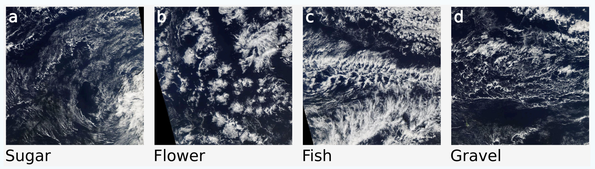
Classifying Cloud Structures using different Semantic Segmentation Architectures

Nathaniel Xu
Computer Science
Tsinghua University
xc-xu19@mails.tsinghua.edu.cn

Matthieu Lin
Computer Science
Tsinghua University
lin-yh19@mails.tsinghua.edu.cn

Background

There are four different types of **cloud formations**, and each image may have all or none of each type of cloud.



Our goal is to explore the performance of classifying cloud formations using **different segmentation architectures**

Some **challenges** in building the best model include:
- Traditional CNN architectures are not rotation-invariant
- Cloud images are composed of two stitched images from two separate satellites. This creates empty 'black' bands in images

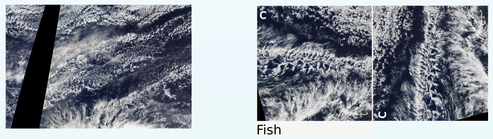
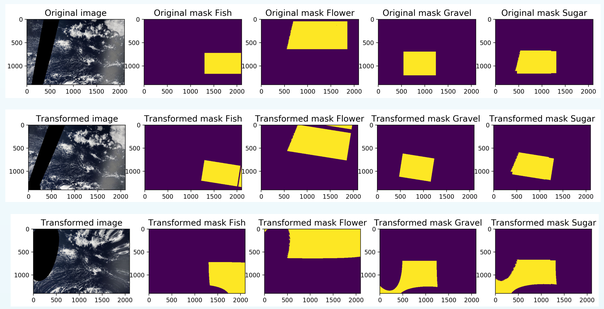


Image Preprocessing

The challenge of classifying affine transformed cloud structures and image black bands is addressed by augmenting the dataset.

The **image augmentations** applied include:
- Horizontal/Vertical flips
- Shift/Scale/Rotations
- Grid Distortion
- Optical Distortion



Loss Function

Binary Cross Entropy:
The cross entropy for a predicted segmentation mask is defined as:

$$BCE(y, \hat{y}) = -\sum_{i \in pixels} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

The corresponding gradient of the cross entropy loss is calculated to be:

$$\frac{\partial BCE(y, \hat{y})}{\hat{y}_j} = -\frac{y_j}{\hat{y}_j} + \frac{1 - y_j}{1 - \hat{y}_j}$$

Dice Loss:
The Dice loss is a measure of overlap between two samples, or in this case, between two segmentation masks.

$$D(A, B) = \left(1 - \underbrace{\frac{2\,|A \cap B|}{|A| + |B|}}_{Dice\ Coefficient}\right) \Longrightarrow D(y, \hat{y}) = \left(1 - \frac{2 \sum_{i \in pixels} y_i \hat{y}_i}{\sum_{i \in pixels} y_i + \sum_{i \in pixels} \hat{y}_i}\right)$$

The corresponding gradient of the Dice loss is calculated as:

$$\frac{\partial D(y, \hat{y})}{\partial \hat{y}_j} = -2\left[\frac{y_j\left(\sum y_i + \sum \hat{y}_i\right) - \sum y_i \hat{y}_i}{\left(\sum y_i + \sum \hat{y}_i\right)^2}\right]$$

Model Inference

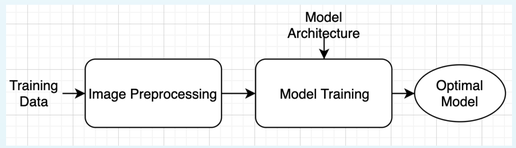
Model inference is run on the validation data to learn the following optimal model parameters:
- **Mask Threshold Value**
  - Predicted mask pixel probabilities greater than the threshold are accepted
- **Minimum Component Size**
  - Predicted segments with fewer number of pixels than the minimum component size are filtered out

