Computer Science
School of Informatics, Computing, and Engineering
Indiana University, Bloomington, IN, USA
Datamining B565 Fall 2017

Kaggle Competitions: Author Identification Statoil/C-CORE Iceberg Classifier Challenge

Joel Park[1]*, Kayl Walton[2] Qingyun Lin[3]

Executive Summary This paper represents entries into two Kaggle competitions. Kaggle is a website that hosts competitions that are intended to be solved via data-mining approaches and rewards monetary prizes to winners. In these two problems, data mining is used to learn a model and then classify new data into given categories. The first of the two problems involves classifying text by its author when given a large quantity of labeled text. The three authors are Edgar Allan Poe, Mary Shelley, and H.P. Lovecraft. The second problem focuses on the classification of satellite images as icebergs or ships from images labeled with iceberg or ship. We approached the first problem with a fully connected neural network and successfully classified 80% of the new text correctly with this method. This algorithm was trained on roughly 20,000 labeled text samples. For the second problem, we used a convolutional neural network. With this method, the algorithm successfully classified 85% of pictures in the test set. This algorithm was trained on over 1600 images.

Keywords
Data mining — Classification — Neural networks

[1] *Computer Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[2] *Computer Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[3] *Computer Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
***Corresponding author**: jgpark@indiana.edu

Contents

1. Introduction

The two datasets we use in this paper come from an online data-mining competition website called Kaggle. Kaggle has hosted nearly 300 competitions since its inception, two of which will be discussed in this paper, and has given away millions of dollars to winners. The Author Identification problem has 1059 teams working on potential solutions, while the Iceberg Classifier competition has 2045 teams. They offer prizes of $25,000 and $50,000 respectively[1]. These two analysis problems are best solved through data mining. At a high level, data mining is the process of examining large quantities of data to generate information that was not originally present. Both competitions have large amounts of data, and the classification of the data from other features is clearly generating new information.

[1] The Iceberg Classifier competition can be found at: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge. The Author Identification competition can be found at: https://www.kaggle.com/c/spooky-author-identification

1.1 Author Identification

Mary Shelley, Edgar Allan Poe, and H.P. Lovecraft are three of the most well-known authors who dealt with the mysterious and macabre. They had distinct styles, but dealt in similar subject-matter. This makes these three authors ideal for classification. The goal of this problem is to use some labeled training text to determine which of these three authors wrote a given piece of (unlabeled) text. Of course, it is important to define how to evaluate the success of the algorithm. The simplest solution to this is to just give the percentage of correct predictions. However, Kaggle uses a different measure for success, the logloss:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij}) \quad (1)$$

Where N is the number of observations in the test set, M is the number of class labels (3 in this case) y_{ij} is 1 if observation i belongs to class j and 0 otherwise. p_{ij} is the predicted probability (from the training set) that observation i belongs to class j. The probabilities are then rescaled by the sum of the row before submission.

The data comes in a simple CSV form with three fields: the ID, the text given as a string, and a three letter code representing the author. EAP is for Edgar Allan Poe, HPL for H.P. Lovecraft, and MWS for Mary Shelley. The given text is simply a selection from one of the author's books, poems, or short stories. For example:

> He had a weak point –this Fortunato –although in other regards he was a man to be respected and even feared. He prided himself on his connoisseurship in wine. Few Italians have the true virtuoso spirit. For the most part their enthusiasm is adopted to suit the time and opportunity, to practise imposture upon the British and Austrian millionaires.

is from Edgar Allan Poe's *The Cask of Amontillado* [2]. This means it would have the label EAP in the dataset. In addition, there is clearly some non-word information in this excerpt, such as the dashes around "this Fortunato." Punctuation like this will need to be removed. Besides this, one may note that "practise" is spelled in the old english manner, and it is important to decide if this word - and other words with newer spellings - should be treated as the same word as its modern counterpart, "practice." Obviously, there will be some preprocessing steps necessary; instead of taking in the text directly, our algorithm will take in a bag of words, the motivations for which will be discussed more in the Data Analysis section.

1.2 Statoi/C-CORE Iceberg Classifier Challenge

Image classification and computer vision is a field with many important uses. A computer can scan many more images in a short amount of time than a human can. In this case, Kaggle is hosting a competition with the purpose of classifying satellite images as icebergs or ships. Similar to the Author Identification problem, it is important to have a method for the evaluation of success. The most obvious way is - once again - to simply examine the percentage of images that are identified correctly. However, Kaggle uses the log loss for evaluation of this problem as well.

The dataset is given as a JSON file in which each image is defined by two bands containing an array of 75x75 pixels encoded as float numbers with decibels as the unit. The two bands are obtained by radar backscatter for two different polarizations at a given incidence angle. The two polarizations are HH, which means that data was transmitted and received horizontally, and HV, which means that data was transmitted and received vertically. When the two bands are used to produce an image, the results look like these:

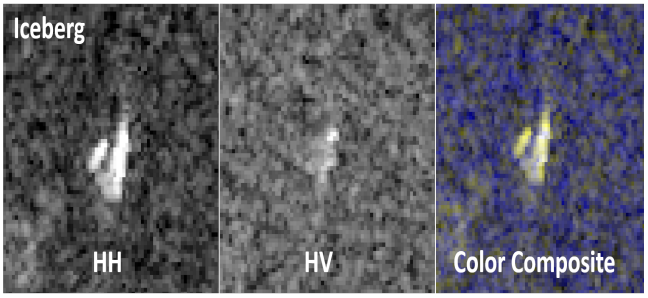

Figure 1. An iceberg.

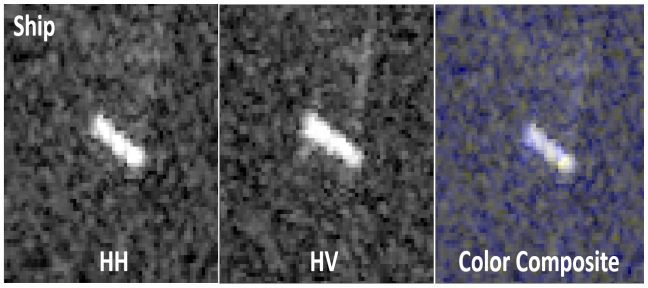

Figure 2. A ship.

The dataset also includes the aforementioned incidence angle. The target variable is a binary variable called "is_iceberg," which is a 1 if the picture is is an iceberg and 0 if it is a ship. From the images, it is clear that some decisions will need to be made about the format of the data our algorithm will take in. Using each of the 5625 float numbers for each band for each data point is most likely unnecessary and computationally overkill. Therefore, the data must be preprocessed, and the details of this will be discussed in the Data Analysis section.

2. Data Mining

Data mining is an iterative process intended to discover patterns in large quantities of data. These patterns are then used

to aid in the answering of questions related to this data. Additionally, data mining is capable of assigning new meaning to data, thus creating information that was not originally present or obvious. While data mining does not solve problems in and of itself, it does answer questions about the data. These answers can lead to more data mining problems, and occasionally to a solution to the original problem. Data mining is done in nine general steps. The steps are: problem statement, get data, enrich data, clean data, integrate data, transform data, mine, validate or interpret results, and make the results actionable. These will be discussed in greater depth below.

