An Iceberg or a Ship?

A Deep learning project
For CSCI6908
At
Dalhousie University

By
Shakti Singh (B00779881)
Nikhil Dhirmalani (B00775542)

Abstract

This project addresses the problem of classification of icebergs from other entities in the sea like a Ship. The problem is borrowed from the original competition page on Kaggle.com. The data comprise of satellite images of objects, having one object per image and their respective labels. To solve this problem, we applied different approaches but first of all, a comprehensive analysis of the images in the dataset is done. This analysis was useful in understanding the data and to gain insight on how to proceed further. During the initial phase we did feature engineering on our channels and found that maximum pixel intensity for band-2 separated the ship-iceberg distribution better, So we implemented MLP to learn the function separating the distribution. But as we were given only 1604 images in train set we were not able to generalize it better. After Analysis, we constructed a CNN model in tensorflow that used some convolution layers and a dense layer to learn the underlying pattern. The results on this network were not so impressive. Further, we used transfer learning to finetune pre-trained VGG16 on our dataset, the results of this approach were much better. We also implemented the architecture of CNN from the kernel that had best results on kaggle and used it as the best model (Ground truth) to compare our results of our models on test set which comprised of 8000 images. After implementing various CNN models we tried to implement deep network using RESNET architecture. We implemented our own RESNET model in keras having 13 layers with skip connections after every two layers. We applied data augmentation techniques like flip, rotation etc to increase our train data and trained our RESNET model which produced good results.

Table of Contents

1. Introduction

The problem of drifting icebergs posing a threat to ships in the sea has been there ever since man decided to cross the seas. This problem is more prevalent in the waters of Atlantic Canada and other coastal areas of North Atlantic. The icebergs pose a threat to not only the ships but also hinder navigation and offshore activities. Several Institutions and companies use air-based monitoring systems to track icebergs in coastal areas, but these methods are ineffective in remote areas of the sea particularly in harsh weather. This is where satellite imagery kicks in. Modern satellites use C-Band radars that can apparently see through darkness, rain or clouds. These radars are mounted on satellites of Sentinel-I constellation, orbiting earth 600 km above the earth surface 14 times a day, monitoring seas and land.

Having satellite images, it is still a tedious task to manually look at the images and classify them as icebergs or ships. There are certain examples where even the human eye fails to identify the object. Looking at the nature of the problem, it seems promising to make use of machine learning to automate the task of classification. This will be helpful in an efficient classification of the object in seas, within a reasonable time. It can save lives in certain cases.

In order to solve this problem, we tried four approaches of deep learning. These approaches include a convolution Neural network in tensorflow, Transfer learning of pretrained VGG16, implementation of ResNet 50 and an MLP based classifier trained on additional extracted features. The best results that we got are from ResNet 50 implementation. It, when trained on original data and augmented data gave out an accuracy of 89%. Before training the models, a comprehensive analysis of data was done in which we found that the data was very noisy. Some methods were applied to eliminate the noise which is discussed in later sections. The CNN implementation in tensorflow, although implemented correctly, failed to learn the underlying pattern. This was not realized until the later days of the project, more about this is talked upon in later sections.

2. Related Work

The paper was published in IEEE for the following competition and we read that paper and found that they implemented SVM and CNN architecture for the same problem and compared their results and concluded that CNN takes more time in training but perform better in comparison to SVM.

Paper: https://elib.dlr.de/99079/2/2016_BENTES_Frost_Velotto_Tings_EUSAR_FP.pdf

We have also skimmed through many kernels on Kaggle and they have used different CNN architecture like Dense Net, etc. but we implemented RESNET 50 in keras and removed the layers and made it to 13 and implemented the network. We have discussed and explained the architecture of the best model in future works in which they implemented extensive CNN architecture that consisted of over 100+ customized CNN's and VGG like architectures and then combined the results from these models using both greedy blending and two-level stacking with other image features.

3. Data

The data that is being used in this project is extracted from the dataset page of the kaggle competition page. The data is presented in JSON format files containing list of images. These images are flattened in the arrays of 5625 values (75x75 images).

	band_1	band_2	id	inc_angle	is_iceberg
0	[-27.878361, -27.15416, -28.668615, -29.537971...	[-27.154118, -29.537888, -31.0306, -32.190483,...	dfd5f913	43.9239	0
1	[-12.242375, -14.920305, -14.920363, -12.66633...	[-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

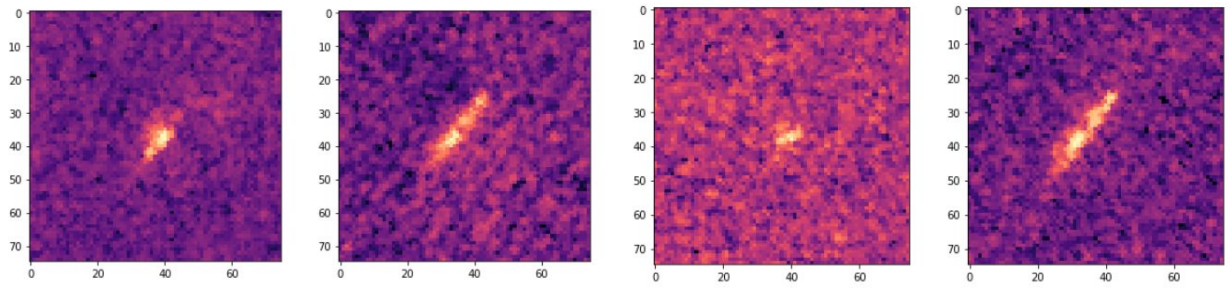
The field in the dataset are described below:

- id : id of the respective image
- band_1 : flattened values of a 75x75 images i.e a vector of 5625 values. These values are not the same as ordinary pixel values, but are negative float integers that have a unit db associated to them. These values are loss in backscatter radiation received by the radar after it is reflected from the object.
- band_2 : similar structure as that of band_1, but the key difference between the two bands is the different polarization at a particular incidence angle. Band_1 corresponds to HH (transmit/receive horizontally) and band_2 corresponds to HV(transmit horizontally receive vertically)
- inc_angle : this field records the incidence angle at which the images were taken.
- 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`.

