CS230

Project CS230 - Spring 2018

Justin Donato
CS230 - Deeplearning
Stanford University
jdonato@stanford.edu

Abstract

Image recognition, powered by deeplearning has the potential to automate and streamline many of the manual task that are performed by today's workforce. Deeplearning algorithms trained to recognize images and initiate advanced decision making processes are driving a paradigm shift that will not only change the way we work, it will improve worker safety and ultimately employee satisfaction. This project uses a convolutional neural network to interpret satellite imagery and determine if a remotely sensed object is an iceberg or a ship. The goal is to automate a manual process and assist companies operating in remote and arctic regions.

1 Introduction

As the accuracy of image recognition continues to improve with advances in deeplearning, we have an opportunity to use this technology to achieve higher levels of worker safety in dangerous and remote locations. This is important as many of the process driven tasks that are focused on compliance and data validation can be overlooked or rushed when left to a human, especially in highly repetitive or stressful environments.

The problem presented in the kaggle.com 'iceberg classifier challenge' is to develop an algorithm that can identify a remotely sensed target as a ship or iceberg using satellite imagery. The goal is to improve safety conditions by replacing the current process of dangerous aerial reconnaissance and lagging shore-based validation, with an automated solution that monitors environmental conditions, and assess risk from icebergs.

The motivation for embarking on this challenge was to better understand the structure and value of satellite imagery, specifically related to its use in deeplearning solutions.

The input to this algorithm is presented in 2 bands of flattened image data with 75x75 pixel values. Each band is characterized by radar backscatter produced from different polarizations at a particular incidence angle, or the angle at which the image was taken. The labels used in training and validation are provided by human experts with geographic knowledge of the target. The output is a classification, i.e. is the target an iceberg or a ship.

CS230: Deep Learning, Spring 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

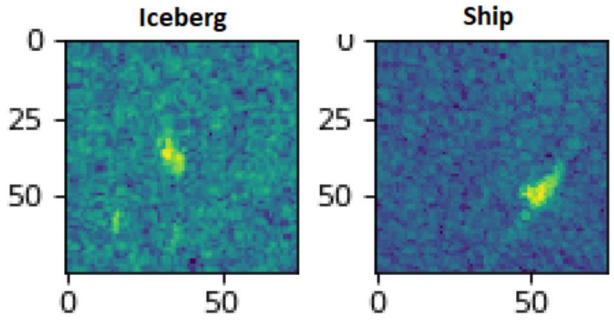
Title	Categorization	Comment
Practical Recommendations for Gradient-Based Training of Deep Architectures	Hyperparameter - selection and tuning	This was a very insightful paper that captured the challenges of tuning hyperparamters. There were now obvious weaknesses and I found this to be a solid paper on the subject matter.
Visualizing and Understanding Convolutional Networks	Understanding how CNN's achieve a prediction	When attempting to use a Saliency Map for the first time, this paper gave some insight into how to interpret the visualizations of a CNN. While it was a good paper I was hoping for a better introduction to the content, and how to apply it in a practical manner.
Food Image Recognition by Using Convolutional Neural Networks (CNNs)	Image Recognition	This paper provided great support in developing my project and provided a framework for how to execute and document this type of solution. It had numerous similarities to my work and was very helpful.
Flexible, High Performance Convolutional Neural Networks for Image Classification	Image Recognition	This work provided good insight into the academic approach to image recognition, and while not immediately relevant to my project, I expect to return to this paper in the near future.
ImageNet Classification with Deep Convolutional Neural Networks	Image Recognition	Excellent paper. While not immediately relevant to my project, it provided a deep insight into how to construct a final report on this topic and how to present my work in a professional and descriptive framework.