Two kinds of problems that often arise in data mining are clustering and classification. Clustering involves grouping objects based on the similarities of their attributes. This can be useful for applications such as image processing or online shopping recommendations. On the other hand, classification is assigning one of a set of definite labels to an object. Clustering is an example of unsupervised learning, while classification is an example of supervised learning. Many approaches to data mining use a loss function to make their predictions. A loss function is a mapping from some event to a real number based on the cost of that event happening. This cost does not have to be strictly monetary; it could be time, lives, or infinite other types of costs. It is important to note that, even once data mining on a given project is complete, it does not definitively tell the data miner what the next steps are. To obtain that information, it is necessary to analyze the results of the data mining.

2.1 Data Mining Steps

The first step is to make the problem statement. This is the most important step, as the problem statement will guide all the later work to be done. The problem statement poses a question about what kind of information can be gleaned from the data. After this, data must be obtained. Obviously, the methods to accomplish this step are vastly different depending on what kind of data is needed. For example, a survey could be taken, or data could be scraped from some online source. Next, the data is usually enriched, meaning that new supplementary data is added to the set. Real data sets are rarely properly formed, whether it's due to human input error, missing data, or duplicated values. Because of this, the data must be cleaned before it can be used. This step can often be very time consuming since data miners often work with huge amounts of data, and this data is frequently riddled with errors and inconsistencies. Once the data has been cleaned, it is time to integrate it. This means that all of the miner's separate sources of data need to be combined into some cohesive form that can be analyzed. Then the data must be transformed into a format that can be used by the chosen algorithm. This often includes binning or clustering the data. For example, a Naive Bayes classifier does not allow continuous features, so some sort of binning is necessary. This step is what requires the vast majority of the time, space, and money reserved for a project, and is therefore the most important step (besides developing a problem statement) in the data mining process. Mining is the next step, and this is when actual results can be obtained. Once the miner has results, s/he validates them, making sure they make sense in the context of the problem and interprets the results into some easy-to-understand form. The final step is to make the results actionable. This involves relating the results to the problem statement and using the results to determine what further actions are necessary.

2.2 Top Ten Algorithms

There are ten commonly used algorithms in data mining. These are:

2.2.1 C4.5

This algorithm is a descendant of the older ID3 and CLS algorithms. Like these, it uses decision trees to classify data. The tree is grown in a divide-and-conquer manner by recursively partitioning the data and then assigning a leaf to a class label once the data cannot be partitioned or subsequent partitions will not aid in the prediction. This algorithm can handle either numeric values or nominal values, and the nature of the data affects how the partitioning is done. Once this is complete, the tree is pruned to prevent overfitting. Unfortunately, C4.5's generation of rulesets is very CPU and memory intensive, taking 9715 seconds to analyze 100,000 samples.

2.2.2 K-means

This algorithm partitions data into a given number of clusters, k. First, k centroids are chosen, and there are many different ways to pick these centroids. One commonly used approach is randomly sampling the data. Once this is complete, each data point is assigned to the nearest centroid. Obviously, there is no single way to decide ”nearness”; as such, the decision of how to compute distance is extremely important for the success of this algorithm. Next, the mean of every cluster is computed, and this mean takes the place of the centroid. The process is complete when the data assigned to a cluster no longer changes. Unfortunately, there are numerous issues with this algorithm. The most obvious of these is that setting k can be difficult, and setting k poorly will dramatically affect the algorithm's performance. Additionally, it is sensitive to outliers.

2.2.3 Support vector machines (SVM)

SVM are a robust and effective approach to classification. SVM are best understood from the standpoint of a linearly separable data set. In this case, the SVM finds a hyperplane separating the data into two classes by maximizing the distance from the line to the nearest points from either class. Of course, data is almost never linearly separable, so part of the SVM formulation is handling the case where some data is on the wrong side of the line. Additionally, a kernel function can be applied when the data is not linearly separable, which is currently an area of active research. SVM struggled with with slow performance, but this is also being actively solved.

This is done mostly by breaking the large optimization problem into many smaller problems. The below graph shows an example of how an SVM classifies.

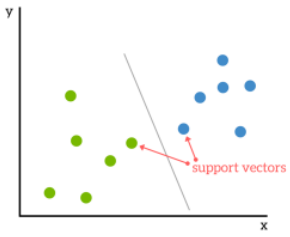

Figure 3. This graph shows the basic principle of an SVM. It finds a line separating the two groups of data, and in this case the data was linearly separable. [13]

2.2.4 Apriori

This algorithm is intended to find frequent itemsets - that is items that occur together frequently - through generating candidates. First, it generates candidates for frequent itemsets of a certain size k+1 from the itemsets of size k. Then it calculates the support for each candidate to determine which of the candidates is an actual frequent itemset. The candidates that meet the minimum support are added to the frequent itemsets of size k+1, and the process is repeated. Unfortunately, the Apriori algorithm generates a large number of candidate sets in some cases, namely when there are many frequent itemsets or low minimum support. Additionally, the algorithm must generate many candidates for itemsets with a large size. For example 2^{100} candidates must be generated for frequent itemsets of size 100.

2.2.5 Expectation-Maximization (EM)

The EM algorithm involves fitting normal mixture models by maximum likelihood. This has numerous advantages, one of which is that it allows the algorithm to model continuous data well. It also allows for the modeling of latent variables, or variables that aren't explicit in the dataset. The algorithm proceeds as one would expect: by finding expectation of the log likelihood function of the conditional distribution of a latent variable given the data and a current estimate of the parameters of the distribution. Then it finds the parameter that maximizes this value. The issue with this algorithm is that the number of clusters must be chosen, and there are many ways to accomplish that.

2.2.6 PageRank

Famously developed by Larry Page and Sergey Brin and used by Google, PageRank ranks web pages by considering a link to a page as a vote for that page. Then it ranks the pages by how many "votes" they get. However, the algorithm also considers the importance of a page doing the voting, since a higher-ranked page should have more clout in what pages it deems important. It accomplishes this task by treating the web as a directed graph and then counting inlinks as part of a vertex's indegree and the outlinks as a vertex's outdegree. The algorithm then simply iterates through the vertices until the ranks do not change very much.

2.2.7 AdaBoost

AdaBoost is an ensemble learning system developed by Yoav Freund and Robert Schapire. It is extremely accurate and simple, with implementation possible in just ten lines of code. First, this algorithm uses a simple learning algorithm to generate class labels for the data. Then it uses the training data to test this learner. It repeats this process T times, and then the result is decided by weighted majority voting by the results of the weak learners. Of course, this does not handle the multiclass case. There is a variant of the algorithm for this case that splits the decisions into a series of binary classification problems. It is also interesting to note that this series of weak learner is comparable in accuracy to a strong learner.

2.2.8 K-nearest neighbors

This algorithm involves finding k items in the data set that are closest to a given object. Like k-means, this "closeness" is not defined, so it is necessary to choose a distance metric carefully. For an unlabeled object, the object's k nearest neighbors are calculated, then those neighbors' class labels vote on the class of the object. One issue with this algorithm is that it is very sensitive to the choice of k. Also, the majority vote is not always a good way to decide the class of an object, since some neighbors may be vastly closer than others. To solve this, a distance-weighted voting system can be used. K-nearest neighbors is well-suited for data in which an object can have multiple labels. The below image shows an example K-nearest neighbors classification.

