Machine Learning Engineer Nanodegree

Capstone Project

Jeongwhan Choi

January 29th, 2018

I. Definition

Project Overview

Deep learning belongs to neural model paradigm in three paradigms of machine learning. It is the latest version of connectionism that has been going on since the 1950s. In connectionism, problems that were previously considered obstacles have been solved one by one, and it has recently become the most destructive technology field in artificial intelligence. [1]In 1979, Kunihiko Fukushima first published Neocognitron model applying neurophysiological theroy to the artificial neural network. This model, inspired by Torsten Wiesel's awrad-winning Nobel Prize in neurophysiology, was the beginning of Convolutional Nerual Network(CNN). CNN, which imitates human visual cognition processes, is inherently used in a unique way in the field of computer vision. Convolutional technology, which shows excellent performance in extracting desired features from various types of data, is used in various fields such as image recognition and speech recognition. The reason for using the convolution technique in image and speech recognition is to separate and extract features contained in signals such as original image or sound wave.

In this Capstone project, I use the CNN model mentioned above to identify icebergs and boats through radar images. The dataset was provided by Kaggle competition "Statoil/C-Core Iceberg Classifier". This goal is to increase the identification accuracy with the CNN mentioned above.

The dataset have three different input: `band_1`, `band_2`, and `angle`. The `band_1` is a 75x75 pixels flattened grayscale image taken by satellite that transmit and receive horizontally. On the other hand, `band_2` is another 75x75 pixel flattened graysclae image taken by satellite that transmit horizontally and receive vertically. Since the image of the SAR radar image of the satellite, the angle at which the image is taken is always different. So the `angle` is added. Finally, the target variable `is_iceberg` is an iceberg at 1, and a ship at 0.

Problem Statement

Drifting icebergs present threats to navigation and activities in areas such as offshore of the Arctic Ocean. As shown in the picture below, the iceberg of this appearance may not be distinguishable from the ship, and it can give a great impact to the passing ship.

[2]

[2]

Currently, many institutions and companies use aerial reconnaissance and shore-based support to monitor environmental conditions and assess risks from icebergs. However, in remote areas with particularly harsh weather, these methods are not feasible, and the only viable monitoring option is via satellite.

They are responsible for maritime safety using SAR for ship and iceberg monitoring. Ships and icebergs can be detected because the response of backscattering is stronger than the surrounding open water.[3] However, the distinction between ship and iceberg is still difficult. Machine learning can be used to solve this problem. The goal is to create an image classification model that finds icebergs among SAR images collected by satellites. We will use the deep learning with CNN for this. As mentioned in the project overview, CNN is excellent for separating and extracting the characteristics of images.

The CNN model receives the flattened iceberg image data as input and the resulting value will indicate whether the image is a iceberg. The result will indicate that the iceberg in case of 1 in a range of from 0 to 1. There are 1604 training data sets, which may not be enough and may require transfer learning. Therefore, I will compare the benchmark model with the enhanced model and the VGG-16 transfer learning model.

Metrics

The evaluation metric used in accuracy.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{CGT}}$$

- SUT—System Under Test
- TP—True positive, an object present in the GT and the SUT (also called Correct Detection or one-to-one match)
- TN—True negative, an element present in neither the GT nor the SUT
- CGT—Complete Ground Truth is the total number of GT objects.

Accuracy = (TP+TN)/CGT, the sum of the true positives and the true negatives relative to the total number of GT objects. This is a measure of the actual performance of the system with regard to both correctly detecting and correctly rejecting targets.[4]

The simple accuracy metrics used as the training dataset has 53.05% "ships" and 46.94% "icebergs ". The classes are close to balanced.

If the training data classes are imbalanced, then F1-score metric is better but, this dataset is almost balanced, so using Accuracy as the metrics.

II. Analysis

Data Exploration

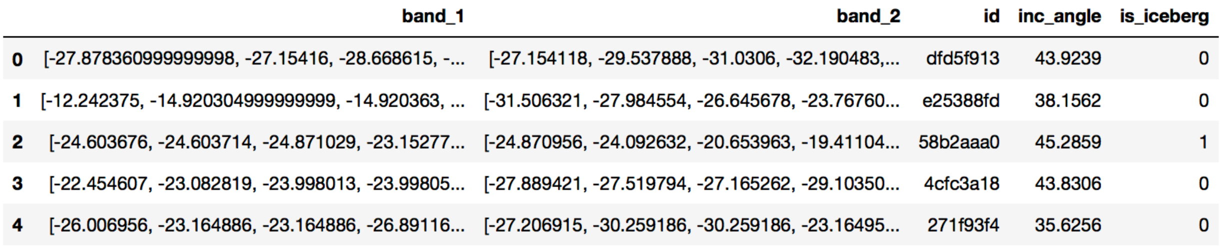
train.json, test.json

The data (`train.json`, `test.json`) is presented in `json` format. The files consist of a list of images, and for each image, you can find the following fields:

- **id** - the id of the image
- **band_1, band_2** - the flattened image data. Each band has 75x75 pixel values in the list, so the list has 5625 elements. Note that these values are not the normal non-negative integers in image files since they have physical meanings - these are float numbers with unit being dB. Band 1 and Band 2 are signals characterized by radar backscatter produced from different polarizations at a particular incidence angle. The polarizations correspond to HH (transmit/receive horizontally) and HV (transmit horizontally and receive vertically).
- **inc_angle** - the incidence angle of which the image was taken. Note that this field has missing data marked as "na", and those images with "na" incidence angles are all in the training data to prevent leakage.
- **is_iceberg** - the target variable, set to 1 if it is an iceberg, and 0 if it is a ship. This field only exists in `train.json`.

There are a total of 1604 training data and 8424 test data. The training dataset has 53.05% "ships" and 46.94% "icebergs". These are close to balanced.

The following shows some of the training data.

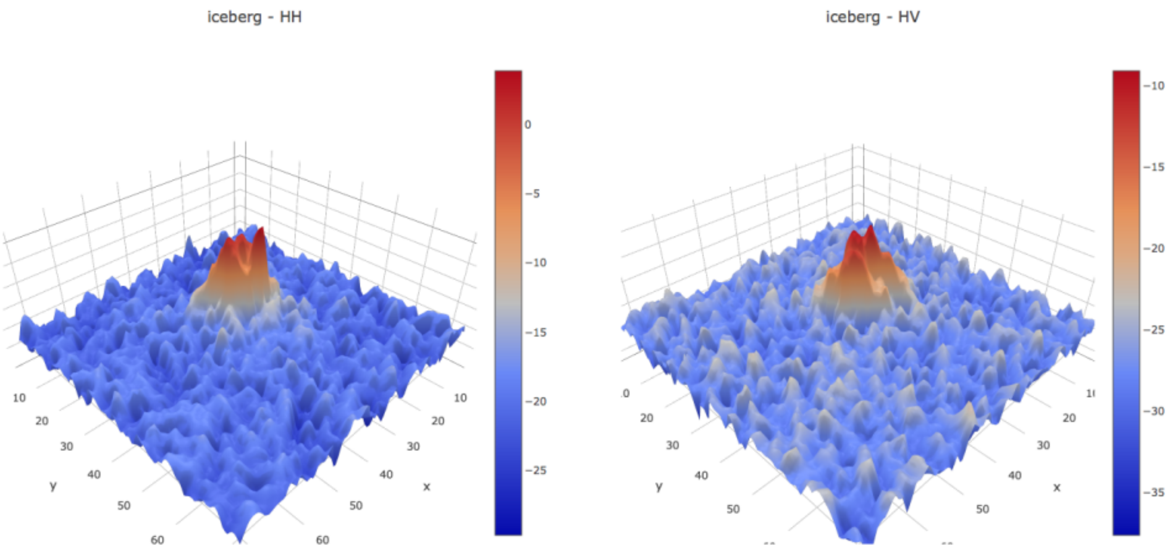


	band_1	band_2	id	inc_angle	is_iceberg
0	[-27.878360999999998, -27.15416, -28.668615, -...	[-27.154118, -29.537888, -31.0306, -32.190483,...	dfd5f913	43.9239	0
1	[-12.242375, -14.920304999999999, -14.920363, ...	[-31.506321, -27.984554, -26.645678, -23.76760...	e25388fd	38.1562	0
2	[-24.603676, -24.603714, -24.871029, -23.15277...	[-24.870956, -24.092632, -20.653963, -19.41104...	58b2aaa0	45.2859	1
3	[-22.454607, -23.082819, -23.998013, -23.99805...	[-27.889421, -27.519794, -27.165262, -29.10350...	4cfc3a18	43.8306	0
4	[-26.006956, -23.164886, -23.164886, -26.89116...	[-27.206915, -30.259186, -30.259186, -23.16495...	271f93f4	35.6256	0

Exploratory Visualization

Iceberg 3D image

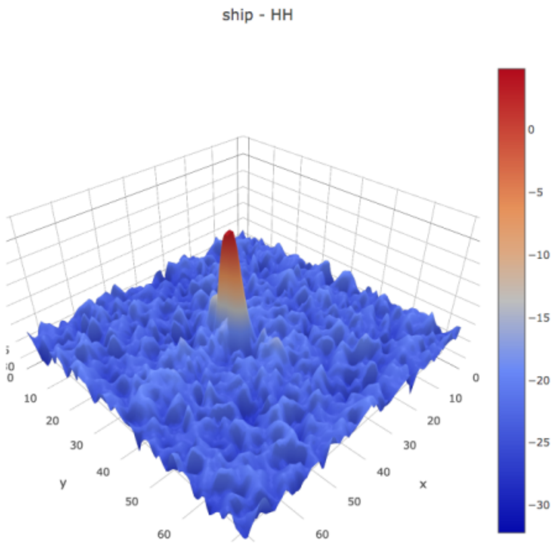
- HH and HV data of the 5th element with `is_iceberg = 1` are shown in 3D.

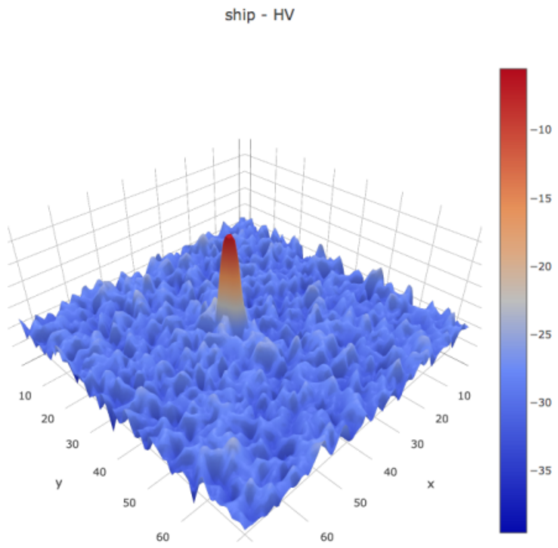


The shape of the iceberg in the radar data is going to be like a mountain, as shown here. This is not an actual image, but rather scattered from the radar, which will cause peaks and distortions in the shape. The shape of the ship will be like a point, it can be like elongated point. There are structural differences here and I can use CNN to take advantage of these differences. Creating a composite image using radar backscatter will be helpful.