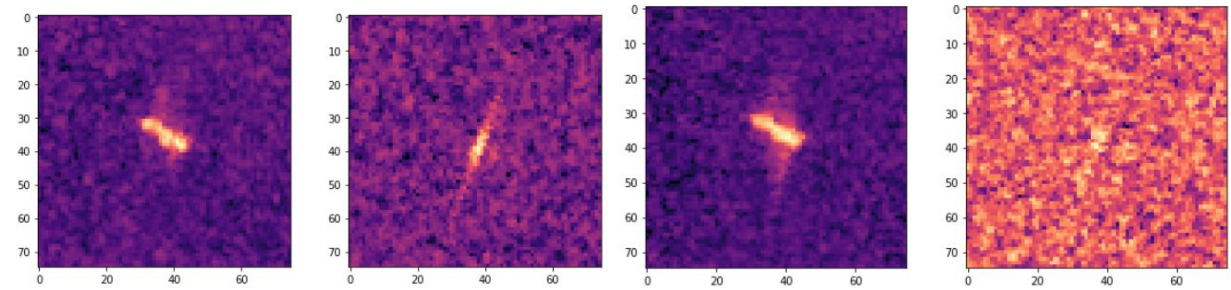
```
Number of examples in training dataset :  1604
Values in band_1 :  5625
Values in band_1 :  5625
Total examples labelled as iceberg :  753
Total examples labelled as not iceberg :  851
```

The data is somewhat evenly distributed among two classes, having a 45%-55% distribution. An even distribution is always helpful in classification as it does not let the model induce a bias for one class. On analysing the data further, we found out that the band_2 of the images is particularly noisy in nature.

Band_1 (iceberg) band_2 (iceberg)

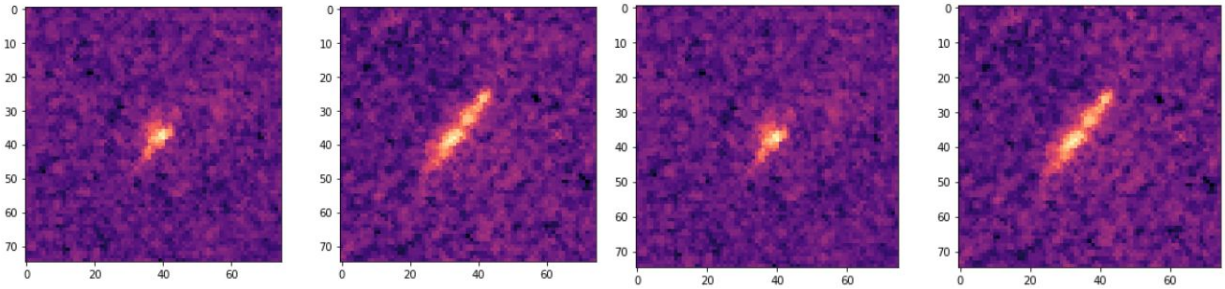


Band_1 (ships) band_2 (ships)



To overcome the problem of noise, we tried to aggregate the values of two available bands into a third channel. Among the mean of the two bands and simple sum of values in two bands, sum was better in eliminating noise according to our point of view.

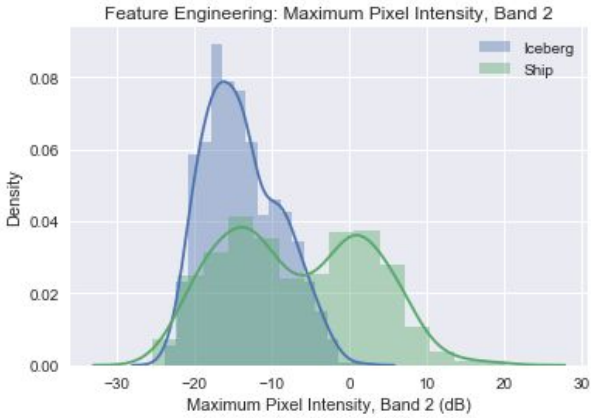
Band_1 + band_2 (icebergs) avg(band_1+band_2)