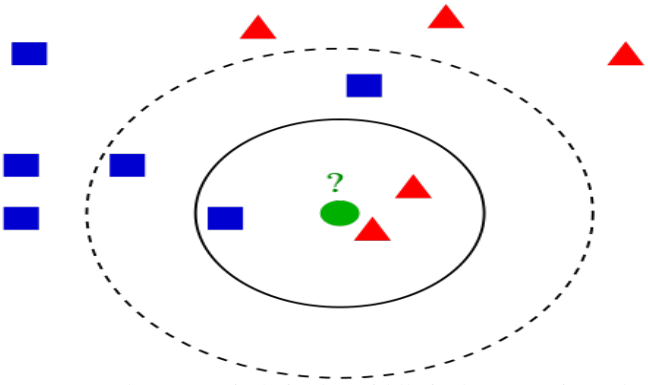

Figure 4. The green circle in the middle is the test point to be classified into either the red triangle class or the blue square class. Depending on k, the test point would be classified differently. If k is 3, it gets assigned to red triangles. If k is 5, it gets assigned to blue squares.[14]

2.2.9 Naive Bayes

Naive Bayes is a commonly used algorithm that, while usually outperformed by more complex algorithms, can be relied

upon to provide good results in a wide variety of problems. Naive Bayes simply calculates the conditional probability of the features given the label and chooses the label with the maximum probability. However, this means that the data must be discretized in some way to ensure that the probability can be computed. Additionally, this algorithm assumes that the features are all independent. They rarely are independent, but this does not make the algorithm perform poorly in practice.

2.2.10 CART

CART, standing for Classification and Regression Trees, was developed by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. The algorithm builds a tree to maximal size and then prunes it back recursively based on which subtree contributes the least to the accuracy of the classification. This accuracy is determined based on external test data or cross validation. CART also supports class balancing, which is an endeavor to make sure the pruning does not unfairly take more occurrences of one class away. The importance of an attribute is calculated by the sum of the improvement in performance in all nodes where the attribute is split upon.

If desired, more information on each of these can be found by referring to [1].

2.3 New Data Mining Problems

Data mining is a field full of active research. There are many problems that are currently being studied in data mining; here are five of them. More information on each can be found by referring to [3].

Scaling Up for High Dimensional Data and High Speed Streams

Data is becoming more and more dense, and there needs to be a way to handle data of this size. For example, biology data often has millions or billions of features. Because of this, more computation power is necessary, or faster algorithms must be developed. Similarly, there are many more ultra-high speed data streams now than there once were, which also poses issues for the old ways of data mining. The data mining algorithms must be fast enough to avoid running behind the stream, which is a difficult problem to solve.

Sequential and Time Series Data

Time series data is extremely valuable, but it can be cumbersome to analyze. An active area of research seeks to answer how to accurately cluster, classify, and predict trends in time series data. One issue faced in time series analysis is that the data is often contaminated by noise, and finding an answer to this problem is important for the future of processing time series.

Mining Complex Knowledge from Complex Data

Data does not always come in the form of a simple table or database. It often comes in the form of a graph or network. Often, this data is neither independent nor identically distributed. This creates problems for most common data mining algorithms. Researchers also want to integrate data mining and knowledge inference, meaning that they want to evolve data mining beyond simply providing the results of algorithms into automatically using those results to make decisions.

Distributed Data Mining

When data mining is done in many areas at once, it is often necessary to relate this data to get a more holistic result. This is difficult right now, but many experts are working hard to solve this issue. One example of this is a network of hundreds of sensors that all provide a piece of the overall picture of the object they're monitoring.

Dealing with Non-static, Unbalanced, and Cost-sensitive Data

Most real-world datasets are massive, having hundreds of thousands of features or more, but unbalanced data has only a few useful features. This means that a large amount of computation time is wasted on features that don't provide value. Additionally, sometimes testing incurs a cost, so using the testing time effectively is extremely important. Clearly, solving these two problems is vital for doing tests on huge data quickly and efficiently.

3. Author Idenfication: Full Problem Description

Identifying the author of a piece of sample text involves finding a probability distribution f that assigns probabilities to each of the authors based on the text excerpts in the training data. These three authors (and their author codes) are H.P. Lovecraft (HPL), Edgar Allan Poe (EAP), and Mary Shelley (MWS). There are 5,635 passages from H.P. Lovecraft's work in the training set, 7,900 examples from Edgar Allan Poe, and 6,044 from Mary Shelley, for a total of 19,579 data points. The inputs and outputs are defined as follows:
Input: A passage from from one of an author's works labeled with the author and an ID.
Output: A probability distribution for each ID that shows the probability that a given passage comes from eacb of the three authors.
The data came from Kaggle pre-split into training and test sets. We initially trained a neural network on a subset of the training data and then tested on another subset since the test data does not contain class labels. After verifying the neural network this way, we submitted our results to Kaggle for evaluation.

3.1 Data Analysis

The data begins as a set of almost 20,000 data points in CSV form. An example row from the dataset is:

id26305, This process, however, afforded me no means of ascertaining the dimensions of my dungeon; as I might make its circuit, and return to the point whence I set out, without

being aware of the fact; so perfectly uniform seemed the wall., EAP

As one can see, the data contains the ID, the text, and the author code. Of course, it is not feasible for the neural network to simply take in all this text. Figures 5-10 show the most commonly used words for each of the authors, displayed in both graphical and word-cloud format.

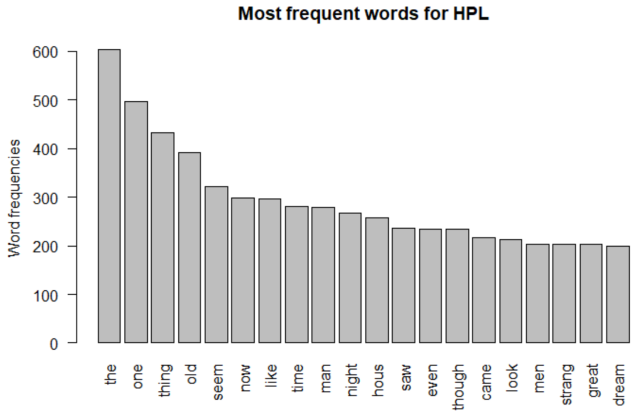

Figure 5. Most commonly used words from H.P. Lovecraft.

Figure 6. Word cloud of most commonly used words from H.P. Lovecraft.

The fact that each author used words with different frequency lends credence to the belief that it is possible to separate the authors based on how often they use specific words. One may also note that the most commonly used words by each author are most often words like ”the” or ”upon.” The first step we took in cleaning the data was the removal of such words, called stop words, as suggested by the Stanford University

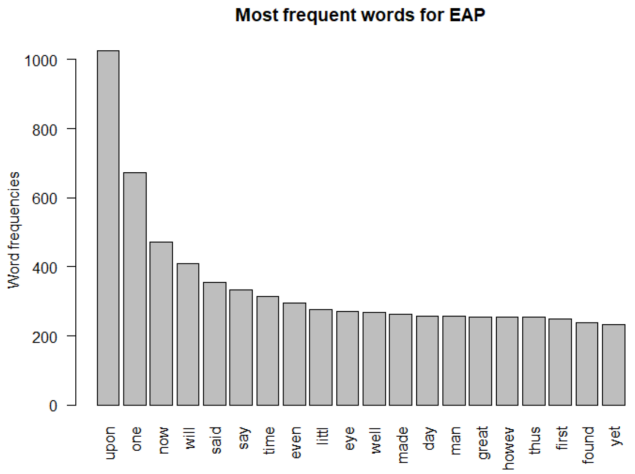

Figure 7. Most commonly used words from Edgar Allan Poe.