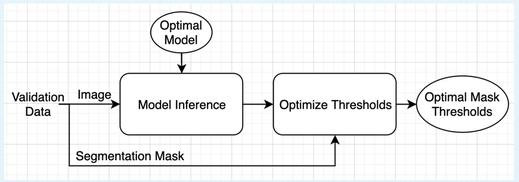
Algorithm 1: Optimize Model Parameters

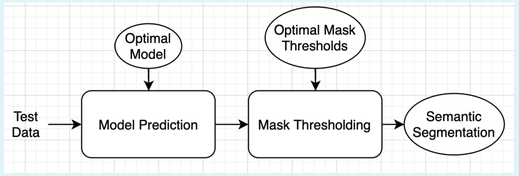
```
Find_Optimal_Class_Parameters (class=k);
    params = ∅;
    for thresh, min_size ∈ [0, 1], [0, 10000] do
        scores = ∅;
        for predicted_mask, true_mask ∈ class k do
            Threshold_Mask (predicted_mask, thresh);
            Filter_Components (predicted_mask, min_size);
            dice_score = Dice (predicted_mask, true_mask);
            scores = scores ∪ dice_score;
        end
        average_score = Average(scores);
        params = params ∪ {average_score, thresh, min_size};
    end
    return Maximum(params, criteria=average_score);
```

Methodology

Classification Pipeline:
The process of training a model to predicting segmentation masks consists of 3 stages:







Training-Validation Split:
- The ratio between training and validation data was taken to be 9:1
- The validation data is a stratified split of the original training data such that both the training and validation data have about the same proportion of images with each cloud type.

Model Architectures

U-Net

The architecture can be broadly thought of as an encoder and decoder network.

Encoder:
- Consists of convolution blocks followed by max pool downsampling
- Encodes the input image into feature representations at different levels
- Usually a pre-trained classification network (backbone)

Decoder:
- Consists of up-convolution (upsampling) blocks and concatenation with higher resolution feature maps from the encoder network.
- Decodes the features learnt by the encoder onto the original pixel space to get a dense classification

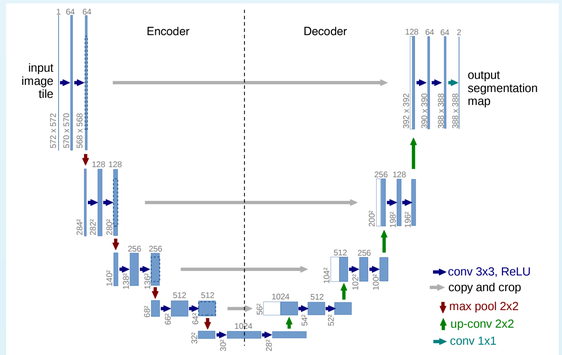


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Feature Pyramid Network (FPN)

The FPN is composed of a bottom-up and top-down pathway, with each pathway shaped as a pyramid of layers.

Bottom-Up Pathway:
- Consists of the feedforward computation of the backbone convolutional network
- Usually a pre-trained classification network (backbone)

Top-Down Pathway:
- Consists of the upsampling of feature maps from higher pyramid levels merged with feature maps of the same spatial size from the bottom-up pathway

Segmentation Masks:
- Segmentation masks are generated at each layer
- The final mask prediction is the merging of masks at each scale

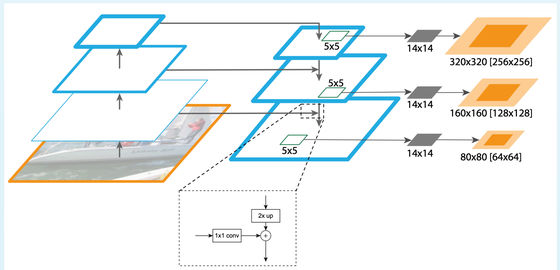
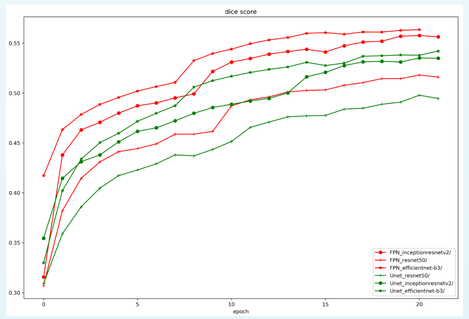


Figure 4. FPN for object segment proposals. The feature pyramid is constructed with identical structure as for object detection. We apply a small MLP on 5×5 windows to generate dense object segments with output dimension of 14×14.

Results

Both the Unet and FPN models were used to classify cloud structures from the satellite image database.



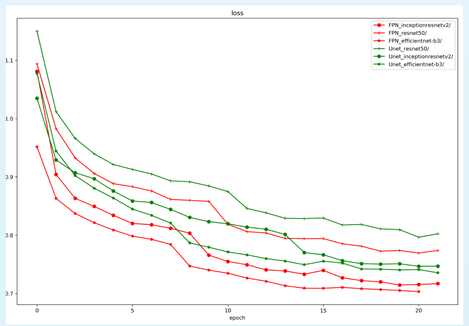


Table 1: Model Dice Scores

Backbone	Unet	FPN
resnet50	0.63438	0.63374
inceptionresnetv2	0.64993	0.65018
efficientnet-b3	0.65032	0.65637