - xhlulu (https://www.kaggle.com/xhlulu)
- Alexander Teplyuk (https://www.kaggle.com/ateplyuk)
- Mutton Biryani (https://www.kaggle.com/gauravtaneja)

And Computer Vision Blog Community:

- Priya Dwivedi (https://towardsdatascience.com/@priya.dwivedi)
-Ilya Michlin (https://towardsdatascience.com/@ilyamichlin)
-Pulkit Sharma (https://www.analyticsvidhya.com/blog/author/pulkits/)
- Vincent Fung (https://towardsdatascience.com/@vincent.fung13)

Cornell University (https://arxiv.org/)