Capstone Project: Prediction of Ice Elevation Data based on Time and Location

Ravel Larose - DSCI 799 – Capstone Formal Project Write-Up

Executive Summary

This report encapsulates the breadth of the ice elevation prediction capstone project, from the early steps of data collection and analysis to the research required to pick a neural network architecture, the process of building and testing models, and finally the results of the final model and a discussion of what the results mean, and what next steps would be viable should the experiment be repeated.

In brief, the data was collected from a National Aeronautics and Space Administration (NASA) project that studied ice elevations over large swaths of Antarctica between the years of 2008 and 2013. The data was exceptional in that it did not contain any missing values. The collection times were bimodal in that they occurred only during the first and last few months of the year, during the winter season; and most data came from the middle years, excluding 2008 and 2013. Due to the data's nature as a time series, the most logical type of network to use in this case would have been either a recurrent or convolutional neural network; a recurrent neural network, specifically a Gated Recurrent Unit network, was ultimately chosen for its relative speed and accuracy compared to the other options. After testing both Tensorflow and PyTorch for this purpose, PyTorch was chosen to build the final model, again due to its speed. After a few failed attempts at model creation, which did not function due to either data mismanagement or memory demand, a final model was created that can predict ice elevations based on location and time.

1: Introduction

The melting of the icecaps in Antarctica is a common topic amongst discussions of rising global temperatures, ocean levels and the planet's climate health. There are often discussions about the rate at which ice is melting, but most sources agree that the ice is melting at an increased rate in recent years; more quickly, in fact, than has previously been predicted.

If it were possible to collect the data and map it over space and time, it would be possible to get a map of the Antarctic ice that shifted over time to show the exact pattern of melting ice; and based on this data, it might be possible to then predict where and when the next major ice melts would be seen.

Advancements in machine learning make it possible to take large datasets, such as observations gathered on the Antarctic datasheets spanning multiple years, and train models to learn from present patterns and observations to project these findings into the future.

2: Data Collection

The project data was collected as part of the IceBridge Riegl Laser Altimeter L2 Geolocated Surface Elevation Triplets V001 study, carried out by the National Aeronautics and Space Administration (NASA)[1], and downloaded from the National Snow and Ice Data Center[2]. The study collected information of the heights of ice and snow in Antarctica over a total of six years, during which it logged the ice elevation values with error up to a meter, of a vast array of locations across the Antarctic contintent. These locations were logged with regards to time, longitude and latitude, and were gathered between December of 2008 and December of 2013. The data collection was undergone exclusively during winter months, to inspect yearly changes in the ice sheets.

The data files themselves are divided initially by year and further by day of collection and then the names of the aircraft that performed the observation, although this last piece of information wasn't deemed useful for the model.

Further information was searched for in the form of temperature data for the corresponding areas of the Antarctic that were surveyed, in order to further expand on the model. However, any data collected had to fit the physical and temporal range of the data set. Unfortunately, all datasets that were available through searches were either stored in inaccessible data formats, fell outside of the temporal window of the project's data, or had physical grains too large to be useful – for example, median temperatures of the entire continent by year. As such, this data was disregarded and the model was build to focus on only the locations of ice elevation and the time of data collection.

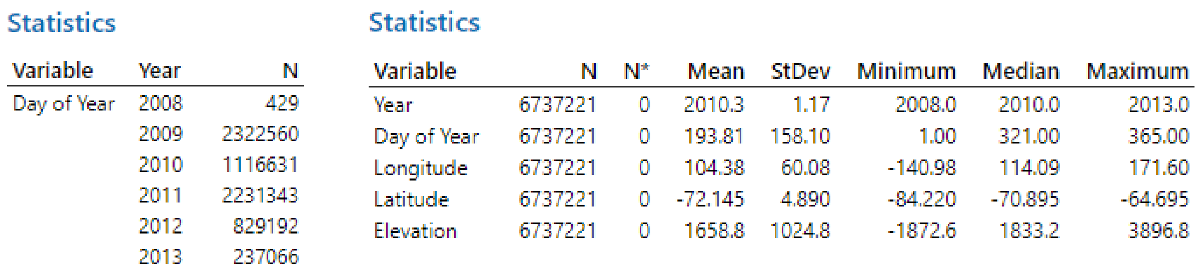
3: Data Profiling

The IceBridge study of the Antarctic ice sheets was carried out over a total of6 years, during which it logged the ice elevation values to the accuracy of within a meter, of a vast array of locations across the Antarctic contintent. These locations were logged with regards to time, longitude and latitude, and were gathered between December of 2008 and December of 2013. The data collection was undergone exclusively during winter months, to inspect yearly changes in the ice sheets.

