CS - 529 : Introduction to Machine learning
Project 4: Statoil/C-CORE Iceberg Classifier Challenge

Pankaz Das, Muntasir Al Kabir

1 Introduction

This project is primarily aimed to classify a ship or iceberg from the remotely sensed satellite images. It is a Kaggle competition and please refer to [1] for detailed description. In summary, we are challenged to build an algorithm that automatically identifies the ship or iceberg to improve and drive the costs down for maintaining safe working conditions in sea.

2 Implementation Design

We use the Tensorflow library with Keras abstraction for implementing neural network. A step by step description of our whole project is given by the following flowchart (figure 1).

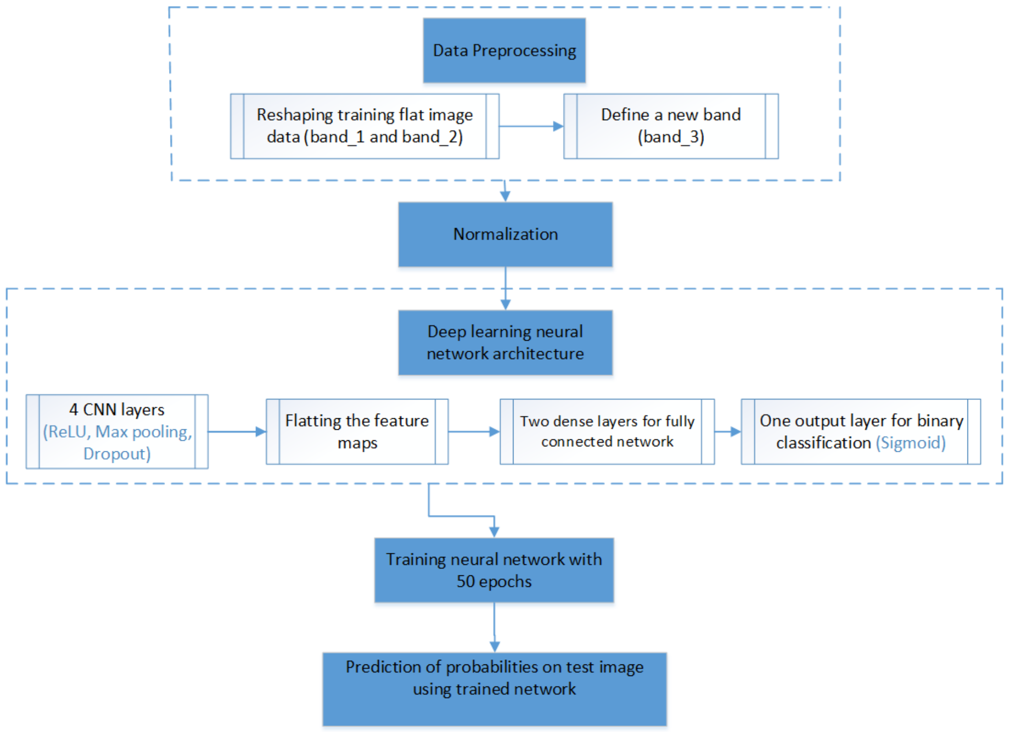

Data Preprocessing
Reshaping training flat image data (band_1 and band_2)
Define a new band (band_3)
Normalization
Deep learning neural network architecture
4 CNN layers (ReLU, Max pooling, Dropout)
Flatting the feature maps
Two dense layers for fully connected network
One output layer for binary classification (Sigmoid)
Training neural network with 50 epochs
Prediction of probabilities on test image using trained network

Figure 1: The step by step flow chart of the whole project implementation.

3 Data Preprocessing

The preprocessing part involved of reshaping the flat image data ($band_1$ and $band_2$) into 75 $\times$ 75 images for both training and test data.

We define a new feature ($band_3$) by averaging $band_1$ and $band_2$, that results in improved classification accuracy. Then we normalize of all preprocessed images data to make their range [0,1].