3 Dataset and Features

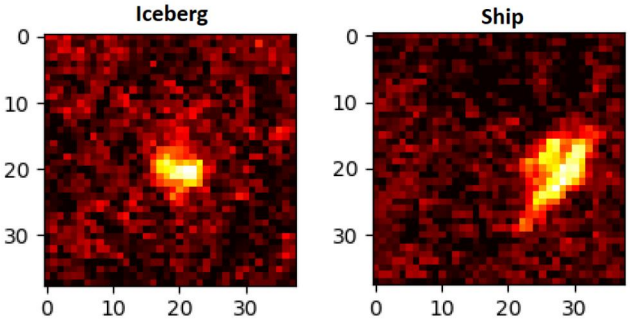
The images used in this solution leverage radar backscatter produced from different polarizations at a particular incidence angle, or the angle at which the image was taken. The labels used in training and validation are provided by human experts with geographic knowledge of the target. All the images are presented in 2 bands of flattened image data with 75x75 pixel values, at 5625 element in total, together with an single angle of incidence value. The original data set was presented in 2 files, across training and test with the following number of records:

- Training: 1103 images
- Test: 8424 images

Given the training set was relatively small, I have used a number of augmentation techniques to expand that data set. The methods used include rotation, flipping, horizontal and vertical shift, zooming and translation along a positive and negative diagonal. As a result I was able to expand the data available for training and validation to 16,545 records. I then split these data at a 75 to 25 percent ratio across training and validation sets. As part of the data preparation process I have normalized all training data using a typical standardization method, i.e. subtracting the mean and diving by standard deviation for the training set, and applying to both training and validation. All data was provided as part of the kaggle.com ‘iceberg classification challenge’. An illustration of an iceberg can be seen below.



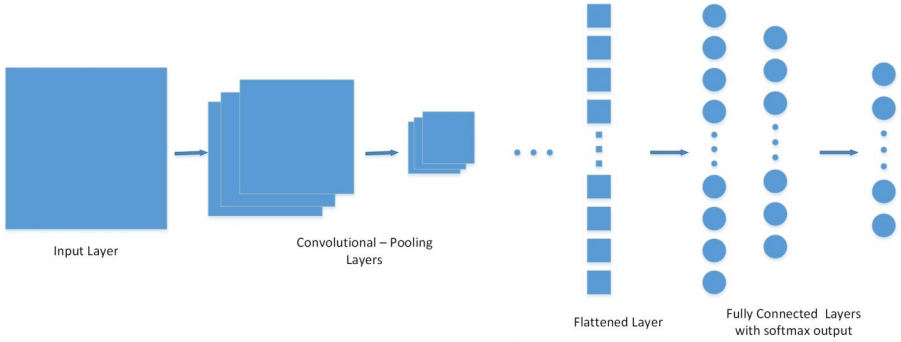
While analyzing how the solution was interpreting these images, I produced various saliency maps throughout the training process. The images below are a sample of that work, and provide some insight into how a ship and iceberg are interpreted by the first layer of the network. The structure of the saliency map quickly became less intelligible in the remaining layers.



4 Methods

The learning algorithm used in this solution was designed take advantage of the proven design patterns established when using convolutional neural networks (CNN's) for image classification. As a result I have used a layered design across convolutional, flattened and two fully connected layers, cross entropy as the loss function and a softmax output layer to drive the final classification. A representation of this framework, together with the loss function is illustrated below.

$$\text{Cross Entropy Loss: } -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\ log(a^{[L](i)}) + (1-y^{(i)})\ log(1-a^{[L](i)})$$



In an attempt to achieve the best prediction possible I have configured a total of four models, and the configurations that were consistent across each model inlcude the following: Convolutional filter size of 3x3 and a max pool filter size of 2x2, with a Stride of 1, and padding of 'SAME'. Unique model configurations were limited to the number of hidden layers and neurons, and the value for dropout regularization. The principal being, as more neurons were added, dropout regularization was increased. The four models are summarized below.

