Image Recognition Project Report

CS 429/ CS 529 Machine Learning

Team Members:

Chiyu Shen

Haijin He

Christina Xuan Yu

Step 1: Reshaping data

We read the band1 and band2 separately from train.json, and reshaped them from (75, 75) to (1604,75,75,1). Then we combined the band1, band2 reshaped versions as well as their average, by grouping them as the new train data. The result has 3 channels to fit as input of convolutional neural network. We then implemented the same process to the test data.

Step2: Normalizing data

We normalized the training and testing data to make the values positive and between 0-1. The purpose of this step is to reduce the difference in background, and make the value normalized between 0 to 1, which makes it easier to visualize and to adopt image processing techniques.

```
for i in range(3):
    for j in range(data.shape[0]):
        m1=data[j,:,:,i].min()
        m2=data[j,:,:,i].max()
        if m2-m1 !=0:
        data[j,:,:,i]=(data[j,:,:,i]-m1)/(m2-m1)
```

Step 3: Augmenting data

We generated flipped versions of images to augment the train data, by using ImageDataGenerator() from Keras. The ImageDataGenerator function provides a convenient way of augmenting image data with a few parameters.

Image augmentation is a way to generate more input data from existing data. It makes the model more generalizable, and prevent overfitting. It is especially useful when training data is not big, as in the case of the iceberg contest.

```
    train_datagen = ImageDataGenerator(
        rotation_range=20,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=True,
        vertical_flip=True )
```

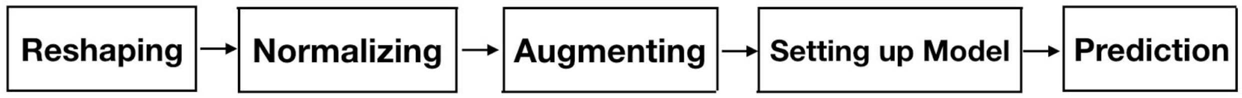
Step 4: Setting up the model

We used a pretrained VGG16 model from Keras with a little modifications to suit our purpose. We reduced the number of nodes in the fully connected layers, because we realized the complexity of this task is much simpler comparing to ImageNet classification task. We also made the final output to contain only 2 nodes since we only need to predict 2 classes. 2 DropOut layer with dropout rate 0.5 is added to prevent overfitting. We did not train from scratch because pretrained model already captures a lot of information for image processing and would accelerate our training process.

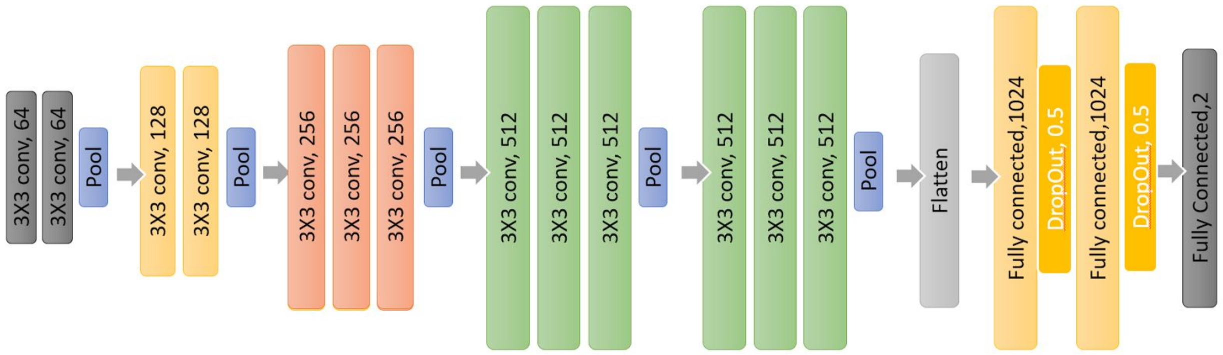
Step 5: Prediction

We Feed the test data into the model we set up, and produced the prediction for the images.

Flow chart:



CNN Model Chart:



Reference:

https://keras.io/

https://keras.io/applications/#vgg16

https://keras.io/preprocessing/image/