Neural Networks for the Automated Identification of Anomalies in Geomagnetic Data

Matthew Johnson, Erik Steinmetz, Augsburg College

Abstract- This paper outlines an attempt to automate the process of identification of anomalous patterns in geomagnetic data. The original goal of the research project this paper describes was to identify in an automated manner such patterns in the data of the MACCS project, a geomagnetic survey conducted jointly by Augsburg University, Boston University, and the University of Alberta. The method to be used for the identification of these anomalies was prescribed by Augsburg faculty as neural networks. Given that neural networks require large amounts of labeled data in order to be trained to identify patterns that correspond to the labeling of the data and that there was an insufficient amount of current MACCS data for such a purpose, the research project instead relied on data from the British Antarctic Survey to train neural networks. The type of neural network that was utilized was a convolutional neural network, an architecture that uses filters on matrix representations of the data in order to intelligently constrain connections between layers of the neural network. The neural network was modeled after LeNet, a convolutional neural network architecture that has been utilized to positive effect in the categorization of images of Hindu-Arabic numerals or more simply, the digits 0-9. Of the neural networks that were produced as part of the research project, the highest accuracy achieved on British Antarctic Survey data was an accuracy of 97%.

I. INTRODUCTION

The MACCS project is a project that hopes to track the behavior of the earth's magnetic field near the northern pole of the earth's magnetic field. Currently anomalous patterns in the data that corresponds to the behavior of the earth's magnetic field near the northern pole are identified by hand. Given that there is a large volume of data, it was determined that it might be helpful to automate the process of identifying these anomalies. One method of automating this process of anomaly identification is to leverage the recent advances made in defining the structure and methods for training of neural networks on the geomagnetic data of the MACCS project. It was determined that the method that might be of the most use towards this end could be a convolutional neural network. The hope of the research project is that by automating the process of identification of anomalies, the investment of time by human annotators might be reduced.

II. METHODOLOGY

There were two main processes that were utilized in the course of the research project. The first process was the formatting and labeling of geomagnetic data. The second process was the training of neural networks on this labeled data set.

A. Data Processing

Neural networks require large sets of labeled data in order to be trained to identify patterns in the data that correspond to the labels of that data. There was not a labeled data set available for MACCS data, nor was there enough current data from the MACCS project to be able to train a neural network. It was determined that a good stand in for MACCS data was geomagnetic data from the British Antarctic Survey, a survey of the environmental conditions of the Antarctic including readings taken by magnetometers that are considered to correspond to the behavior of the earth's magnetic field around the Antarctic environment.

The data from the British Antarctic Survey was acquired from their publicly available servers [1]. The data acquired was the ASCII representation of magnetometer readings from the Halley IV SCM. The readings represent the sampling of the magnetic field near the magnetometer in three dimensions, sampled at a rate of one sample every tenth of a second. This data can be decomposed from the complex waveform

represented in the ASCII to a composition of different frequencies and magnitudes at each frequency by means of a Fourier Transform. The data plots representing this decomposition as performed by the British Antarctic Survey were also acquired, with each data plot consisting of measurements of magnetic intensity in three dimensions labeled in the data as X, Y, and Z axes. These data plots were then sectioned into windows two hours wide by a python script run at the console, and presented to the user via the default image viewing software of the system the python script was run on. The user would close out of the window and enter into the console a yes, no, or discard response based on certain criteria. The response was then recorded as the label for that two hour window in a text file. This process was repeated for each two hour window in each of the three dimensions recorded by the magnetometer for a total of 36 labels for each day of data. In total 1527 days of data were labeled in this manner.

The criteria for classification were determined by the researcher with the assistance of Augsburg faculty. The criteria for a non-anomalous readings were as follows:

- There was no event. An event is categorized as a change in excess of a factor of 10 over either a 50 Hz interval or a 10 minute interval.
- If an event occurred, the event is continuous from 0 Hz. An event is continuous if any drop in the readings in excess of a factor of 10 is not reversed at a frequency 25 Hz greater than the frequency at which the drop occurs.

The criteria for an anomalous readings were as follows:

- The event contains a change of power of at least a factor of 100 over either a 50 Hz interval or a 10 minute interval.
- The event is strongly discontinuous from 0 Hz and 1000 Hz. The event is separated from activity which is continuous from 0 Hz or 1000 Hz by a range of at least 75 Hz.