Ship 3D image

- HH and HV data of the 5th element with `is_iceberg = 0` are shown in 3D.



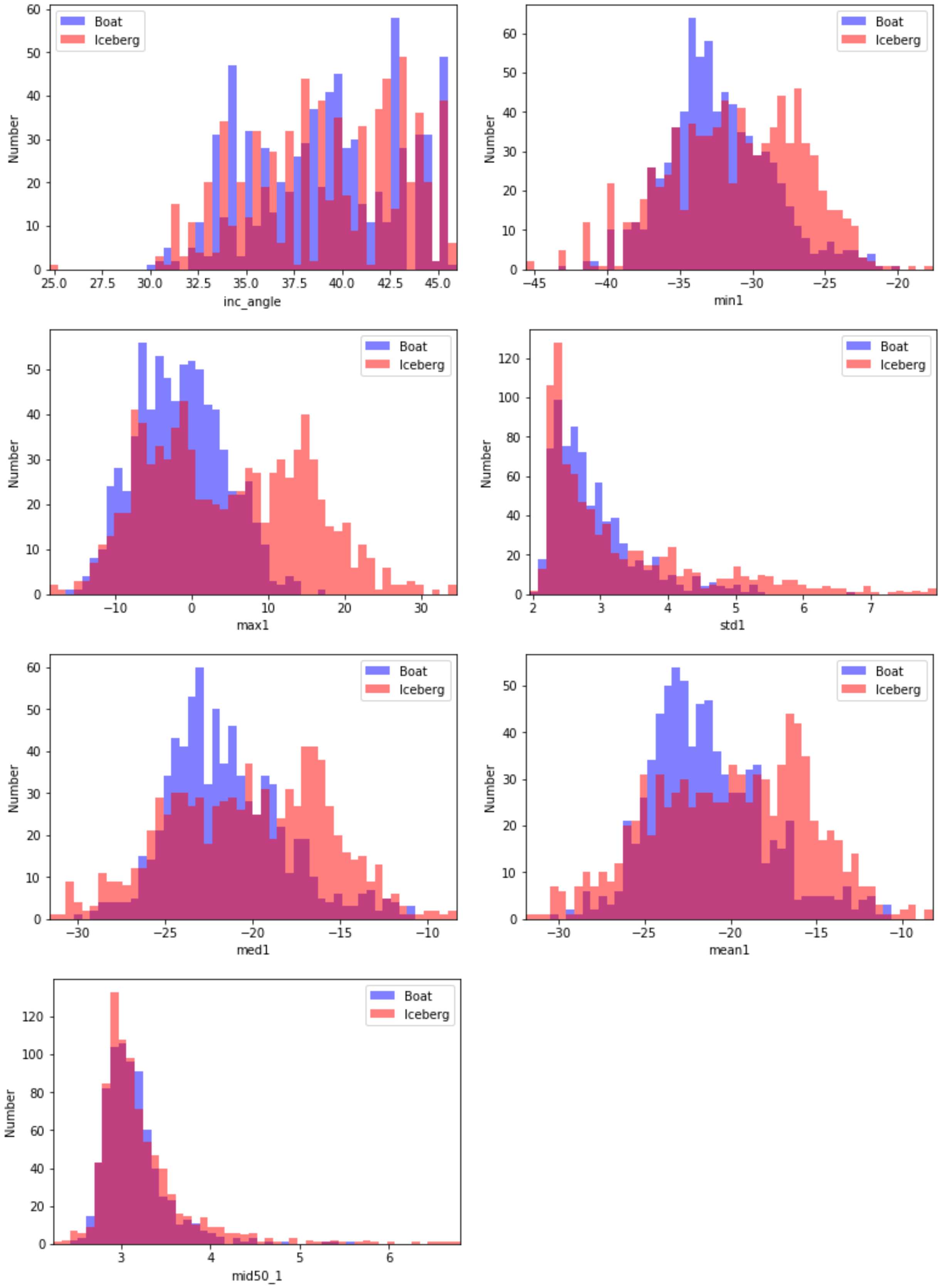


Judging by the 3D image of the two sample data, the iceberg is a rugged graph. In the case of a ship, it is a soaring graph with a simple shape and looks like a elongated point.

Global Image Stats

Below is a histogram of some global attributes of the image. These are the minimum and maximum values, the average, the median, and the 50% median range of the signal values. Because the file uses two different bands, this will be done separately for each band.

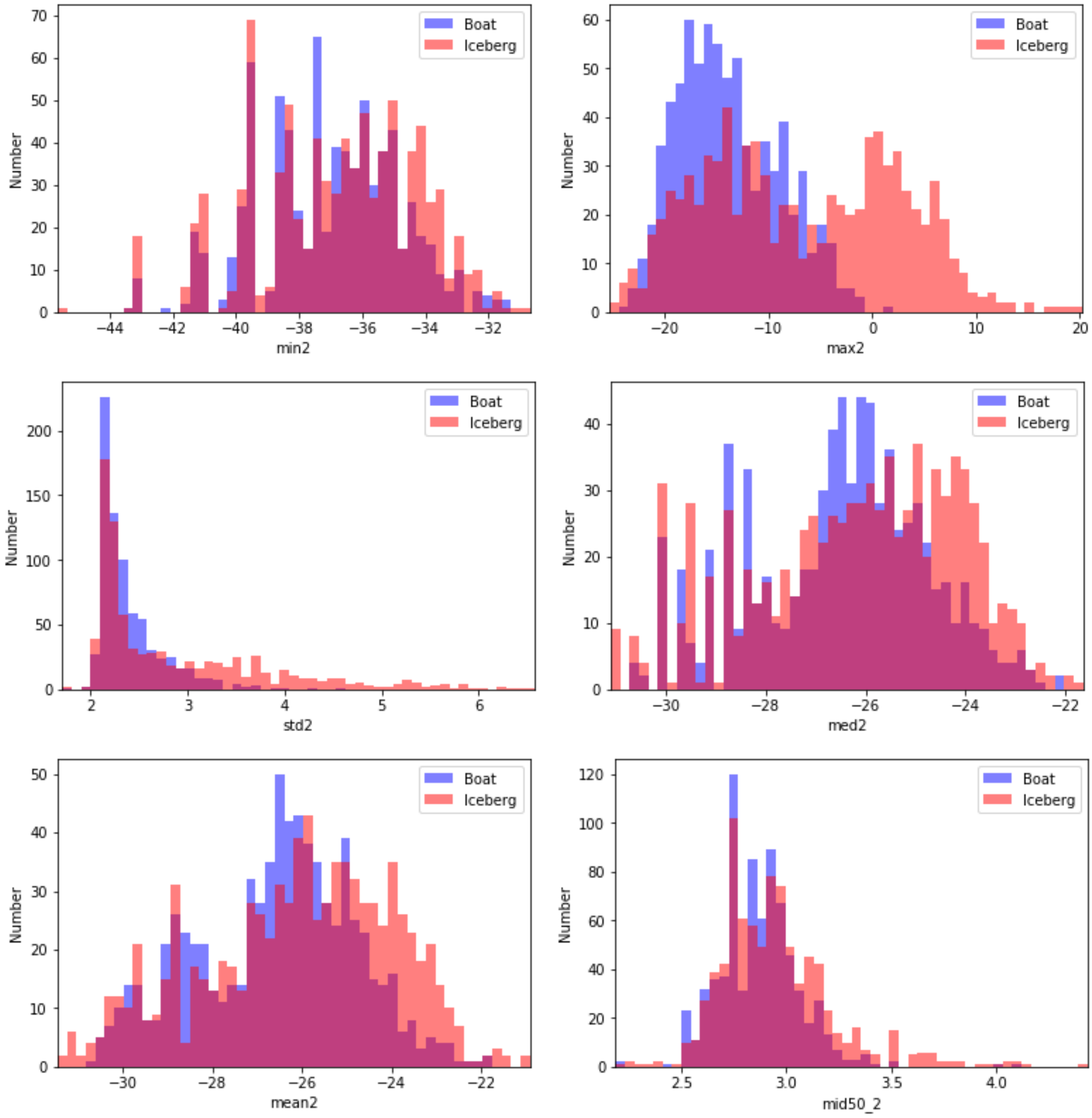
Histograms for band_1



There are some important differences for the first band. Medium 50% ranges are both the same size, but the minimum, maximum, standard deviation, median, and mean values all indicate a noticeable difference in some value ranges. Obviously, these basic variables are sensitive to what we are trying to measure. For example, if an iceberg is much larger than a ship, it could be expected to contain more pixels.

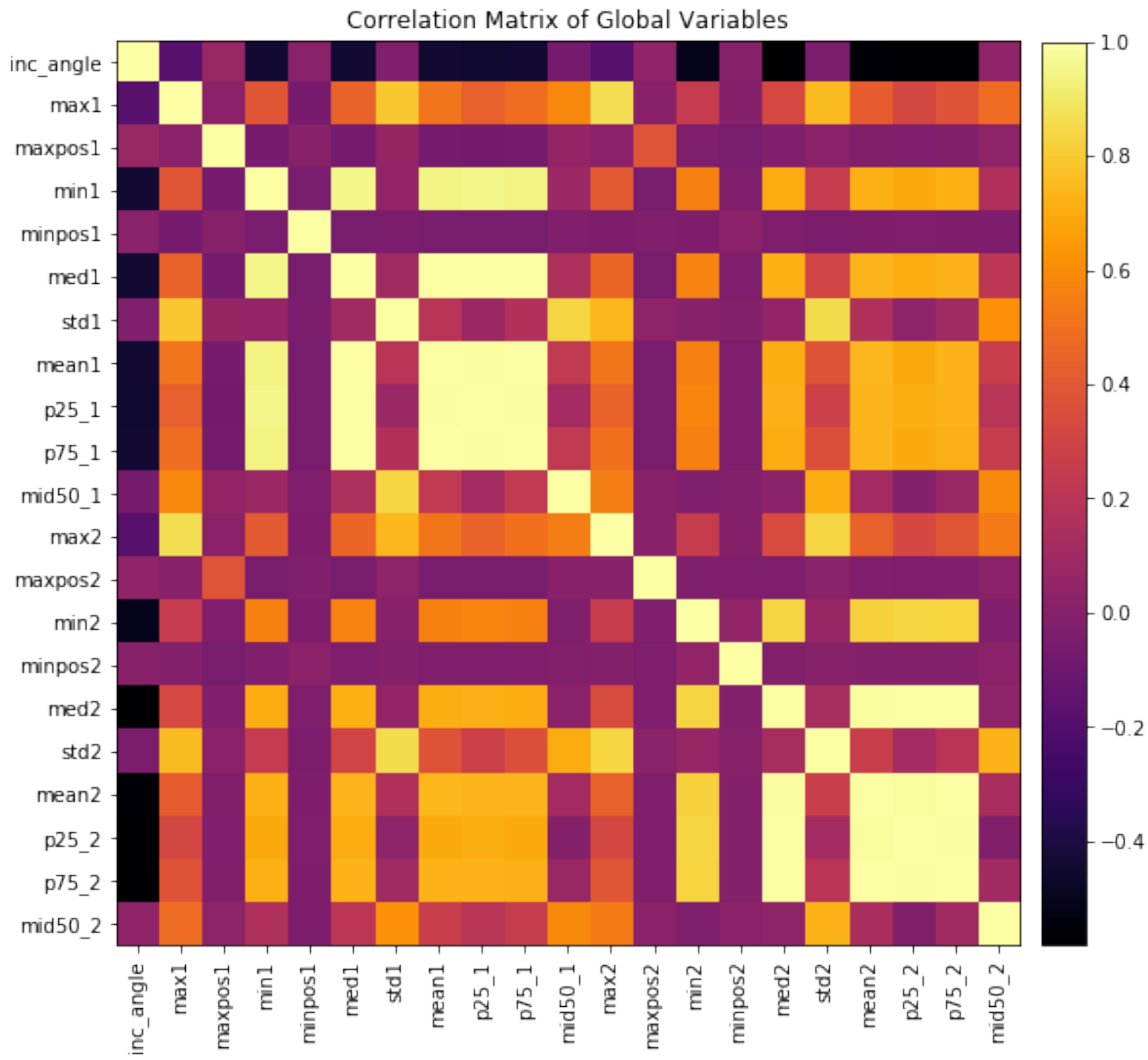
The second band is similar to the above.