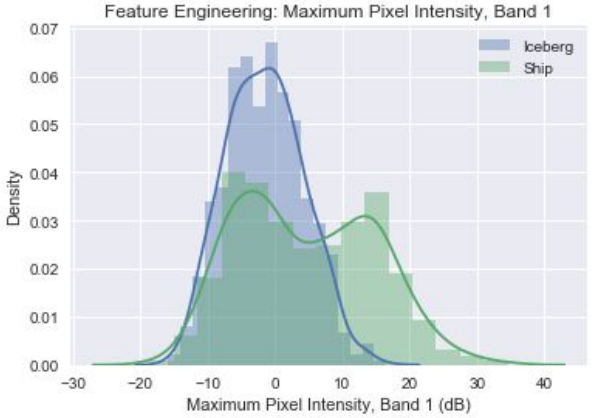


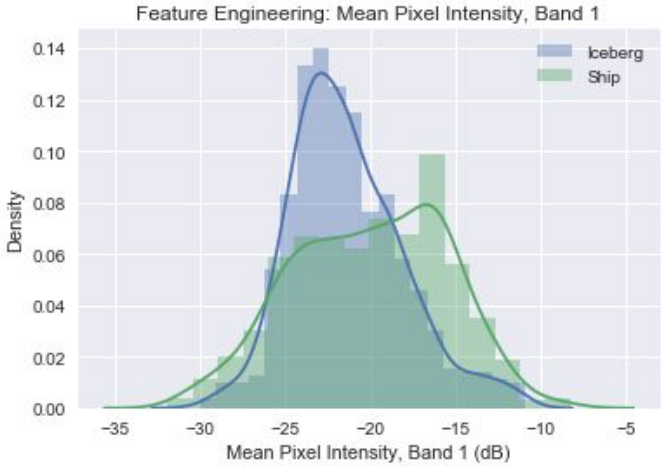
We used band_1 , band_2 and (band_1+ band_2) as three channels of images for all of our models.

We also tried to extract some extra features other than the given images through plotting the distribution of maximum, mean, median and minimum pixels intensity of both the

channels. The results of the plots for median and minimum were of not so much importance w.r.t the features, but the distribution of maximum and mean pixel intensity proved out to be quite insightful.





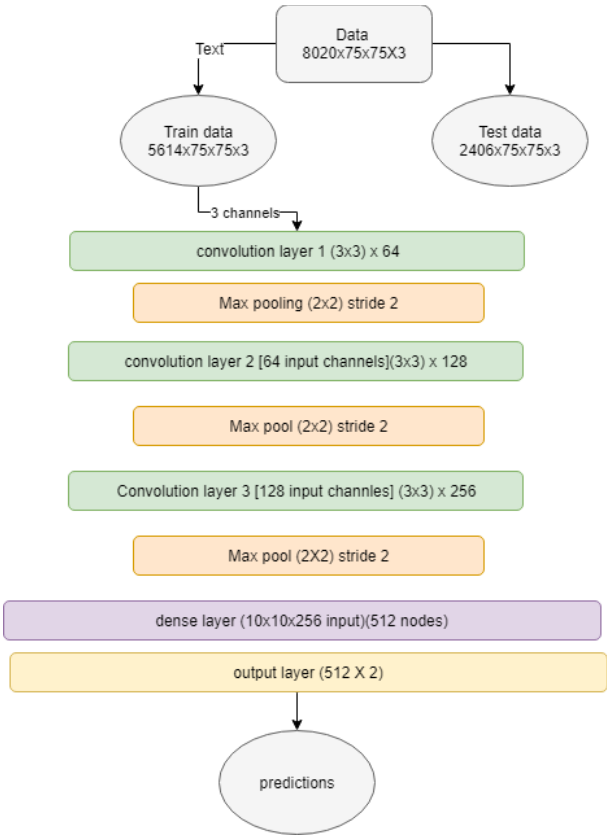


We can see in the above distributions that maximum pixel intensity helps distinguish between the two classes. The by looking at the distributions, we can deduce that images with icebergs have more dark pixels than in pixels of images of ships. Although it cannot be generalized to the whole dataset, but it surely forms a basis for classification of the classes. These features are used in training the MLP classifier that is discussed in the next section.

4. Methods

4.1 Approach 1 : CNN in Tensorflow

Our first approach was to construct a simple convolutional network in tensorflow that takes the input of three channels of images and their respective labels. Since these satellite images have similar kind of structural differences as an RGB image have, we can use convolution on them to find the perfect filters that can classify these images. The implemented architecture is depicted below.

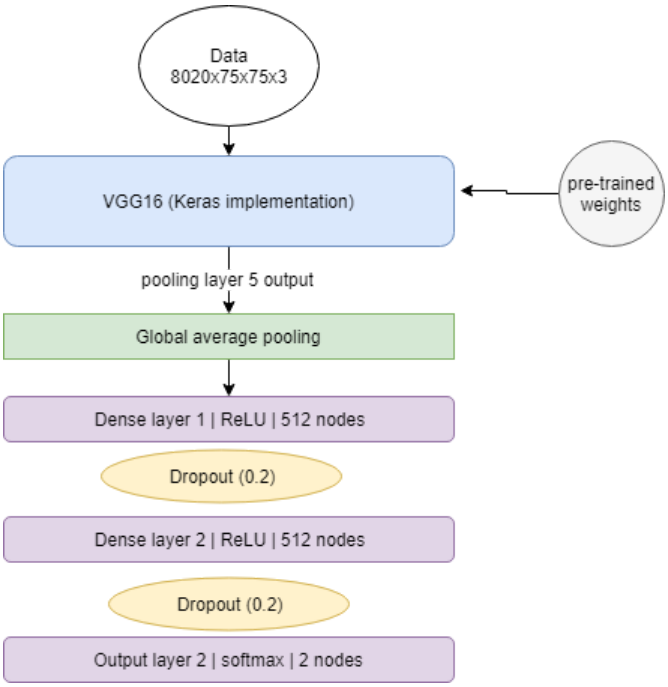


CNN architecture

The results on training this model are discussed in the next section.