For events that did not fit either of these criteria, the readings were discarded. There were two sets of labels that were created as part of this process, one set of 1117 days of labels and one set of 628 days of labels. There was an overlap of 218 days between these sets of labels, with a subset of those days existing in both sets of labels compared for the purposes of consistency in labeling. A subset of the total 1527 days of labeled data were also reviewed by Augsburg faculty for accuracy in labeling.

There were several methods used to represent the readings from the magnetometers at BAS Halley Site. The first step regardless of the method was to decompose the complex waveforms represented by the ASCII files. This decomposition was performed by first computing the numerical derivative of the data in the ASCII file, normalizing the complex waveform to the magnitude of the first reading. The average of every ten samples was then taken on this numerical derivative, providing the average value of the wave form measured in one second intervals. Next, the normalized and averaged complex wave form was decomposed into 256 different frequencies with magnitudes provided at each frequency for each minute and a half long interval in the readings. This decomposition was performed using the Fast Fourier Transform software provided with the python software library, numpy [4].

The simplest method of presenting the data to the neural network was to use the data provided by the Fast Fourier Transformation without any additional modification. One method of representing the data after it had been decomposed by a Fast Fourier Transform with additional modifications was to take the log base 10 of each of the magnitudes provided as output from the Fast Fourier Transform. This method is referred to here as the log10 method. Another method was to represent the values output from the Fast Fourier Transform as discrete values, wherein the integers $\{m \in \mathbb{Z} : 1 \leq m \leq n\}$ were utilized to represent *n* different ranges of magnitudes and each magnitude from the output of the Fast Fourier Transform placed into one of these ranges. This method is referred to here as the bucket method. The last method utilized was to take the log base 10 of each of the magnitudes as output from the Fast Fourier Transform, and then to place them into the discrete buckets described above. This method is referred to here as the log10bucket method.

B. Neural Network Training

After having labeled and transformed the BAS Data, several different neural network architectures were implemented using the python library keras [2], which was used as an API for the TensorFlow software library [5]. The neural network architectures used closely mirror LeNet, a neural network architecture first specified by LeCun, et al [3].

We might note here for the purposes of review that a neural network can be considered as a directed graph wherein nodes are separated into different sets, each of these sets being a layer of the neural network. Each of the nodes in one layer are connected by directed edges to the nodes in another layer that is considered as a subsequent layer of the neural network. The initial layer of the neural network is the input layer, and each of the nodes in the input layer is assigned a numerical value corresponding to a value in the input data. The terminal layer of the neural network is the output layer and is preceded by a function that computes a probability distribution over the number of nodes in the output layer. For the purposes of this project the output layer consisted of two nodes, one corresponding to a labeling of the input data as anomalous and one corresponding to a labeling of the input data as non-anomalous. The function for computing the probability distribution over these two nodes was the softmax function. The directed edges that connect each of the layers represent computations that are performed on the numerical values of each of the nodes in the layer on the front side of the directed edge. The computations can be as simple as multiplying the value of the input to the computation by a constant referred to as a weight, then adding to the result a constant referred to as a bias. The result of this computation is then passed through an activation function that normalizes the output of the computation to a specific range. For the purposes of this project the activation function was the Rectified Linear Unit. The weights and biases of each of the directed edges in the graph representation of the neural network are adjusted as part of the training process, and the final values arrived at for each of the weights and biases stored in an output file. Together with the structure of the graph representation of the neural network, these weights and biases form the definition of a neural network.

We might also note that there are different ways of training neural networks. In order to train a neural network, one must first define a way of measuring the accuracy of the outputs of the neural network. This method of measuring the accuracy is referred to as a loss function, which for the purposes of this research project was the categorical cross-entropy function. Once a measure of accuracy has been defined, one can take the derivative of the a given weight or bias with respect to the output of the loss function. These derivatives are then used to adjust the weights of the neural network according to different algorithms, called optimizers. The two optimizers that were used in the research project were the Stochastic Gradient Descent algorithm and the optimizer defined by Google and referred to by the name Adam. The optimizers are used to adjust the network with respect to each of the inputs in the training data set, the data set passed in to the training algorithm in batches of a fixed size that allow for all training data in the batch to be held in memory simultaneously. One iteration over the full data set is referred to by the keras API as a training epoch, and multiple epochs may be run on the same network to increase the accuracy of the network. A set of epochs is referred to here as an instance of training and produces one set of weights. A set of training instances is referred to here as a training session, where the training parameters (optimizer and its settings, input data and its format) for the training session are the same for all training instances in that session.