Histograms for band_2



Correlations Between Features

The following is a correlation matrix for examining the correlation between variables.

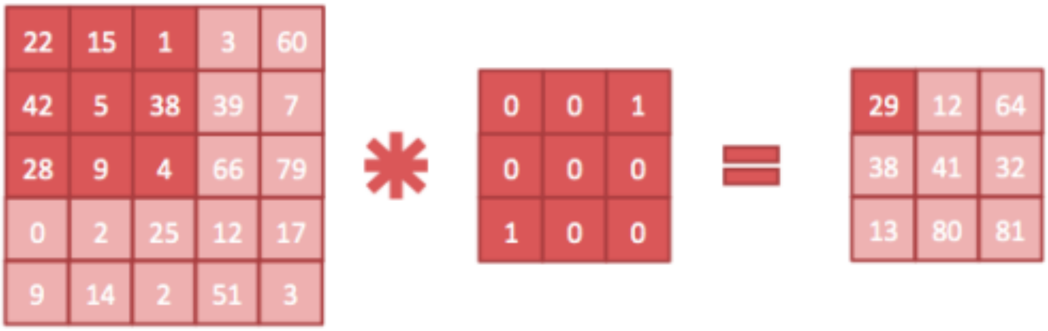


The above matrix graph shows which variables have a strong correlation. In particular, the mean, median, 25% and 75% signals are closely related and almost 75% correlated. Both minimum and maximum values have a very high correlation for band_1 and indicate that the signal may have been adjusted in some way to enhance this correlation and the minimum and middle values of both bands. The above matrix shows that there is a correlation between the two bands, and finally shows a correlation of -0.5 between band_2 average and angle and a weaker correlation to band_1.

Algorithms and Techniques

1. Convolution operation[5]

The convolution operation corresponds to a filter operation in the image processing. It moves the window of the filter at regular intervals and applies it to the input data.

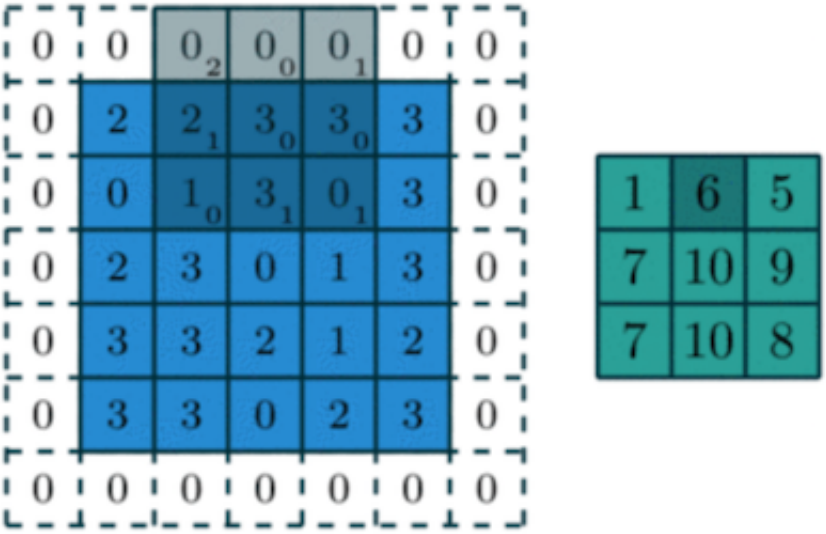


As shown in the figure above, the convolution operation applies a filter to the input data. In this example, the input data has dimensions in the vertical and horizontal directions. In this example, the input is (5,5), the filter is (3,3), and the output is (3,3). Filters may also be referred to as kernels, depending on the literature.

The convolution operation moves the window of the filter at regular intervals and applies it to the input data. The window here refers to the dark 3x3. As shown in this figure, multiply the input elements by the corresponding elements in the filter and obtain the sum. This process is performed everywhere and stored in the corresponding place in the output, and the output of the convolution operation is completed.

2. Padding

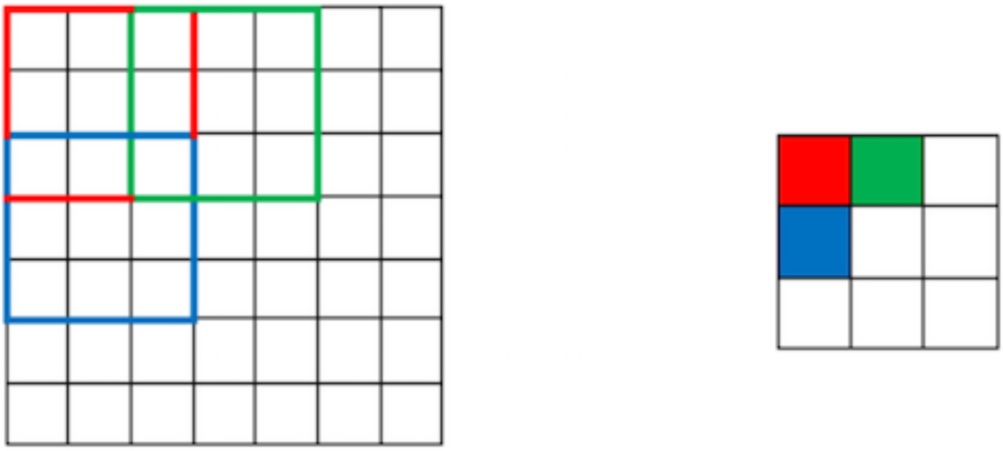
The surrounding of the input data may be filled with a specific value (e.g., 0) before performing the convolution operation. This is called padding, and this technique is often used in convolution operation.



Padding is mainly used to adjust the output size. For example, if you apply the (3,3) filter to (4,4) input data, the output will be (2,2), which is two less than the input. This can be a problem in deep-layer neural networks that repeatedly multiply the convolution operation.

3. Stride

The spacing of the locations where the filter is applied is called stride. For example, if stride is set to 2, the window to which the filter is applied moves by two spaces.



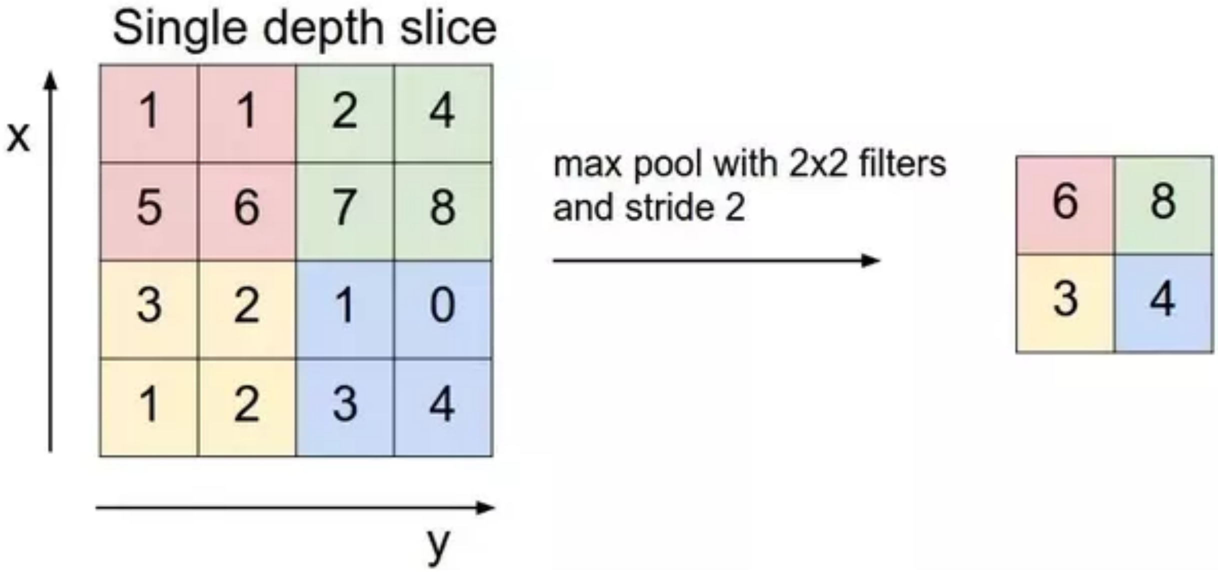
In the above figure, a filter with stride set to 2 is applied to input data of size (7, 7). In this way, stride specifies the interval at which to apply the filter. But the stride is 2, so the output is (3,3). This stride increases the output size. On the other hand, the larger the padding, the larger the output size.

4. Batch

In the neural network processing, input data is grouped into batches. Thus, the dimension of the data flowing through each layer is increased by one and stored as four-dimensional data. Specifically, data is stored in order of (data number, channel number, height, width). If the data is N, then every time four-dimensional data flows through the neural network, a convolution operation is performed for N data.

5. Pooling Layer

Pooling is an operation that reduces space in the vertical and horizontal directions. For example, as shown in the figure below, the 2x2 area is aggregated into one element to reduce the space size.