4.2 Approach 2 : Transfer learning of VGG16

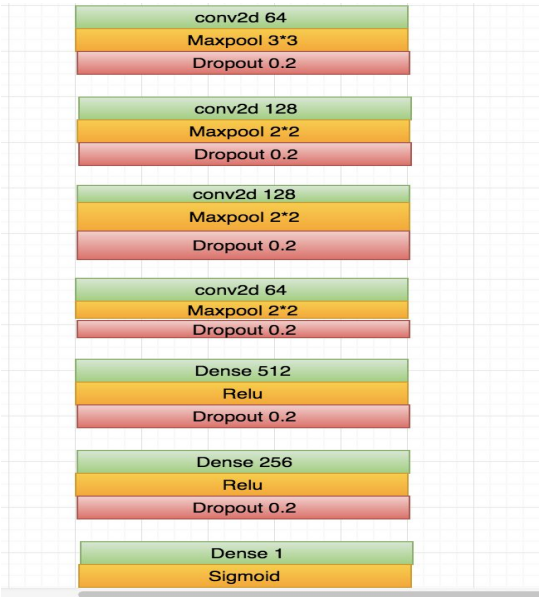
Since we are treating the given images as RGB images, we also tried transfer learning of VGG16 network that was pre-trained on imagenet dataset. We used the pre-trained weights and first five convolution layers of the network. First five layers must have captured the basic structural features like edges and curves in an image. We added two dense layers after these 5 convolution layers.



VGG16 transfer learning architecture

4.3 KAGGLE BASELINE MODEL

We implemented the same CNN architecture in Keras with same hyper parameters from the kernel that got first position in Kaggle public submission board. We only added the augmented data while training the model which is only thing that we did different from the kernel implementation. This model got good results on keras which we will discuss and we consider this model as Ground truth to compare our models results on test data.



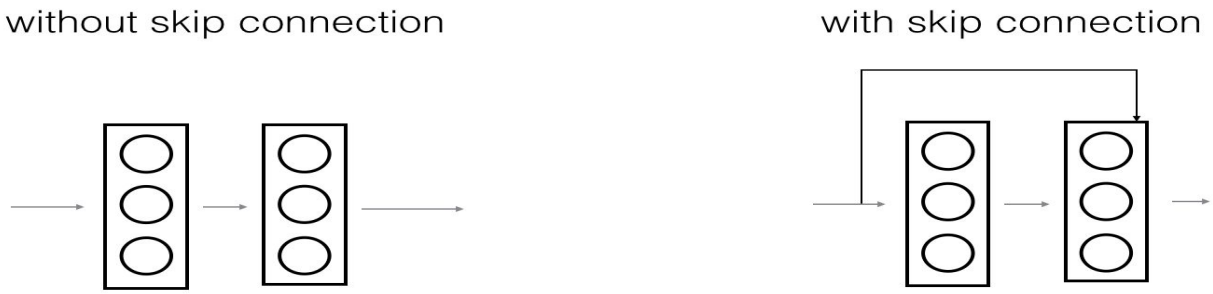
4.4 MLP (Multilayer Perceptron)

During the data exploration phase, as mentioned earlier we found that the maximum pixel intensity for band-2 separated the ship iceberg distribution perfectly. So we implemented MLP network and started our experiments by adding more layers, increasing the neurons in hidden layers, using different activation functions etc. We also added dropout and weight decay regularization to improve our model and make it less prone to overfitting, as we were only having 1604 images i.e. 1604 rows in training set.

Hidden Layers	Activation	No of neurons in layers
4	relu	100,200,50,40
3	relu	100,100,50
5	relu	100,200,100,50,40

4.5 RESNET (Residual Network)

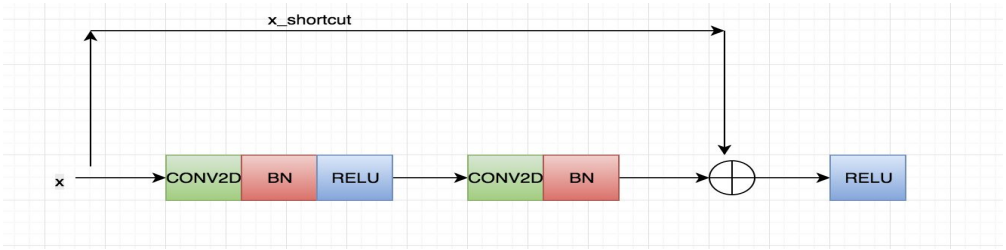
Deep networks are hard to train because of the vanishing gradient problem—as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitely small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly. In ResNets, we make network deeper by using a "shortcut" or a "skip connection" which allows the gradient to be directly back propagated to earlier layers:



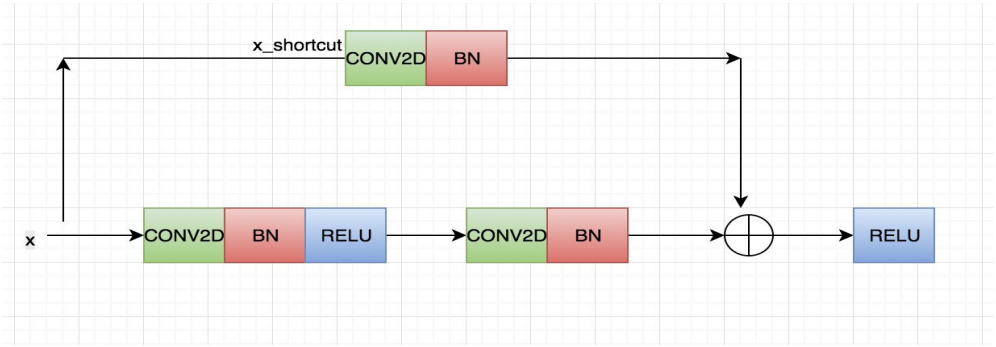
The image on the left shows the "main path" through the network. The image on the right adds a shortcut to the main path. By stacking these ResNet blocks on top of each other, you can form a very deep network.