For the study at hand, the data is used to inspect how ice elevations change with respect to time and location, to possibly identify areas that are at risk and project future ice elevations. The total number of variables in the downloaded dataset included the year, day of the year, second of the day, longitude and latitude of the observation, and then finally the observation of the height of the ice. The model did not include the variable 'Second of Day', as it did not intend to factor time of day into its ultimate analysis; therefore, the finest grain in the data is the Day of Year that the data was collected.

The data of the project was divided into a total of 726 different .txt files, organized by year, day of collection and name of the aircraft. The data was first read into Jupyter Notebook running Python, where it was consolidated into a single DataFrame object over the course of two and a half weeks. Due to the amount of time required to read all of the data, early analysis was performed on only a small sample of the data that was read in within a day; later analysis went back to repeat the analysis done on the sample data. Preliminary analysis of the total dataset, such as inspecting the shape of it, were performed on this DataFrame. The frame was then exported to a single .csv file, “iceData.csv,” and Minitab was used to perform more complicated data analysis.

The dataset had a total of 6,737,221 observations, with a small amount coming from the first year of 2008 and the vast majority coming from the following years, as can be observed in Figure 1.



Statistics

Variable	Year	N
Day of Year	2008	429
	2009	2322560
	2010	1116631
	2011	2231343
	2012	829192
	2013	237066

Statistics

Variable	N	N*	Mean	StDev	Minimum	Median	Maximum
Year	6737221	0	2010.3	1.17	2008.0	2010.0	2013.0
Day of Year	6737221	0	193.81	158.10	1.00	321.00	365.00
Longitude	6737221	0	104.38	60.08	-140.98	114.09	171.60
Latitude	6737221	0	-72.145	4.890	-84.220	-70.895	-64.695
Elevation	6737221	0	1658.8	1024.8	-1872.6	1833.2	3896.8

Figure 1. Count of data entries by year (left) and descriptive statistics for relevant data variables (right). N* in the descriptive statistics demarks the count of missing data observations.

One exceptional thing to note regarding the dataset is the lack of missing values, shown in Figure 1 in the column labelled N*. This is most likely due to the data collection being performed automatically by remote aircraft, and not being collected by human hands. This is one factor that will simply data analysis in the future.

The variable used to most broadly split up the observation is the Year of Collection. This is the variable most useful for tracking change over a long period of time. Much of the analysis of the other variables will be divided into their respective years of collection.

As seen in Figure 1, the vast majority of the data comes from years 2009 through 2013, excluding the year 2008 due to its single day of collection. The same is seen to a lesser extent in the year of 2013, as it ceases data collection earlier than the other years (in January as opposed to December.) Figure 2 shows that the years of 2009 and 2011 present the most data observations.

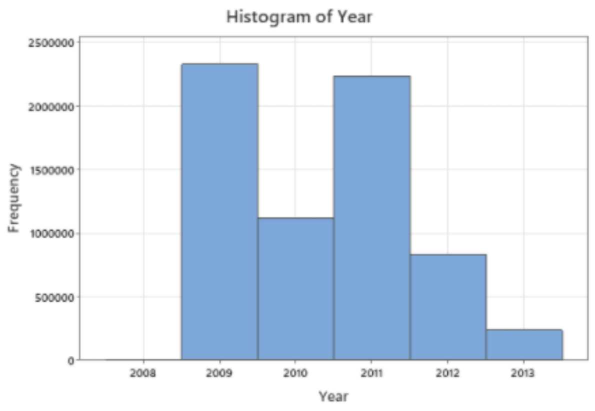


Figure 2. Distribution of the year of collection in the dataset.

