Capstone proposal

Domain Background

Icebergs presents threats to the ships navigation and various offshore activities. Especially, it as actual problem for the area offshore to Newfoundland and Labrador known as Iceberg Alley. The primary iceberg detection method for now is aerial reconnaissance using vessel-based monitoring data. Also, data received though satellites are widely being integrated now onto the monitoring systems greatly reduce monitoring cost. Additionally, Synthetic Aperture Radar (SAR) satellites can still monitor in various weather conditions such as clouds and fog.

However, manual visual classification of SAR images to identify iceberg is very time-consuming process. So, C-CORE company (https://www.c-core.ca/) has developed a computer vision system that analyzes SAR data to automatically detect and classify icebergs and vessels. Now it challenges ML community to build effective classification algorithm for their detection system [1]:

Problem Statement

The goal of the project is to build an algorithm which can reliably classify data to identify either it is iceberg or ship, based on given Synthetic Aperture Radar data. Also, the results are clearly measurable using prediction accuracy and it is important to have classifier with higher accuracy (ideally 100%).

Additionally, analysis and classification SAR data is interesting problem. Even if it seems like standard image classification task it has some important differences which makes it challengeable to use pre-trained neural networks with transfer learning for the image classification such as VGG [2] or Inception [3]:

- SAR data is not a three-channels regular image
- Radar detected shapes are different than visually detected shapes.
- Data set has additional incidence angle parameter of which the image was taken. So, it is additional interesting task to figure out how to include this into the model and how beneficial this parameter is.

However, as SAR data is similar to image data so the most suitable potential solution is to use custom CNN architecture which fits SAR data best.

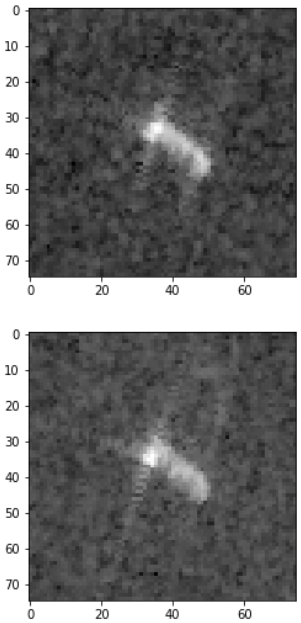
Datasets and Inputs

CORE provided dataset of satellite SAR images containing either a ship or an iceberg including 1604 training samples and 8424 test samples (5000 from them are autogenerated)

Data was collected from SAR which bounces a signal off an object and records the echo, then that data is translated into an image. Two channels of image are provided: HH (transmit/receive horizontally) and HV (transmit horizontally and receive vertically). See [4] for more details.

Data was packed to the two JSON files (train.json, test.json). Each file consists of list of satellite images in following format:

- **Id** of the image
- **band1** – flatten image data of the HH channel (5625 elements, 75x75 image), each element is float value measured in dB.
- **band2** – flatten image data of the HV channel (5625 elements, 75x75 image) , each element is float value measured in dB.
- **inc_angle** - the incidence angle of which the image was taken
- **is_iceberg** – classification label of the image. 1 is for iceberg, 0 for ship.



HH and HV channels data of the ship.

Training dataset the only dataset which has labels assigned – so it will be used for training and validation. Test dataset does not have labels and will be used for the model evaluation.

For the model features we are going to use band1, band2 and inc_angle data as features and is_iceberg field as labels.

As we can see 1604 images for both training and validation is small number – so it requires to apply data augmentation techniques to expand training dataset size to be able to train solid model – otherwise there is a big risk of overfitting. Additionally, semi-supervised learning might be considered as additional extension of algorithm to improve prediction accuracy.

Solution Statement

An algorithm should be designed to get input data provided, extract necessary features and answer the question if this object is an iceberg or ship.

The Deep convolution neural network is planned to be built as a main solution using Keras or/and TensorFlow to. Log loss will be used as primary metric to evaluate model quality.

Both model architecture and best pre-trained model will be saved – so it will be easy to either re-train model or load already trained model and use it for the iceberg predictions in the real-time.

Benchmark Model

We will be using two benchmark models. As it is Kaggle competition the score of the best models is publicly available – so we can compare our model performance with the Kaggle leaderboard [5] to check how well our model performs.

As no model architecture is publicly available for the top Kaggle leaderboard models it would be great to have some real model to compare with which has both model architecture defined and log loss score. The good candidate is simple CNN implemented with Tensorflow with the log loss = 0.27 [6]. So, we can see if we can achieve better results with the proposed algorithm.

Evaluation Metrics

As this problem is a standard binary classification problem and while considering problem domain it is obvious that accuracy of the prediction is an essential metric. It is just calculated as a fraction of correctly classified images over database size.

Additionally, we are going to use Log loss [7] as additional evaluation metric. Firstly, because it is official evaluation metric for the Kaggle completion and its value available for the benchmark model and can be compared. Secondly, it provides additional information compared to accuracy: how “sure” model is about its prediction. Lower log loss corresponds to models which more solid prediction boundary even if accuracies of the models are close.

Project Design

Project will be implemented in the Python 3.5 using Machine learning libraries such as: sklearn for simple classifiers as a starting point (for example Logistic Regression or SVC), TensorFlow, Keras for CNN approach. Additionally, OpenCV might be required for some experiments with image pre-processing techniques like noise reduction or gradients.

Following workflow is designed to complete the project:

- Load data from the input files
- Visualize and analyze dataset to identify suitable data normalization and augmentation techniques.
- Perform necessary data transformation to fit it to the selected model including:
    - Reshaping data to get square images if we use CNN approach.
    - Consider a way to use **inc_angle** as additional feature. For example, we can use its value for whole additional channel for the CNN models.
    - Perform data normalization using either Min-Max scaling or Z-score normalization [8].

- Split training data into training and validation set.
- Use data augmentation to extend dataset including rotation, scaling, flipping. (only for training data)

- Build simple classification model, I would start from sklearn logistic regression, SVC or Decision tree model as initial point.

- The next step to build simple CNN architecture and compare it with the simple classifier.

- Play with two options: 2- channel image (HH, HV) and 3 -channel image (HH, HV, incidence angle value) to see if there is any improvement.

- Try to build more complex models increasing and varying number of CNN layers.

- Try various regularization techniques as batch Normalization, L1 and L2 regularization, different activation functions and different optimizers to find best parameters.

- As an advanced optional step try semi-supervised learning with GAN to see if it provides additional improvement.

Best model will be submitted to the and compared to the leaderboard models and previously selected benchmark.

References

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

[2] VGG: http://www.robots.ox.ac.uk/~vgg/research/very_deep/

[3] Inception: https://arxiv.org/pdf/1409.4842.pdf

[4] Polarization: http://learningzone.rspsoc.org.uk/index.php/Learning-Materials/Radar-Imaging/Image-Interpretation-Polarisation

[5] Kaggle leaderboard: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge/leaderboard

[6] Benchmark model: https://www.kaggle.com/camnugent/convolutional-neural-net-tensorflow-lb-0-27

[7] Log loss: http://www.exegetic.biz/blog/2015/12/making-sense-logarithmic-loss/

[8] Data Normalization: http://sebastianraschka.com/Articles/2014_about_feature_scaling.html