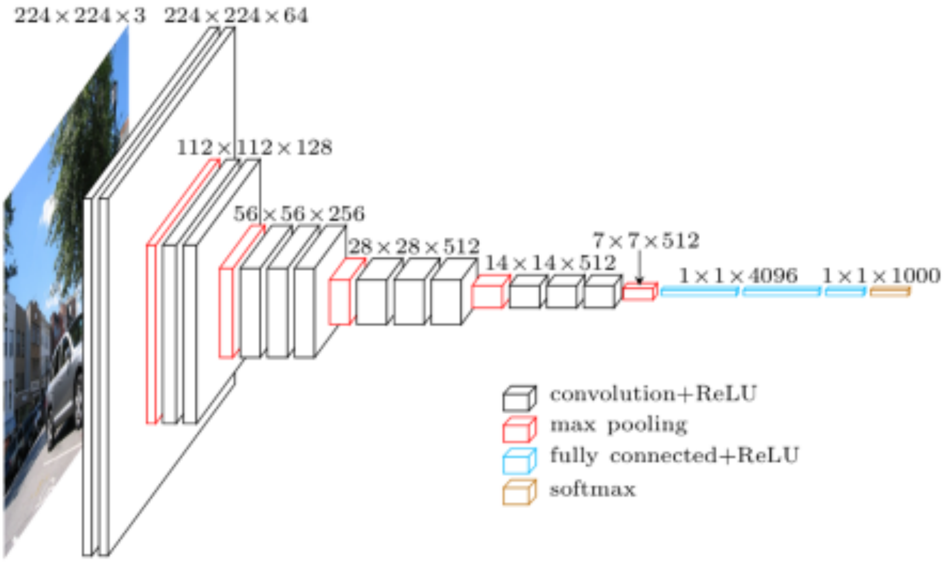


The above figure is the order of processing 2x2 max pooling with stride 2. The max pooling is the operation to obtain the maximum value, and '2x2' means the size of the target area. That is, 2x2 max pooling extracts one of the largest elements in the 2x2 area as shown in the figure. In addition, stride is set to 2 in this example, so a 2x2 window is moved by 2 elements of element. For reference, it is common to set the window size and stride of the pooling to the same value. For example, if the window is 3x3, the stride is set to 3.

The max pooling is the operation that takes the maximum value in the target area, while the average pooling computes the average of the target area. In the field of image recognition, the max pooling is mainly used.

6. VGG 16[6]

VGG is a 'basic' CNN consisting of a convolition layer and a pooling layer. However, as shown in the figure, both the convolution layer and the fully collected layer are deepened to 16 layers.



[7]

The point to note in VGG is that the convolution layer using a 3x3 small filter runs continuously. As shown in the figure, the pooling layer is repeated two or four times in succession to reduce the size to half. At the end, it passes through the fully connected layer and outputs the result.

7. Transfer Learning

Transfer learning is useful when there are few training dataset. It uses the weight values learned from the huge dataset provided by ImageNet for the dataset of this project. Weights copied from the ImageNet are copied to another neural network, and fine tuning is performed in that state.

In this project, a neural network with the same configuration as the VGG is prepared, the previously learned weight is set as the initial value, and fine tuning is performed on the new dataset. Therefore, this is a useful method when it is judged that training data is insufficient.

8. SGD

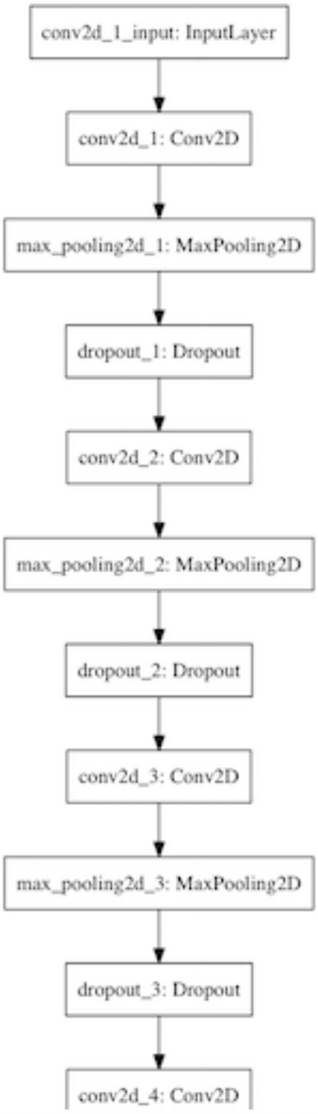
When calculating the loss function, using the whole train set is called Batch Gradient Descent. However, if you do this calculation, you will have to calculate the loss function for the whole data when you take a step. To prevent this, we usually use the method called Stochastic Gradient Descent (SGD). In this method, when calculating the loss function, the loss function is computed only for a collection of some small data (mini-batch) instead of whole data. This method may be somewhat inaccurate than the batch gradient descent, but because it is much faster to calculate, more steps can be taken at the same time, and if it is repeated many times, it usually converges to the result of the batch. Also, when using SGD, there is a possibility to converge to a better direction without falling into the local minima to be missing from Batch Gradient Descent.

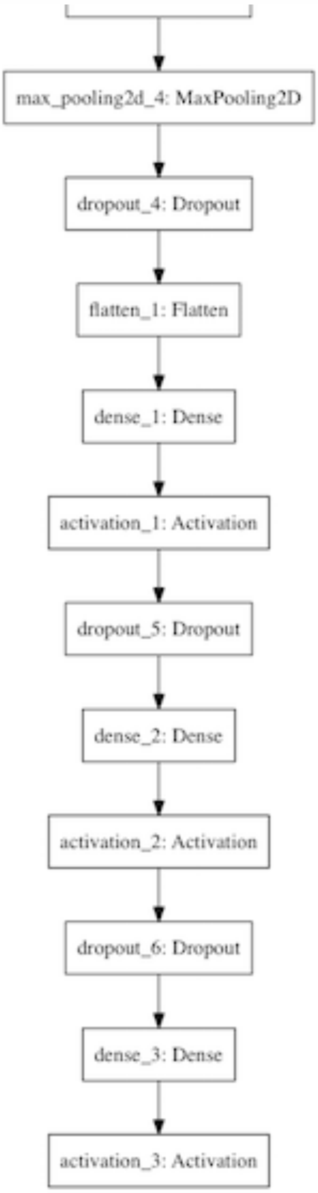
Usually, this SGD is used when training a Neural Network. However, there is a limitation in learning a network using a simple SGD. If you use the simple SGD, if it hits the local minima, the gradient becomes 0 and it can not move. However, if the Nesterov Momentum is applied, it is inertial to the direction of movement, so it can be expected to get out of this local minima and move to a better minima. It is also easier to braking at the right time to stop, while still enjoying the benefits of rapid movement.

Benchmark

The benchmark model is proposed by DeveshMaheshwari. This model consists of four convolution layers. The max pooling is performed for each convolution layer. Of the last three output layers, the first two layers use the ReLu activation function and the last layer uses the Sigmoid activation function.

The figure below shows the CNN architecture.





The accuracy for the benchmark model is 87.03%.

III. Methodology

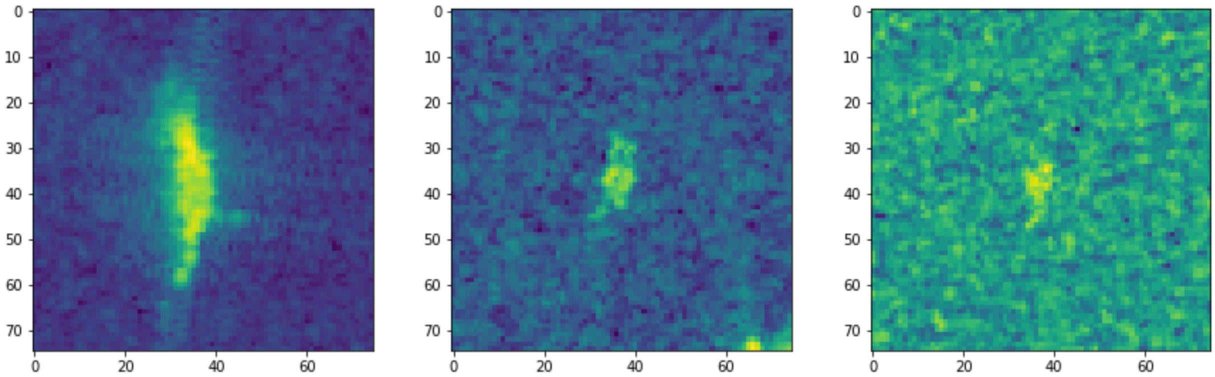
Data Preprocessing

Image Processing

The VGG-16 architecture requires images to be color images. That is, the image means the third dimensional dimension of the color. For example, an image 224 x 224 x 3 in size would mean an image with a height of 224 pixels, a weight of 224 pixels, and 3 RGB color information. However, band_1 and band_2 are both grayscale images and are 75 x 75 x 1 in size. In other words, it does not have three dimensions. So I need to find a way to format 75 x 75 x 3 images.

Frankly, the method I used to format the 75 x 75 x 3 image is actually a sample. Use band_1 as the first color dimension and band_2 as the second color dimension. I formatted the third color dimension using the average of band_1 and band_2. Finally, use the connection method inside numpy to format the 75 x 75 x 3 image.

The following is sample images after image processing.



Angle Processing

Because 133 angle values in the training dataset do not appear as 'na', I need to find a way to meet these missing values. These 'na's filled by pad method, which fill values forward.

Image Augmentation

Increase the number of training data for better learning and better predictions. To do this, augment the number of images through a transformation that flips or rotates the image. The Keras ImageDataGenerator used to transform the images. The parameters of the ImageDataGenerator are listed below.

- horizontal_flip: Boolean. Randomly flip inputs horizontally.
- vertical_flip: Boolean. Randomly flip inputs vertically.
- width_shift_range: Float (fraction of total width). Range for random horizontal shifts.
- height_shift_range: Float (fraction of total height). Range for random vertical shifts.
- channel_shift_range: Float. Range for random channel shifts.
- zoom_range: Float or [lower, upper]. Range for random zoom. If a float, `[lower, upper] = [1-zoom_range, 1+zoom_range]`.
- rotation_range: Int. Degree range for random rotations.