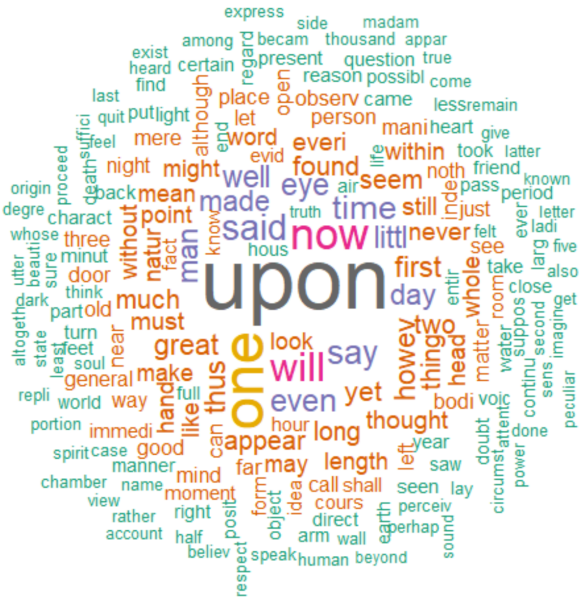

Figure 8. Word cloud of most commonly used words from Edgar Allan Poe.

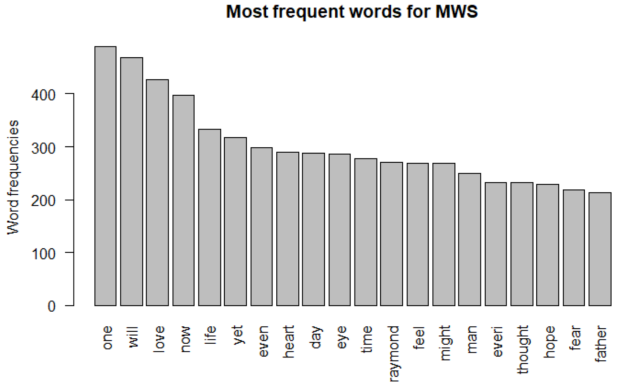

Figure 9. Most commonly used words from Mary Shelley.

Figure 10. Word cloud of most commonly used words from Mary Shelley.

NLP group [6]. Stop words are words like ”the,” ”and,” and ”a” that provide very little meaning for the sentence. They are also the most-used words in the English language, so removing them dramatically decreases the amount of text. Not only are they the most common words in the English language, but they are similarly the most commonly used words by these three authors, as evidenced by the above figures. The next step we took was to remove all punctuation. This does not necessarily decrease the amount of data, but the authors use punctuation differently. For example sometimes one author will use single-quotes and sometimes double, and we did not want to obfuscate the algorithm by modeling punctuation. We also made all letters lowercase to avoid treating words differently if they came at the beginning of a sentence. The removal of punctuation and uppercase letters is suggested here [7]. Once these steps were complete, the above example text looks like this:

process afforded means ascertaining dimensions dungeon might make circuit return point set out without aware fact perfectly uniform seemed wall

However, it is still necessary to turn the text into a form the neural network can work with. To do this, we translated the text to a probability distribution. We counted the number of occurrences of each word in the passage then divided it by the total number of words in this passage. The word is then associated with this probability. This approach was recommended here: [8]. The above example text now looks like this:

{
process: 0.052
afforded: 0.052
means: 0.052
ascertaining: 0.052
dimension: 0.052
dungeon: 0.052
might: 0.052
make: 0.052
circuit: 0.052
return: 0.052
point: 0.052
set: 0.052
out: 0.052
without: 0.052
aware: 0.052
fact: 0.052
perfectly: 0.052
uniform: 0.052
seemed: 0.052
wall: 0.052
}

One may note that each word only occurred once, so they all had the same probability. It was also necessary to put this data into a database for efficiency, since the files were large.

3.2 Methods

All training and testing was done on a PC running Windows 10 with an Intel Core i7 6700k processor, an Nvidia Geforce GTX 980TI graphical processing unit, and 32 GB of 3000 MHZ RAM. The code was written in Python version 3.5, specifically for compatibility with TensorFlow[2]. The neural network was modeled and executed using TensorFlow GPU version 1.4.0 [4]. The database was made in the PostgreSQL database using version 9.6.

The algorithm that we chose to implement was a fully connected neural network. Neural networks have been a popular choice for natural language classification in the past [19]. We chose to implement a simpler, fully connected neural network. As can be seen in Figure 11, the model consisting of two fully connected layers which then produce an output vector, with the probabilities of each author having made the initial text.

The input to the model is a frequency distribution of each words over the given text. We chose this over a simple binary occurrence approach due to the fact that some authors may use certain words significantly more frequently than others, which would not be accounted for when just checking for the presence of a word. Each layer of the model has a set of weights and biases, which determine that amount of activation for each node in the neural network at any given point. The purpose of training our neural network is to continually update these weights and biases after each training step. This is done

[2]Documentation and a high level overview for TensorFlow can be found at https://www.tensorflow.org/

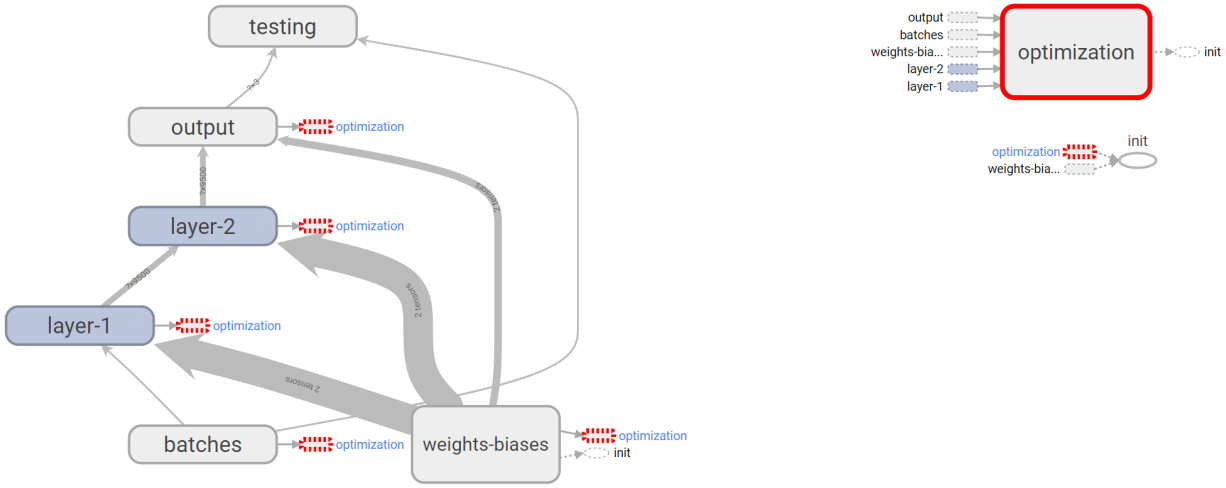

Figure 11. The model.

through an adaptive version of stochastic gradient descent called Adam [18]. The process is handled for us automatically during each training step through the AdamOptimizer class in TensorFlow. During the training of our algorithm we also tested to see the accuracy of our model on a partition of 1/10 of the data that was not used in the model. As Figure 12 shows, we simply calculate the percentage of time that the most likely outcome is not the same as the actual labels for our data.

As seen in Figure 13, we ran into some issues with variations in performance between different runs. In order to combat this, we just chose the best runs out of them, and saved the weights to classify the Kaggle data. We also discovered that certain runs were more prone to overfit the training data than others, which was also alleviated by taking the best model after 200 epochs of training.