The data is further divided into the Day of the Year of its collection (DOY). The DOY tracks the day of the year that each observation was documented. This method of date tracking was picked over the use of Month of the Year presumably for two reasons – first that the month of the year can be deduced from the year and day information, and secondly that the date would provide a finer grain for the data observations. However, as the original authors of the paper have declined to comment, this is strictly speculation.

By only inspecting the maximum and minimum DOY values, it might be mistakenly assumed that the DOY is distributed uniformly, but Figure 3 makes it clear that this assumption isn't true; the data is extremely bimodal. By inspecting the data and cross-referencing the DOY values with a Day of Year chart that includes which months contain which days [3], it can be seen that the observations of the ice elevation last from January into early February, stop entirely, and then start again in the months of November and December. This was most likely done to prevent the yearly temperature cycle from introducing variance to the elevation data; it would be expected, after all, that ice would melt more in the summer months and build up in the winter months. By only measuring the ice elevations during winter, the study eliminated the varying conditions due to warmer weather.

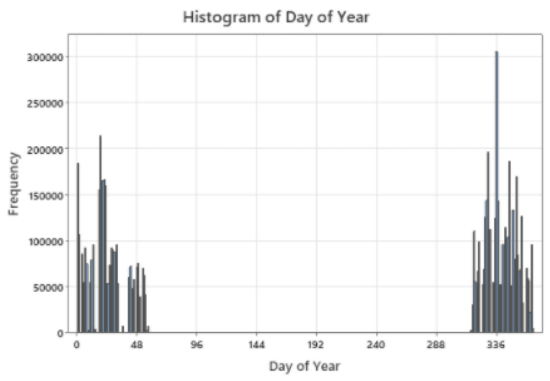


Figure 3. Distribution of the day of the year of collection

Although latitude and longitude metrics are recorded as separate variables, they are used together to denote locations on the globe and represent very similar information; therefore, they will be analyzed together. Although these values are technically numerical, it makes more sense to think of them as very finely divided categories; as such, studying their descriptive statistics is not entirely useful. Instead, they can be best represented by plotting them on a scatterplot against one another, the way one would plot points on a map. Figure 4 illustrates the locations of data collection.

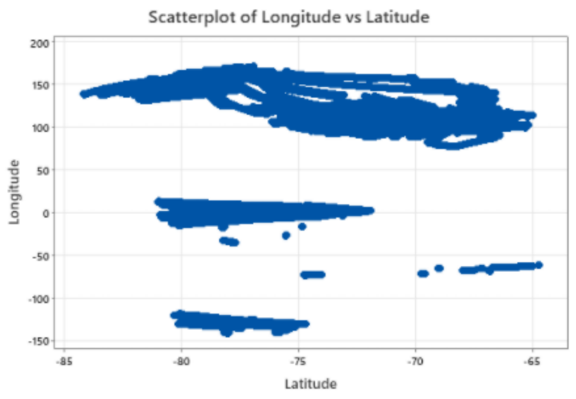


Figure 4. Latitudinal and longitudinal values of the dataset.

For a more accurate visualization of what the longitude and latitude values look like as plotted on a map, the Python library Basemap was used to plot the longitude and latitude values based on a particular map projection that centered the Antarctic. The results are shown in Figure 5.

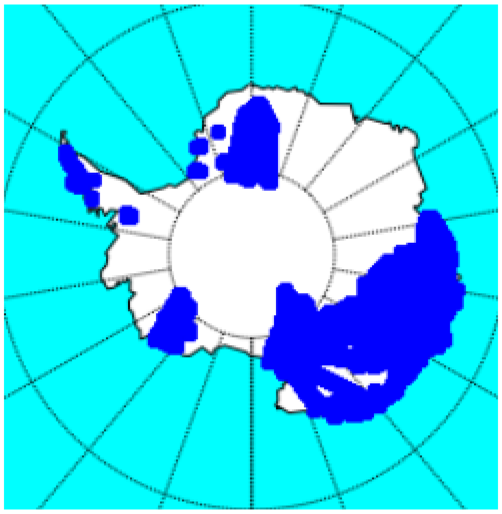


Figure 5. Projection of the longitude and latitude values of the dataset, plotted with the Python library Basemap.