Implementation

Environment

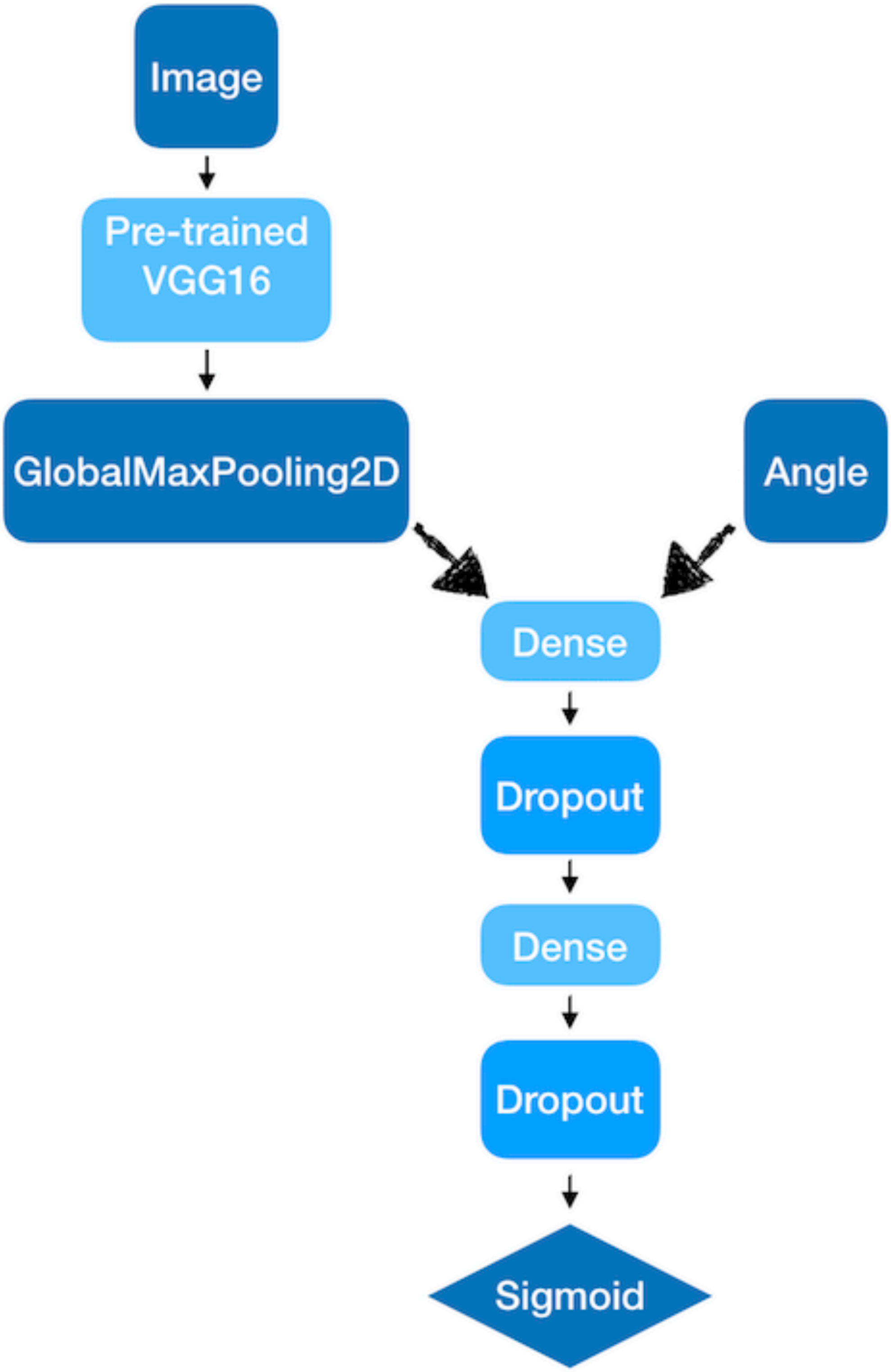
I used AWS EC2 for faster model training. I then build the model using Keras. The backend for the Keras in Tensorflow. The list for the required library is below.

- AWS EC2
    - Deep Learning AMI Ubuntu Linux
    - Instance type: p2.xlarge
- Jupyter Notebook
- Python 3.6
- Keras
- Tensorflow

- numpy
- pandas
- matplotlib
- plotly

Transfer Learning

As a final model, this project was carried out using the transfer learning method. The following is a detailed structure of the VGG16 model selected for transfer learning.



1. Pre-trained VGG16

The following arguments are set for the VGG16 model.

```
VGG16(weights='imagenet', include_top=False, input_shape=X_train.shape[1:],
classes=1)
```

- weights: `'imagenet'` (pre-training on ImageNet).
- include_top: whether to include the 3 fully-connected layers at the top of the network.
- input_shape: optional shape tuple, only to be specified if `include_top` is `False` (otherwise the input shape has to be `(224, 224, 3)` (with `'channels_last'` data format) or `(3, 224, 224)` (with `'channels_first'` data format). It should have exactly 3 inputs channels, and width and height should be no smaller than 48. E.g. `(200, 200, 3)` would be one valid value.
- classes: optional number of classes to classify images into.

2. GlobalMaxPooling2D

The data is extracted through the GlobalMaxPooling2d layer and concatenated with the angle data.

3. Dense

The following arguments are set for the Dense().

```
Dense(512, activation='relu', name='fc2')
```

- units: Positive integer, dimensionality of the output space.
- activation: Activation function to use (see activations). If you don't specify anything, no activation is applied (ie. "linear" activation: `a(x) = x`).

4. Dropout

Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

The following arguments are set for the Dropout().

```
Dropout(0.3)
```

- rate: float between 0 and 1. Fraction of the input units to drop.

5. Sigmoid

This model uses the sigmoid layer as the last layer, and it is appropriate to use the sigmoid layer because it is a binary classification that identifies whether the input image is a ship or an iceberg.

6. SGD

In order to find the parameters that make the value of the loss function as low as possible, it is important to find the optimal value of the parameter. Solving this problem is called optimization. This model will use SGD (Stochastic Gradient Descent) for this purpose.

I also compare it with SGD using Adadelta, another optimizer. This will be mentioned later.

Below are the arguments for SGD below.

```
 sgd = SGD(lr=1e-3, decay=1e-6, momentum=0.9, nesterov=True)
```

- lr: float >= 0. Learning rate.
- decay: float >= 0. Learning rate decay over each updat
- momentum: float >= 0. Parameter that accelerates SGD in the relevant direction and dampens oscillations.
- nesterov: boolean. Whether to apply Nesterov momentum.

7. StratifiedKFold

This cross validation object is a KFold variant that returns a stratified fold. Folds are created by maintaining a sample rate for each class.

```
folds = list(StratifiedKFold(n_splits=K, shuffle=True,
random_state=16).split(X_train, target_train))
```

Below are the parameters for StratifiedKFold().

- n_splits : Number of folds. Must be at least 2.

- shuffle : Whether to shuffle each stratification of the data before splitting into batches.

- random_state : If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random. Used when `shuffle` == True.

I tried to compare different folds to choose the most appropriate K size for the model.

Complications that occurred during the coding process

The following is about difficulties encountered during the coding process.

1. Time consumption

When using my MacBook Pro, the learning time was spent about two hours. Also, CUDA could not be installed because the MacBook Pro GPU is not NVIDIA. To solve this waste of time, I used AWS EC2 (Deep Learning AMI Ubuntu Linux) recommended by Udacity. This reduced model learning time by half.

2. Unfamiliar AWS EC2

I was trying AWS EC2 for the first time. Particularly, in EC2, iPython Notebook was used for coding. However, as I was doing this project, I became more and more accustomed to improve productivity on the time to learn the CNN model.

3. VGG16

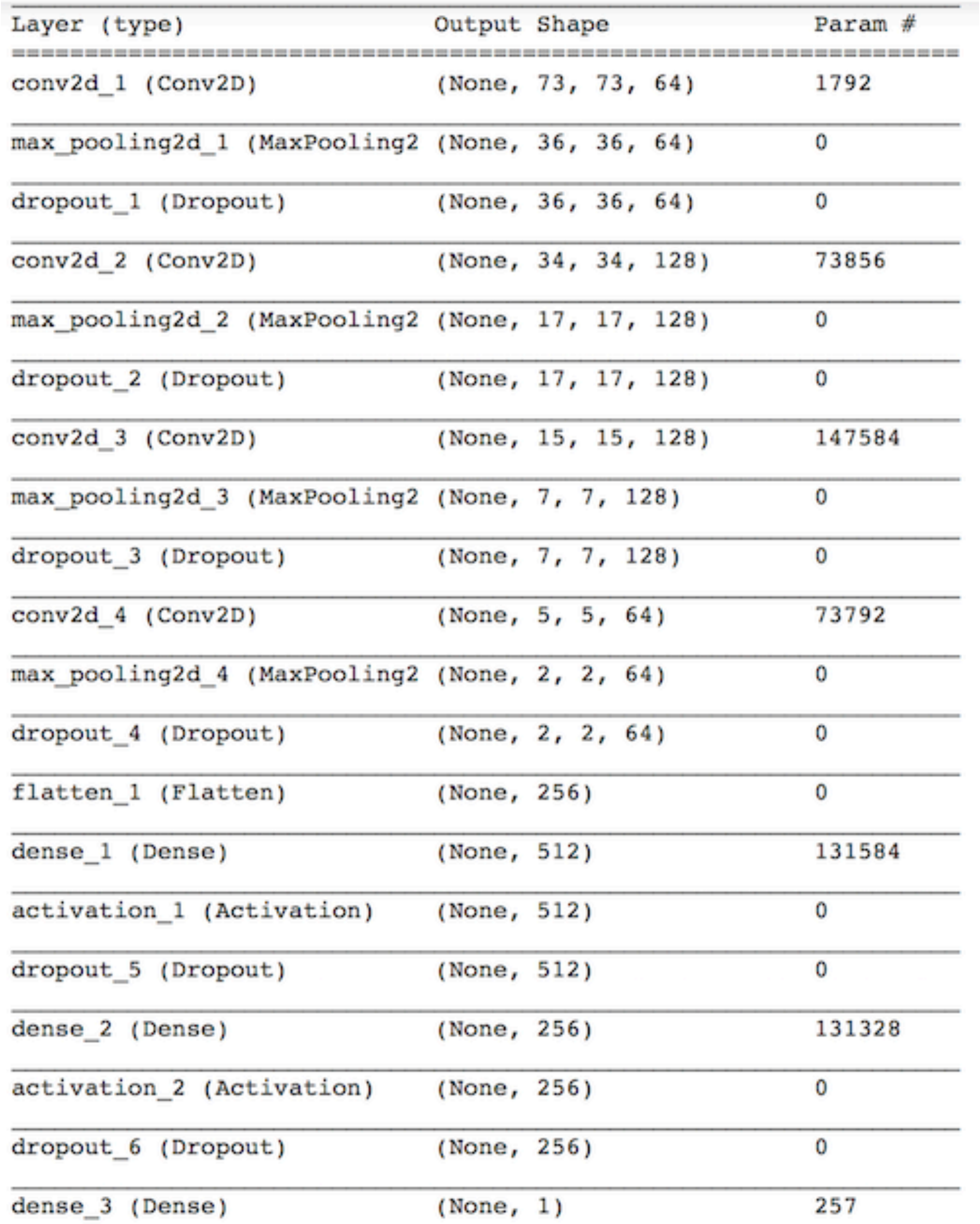
Finally, I have encountered difficulties in applying the VGG16 model to transfer learning. I tried to refer to Keras Documentation and the previous project, Dog Breed Classifier, but the project did not go on due to an error. In the meantime, I was able to transfer learning to VGG16 model with help of Kaggler DeveshMasheshwari kernel.