3.3 Results

During the testing of the neural network, the average error was around 20%, meaning that 80% of the test set was classified correctly. The training and testing of the neural network took between 15 and 20 minutes, so it was not overly computationally intensive. The accuracy across different epochs can be seen in Figure 14. From this graph, it is clear that the accuracy did not increase at each step, but there was an overall upward trend in accuracy.

A random selection of authors would only classify 33% of the text correctly, making our results fairly successful. However, other competitors on Kaggle were able to achieve over 95% accuracy by using more Natural Language Processing techniques. The results of our classification suggest that these three authors, though they write about similar subjects, have very distinct writing styles. More specifically, each author

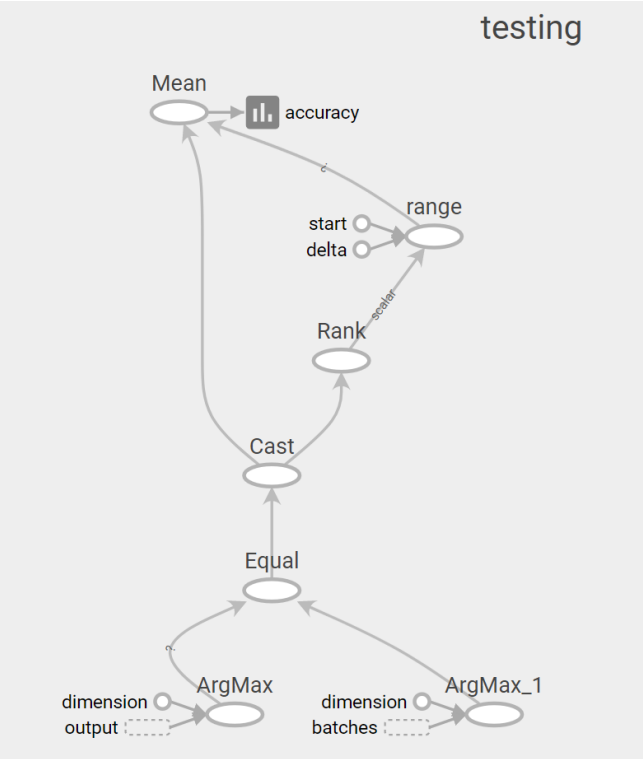

Figure 12. Author data testing process.

uses different words with different frequencies.

There were some challenges with this problem, though. Our first issue was deciding how to represent the text, since we couldn't just use the plain text. We first used a simple bag of words with the counts of each word, but we realized that the text samples had wildly disparate word counts. So instead, we normalized by the total number of words in the excerpt. The

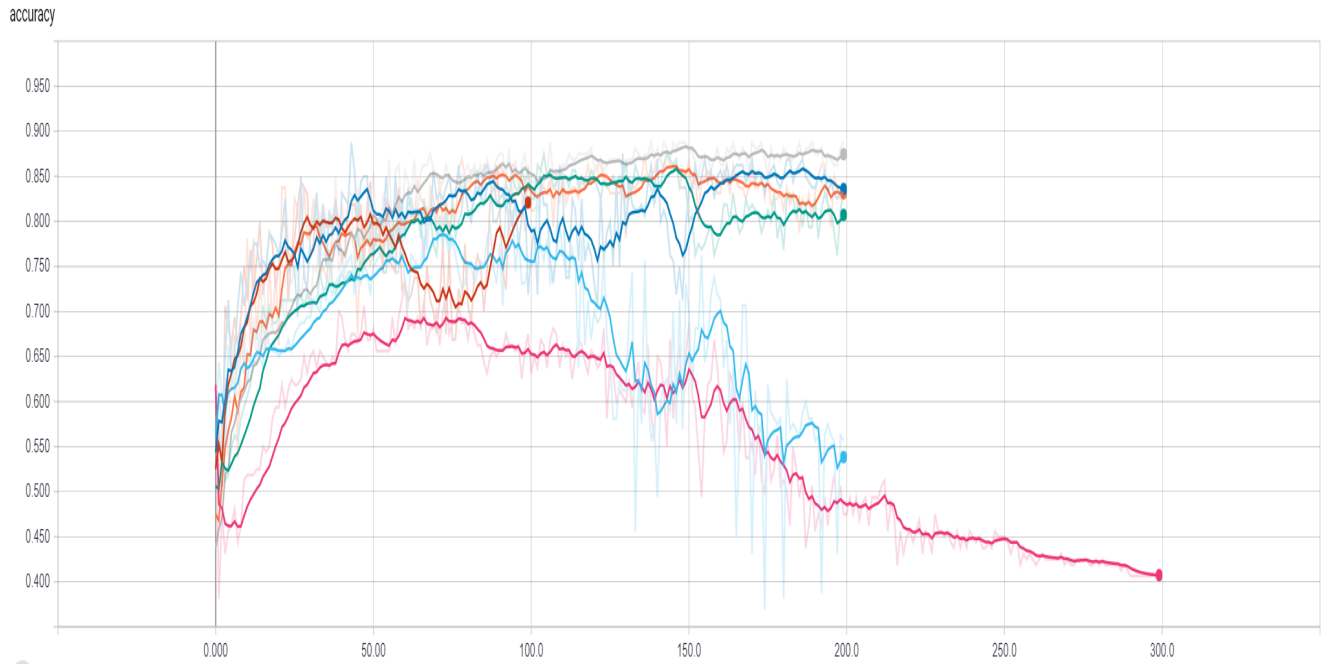

Figure 13. Author testing diagram.

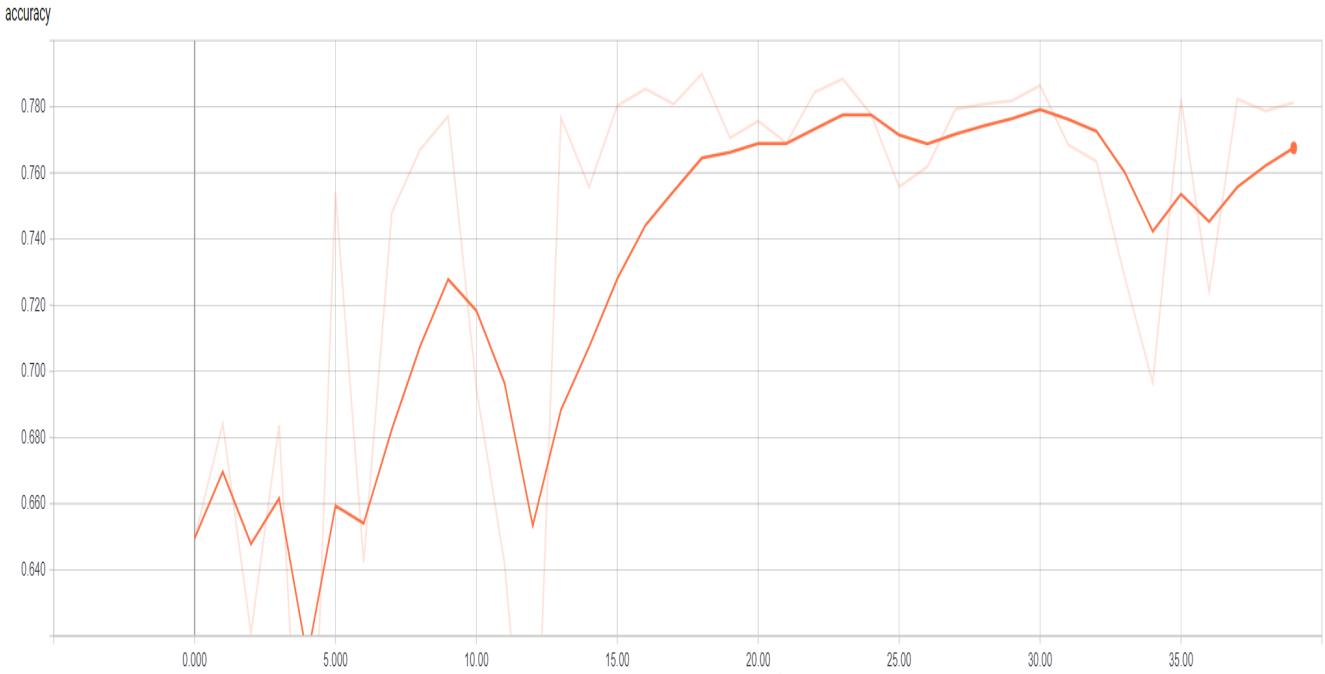

Figure 14. Accuracy at each step.