Elevation is the target variable of the future model and study, and therefore will be subject to the most inspection. The distribution of the Elevation values can be inspected in Figure 5. The sample of data that was initially used for data profiling displayed a sample roughly Gaussian in shape, but on further examination of the entire dataset, it's clear that the Elevation is largely under a uniform distribution, with a much higher than usual count of values around 0.

Cross-referencing these unusual values with the observation paths shown in Figure 7, it seems likely that the high number of elevation measurements of near or at 0 meters come from the observations actually not occurring over ice floes, but over the ocean surface instead. Whether this indicates that the ice melt is occurring more quickly than expected and there should have been ice present over that patch of ocean, or if the aircraft was designed to capture data over the ocean as well as the ice, is unclear.

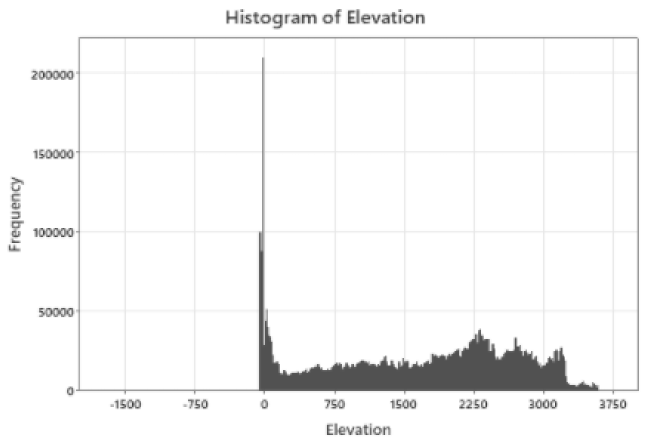


Figure 6. Distribution of the ice elevation values of the dataset.

The initial data profiling inspected the minimum, maximum and center values of the ice elevations separated by year; however, this did not make it clear if the years were meaningfully decreasing or were skewed. As such, more gainful inspection of the ice elevation can by inspecting each year's ice elevation values as histograms. This is illustrated in Figure 6.

By inspecting Figure 6, some earlier observations are reinforced; particularly that very little data is coming from the years 2008 and 2013. It can also be seen that a very large amount of the data that was collected over open ocean comes from the year 2012, which is very interesting. Between the years of 2010 to 2012 there appears to be a trend of overall ice elevations dropping towards 0, with much more observations over the ocean each subsequent year until 2013. This might suggest a trend of ice melting over the years, although with 2013 having collected much less data in comparison, it's hard to tell if this trend holds.