Network Implementation:-

Identity Block : The identity block is the standard block used in ResNets, and corresponds to the case where the input activation (say a_l) has the same dimension as the output activation (say a_{l+n}). To flesh out the different steps of what happens in a ResNet's identity block, here is an alternative diagram showing the individual steps:

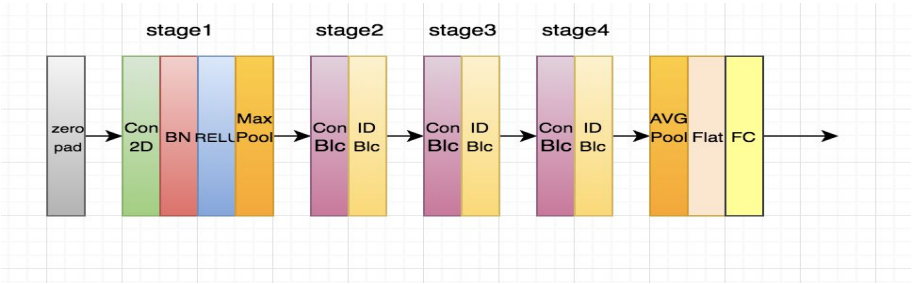


Convolution Block: This type of block when the input and output dimensions don't match up. The difference with the identity block is that there is a CONV2D layer in the shortcut path:



The CONV2D layer in the shortcut path is used to resize the input to a different dimension, so that the dimensions match up in the final addition needed to add the shortcut value back to the main path. For example, to reduce the activation dimensions height and width by a factor of 2, you can use a 1x1 convolution with a stride of 2. The CONV2D layer on the shortcut path does not use any non-linear activation function. Its main role is to just apply a (learned) linear function that reduces the dimension of the input, so that the dimensions matchup for the later addition step.

The following figure describes in detail the architecture of RESNET. “IDBlc” in the diagram stands for Identity block and “ID Blc x 1” means you should stack one identity block, similarly “ConBlc” stand for convolution block.



The details of this ResNet model are:

- Zero-padding pads the input with a pad of (3,3)
- Stage 1:

    - The 2D Convolution has 64 filters of shape (7,7) and uses a stride of (2,2). Its name is "conv1".
    - BatchNorm is applied to the channels axis of the input.
    - Max Pooling uses a (3,3) window and a (2,2) stride.
- Stage 2:
    - The convolutional block uses two set of filters of size [64,128].
    - The identity block use two set of filters of size [64,128].
- Stage 3:
    - The convolutional block uses two set of filters of size [128,256].
    - The identity block use two set of filters of size [128,256].
- Stage 4:
    - The convolutional block uses two set of filters of size [256, 512].
    - The identity block use two set of filters of size [256,512].
- The 2D Average Pooling uses a window of shape (2,2)
- The flatten doesn't have any hyperparameters or name.
- The Fully Connected (Dense) layer reduces its input to the number of classes using a softmax activation.

4.6 Data Augmentation

The data that we are having for this problem only have 1604 labelled images in it. After training the models for a while we realized that this amount of data is not sufficient to successfully learn the underlying pattern for classification. Hence we decided to perform data augmentation on the available data to increase the size of data for training.

We applied the following transformation on every image of the available data:

- Horizontal flip
- Vertical flip
- Rotate CW/ ACW
- Shift along X and Y axis

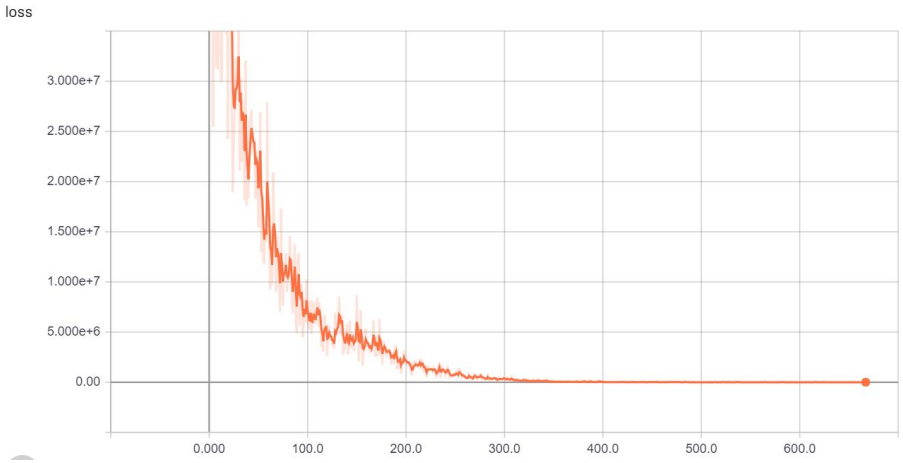
After applying these transformations, we managed to increase the size of dataset to 8020 images. The results of training after data augmentation are reported in the next section.