second challenge was deciding how to implement the neural network. We eventually decided on the implementation of a neural network after observing initial good performance. ‘

3.4 Summary and Future Work

The goal of the project was to accurately classify text by its author based on a set of labeled text. The three authors in both the training and test set (H.P. Lovecraft, Edgar Allan Poe, and Mary Shelley) all wrote in the horror genre. The project was mostly successful, as we were able to predict the label of 80% of the text correctly. This vastly outperforms random guessing, or even simple algorithms like looking for words that only a certain author would use (such as ”Frankenstein” in Shelley’s work).

However, there are numerous possible improvements in the

future. The first of these is to supplement the data with more labeled text from similar authors. For example, while Shelley and Poe were active in a similar time (the mid 19th century), Lovecraft did not begin writing until the early 20th century. Because of this, it may be possible to find words that were not widely-used until Lovecraft's time to further differentiate him from the other two. Additionally, Lovecraft and Poe lived in the Northeast United States, while Shelley lived in England, and there are likely many different word usages in British English versus American English. Beyond that, it may be beneficial to explore the use of other natural language processing techniques. One example technique is to add a feature that tags the text by its parts of speech to train the neural network on the text as well as the underlying structure. An interesting adaption of our algorithm could be to switch from the frequency vector representation of our words, to a vector space representation. This would allow for many more techniques, such as convolutional networks being applied to the problem.

4. Iceberg: Full Problem Description

The objective in this Kaggle competition is to classify satellite images into two categories: pictures of icebergs, and pictures of ships. Each picture is defined by two bands, and each band is 75x75 pixels. The pixels are given as decibel values obtained from radar backscatter. There is also the incidence angle, or the angle from which the images were taken. There are 1604 pictures in the training set. The inputs and outputs are defined as follows:

Input: A labeled image given in two bands, each with 5625 floating point numbers and an incidence angle given in degrees.

Output: A number 0 or 1 for each image. 1 indicates that the image is an iceberg, 0 indicates that it is a ship.

The data came from Kaggle pre-split into training and test sets. We initially trained a neural network on a subset of the training data and then tested on another subset since the test data does not contain class labels. After verifying the neural network this way, we submitted our results to Kaggle for evaluation. All computation was performed on the same PC as the author identification problem.

4.1 Data Analysis

The data comes from Kaggle as JSON files. The training file contains 1604 images represented as 5625 floating point numbers in each of two bands. Each of these bands shows the same picture, but they show the horizontal components and the vertical components of the image respectively. The file also contains the incidence angle. The target variable is "is_iceberg" which is 1 if the picture is an iceberg and 0 if it is a ship. The bands begun as two separate arrays of 75x75 floating point numbers. To better fit our needs, we transformed the two arrays into a single three-dimensional array of size 75x75x2. This will allow the convolutional neural network to treat the image as three dimensional. It was also necessary to put this data into a database since the JSON files were very large.

4.2 Methods

All training and testing was done on a PC running Windows 10 with an Intel Core i7 6700k processor, an Nvidia Geforce GTX 980TI graphical processing unit, and 32 GB of 3000 MHZ RAM. Code was written in Python version 3.5. The neural network was modeled and executed using TensorFlow GPU version 1.4.0 [4]. The database was made in the PostgreSQL database using version 9.6.

The algorithm that we chose to implement was a convolutional neural network with max pooling layers applied after each convolution, and a fully connected layer with dropout before the classification. It has become a common method to implement convolutional neural networks when processing complex and unstructured data of all kinds. They have particularly been shown to outperform fully connected neural networks, and many other algorithms, consistently when applied to image classification tasks [16]. Different types of pooling layers are extremely common in convolutional neural networks [15], which is why we have also incorporated max pooling into our model. The final component of the model is the activation function for each layer. One particularly effective activation function for imagine classification is the rectified linear unit activation function [17]., which is what we use after each layer of the network.

As can be seen in Figure 16, the model is relatively simple. It consists of two convolutional layers with max pooling, and a final fully connected layer. One can see the fully connected layer diagram in Figure 15. After each layer, a relu activation is used as a dropout. We chose to implement a relatively shallow neural network because the image data is relatively simple. None of images appear to have complex features to be gleaned from a large number of layered convolutions. Instead the important features mainly consist of edges and simple geometric shapes, as can be seen in Figure 1 and Figure 2.

The data was stored as single dimensional arrays of floating point numbers in the database, and reshaped as it was run through the neural network. The pixel values are initially of the shape 75 x 75 x 2 when the first convolutional layer acts on them. The first dimension being the 75 x 75 pixel band1 image, and the second dimension being the 75 x 75 pixel band2 image. As can be seen in Figure 17, The convolution 1 node in Figure 16, applies the convolution to the multi-layered image, adds biases, applies a rectified linear activation function, performs max pooling, and then sets a certain percentage of the values to zero before passing the data on to the next layer. The most successful dropout rate throughout our testing seemed to be 5 percent. The application of the convolution involves running 64 filters over the data. All of the filters are represented as 4 x 4 matrices which are stepped one unit at a

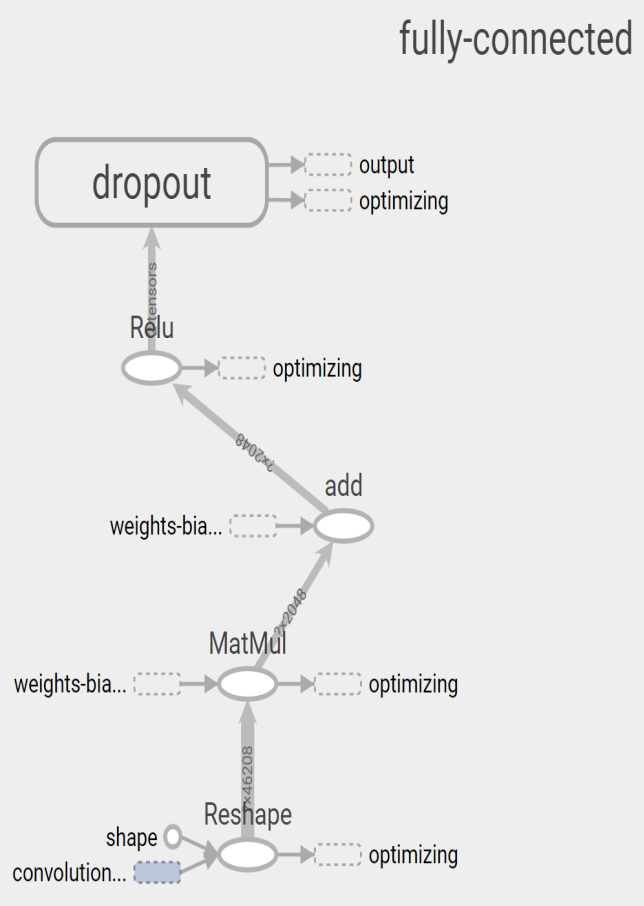

Figure 15. Fully connected layer computation diagram.

time over each of the 75 x 75 pixel images, transforming each 4 x 4 sub matrix into a single value. In our implementation padding is used in the form of repeated values filling open space on the edges, making the resulting convolution is the same size as the original data. The second convolutional layer acts on the result of the first layer, and operates in the same way. With the only difference being that it uses 128 4 x 4 filters. The result of the output is then run through a fully connected layer with the same dropout probability. Finally, it is connected to the output layer, which creates a 2 entry vector, with the likelihood of an iceberg or ship in each entry.