All of the neural network architectures used in the research project can be considered to be deep neural networks as they contain 3 or more layers in the network. Each of the architectures used also contained convolutional layers along with max pooling layers. We can consider convolutional layers to be intelligent methods of restricting the connection of nodes in one layer to the nodes of the subsequent layer, the method being derived from a geometric representation of the input data. Each of the architectures had an input layer of a size 256 by 80. There were in total 7 different architectures that were developed in the course of the research project. Some of the architectures incorporated dropout layers, a specification in the keras API that allows the designer to randomly drop a certain percentage of the weights that have been trained

between training epochs. The different architectures used were named by the date they were developed along with an enumeration for that date and are detailed in the appendix at the end of this paper.

III. RESULTS

The data of the results is located in an appendix at the end of this paper. The data that was recorded as the measure of the success of a given training instance was the accuracy of a given network architecture in a given training session. These training instances were conducted by first training a network on a randomly selected subset of the labeled data produced as described above. The portion of the labeled data not used in training was then used to test the accuracy of the network by providing them as input to the network and then observing if the classification provided by the network was consistent with the label provided. The accuracy of the network is the percentage of samples in this testing set that are correctly classified according to the provided label. Multiple training instances, each containing multiple epochs, were run for each given set of training specifications, and the average of the accuracies of these training instances was used as the measure of success for a given set of training specifications. The average accuracies of each network for a given set of training specifications is referred to here as a training session.

The process used to acquire the data was to first run training sessions on a network directly resembling LeNet, as is found in network 170627. Subsequent training runs varied the architecture of the network, initially by altering hyper parameters and then by altering the sequence and number of layers. The result of altering the hyper parameters, that is to say by altering the number of convolutional filters in a given convolutional layer, appears to have had a positive effect on accuracy when a smaller number of convolutional layers were used. The result of varying the architecture revealed that a smaller number of layers did not have a positive effect on accuracy. These results seemed to suggest that an overfitting of the network's weights to patterns that were present in the data, but not the patterns that were wanting to be identified was occurring. In light of this, the architecture was altered to include drop out layers, a method that can be utilized to prevent the severity of the overfitting of the weights to unwanted patterns. The alteration of the architecture to include dropout layers did appear to have a positive effect on accuracy, though the effect was more notable between a convolutional layer and a dense layer than between two convolutional layers.

The training sessions also varied the optimizers that were used for the purposes of training, as well as the learning rates used by these optimizers. Stochastic Gradient Descent was utilized with all of the network architectures with learning rates of .005, .01, and .1. Adam was utilized with all of the network architectures with learning rates of .001, .005, .01, and .1. The pattern observed in the data was that the optimizer Adam consistently produced networks with a higher prediction accuracies than SGD. The learning rate of .001 also seemed to more consistently produce networks with higher prediction accuracies than learning rates of .005, .01, and .1. This was consistent with the finding of the publishers of the Adam optimization algorithm.

Once a network and training specification had been identified, specifically architecture 170705-1 with optimizer Adam at a learning rate of .001, variations on the input data was performed. Testing on the first set of labeled data was done (set size of 1117), and it was observed that the first set of labels tended to produce less accurate networks than the second set of labeled data (set size of 628). Testing on the first and second set of labels combined was more accurate than either the first set or the second set alone (combined set size of 1527). Variations on the representation of the input data were also made. The pattern in the data seemed to suggest that the log10 representation of the input data was less accurate than either of the discrete representations we have here titled the bucket representation and the log10bucket representation. It also appeared that the bucket representation produced more accurate networks than the log10bucket representation. Finally, it was observed that a discrete representation with a smaller number of ranges was more accurate than discrete representations with a larger number of ranges. The final set of training parameters was the architecture 170705-1 with

optimizer Adam at a learning rate of .001 and with the input data transformed and placed into buckets with ranges of length 100000. This final set of training parameters produced networks with an average accuracy of 96.58%.

IV. FUTURE WORK

Were this project to be continued in the future, strong avenues of investigation might be to test the efficacy of different sets of labels, translation of these networks trained on BAS data to be able to make predictions on MACCS data, or for different alterations of the input data to the networks to be performed. Another attempt at classifying data in a manner similar to the networks described in this paper might do away with the convolutional architecture entirely and instead opt to connect layers of a network using a description of a general shape as the criteria for connecting the nodes of two adjacent layers. Other attempts still might use techniques that are not neural networks at all.

ACKNOWLEDGEMENT

A special thank you to the Augsburg Office for Undergraduate Research and Graduate Opportunities for the funding to be able to conduct this research during the summer of 2017, Jennifer Posch of the Augsburg Physics department for her assistance in finding adequate data for the training of a neural network as well as assistance in labeling the data, and to Professor Erik Steinmetz who supervised the research project and consistently provided support to the author in developing an ability to ask the right questions.