5. Experiments

Discuss any experiments that you carried out. If your problem allowed for sufficient compute The exact experiments will vary depending on the project, but you might compare with previously published methods, perform an ablation study to determine the impact of various components of your system, experiment with different hyperparameters or architectural choices, use visualization techniques to gain insight into how your model works, discuss common failure modes of your model, etc. You should include graphs, tables, or other figures to illustrate your experimental results. Include the relationship of your results to existing baselines on Kaggle, if you are using a Kaggle set.

CNN model

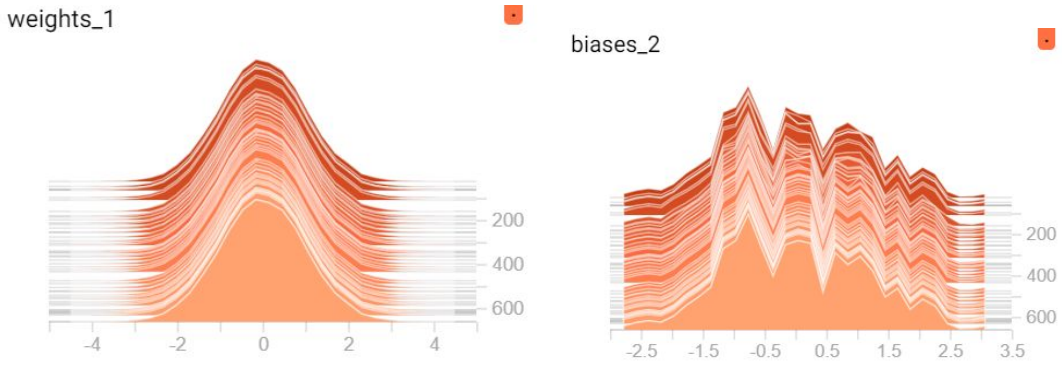
The CNN model on tensorflow did not work so well on the given problem.



The loss initially went down when ran for few epochs (x-axis is the number of steps in the above chart), but became steady after training for a while. The loss is also pretty high (seven digit). Initially it appeared to be a problem of weights initializations, but even after certain random initializations, similar results were observed.

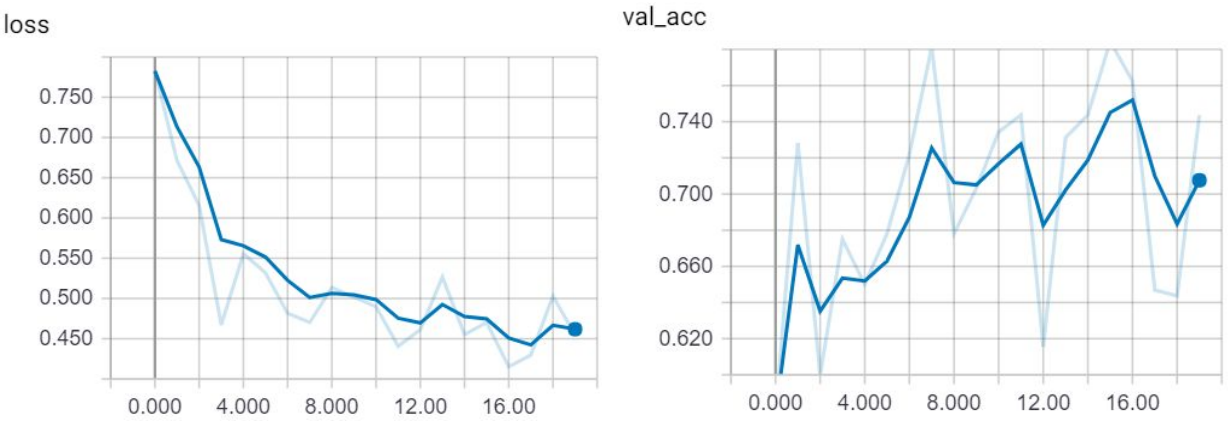
One more interesting fact that we learned about the training of CNN is that the distribution of weights is not changing much, as the training progress. Although, changes

were observed in bias distributions. (z-axis depicts the number of steps of training/batches). Accuracy on this model was ~68%(batch_size=20, n_epochs=20).



VGG transfer learning :

Training results of the VGG network started to show some promising results since the beginning. The loss decreased constantly, increasing the accuracy after each epoch.



The above depicted graphs show the results of training VGG16 (batch_size = 40, epochs=20) on cpu for 2 hours (Learning rate : 0.0001). Learning rate was kept low because in fine tuning, gradients can become large and can overshoot the minima.

MLP

The following are results of MLP on 50 epochs. We did not perform extensive experiments on MLP as we had very small amount of data which had a risk of overfitting.