Each of these steps requires a number of parameters such as filter weights, biases, and edge weights. These are all initialized randomly with a normal distribution. However, the heart of neural networks is the updating of these weights. This is done through an adaptive version of stochastic gradient descent called Adam [18]. The process is handled for us automatically during each training step through the AdamOptimizer class in TensorFlow. The resulting optimal weights are what make a trained neural network so much more effective than one initialized from some random distribution.

The same issues of variance came up when training our convolutional network, the difference in performance between different runs was significant. In order to combat this, we just chose the best runs out of them, and saved the weights to classify the Kaggle data. We also discovered that certain runs were more prone to overfit the training data than others, which was also alleviated by taking the best model after 200 epochs of training.

4.3 Results

Our convolutional neural network was able to correctly classify 85% of the test set. This is a fairly successful result, vastly outperforming the random guess accuracy of 50%. The accuracy at each step can be seen from Figure 18. From this graph, it is clear that the accuracy fluctuated much more than in the author identification problem, but still maintained a mostly upward trend. However, similar to the author identification problem, the top teams on the Kaggle leaderboard were classifying over 95% of the images correctly. They may have used some different algorithm, such as a support vector machine, or found some supplementary data that dramatically improved prediction. These results suggest that icebergs look sufficiently different from ships, even from a satellite's view. When visually inspecting the images, one can see that the icebergs are generally much more jagged and irregularly shaped, which makes intuitive sense.

We faced various challenged when working on this project. The first issue we faced was deciding on what algorithm to use. We knew we wanted to use a neural network, but didn't know what kind. After some research, we saw that convolutional neural networks were recommended for image classification [11] and decided to use a convolutional neural network. Then we saw that it would not be possible to automatically put the test file into a database since the file was larger than a gigabyte. To fix this issue, we used a feature of Python which allows the user to load a JSON file as floating point numbers instead of a large string. This dramatically reduced the amount of memory needed, so much so that it easily fit in the memory allotted to Python, and we inserted the values into the database using Python.

4.4 Summary and Future Work

The goal of this project was to accurately predict whether a given image was of a ship or an iceberg based on a set of training images and the incidence angle from which the images were captured. Our convolutional neural network had a fairly good success rate of 85%.

While this does vastly outperform simpler algorithms, there are numerous improvements we could make. One improvement we could make, as suggested here [12], is to normalize the dataset for brightness. We know that this is a problem with the dataset because the authors of the competition on Kaggle stated that high winds generate a brighter background [10]. We would do this by subtracting each value from the mean decibel values. It also might be possible to use the incidence

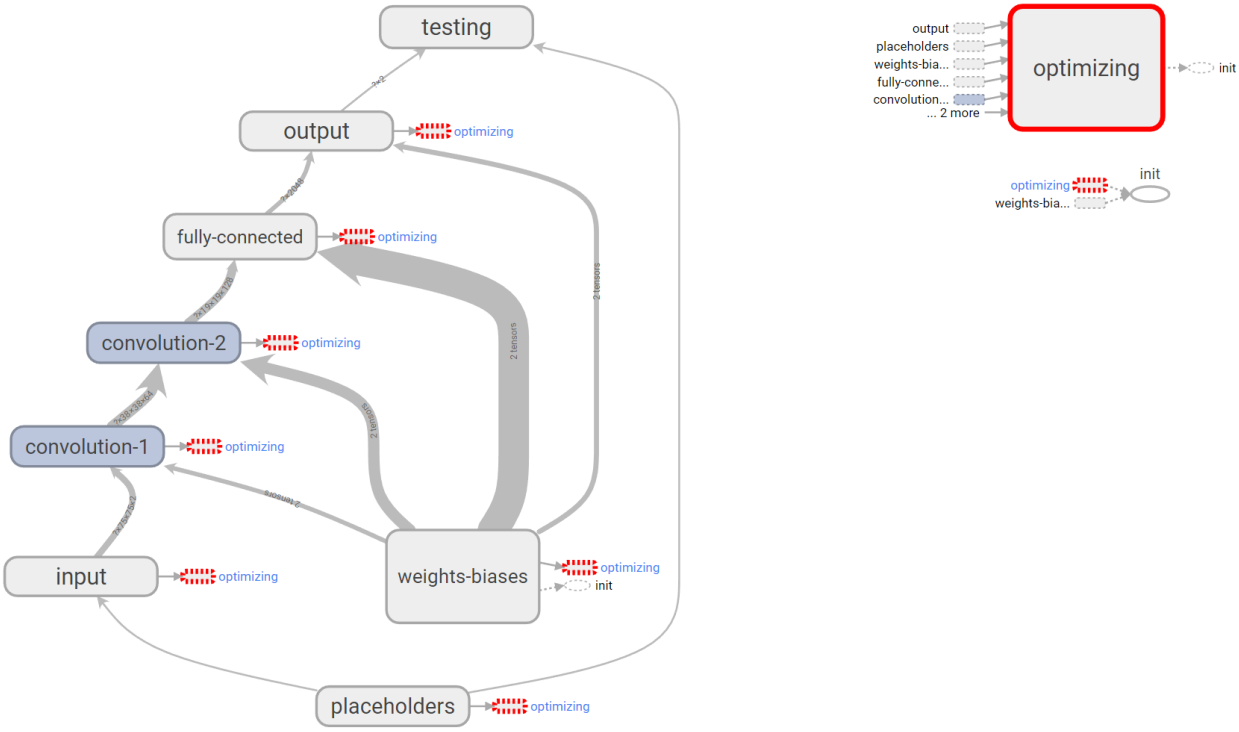

Figure 16. The model.