REFERENCES

[1] British Antarctic Survey. Halley SCM Dataset. http://psddb.nerc-bas.ac.uk/data/access/coverage.php?menu=1\&source=1\&class=101\&script=1. Retrieved 27 July 2017.

[2] Keras, The Python Deep Learning Library. https://keras.io/.

[3] Y LeCun, L Bottou, Y Bengio, P Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, volume 86, issue 11, pages 2278-2324. (1998).

[4] SciPy. One-dimensional Discrete Fourier Transform. https://docs.scipy. org/doc/numpy-1.12.0/reference/generated/numpy.fft.fft.html.

[5] TensorFlow, An open source software library for Machine Intelligence. https://www.tensorflow.org/.

APPENDIX

The data of this appendix are records of the average accuracies of different training sessions across different neural network architectures, where the accuracy is understood as being the percentage of samples that were correctly identified from a testing set of samples, a training session is understood to be a set of training specifications, and a network architecture is understood to be the type and order of layers in the neural network.

The network architectures are as listed below:

- 170627: Convolutional Layer (20 filters, kernel 5x5), Max Pooling Layer (4x4), Convolutional Layer (50 filters, kernel 5x5), Max Pooling Layer (4x4), Dense Layer (500 nodes), Softmax Classifier (2 outputs)
- 170702-0: Convolutional Layer (20 filters, kernel 5x5), Max Pooling Layer (4x4), Convolutional Layer (20 filters, kernel 5x5), Max Pooling Layer (4x4), Dense Layer (500 nodes), Softmax Classifier (2 outputs)
- 170702-1: Convolutional Layer (20 filters, kernel 5x5), Max Pooling Layer (4x4), Dense Layer (500 nodes), Softmax Classifier (2 outputs)
- 170705-0: Convolutional Layer (20 filters, kernel 5x5), Max Pooling Layer (4x4), Dropout Layer (25\% drop rate), Convolutional Layer (20 filters, 5x5 kernel), Max Pooling Layer (4x4), Dense Layer (500 nodes), Softmax Classifier (2 outputs)
- 170705-1: Convolutional Layer (20 filters, kernel 5x5), Max Pooling Layer (4x4), Convolutional Layer (20 filters, kernel 5x5), Dropout Layer (50\% drop rate), Dense Layer (500 nodes), Softmax Classifier (2 outputs)

- 170705-2: Convolutional Layer (20 filters, kernel 5x5), Max Pooling Layer (4x4), Dropout Layer (25\% drop rate), Convolutional Layer (20 filters, kernel 5x5), Max Pooling Layer (4x4), Dropout Layer (50\% drop rate), Dense Layer (500 nodes), Softmax Classifier (2 outputs)
- 170705-3: Convolutional Layer (20 filters, kernel 5x5), Max Pooling Layer (4x4), Dropout Layer (50\% drop rate), Dense Layer (500 nodes), Softmax Classifier (2 outputs).

The training sessions and their parameters are as listed below:

session170705	session170706	session170707_0	session170707_1	session170708_0
Number of Instances=15	Number of Instances=15	Number of Instances=15	Number of Instances=15	Number of Instances=15
Optimizer = SGD	Optimizer=SGD	Optimizer=SGD	Optimizer=SGD	Optimizer=adam
Learning Rate = 0.01	Learning Rate=0.01	Learning Rate=0.1	Learning Rate=0.005	Learning Rate=0.005
Shuffle = False	Shuffle=True	Shuffle=True	Shuffle=True	Shuffle=True
Input Size = 48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256
Input Type = set0(partial)	Input Type=set0(partial)	Input Type=set0(partial)	Input Type=set0(partial)	Input Type=set0(partial)
Number Inputs = 562	Number Inputs=562	Number Inputs=562	Number Inputs=562	Number Inputs=562

session170708_1	session170709_0	session170709_1	session170711_0	session170711_1
Number of Instances=15	Number of Instances=35	Number of Instances=35	Number of Instances=35	Number of Instances=35
Optimizer=adam	Optimizer=adam	Optimizer=adam	Optimizer=adam	Optimizer=adam
Learning Rate=0.1	Learning Rate=0.005	Learning Rate=0.1	Learning Rate=0.005	Learning Rate=0.01
Shuffle=True	Shuffle=True	Shuffle=True	Shuffle=True	Shuffle=True
Input Size=48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256
Input Type=set0(partial)	Input Type=set0(partial)	Input Type=set0(partial)	Input Type=set0(partial)	Input Type=set0(partial)
Number Inputs=562	Number Inputs=562	Number Inputs=562	Number Inputs=562	Number Inputs=562