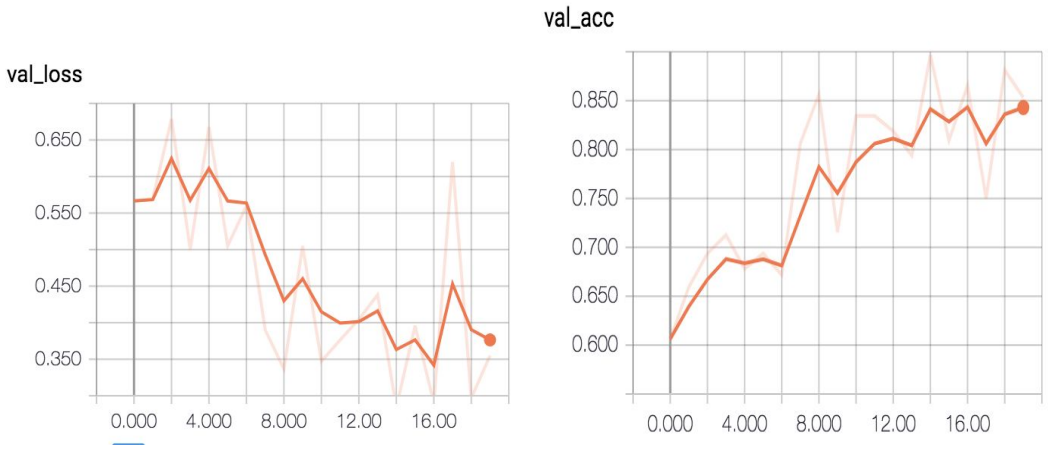
Hidden Layers	Activation	Train Accuracy	Test Accuracy

4	relu	0.671	0.64
3	relu	0.68	0.67
5	relu	0.52	0.49

RESNET:

As shown below are the results of our model on 2o epochs. The validation loss decreases and validation accuracy increases.

```
('Test loss:', 0.28499407
('Test accuracy:', 0.8812
```

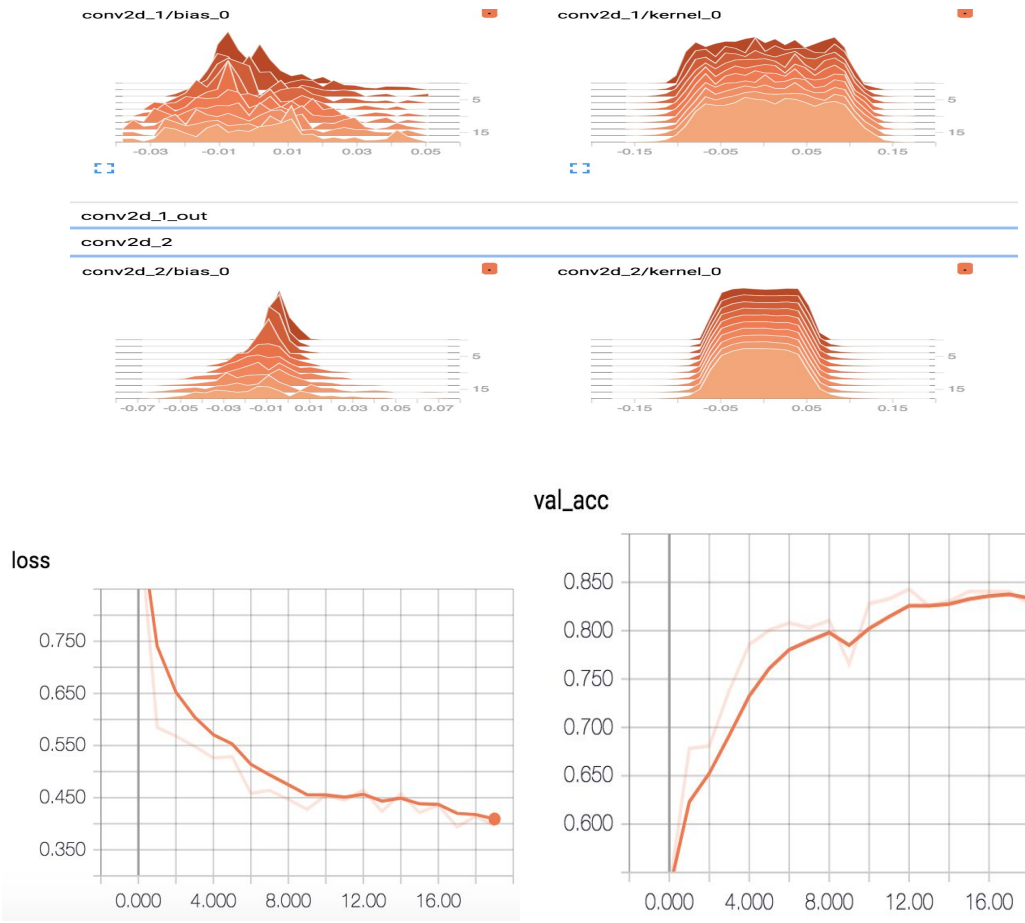


Kaggle Baseline:

As shown below are the results of our model on 2o epochs. The validation loss decreases and validation accuracy increases.

```
('Test loss:', 0.3778225
('Test accuracy:', 0.840
```

One interesting finding on Kaggle baseline model was that weight distribution changes in initial convolutional layers as shown but it changes very less in the end layers which we were not able to understand. Results are shown on Conv_2d1 and Conv_2d2 layers of baseline model.