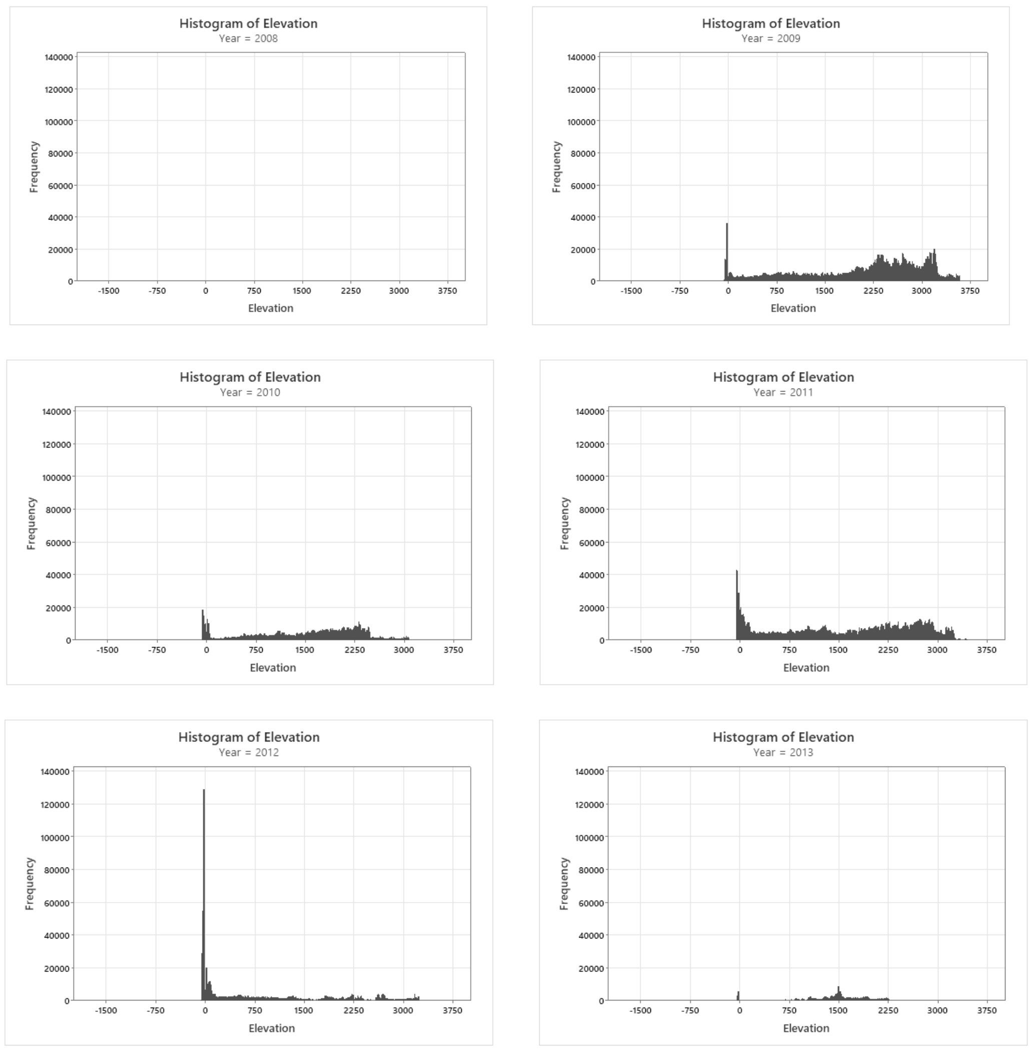


Figure 7. Distributions of ice elevations of the dataset, separated by year.

4: Technical Research

There are many types of neural network architecture that could have been used for the project, from 'normal' artificial neural networks (ANNs) to more complex models such as LSTM and convolutional models. In order for the project to work as well as possible, a good model had to be fit to the collected data.

As discussed above, the dataset is formatted so that individual observations occur at different days, or different "time steps," and each log a series of different variables from that "time step." This means that the dataset is categorized as a type of dataset called a time series, with days as its smallest grain. Time series prediction, the core of the problem, is most often tackled by two overarching types of neural networks – these being recurrent neural networks (RNNs) and convolutional neural networks (CNNs). Included in these two categories are a wide array of possible architectures available.

Of the RNN structures, there are three main architectures available – the standard recurrent neural network, the Long Short-Term Memory RNN (LSTM), and the Gated Recurrent Unit RNN (GRU). These last two neural networks feature 'memory cells' as their main neuron function; unlike a basic neuron, with a single value and activation function, memory cells include layers of mathematical calculation. These functions, called 'gates', change how memory is stored in the cell, how much the memory affects the current value, and then calculates the value based on the memory. These structures are unique to RNN's because they incorporate the recurrent nature of the network into their design; however, some other networks do use architectures inspired by them.

A paper by Wei et. al. covered a comparison of these three types of RNN's as well as a standard ANN as applied to a time series prediction [4]. The paper compares the three types of RNNs and the ANN on the basis of each network's accuracy, measured as the coefficient of determination, or R^2, and the root mean square error or RMSE. The paper found that the RNN trained significantly quicker than the ANN; due to this factor, combined with the fact that the Antarctic dataset has a total of length of 6,737,221 distinct entries, it's clear that the RNN approach is better suited to this particular dataset.

Both the remaining RNN types were shown to be highly precise and robust against issues like vanishing gradient. Due to the two types of gated RNN's being nearly identical when compared via R^2 and RSME, comparison of the two proceeded to measures of efficiency. Here, due to its simpler structure, the GRU was shown to be preferrable: the average training time of the GRU was shown to be almost 30% faster than the average training time of the LSTM model. Once again, due to the considerable size of the Antarctic ice data set, training time is an important consideration. Therefore, the natural choice for implementation from the selection of RNN's is the GRU network.