Refinement

After running the project first with the benchmark model, I made improvements in two ways. The first method is to increase the training data and the second is to perform the transfer learning using the VGG16 model.

1. Benchmark CNN Model: 87%
2. CNN Model with more train data images: 90.7%
3. Transfer learning with VGG16 model: 92.27%

Benchmark CNN Model



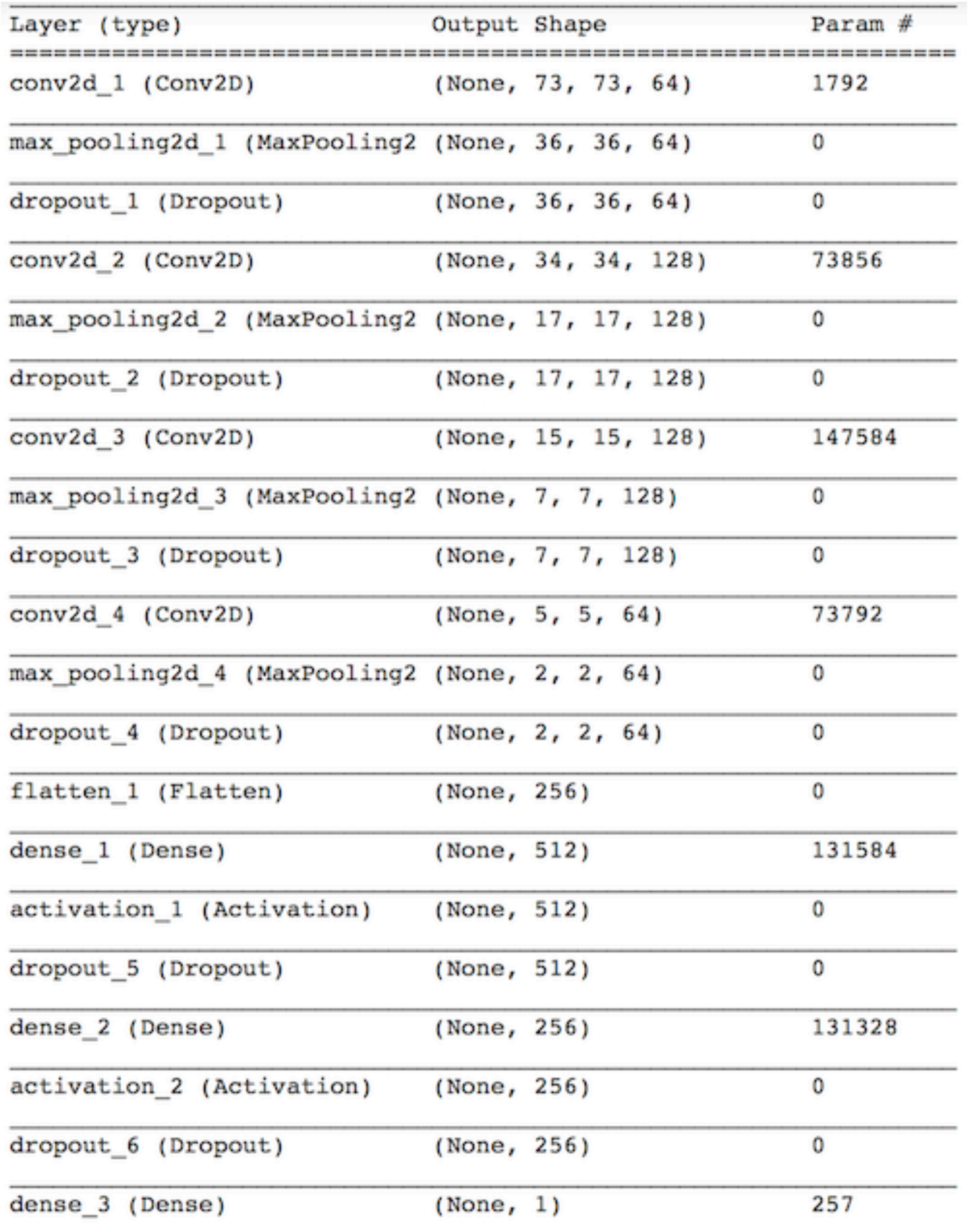
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 73, 73, 64)        1792
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 5, 64)          73792
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 2, 2, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
activation_2 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
```

The accuracy of the benchmark model can not exceed 90%. Accuracy in the two advanced models below shows that the accuracy has increased gradually.

```
Validation accuracy: 0.8703241902693846
```

CNN Model with more train data

This model increased training images through flip and rotate to increase insufficient training data. Therefore, the accuracy was improved by 3%.



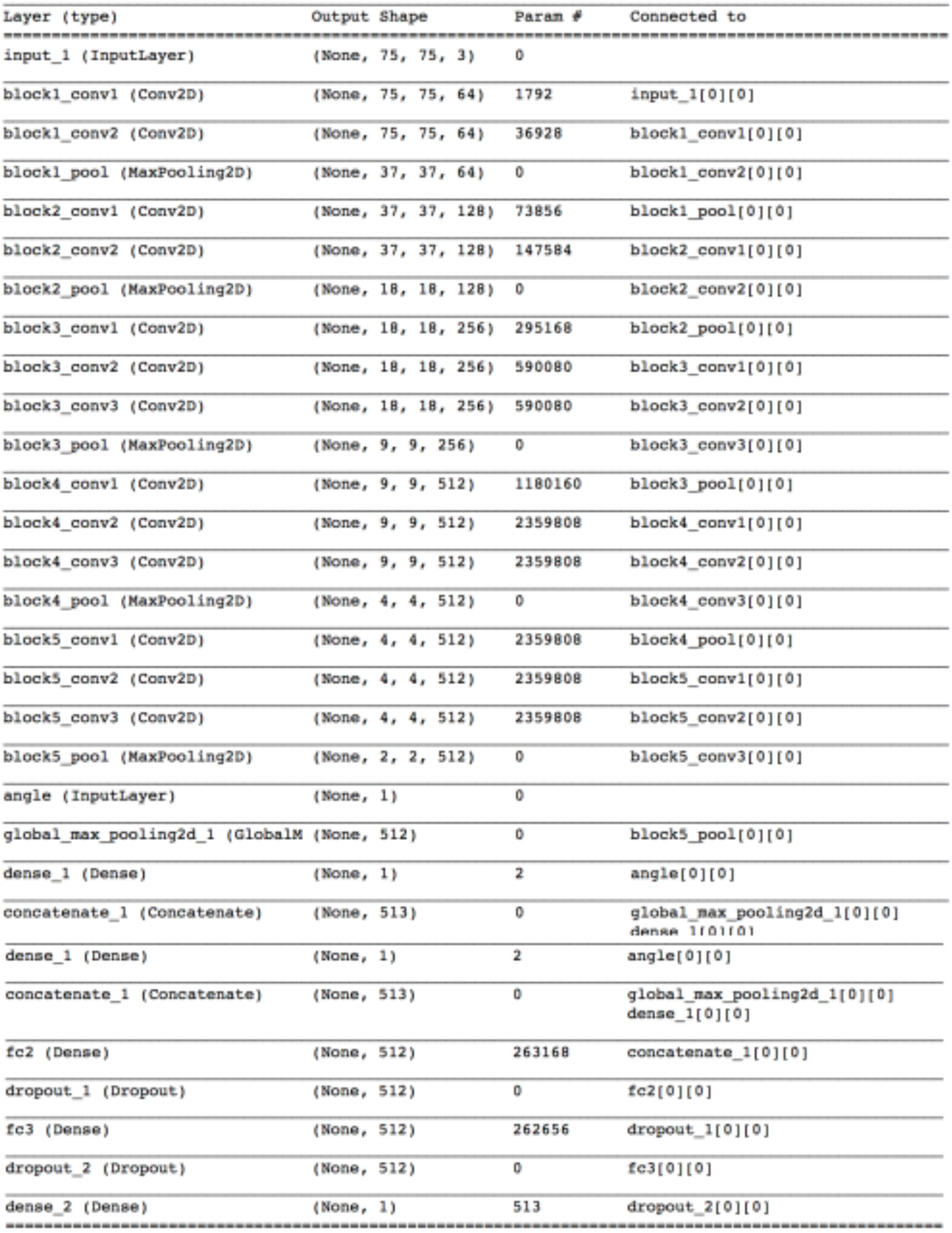
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 73, 73, 64)        1792
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 5, 64)          73792
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 2, 2, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
activation_2 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
```

```
Validation accuracy: 0.907608695652
```

VGG16 Model

As a final model, transfer learning was performed using VGG16 model. This resulted in a 2% increase in accuracy. It also uses K-fold cross validation to prevent over-summing and solve the problem of insufficient data volume.