session170712_0	session170712_1	session170713_0	session170713_1	session170717_0
Number of Instances=100	Number of Instances=50	Number of Instances=175	Number of Instances=56	Number of Instances=75
Optimizer=adam	Optimizer=adam	Optimizer=adam	Optimizer=adam	Optimizer=adam
Learning Rate=0.005	Learning Rate=0.01	Learning Rate=0.005	Learning Rate=0.005	Learning Rate=0.005
Shuffle=True	Shuffle=True	Shuffle=True	Shuffle=True	Shuffle=True
Input Size=48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256
Input Type=set1(partial)	Input Type=set1(partial)	Input Type=set1(partial)	Input Type=set0	Input Type=set0
Number Inputs=~120	Number Inputs=~120	Number Inputs=270	Number Inputs=1268	Number Inputs=~1300

session170717_1	session170719_0	session170719_1	session170723_2	session170723_3
Number of Instances=3	Number of Instances=25	Number of Instances=10	Number of Instances=1	Number of Instances=1
Optimizer=adam	Optimizer=adam	Optimizer=adam	Optimizer=adam	Optimizer=adam
Learning Rate=0.1	Learning Rate=0.05	Learning Rate=0.05	Learning Rate=0.001	Learning Rate=0.001
Shuffle=True	Shuffle=True	Shuffle=True	Shuffle=True	Shuffle=True
Input Size=48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256
Input Type=random	Input Type=set1	Input Type=set0	Input Type=set1(partial)	Input Type=set1(partial)
Number Inputs=18000-22000	Number Inputs=629	Number Inputs=1527	Input Formatting=log10buckets	Input Formatting=log10
			Number Inputs=614	Number Inputs=614

session170723_4	session170723_5	session170724	session170725	session170727
Number of Instances=10	Number of Instances=15	Number of Instances=5	Number of Instances=20	Number of Instances=10
Optimizer=adam	Optimizer=adam	Optimizer=adam	Optimizer=adam	Optimizer=adam
Learning Rate=0.001	Learning Rate=0.001	Learning Rate=0.001	Learning Rate=0.001	Learning Rate=0.001
Shuffle=True	Shuffle=True	Shuffle=True	Shuffle=True	Shuffle=True
Input Size=48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256	Input Size=48x256
Input Type=set0,set1	Input Type=set0,set1	Input Type=set0,set1	Input Type=set0,set1	Input Type=set0,set1
Input Formatting=log10buckets	Input Formatting=log10	Input Formatting=log10	Input Formatting=buckets	Input Formatting=buckets,fewer
Number Inputs=1527	Number Inputs=1527	Number Inputs=1527	Number Inputs=1527	Number Inputs=1527

The average accuracy of each network architecture for each training session are as follows:

	170702-0	170702-1	170705-0	170705-1	170705-2	170705-3
session170705	52.38%	80.16%	84.76%	84.76%	80.12%	84.76%
session170706	71.19%	66.58%	80.20%	84.76%	84.76%	75.56%
session170707_0	66.06%	84.73%	71.13%	84.76%	84.76%	80.12%
session170707_1	75.63%	80.24%	84.76%	80.20%	80.12%	80.20%
session170708_0	84.77%	80.45%	72.05%	84.76%	84.76%	84.76%
session170708_1	75.59%	80.04%	80.12%	84.76%	80.12%	84.76%
session170709_0				84.76%	82.77%	82.77%
session170709_1				80.78%	80.78%	78.80%
session170711_0	82.84%	72.95%	80.49%			
session170711_1				84.76%	80.78%	82.80%
session170712_0				73.15%	73.62%	73.15%
session170712_1	60.01%	61.91%	65.27%	73.62%	73.62%	72.67%
session170713_0				73.86%	73.03%	73.02%
session170713_1				84.35%	83.17%	81.92%
session170717_0				82.70%		
session170719_0				76.15%	74.06%	74.06%
session170719_1				83.45%	83.45%	83.45%
session170723_2				74.40%	74.40%	74.40%
session170723_3				74.40%	74.40%	74.40%
session170723_4				83.67%	83.67%	83.67%
session170723_5				83.67%	79.25%	74.83%
session170724				83.67%	83.67%	70.40%
session170725				87.42%		
session170727				96.58%		