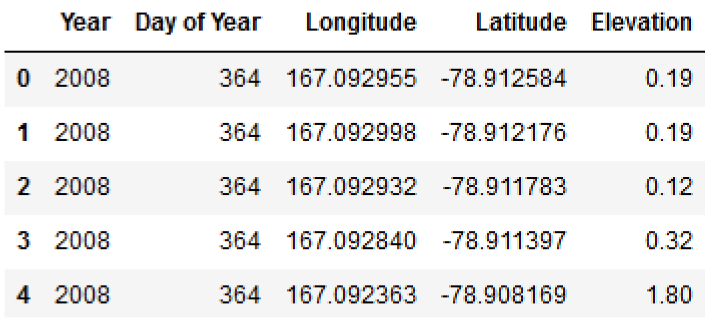
CNN networks, as mentioned above, were originally designed for use with image processing and computer vision, and as a result, the CNN neuron units function as matrices of values that can mimic the values of colour channels in digital images. The initial matrix is then passed through layers of filtering, pooling or subsampling to reduce the matrix back down to a one by one value, where it can be further processed by the network before reaching an outcome.

Another paper found that of the models tested on the stock data, the LSTM model scored consistently lower for overall return value than the paper's suggested model[5]. However, it did outperform all of the other models that the proposed model was tested against. Furthermore, although the performance of the converted CNN model may have been better in the paper's case, if it were to be converted for use with the Antarctic dataset, it would have to account for both the relatively small amount of variables in the dataset and the extreme length of it. The much smaller number of variables in the Antarctic dataset could pose a difficulty when trying to convert the values into meaningful images for processing. Finally, considering the dimensional nature of a convolutional dataset, the amount of parallel nodes required to process a dataset the length of the Antarctic data set borders on unreasonable, and likely would take a significantly longer time to compile than a GRU model on the currently available hardware. Therefore, although the CNN model is very interesting and further innovations will be fascinating to see, the GRU model is the most well-suited model of the options discussed.

5: Neural Network Creation Process

Methodology

The first necessary step to creating the neural network was to transform the data until it was ready to be worked on. Luckily, the first step of this was accomplished during the data profiling step – the data was already read into a single .csv, which made importing the data into a single DataFrame much easier and quicker than reading in the data from a variety of separate files. Reading the data from the single .csv took only about a minute, significantly less time compared to the time required to read in the data for the initial profiling. This form of the data can be observed in Figure 7.



	Year	Day of Year	Longitude	Latitude	Elevation
0	2008	364	167.092955	-78.912584	0.19
1	2008	364	167.092998	-78.912176	0.19
2	2008	364	167.092932	-78.911783	0.12
3	2008	364	167.092840	-78.911397	0.32
4	2008	364	167.092363	-78.908169	1.80

Figure 8. The first rows of the initial DataFrame that the data was collected in, before transformation.

With the data gathered, the transformation step could begin. Originally, a neural network was built immediately using the data collected here, only after performing scaling and splitting it into training and test sets. This neural network, however, was revealed to be junk due to the way that the dates of the original data were collected.

As can be observed in Figure 7, the date of each observation is split into 'Year' and 'Day of Year.' These two columns create a series of unique dates when taken together; however, the network isn't necessarily capable of understand that. What then results is that, because of the cyclical nature of the 'Day of Year' variable, the datapoints that are meant to fall along a line end up piling on top of each other, leading to the model outputting junk, as can be seen in Figure 8.