We compared our results of RESNET and MLP considering the Kaggle model as Ground truth on test set. The similarity scores of our model with baseline model is given as follows: RESNET- 62 %, MLP- 65%

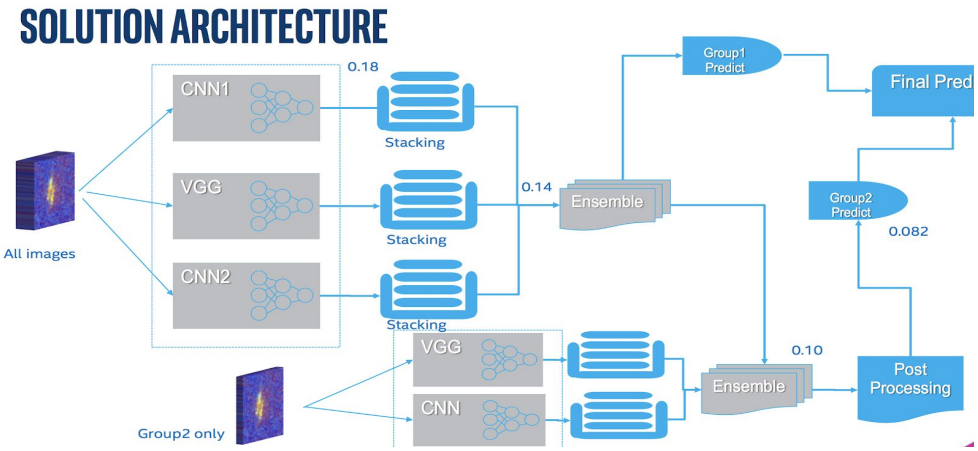
It was shocking when we found that though we achieved higher accuracy then Kaggle model when applying data augmentation on Kaggle model but the similarity of results on test set which has around 8000 images is less of RESNET than MLP.

6. Future Work

- We will use image visualization method Grad-Cam to open up the black box (CNN network) and see what our model is learning. Using this method we can activate each of the filters and check what is causing it to predict a certain class.

  Research paper: https://arxiv.org/pdf/1610.02391.pdf

- Solution architecture of the winning model which stood first in the competition is given below. So here they took all images and ran it through deep ensembles of multiple CNN's. So in each of the blocks CNN1, VGG there are 50-100 models behind them, So in solution, there are somewhere 200-400 total models that are assembled together. The way they are assembled together is they ran each model and did a grid search, So they kept only that model in which cross-validation score was below certain threshold. So they ran an infinite loop until they got their 50 models or whatever they defined as a minimum to be able to include them in the final model. Then they have used stacking as shown below. So we also try to employ stacking in our architecture as it becomes computationally costly but we will try it in our VGG and CNN architectures.



7. Conclusion

After seeing the results, it can be concluded that the underlying problem can be solved with deep convolutional networks. As the depth of the network was increased (from 3 layered CNN in tensorflow to ResNet50 implementation in keras) the performance of the model also got better. More promising results could have been achieved with higher computation power at hand. Though when we compared our RESNET model with baseline model on test set we found the similarity score of 60 % which can lead us to fact that our RESNET model (13 layers) might be overfitting the data.

Visualizing the performance of network through tensorboard provided us a lot of insight. We learned how to monitor the training of the model and to deduce whether a model is training or not by taking a look at the distributions.

We learned how to perform transfer learning of the pretrained models on a custom dataset. This approach helps save training time and encourage reuse of trained weights and filters.

References

1. *[Dataset]. Available: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge/data.*
2. *"Keras Model for Beginners (0.210 on LB)+EDA+R&D | Kaggle", Kaggle.com, 2018. [Online]. Available: https://www.kaggle.com/devm2024/keras-model-for-beginners-0-210-on-lb-eda-r-d.*
3. *"A Comprehensive guide to Fine-tuning Deep Learning Models in Keras (Part I) | Felix Yu", Flyyufelix.github.io, 2018. [Online]. Available: https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html*
4. *[A Comprehensive guide to Fine-tuning Deep Learning Models in Keras (Part II) | Felix Yu Available: https://flyyufelix.github.io/2016/10/08/fine-tuning-in-keras-part2.htm*
5. *Brownlee, "How to Use The Pre-Trained VGG Model to Classify Objects in Photographs", Machine Learning Mastery, 2018. [Online]. Available: https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/.*
6. *"Fine-tuning Convolutional Neural Network on own data using Keras Tensorflow - CV-Tricks.com", CV-Tricks.com, 2018. [Online]. Available: http://cv-tricks.com/keras/fine-tuning-tensorflow/.*
7. *"Keras CNN - StatOil Iceberg LB 0.1995 (now 0.1516) | Kaggle", Kaggle.com, 2018. [Online]. Available: https://www.kaggle.com/cbryant/keras-cnn-statoil-iceberg-lb-0-1995-now-0-1516/.*
8. *"Kulbear/deep-learning-coursera", GitHub, 2018. [Online]. Available: https://github.com/Kulbear/deep-learning-coursera/blob/master/Convolutional%20Neural%20Networks/Residual%20Networks%20-%20v1.ipyn*
9. *"Iceberg Challenge – Deliverable Insights", Deliverableinsights.com, 2018. [Online]. Available: http://deliverableinsights.com/index.php/2018/02/04/iceberg-challenge/*
10. *"Winning Solution to Kaggle's Most Popular Ever Image Classification Contest: Ships vs Icebergs", Aidc.gallery.video, 2018. [Online]. Available: http://aidc.gallery.video/detail/video/5789369111001/winning-solution-to-kaggle%E2%80%99s-most-popular-ever-image-classification-contest:-ships-vs-icebergs.*
11. *"VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION", Arxiv.org, 2018. [Online]. Available: https://arxiv.org/pdf/1409.1556v6.pdf.*