	Dropout	Convolutional Layers						FC Layers	
Model 1	0.2	64	128	128	256	128	64	512	256
Model 2	0.7	64	128	256	512	128	64	512	256
Model 3	0.2	64	128	128	256	0	0	256	128
Model 4	0.7	64	128	128	256	0	0	256	128

The design methodology in developing and configuring these models has been to implement as many layers as possible – given available computational power, to achieve a prediction that is as detailed as possible. For example, as the network trains and traverses the various layers, there is the expectation it will move from a more generalized interpretation of the image to a 'deeper' understanding of the nuanced characteristics that differentiate an iceberg from a ship. This is evident is the saliency map above, taken from the first later of the network, and showing a graphic that is very close to the initial image. The hope is that later layers focus on the specific and detailed characteristics of differentiation.

5 Experiments/Results/Discussion

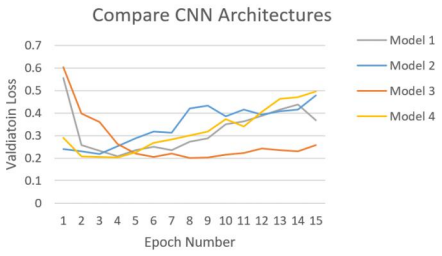
I used the test error as the key metric, and validated each of the models using 15 epochs of the training and validation data to produce the results below. Model 3 had the best overall score and was used for the final submission to the kaggle.com competition. The final prediction was produced using 50 epochs and generated a training error of 0.1860 and test error of 0.2005. My final place on the leader board progressed from a starting position of 1519 to 1145, with a public score of 0.1657. The competition had 3343 entrants.
Hyperparameter tuning focused on the three elements below. Tuning the learning rate was a quantitative, data driven exercise whereas tuning the batch size and dropout metrics were notably more qualitative. For example, a combination of monitoring CPU utilization, execution times and ultimately results led to the final configurations for batch size and dropout regularization. In an effort to make the most of available resources, Adam optimization was applied consistently across all models.

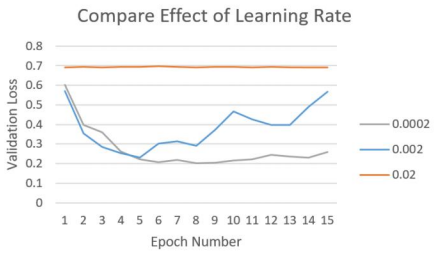
- Learning Rate: Through a process of extensive trial and error I determined that 0.0002 was the most effective rate. To formalize my finding I carried out a standard learning rate analysis from 0.02 to 0.0002 as documented in the chart below.
- Batch Size: While I attempted a variety of batch sizes the smallest value I could implement, and not over utilize my computational resources, was 256 records.
- Dropout Regularization: Through trial and error I arrived at a keep probs value of 0.2. The impact of an increasing number of hidden layers while applying dropout regularization made this selection more of a judgement call, balancing configurations that would execute to completion, against an attempt to achieve the best results.

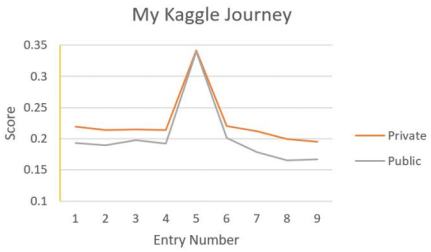
Test error was used as the success metric and the values below record the most successful iteration of each model during the model validation phase of development.

Training Samples: 16,545
Test Samples: 368

	Training Error	Test Error
Model 1	0.2828	0.2083
Model 2	0.1088	0.2183
Model 3	0.2001	0.2013
Model 4	0.2184	0.2031