4 Deep Learning Neural Network Architecture

In this section, we provide a brief demonstration of our deep learning architecture as below [2,3]. We used CNN deep learning framework as it is vastly used for image classification.

a. Four Convolution Neural Network (CNN) layers are used for extracting the Feature Map from the images. The activation function we used is Rectified Linear Unit (ReLU). We also used max pooling to reduce the dimensionality of each feature map and random Dropout (0.2) regularization for minimizing the over-fitting. The depth of the four Kernals/filters are 64, 128, 128, 64, respectively and size is (3$\times$3).

b. Flatting the feature maps for using those data as the input for the dense layers.

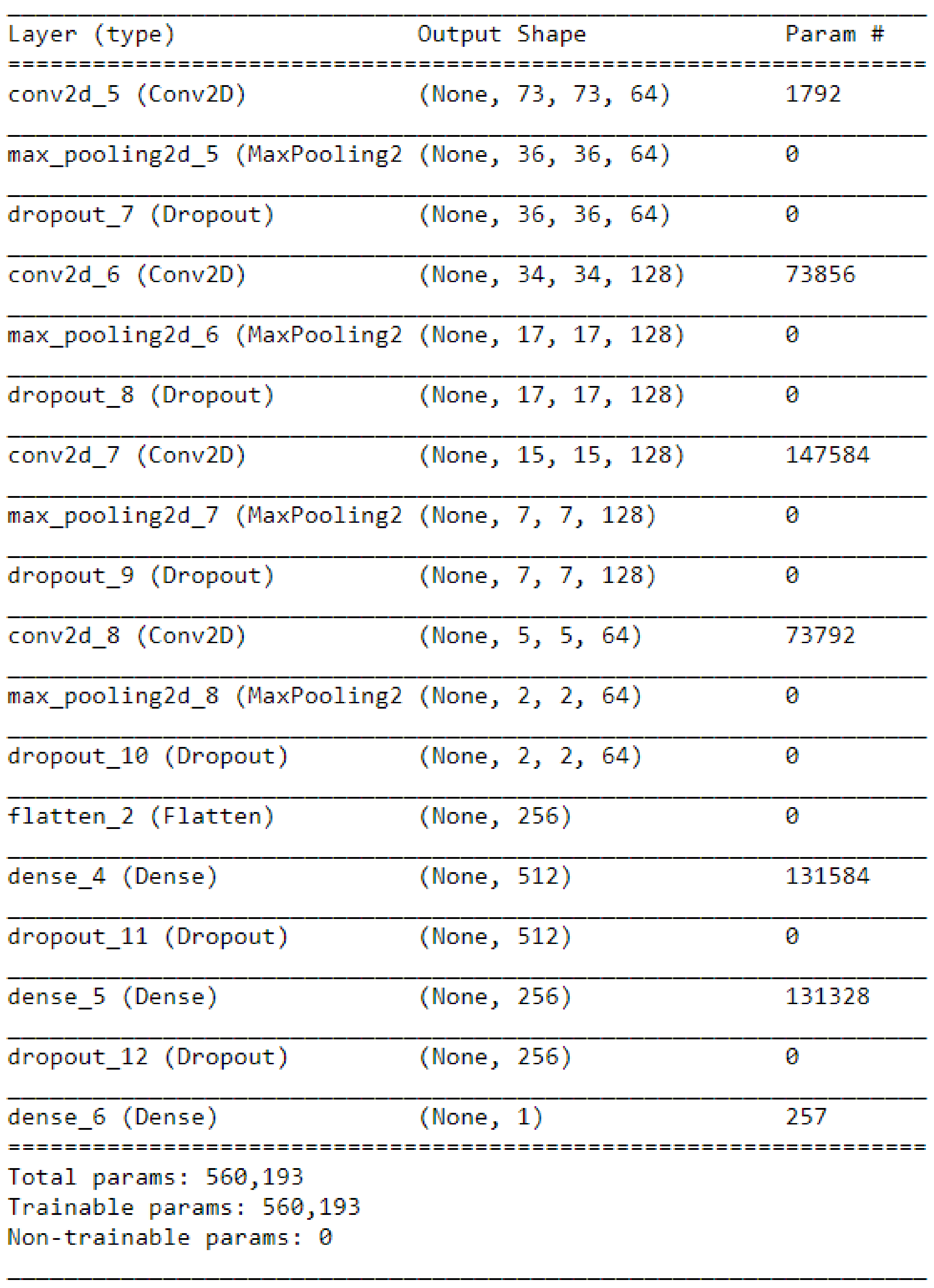
c. Two dense layers for the fully connected layer where activation function is ReLU.

d. One output layer for binary classification where we used stochastic Adam optimization (with learning rate = 0.001) and Sigmoid activation function.