Since data is only training set and test set, we use k-fold cross validation, a resampling technique, to increase the statistical reliability of the classifier performance measurement.



```
Layer (type)                     Output Shape         Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 75, 75, 3)    0
____________________________________________________________________________________________________
block1_conv1 (Conv2D)            (None, 75, 75, 64)   1792        input_1[0][0]
____________________________________________________________________________________________________
block1_conv2 (Conv2D)            (None, 75, 75, 64)   36928       block1_conv1[0][0]
____________________________________________________________________________________________________
block1_pool (MaxPooling2D)       (None, 37, 37, 64)   0           block1_conv2[0][0]
____________________________________________________________________________________________________
block2_conv1 (Conv2D)            (None, 37, 37, 128)  73856       block1_pool[0][0]
____________________________________________________________________________________________________
block2_conv2 (Conv2D)            (None, 37, 37, 128)  147584      block2_conv1[0][0]
____________________________________________________________________________________________________
block2_pool (MaxPooling2D)       (None, 18, 18, 128)  0           block2_conv2[0][0]
____________________________________________________________________________________________________
block3_conv1 (Conv2D)            (None, 18, 18, 256)  295168      block2_pool[0][0]
____________________________________________________________________________________________________
block3_conv2 (Conv2D)            (None, 18, 18, 256)  590080      block3_conv1[0][0]
____________________________________________________________________________________________________
block3_conv3 (Conv2D)            (None, 18, 18, 256)  590080      block3_conv2[0][0]
____________________________________________________________________________________________________
block3_pool (MaxPooling2D)       (None, 9, 9, 256)    0           block3_conv3[0][0]
____________________________________________________________________________________________________
block4_conv1 (Conv2D)            (None, 9, 9, 512)    1180160     block3_pool[0][0]
____________________________________________________________________________________________________
block4_conv2 (Conv2D)            (None, 9, 9, 512)    2359808     block4_conv1[0][0]
____________________________________________________________________________________________________
block4_conv3 (Conv2D)            (None, 9, 9, 512)    2359808     block4_conv2[0][0]
____________________________________________________________________________________________________
block4_pool (MaxPooling2D)       (None, 4, 4, 512)    0           block4_conv3[0][0]
____________________________________________________________________________________________________
block5_conv1 (Conv2D)            (None, 4, 4, 512)    2359808     block4_pool[0][0]
____________________________________________________________________________________________________
block5_conv2 (Conv2D)            (None, 4, 4, 512)    2359808     block5_conv1[0][0]
____________________________________________________________________________________________________
block5_conv3 (Conv2D)            (None, 4, 4, 512)    2359808     block5_conv2[0][0]
____________________________________________________________________________________________________
block5_pool (MaxPooling2D)       (None, 2, 2, 512)    0           block5_conv3[0][0]
____________________________________________________________________________________________________
angle (InputLayer)               (None, 1)            0
____________________________________________________________________________________________________
global_max_pooling2d_1 (GlobalM  (None, 512)          0           block5_pool[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 1)            2           angle[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 513)          0           global_max_pooling2d_1[0][0]
                                                                  dense_1[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 1)            2           angle[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 513)          0           global_max_pooling2d_1[0][0]
                                                                  dense_1[0][0]
____________________________________________________________________________________________________
fc2 (Dense)                      (None, 512)          263168      concatenate_1[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 512)          0           fc2[0][0]
____________________________________________________________________________________________________
fc3 (Dense)                      (None, 512)          262656      dropout_1[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 512)          0           fc3[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 1)            513         dropout_2[0][0]
====================================================================================================
```

```
FOLD= 0
Test accuracy: 0.938271604938
FOLD= 1
Test accuracy: 0.95652173913
FOLD= 2
Test accuracy: 0.913043478261
FOLD= 3
Test accuracy: 0.91875
FOLD= 4
Test accuracy: 0.94375
FOLD= 5
Test accuracy: 0.9375
FOLD= 6
Test accuracy: 0.90625
```

```
FOLD= 7
Test accuracy: 0.925
FOLD= 8
Test accuracy: 0.875
FOLD= 9
Test accuracy: 0.9125

Test Log Loss Validation=  0.193213699335
```

The average of the accuracy at each fold is:

92.27% (0.9226586822329)

This is a much better result than the previous models.

IV. Results

Model Evaluation and Validation

Both of the improved models used techniques to solve the less training data. Technologies for this are as follows.

- Image Augmentation
    - Gain insufficient training data through image augmentation
- K-fold Cross Validation
    - To improve the statistical reliability of the classifier performance measurement by resampling.

In addition to the improvements through the above two methods, I think that using the VGG16 model to increase the test set accuracy to 92.27% is a reasonable project result.

Choise of K in K-Fold Cross Validation

In order to select the most suitable number of k for the model, we derive the result by changing the value of k as follows.

1. K = 3

```
FOLD= 0
Test accuracy: 0.927102805521

FOLD= 1
Test accuracy: 0.934579440589

FOLD= 2
Test accuracy: 0.902621723293

Test Log Loss Validation=  0.206955990373
```

The mean of the accuracy at each fold is:

92.14% (0.921434656467667)

The variance value is:

0.000186279905974

The iPython notebook for this result can be found here.

2. K = 5

```
FOLD= 0
Test accuracy: 0.934782608696
FOLD= 1
Test accuracy: 0.529595015576
FOLD= 2
Test accuracy: 0.93769470405
FOLD= 3
Test accuracy: 0.896875
FOLD= 4
Test accuracy: 0.8875

Test Log Loss Validation=  0.307833130261
```

The mean of the accuracy at each fold is:

83.73% (0.8372894656644)

The variance value is:

0.0240667066218

The iPython notebook for this result can be found here.

3. K = 7

```
FOLD= 0
Test accuracy: 0.926086956522
FOLD= 1
Test accuracy: 0.934782608955
FOLD= 2
Test accuracy: 0.921739130435
FOLD= 3
Test accuracy: 0.960869565736
FOLD= 4
Test accuracy: 0.899122807018
FOLD= 5
Test accuracy: 0.899122807018
FOLD= 6
Test accuracy: 0.912280701754

Test Log Loss Validation=  0.194352943866
```

The mean of the accuracy at each fold is:

92.20% (0.922000653919714)

The variance value is:

0.00040460088166

The iPython notebook for this result can be found here.

4. K = 10

```
FOLD= 0
Test accuracy: 0.938271604938
FOLD= 1
Test accuracy: 0.95652173913
FOLD= 2
Test accuracy: 0.913043478261
FOLD= 3
Test accuracy: 0.91875
FOLD= 4
Test accuracy: 0.94375
FOLD= 5
Test accuracy: 0.9375
FOLD= 6
Test accuracy: 0.90625
FOLD= 7
Test accuracy: 0.925
FOLD= 8
Test accuracy: 0.875
FOLD= 9
Test accuracy: 0.9125

Test Log Loss Validation=  0.193213699335
```

The mean of the accuracy at each fold is:

92.27% (0.9226586822329)

The variance value is:

0.000481258375864

The iPython notebook for this result can be found here.

The highest test accuracy among the total of 4 experiments is when k is 10. The smaller the Log Loss Validation value is, the better the result. When k is 7 and 10, the latter has slightly better results. Therefore, k-fold CV when k is 10 is the most robust model.

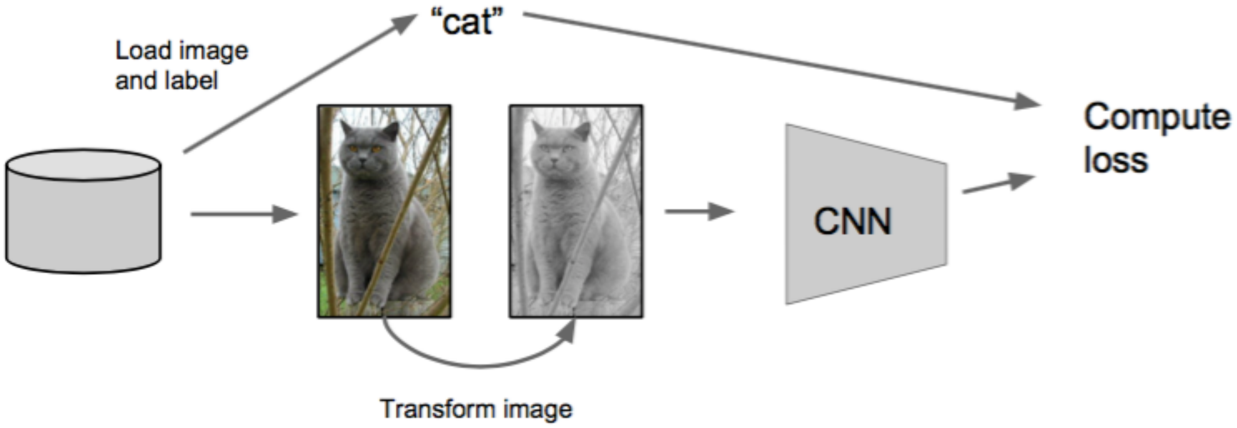
However, when I check each variation value, the variation is smallest at k = 3, and 7 and 10 are not much different from each other. This result is contrary to the common saying that "the smaller the k, the greater the variance". Nevertheless, since the lowest value of Test Log Loss is 10, I think that the model with k = 10 is the most robust.

Justification

The final results are found stronger than the benchmark result reported earlier. The accuracy of the benchmark model was 87%, but it improved to 92.27% accuracy with two model development.

Why Image augmentation is an important solution in solving this project.

Image augmentation, or data augmentation, is a method used to improve CNN performance. If you move all the pixels in the cat image one space to the right, it looks the same in the human eye. However, because the computer expresses and recognizes the image as a pixel vector, the cat image moved by one pixel is recognized as different from the original image. To solve this problem, we use data augmentation.



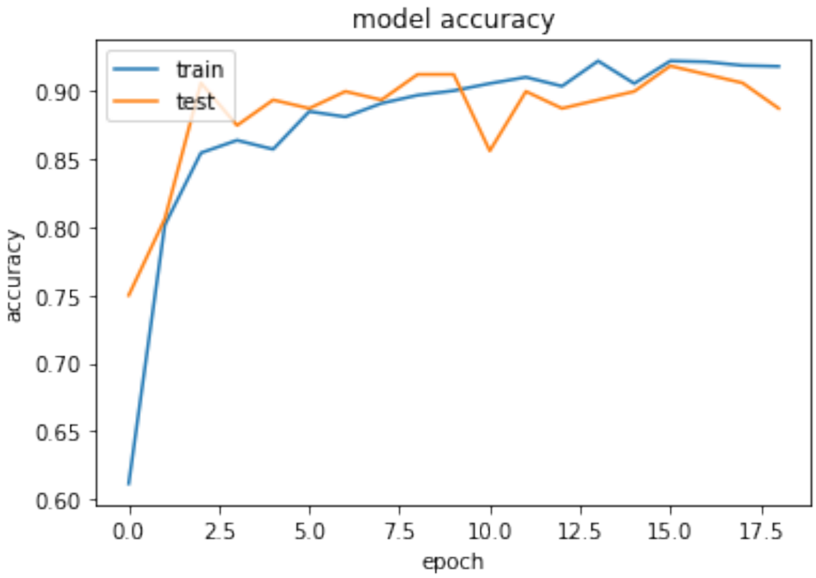
8

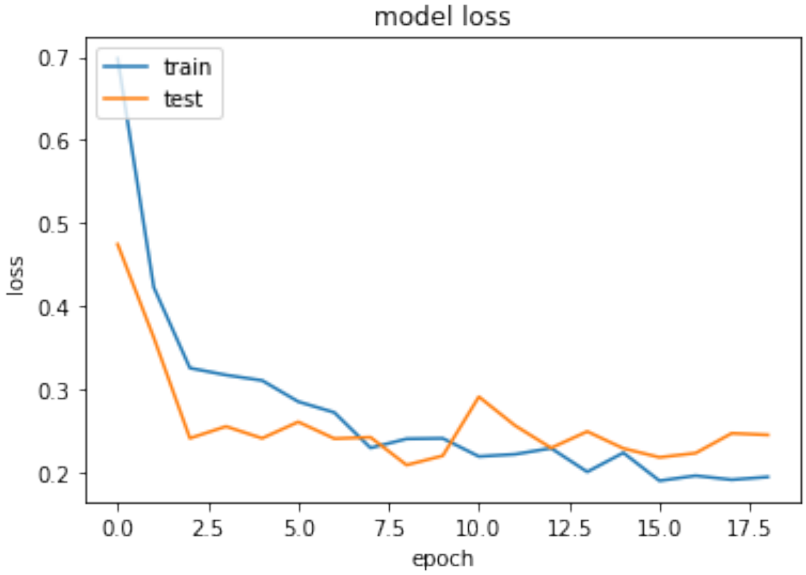
To summarize again, data augmentation is a method of changing a pixel without changing the label of the image, and proceeds with learning using the modified data. Also, almost all network models to date have used data augmentation, which is a very common method.