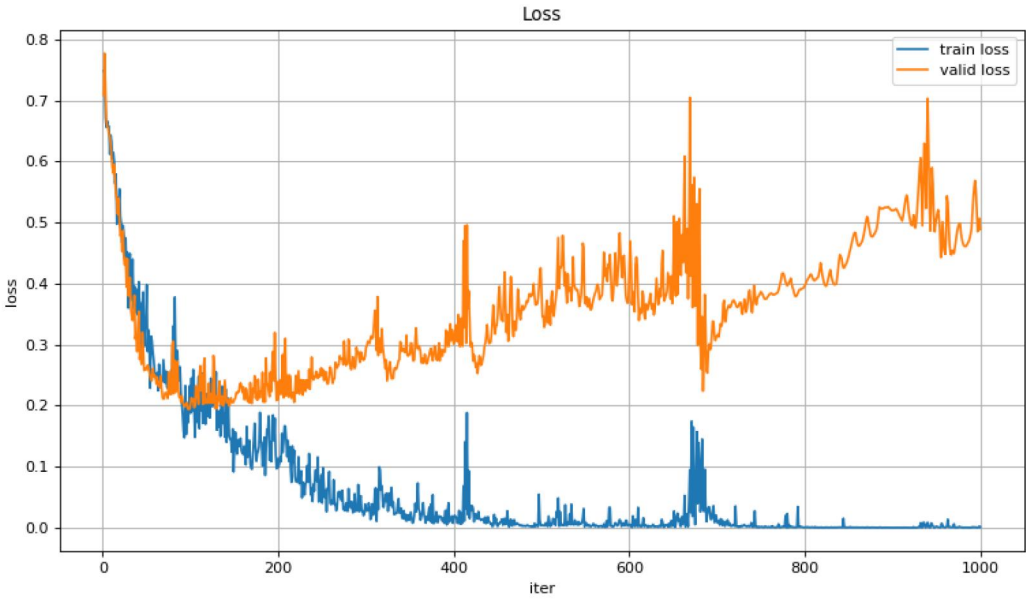






Other configurable hyperparamters, such as number of layers and associated neurons were selected based on trial and error, and as a result I implemented multiple models to ensure a reasonable coverage of options. Accuracy provided some qualitative insight into which configurations were likely to provide the best results.

The final result produced using model 3 and 50 epochs did generate the best result, but still suffered considerably from overfitting. The chart below clearly highlights that relatively early in the training process the model had achieved its lowest validation error and there after began to overfit the training set. In an attempt to address this overfitting I tried various configurations of the dropout metric and swapping out different elements of the augmented data.



6 Conclusion/Future Work

The convolutional neural network delivers a powerful platform to analyze and interpret images, and with a reasonable effort I have been able to put together a solution that has had some success in classifying an iceberg when compared to a ship. However, to develop a model that is capable of winning a kaggle.com challenge like this, does require an in-depth knowledge of how a CNN works and how to combine it with other complimentary algorithms, such as clustering. Given another 6 months I would attempt the following:

- Apply different regularization techniques to address the overfitting evident in the error analysis. Specifically an exploration of L2 regularization.
- Develop a better understanding of the data, and explore the implementation of clustering algorithms as suggested by the competition winner.
- Leverage a more powerful platform for model training, such as AWS or google cloud.

7 Contributions

This project was an individual effort, undertaken by the author of this paper.

References

[1] Iceberg Classifier Challenge [Online]. Available: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge

[2] Yoshua Bengio, Y. (2012) Practical Recommendations for Gradient-Based Training of Deep Architectures, Version 2, Sept. 16th, 2012
[3] Zeiler, M.D., & Fergus, R. (2013) Visualizing and Understanding Convolutional Networks, Dept. of Computer Science, Courant Institute, New York University.
[4] Yuzhen Lu, (2004) "Food Image Recognition by Using Convolutional Neural Networks (CNNs)" Department of Biosystems and Agricultural Engineering, Michigan State University, East Lansing, MI 48824, USA Feb.
[5] Cires, D.C., Meier, U., Masci, J., Gambardella, L.M., & Schmidhuber, J., Flexible, High Performance Convolutional Neural Networks for Image Classification. IDSIA, USI and SUPSI Galleria 2, 6928 Manno-Lugano, Switzerland
[6] Krizhevsky, A., Sutskever, I., & Hinton, G., ImageNet Classification with Deep Convolutional Neural Networks