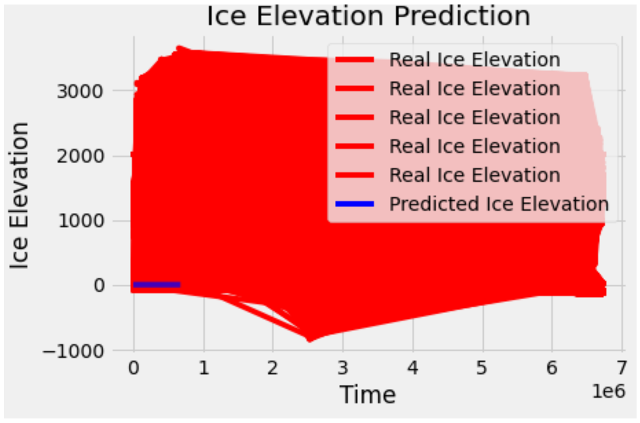
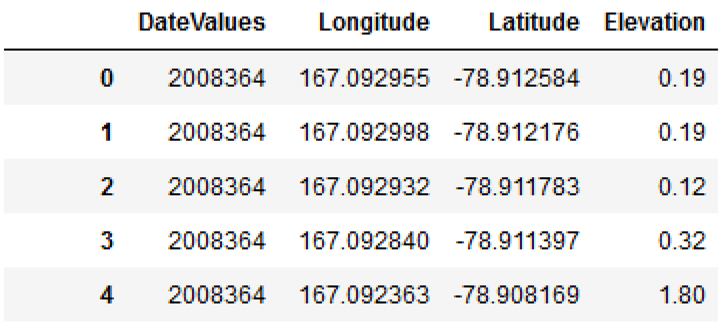


Figure 9. Failed outputs of a Tensorflow neural network due to badly formatted data.

After observing this, it was resolved to change the format of the dates in the dataframe into unique observations. Naturally, due to several observations being made on singular days across the years, the observations are not truly 'unique,' but the distinction between the same days of different years should be made for the benefit of the model and readability of the output.

The two columns, 'Year' and 'Day of Year,' were iterated through and had their data changed from an integer format into a string. The values of each row were then appended to each other (specifically, 'Day of Year' was appended to the back of the 'Year'); then the values were converted back into integers so that the numerically based model would be able to understand them, and sorted into their own Dataframe column, dubbed "DateValues." This created a column of unique date values, where the first four digits correspond to the year and the last three to the day. Once this new column was created, the columns of 'Year' and 'Day of Year' were removed in order to prevent the neural network from accounting for these data values more than once per observation. The final form of the data frame columns can be seen in Figure 9.



	DateValues	Longitude	Latitude	Elevation
0	2008364	167.092955	-78.912584	0.19
1	2008364	167.092998	-78.912176	0.19
2	2008364	167.092932	-78.911783	0.12
3	2008364	167.092840	-78.911397	0.32
4	2008364	167.092363	-78.908169	1.80

Figure 10. The first rows of the final data frame column that was ultimately fed into the neural network.

Next in the process of creating the neural network was to become familiar with the required software. For this project, the Python libraries of PyTorch and Tensorflow were both tested in creating a neural network. Although both were promising in their implementations of neural networks, ultimately one of them had to be chosen due to the incompatibility of PyTorch and Tensorflow tensors; this would cause the data, once properly formatted for one model, to be unable to transfer to the other. PyTorch was ultimately chosen, as the first models testing with Tensorflow showed themselves to be significantly slower in performance than the PyTorch models.

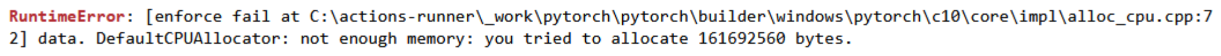
A few different models were tested before a single model was settled on.

The first model tested was a Tensorflow model, the results of which are displayed in Figure 8 and prompted the cleaning of the data. Once the data was cleaned and it was resolved to work only with PyTorch assets, two other models were tested out.

Results & Discussion

Tutorials outlining the process of building models in PyTorch were helpful in the development of the models tested for this project[6][7][8]. The first PyTorch model was built by defining custom classes in the Jupyter Notebook, specifically a class GRUNet that accepted a PyTorch model and defined some methods for forward passing through the network and creating the hidden layers. Two other methods, train and evaluate, were created to complement it. The model itself appeared to work acceptably and was able to train on the data.