So the image data of this project also helped to supplement the lack of training data through data augmentation and improve the performance of CNN.

Model Accuracy and Model Loss

The following two graphs show the performance of the final model. Accuracy increases with each epoch. The loss value indicates how good or bad a particular model is each time the optimization is repeated. Ideally, the loss value is expected to decrease after each or several repetitions.

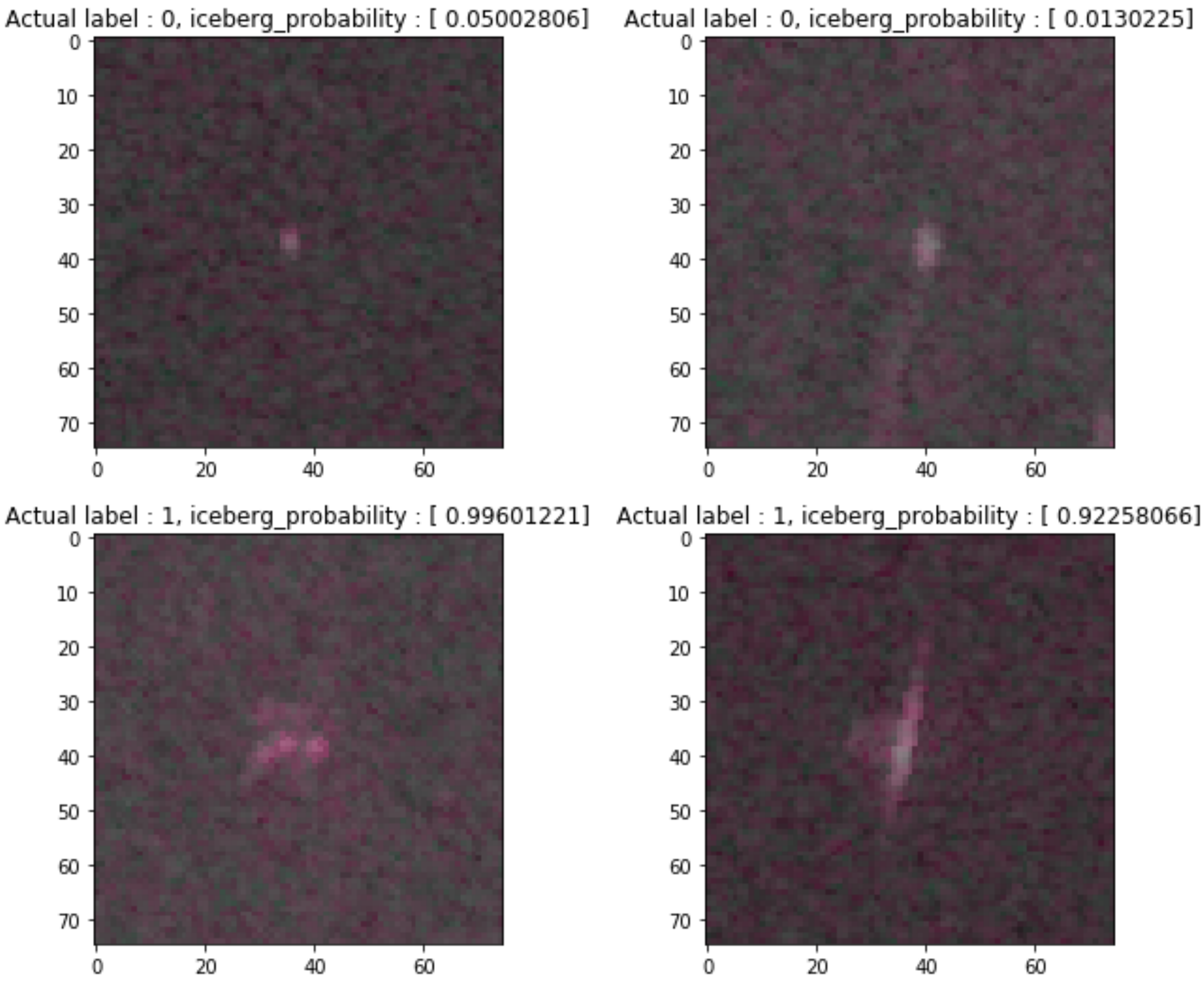




V. Conclusion

Free-Form Visualization

The predicted results of the final model were compared with the actual label values. Once again, when the label value is 1, it is iceberg. Let's look at the predictions for the four image data below. It can be seen that iceberg_probability is not much different from the actual label value. In other words, all of the following results are successful predictions.


Actual label : 0, iceberg_probability : [0.05002806]
Actual label : 0, iceberg_probability : [0.0130225]
Actual label : 1, iceberg_probability : [0.99601221]
Actual label : 1, iceberg_probability : [0.92258066]
0
10
20
30
40
50
60
70
0
20
40
60

Reflection

First, I found an interesting data set in Kaggle, and I was able to do a good project. This project is a binary image classification problem. So, first, I decided to use the CNN model. We then navigated through the data set, preprocessed all the data, and then visualized the data set. Visualization gave me a deeper understanding of the data.

To improve the existing benchmark model, we tried to change the image augmentation and various parameter values, but decided to use transfer learning using the VGG16 model in a more innovative way. Comparing the results of this model with previous models, we can see that the performance is better.

A key part of this project is finding ways to produce classifiers with good accuracy through insufficient training data. I think this has been solved through transfer learning, k-fold cross validation and image augmentation.

Improvement

Change Optimizer

I used Adadelta, an optimization technique other than SGD, to improve performance.

```
adadelta = Adadelta(lr=1e-3, rho=0.95, epsilon=1e-6, decay=1e-6)
model.compile(loss='binary_crossentropy', optimizer=adadelta, metrics=
['accuracy'])
```

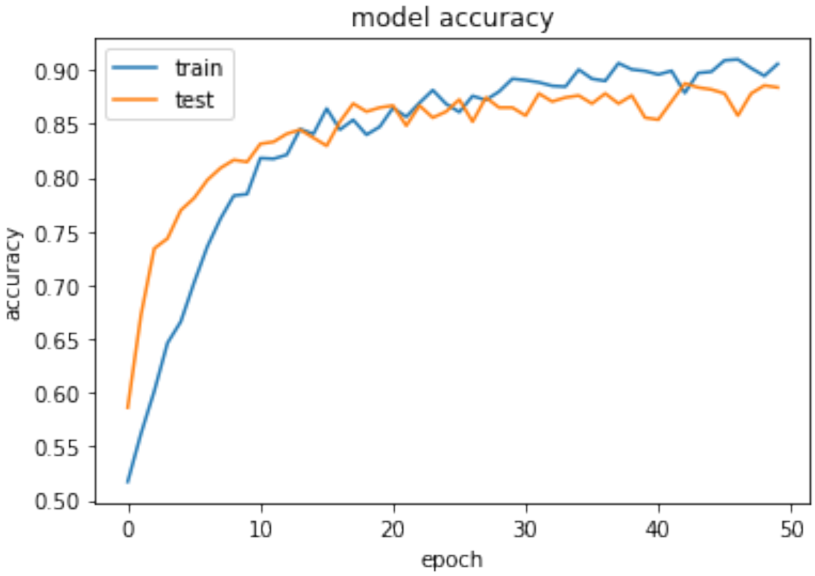
Below is the test accuracy of each fold of K-fold cross validation.

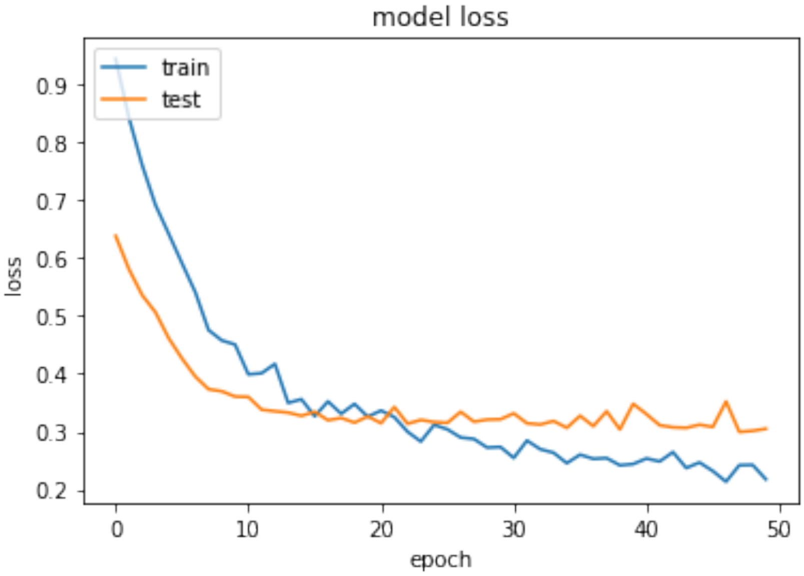
```
FOLD= 0
Test accuracy: 0.914018692146
FOLD= 1
Test accuracy: 0.923364487318
FOLD= 2
Test accuracy: 0.878277154674
```

The average of the above three accuracies is:

90.52% (0.905220111379333)

Based on test accuracy, the above improvements are not likely to improve on the final model, but are likely to have higher performance. In fact, there is potential to be a better model when looking at model loss and accuracy graphs. Therefore, I plan to find a better model by experimenting with various parameters for this.





1. Fukushima, K. (1979). Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position- Neocognitron. *Electron. & Commun. Japan*, *62*(10), 11-18.↩

2. Alexandrov, V., Volkov, V., Sandven, S., & Kloster, K. (2008). Detection of Arctic icebergs on the basis of satellite SAR. In *The 2 intern Workshop on advances in SAR Oceanography from Envisat and ERS missions. ESA-ESRIN. 21-25 January*.↩↩

3. C. Bentes et al. (2016). Ship-Iceberg Discrimination with Convolutional Neural Network in High Resolution SAR Images.↩

4. Godil, A., Bostleman, R., Shackleford, W., Hong, T., & Shneier, M. (2014). *Performance Metrics for Evaluationg Object and Human Detection and Tracking Systems*. US Department of Commerce, National Institute of Standards and Technology.↩

5. WANG, Haohan; RAJ, Bhiksha; XING, Eric P. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017.↩

6. SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.↩

7. A BRIEF REPORT OF THE HEURITECH DEEP LEARNING MEETUP #5. (2016). Heuritech Blog. Retrieved January 28, 2018, from https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/↩

8. CS231n: Convolutional Neural Networks for Visual Recognition (2017). Stanford Vision Lab. Retrieved January 28, 2018, from http://cs231n.stanford.edu/syllabus.html↩