Finally we compile the model with loss function was the binary cross-entropy. Note that this is sequential model where layers come sequentially one after another. A summary of the model is given in figure 2 and the model has 560,193 parameters to train and optimize.

5 Training the Model

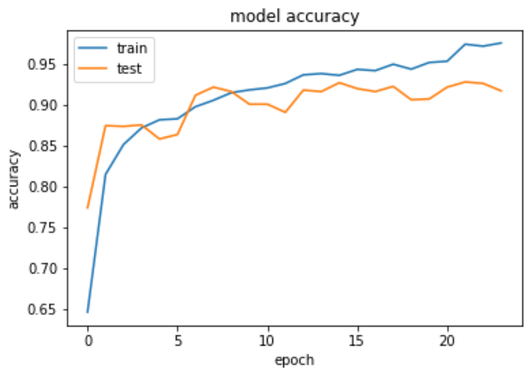
We trained the model with 50 epochs considering the computation speed as we were only using CPU. With random initialization of all model parameters, some other following parameters are also used for training purpose:
a. **Batch size** = 32. We have tried with several other batch sizes, however we found 32 is giving good accuracy and low value loss as well as reasonable computational speed.
b. **Learning rate reduction:** we reduce the learning rate if no reduction of value loss metric for consecutive 7 epochs. Number 7 epoch is chosen randomly for computation speed.
c. **Early stopping:** we also used early stopping criteria if no improvements of value loss metric for consecutive 10 epochs. Number 10 epoch is chosen randomly.



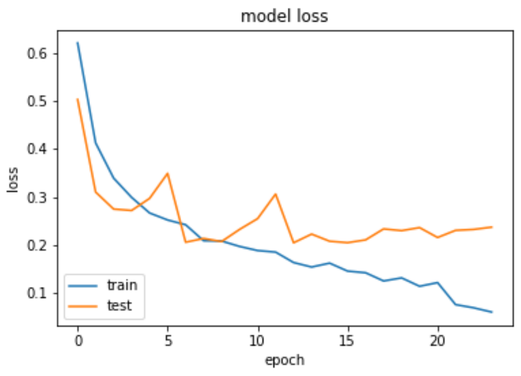
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 73, 73, 64)        1792
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
dropout_8 (Dropout)          (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
dropout_9 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 5, 5, 64)          73792
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
dropout_10 (Dropout)         (None, 2, 2, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_11 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_12 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 257
=================================================================
Total params: 560,193
Trainable params: 560,193
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: The model summary.

Figure 3 shows the accuracy and loss of our model with training epochs. As we can see the accuracy is increasing and loss is decreasing with training epochs for both training and validating sets of data.



(a) Model accuracy with training epochs



(b) Model value loss with training epochs

Figure 3: Accuracy and value loss of the model with training epochs

6 Results

We find the accuracy using after splitting the training data into 0.75-0.25. The Accuracy and score on training data are given below in Table 1. The scores from Kaggle submission with three preprocessed data are shown in Table 2.

Table 1: Accuracy and score on the training data

Train accuracy	Score
95%	0.11

Table 2: Accuracy and score on test data from Kaggle submission

Techniques	Score
Without Normalization	0.2163
With Normalization	0.1911
With Normalization and third feature	0.1694

7 Future Work

We are still working on this project to improve the accuracy and to win Kaggle competition!

References

[1] https://www.kaggle.com/c/statoil-iceberg-classifier-challenge.

[2] https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[3] https://deeplearning4j.org/convolutionalnetwork