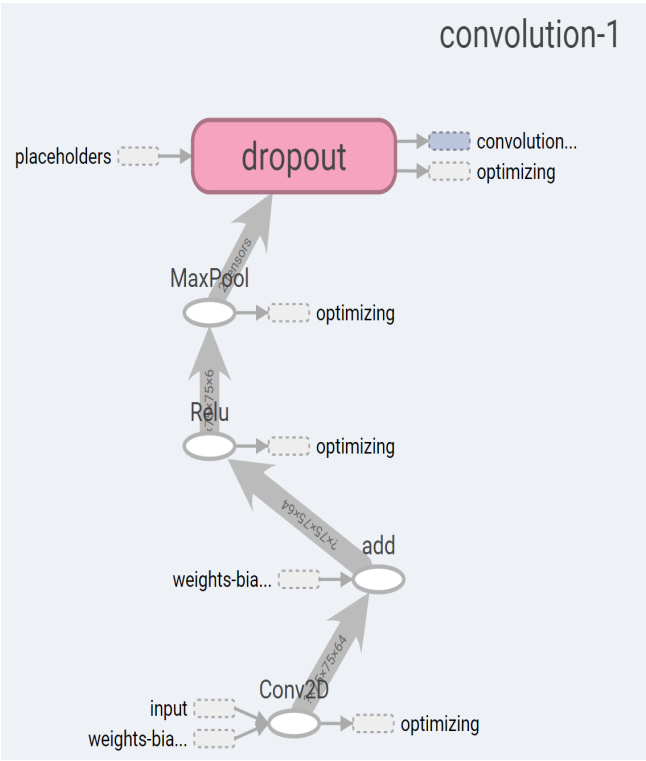

Figure 17. Convolution computation diagram.

angle to determine if the picture was taken in a place that ships never traverse. For example, a picture taken in the arctic is almost definitely an iceberg. However, this would need to be researched extensively to ensure that we could accurately deduce where the picture was taken. It would also be possible to do more preprocessing on the images themselves, such as blurring or removing the edges so the iceberg/ship is the only non-ocean element of the picture. We also could have attempted to add several predetermined filters for our data. We know a good amount about the structure of our data, and how icebergs naturally differ from ships. For example, icebergs tend to have much more jagged edges than ships. We could leverage this information by seeding some of the trained filters with what we think would be successful. This would help to alleviate some of the issues with the randomized initial conditions being stuck in local minimums.

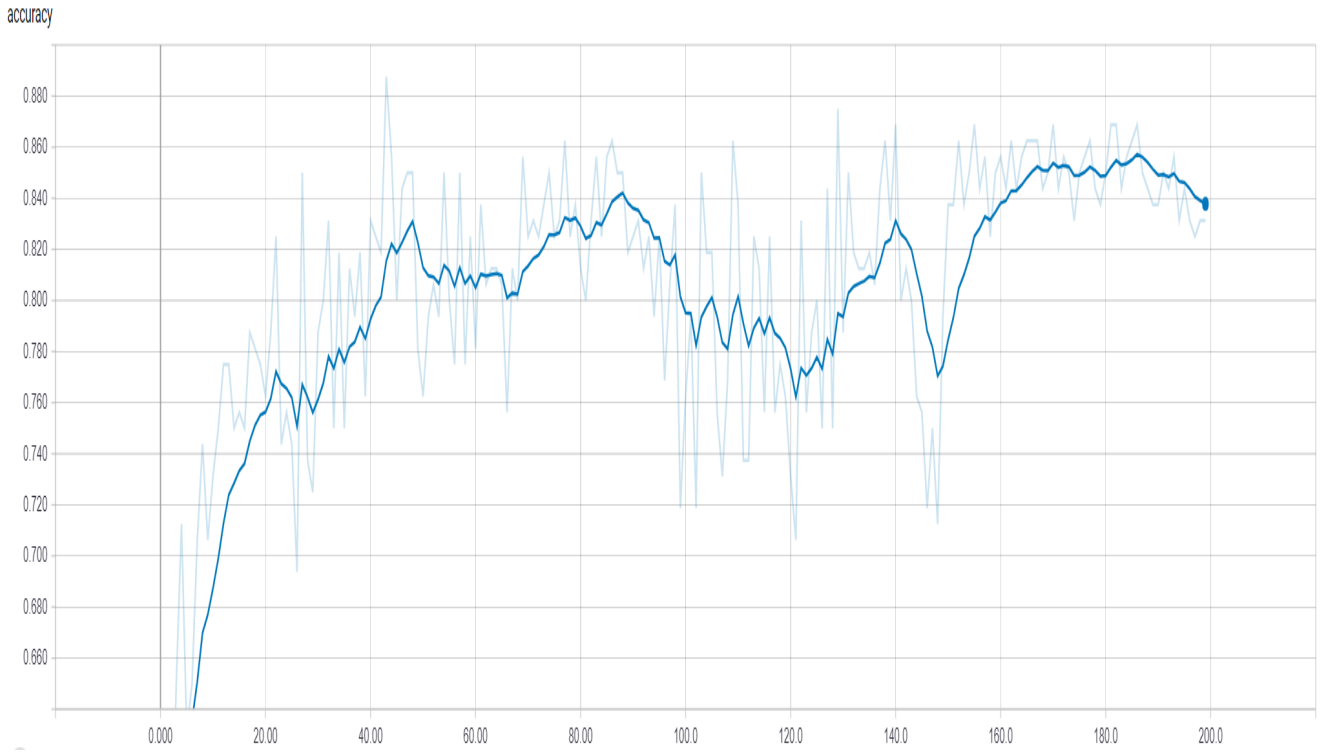

Figure 18. Accuracy at each step.

Acknowledgments

Thanks to Indiana University for giving us the opportunity to work on this. Thanks to Marcin Malec for his advice on various aspects of these two projects. Thanks to Dr. Mehmet Dalkilic for teaching us what data mining is and how to do it.

References

[1] Xindong Wu et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14:1–37, 2008.

[2] Edgar Allan Poe. *The Cask of Amontillado*. 1846.

[3] Qiang Yang et al. 10 challenging problems in data mining research. *International Journal of Information Technology and Decision Making*, 5:597–604, 2006.

[4] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems r1.4, 2017. Software available from `tensorflow.org`.

[5] Steven Bird. Natural language toolkit version 3.2.5, 2017. Software available from `nltk.org`.

[6] Dropping common terms: stop words. `https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html`, 2008. Accessed: 2017-12-10.

[7] Leila Arras et al. What is relevant in a text document?: An interpretable machine learning approach. *PLOS One*, 12, 2017.

[8] Bag of words (bow). `https://ongspxm.github.io/blog/2014/12/bag-of-words-natural-language-processing/`, 2014. Accessed: 2017-12-10.

[9] Spooky author identification. `https://www.kaggle.com/c/spooky-author-identification#description`, 2017. Accessed: 2017-12-10.

[10] Statoil/c-core iceberg classifier challenge. `https://www.kaggle.com/c/statoil-iceberg-classifier-challenge#Background`, 2017. Accessed: 2017-12-10.

[11] Xin-Yu Ou, He-Fei Ling, Ling-Yu Yan, and Mao-Lin Liu. Multi-column deep neural networks for image classification. *Asia-Pacific Signal and Information Processing Association 2014 Annual Summit and Conference (APSIPA)*, pages 1–10, 2014.

[12] Emine Cengil et al. Preprocessing for image classification by convolutional neural networks. *Computer Science and Engineering (UBMK) 2017 International Conference*, pages 440–444, 2017.

[13] Noel Bambrick. Support vector machines: A simple explanation, 2016. image.

[14] Wikimedia Commons. K-nearest neighbors algorithm, 2017. image.

[15] D. Scherer, A. Muller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object

recognition. *Proc. of the Intl. Conf. on Artificial Neural Networks*, pages 92–101, 2010.

[16] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. *Seventh International Conference on Document Analysis and Recognition*, pages 958–963, 2003.

[17] George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. *Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[18] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[19] Yoon Kim. Convolutional neural networks for sentence classification. *Conference on Empirical Methods in Natural Language Processing*, 2014.

[20] Harrison. Deep learning with tensorflow - creating the neural network model, 2016. video.