However, the issue with this version of the model came into sharp relief when it came time to evaluate the model. The evaluation method for this model called <<TALK ABOUT THE METHODS>>, which required an extreme amount of computer memory. It was assumed that this was due to the extreme size of the dataset, and a very small subset of the data, specifically the first 200 rows, were selected for testing instead. However, the issue with memory persisted – it was simply demanding more memory than was available on the current machinery, as can be seen in Figure 10. Because of this, this model was discarded, and the final model was ultimately chosen.



```
RuntimeError: [enforce fail at C:\actions-runner\_work\pytorch\pytorch\builder\windows\pytorch\c10\core\impl\alloc_cpu.cpp:7
2] data. DefaultCPUAllocator: not enough memory: you tried to allocate 161692560 bytes.
```

Figure 11. Failure of the first PyTorch model to evaluate its outputs, due to extreme memory demand.

The final model was built using inbuilt methods offered by PyTorch. The model was defined as a PyTorch.Sequential() model, meaning it would be a model specialized for sequential data such as time series. The model architecture was built with the model.add() methods using GRU nodes. The nodes specified a tanh activation function and sigmoid activation for recurrent connections within the node; this is the common approach to building a GRU node. The model consisted of 15 GRU nodes, a dropout layer to optimize performance and minimize overfitting, and a single output layer with a single node; since the problem is a regression task, there is no need to define more than a single output node. The model evaluated itself on the backward pass using mean squared error as loss, along with the popular Adam optimizer.

The evaluation of the second model was able to complete on a small sample of data without causing issue, but could not handle evaluation of the entire dataset at once. As a result, the data was split into two distinct datasets, and the same model was run on both in order to gather matching outputs. Once the data was split in half, each distinct half was prepared separately.

Both data halves were split into training and testing sets. These training sets contained a set number of timesteps that it would be able to recall for the recurrent network function, and a set number of steps in the future that it would be able to feed data into. For this model, both of these values were set to 10. The data was sorted into training and set sets, on a 90% split.

Here, the x axis represents the number of timesteps completed by the network, which work as an effective proxy for time as they order the data in sequential order; and the y axis represents the ice elevation in meters.

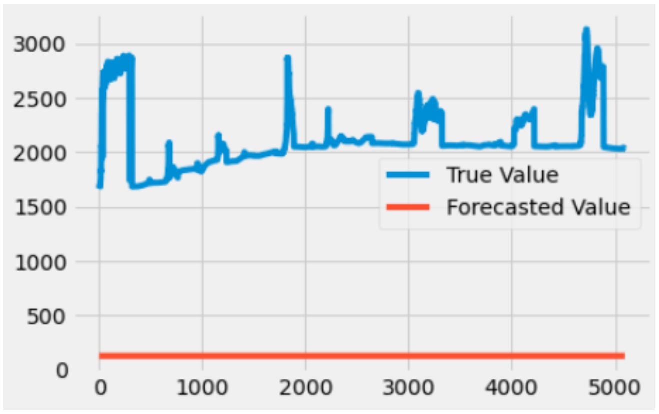


Figure 12. Preliminary outputs of true and forecasted ice elevations of the first half of the dataset.

As can be seen in Figure 11, the forecasted values don't appear to be fitted to the pattern of the data at all. This is due to a mistake during the data processing. During the preliminary run of the model, the section of data that was testing the performance of the model had not been scaled properly; as such, due to the very large values of the ice elevation being handled, the model encountered a problem known as exploding gradients, in which the difference between the predicted values and the true values in the network get too big too quickly; because gradients are calculated based on other gradients, if even one is larger than expected, the problem can spiral out of control. This problem led to the model being unable to effectively train, and as such, the forecasted values don't follow the pattern of the data.

The data halves were scaled so that all values were between 0 and 1 in magnitude; this would help the model train more efficiently, while also avoided the common problem of 'exploding gradients' when training neural networks, in which gradients get far too large. Two scalers were required to scale the first and second halves of the dataset independently, and then a third scaler was defined for the purposes of restoring the outputs back to their true values once the model was finished running. After properly scaling the data and ensuring that all data was in the proper form, the model was run once again; on the first and second halves of the data independently, as before. The final, rescaled outputs of the model, with the data correctly scaled, can be seen in Figure 12.

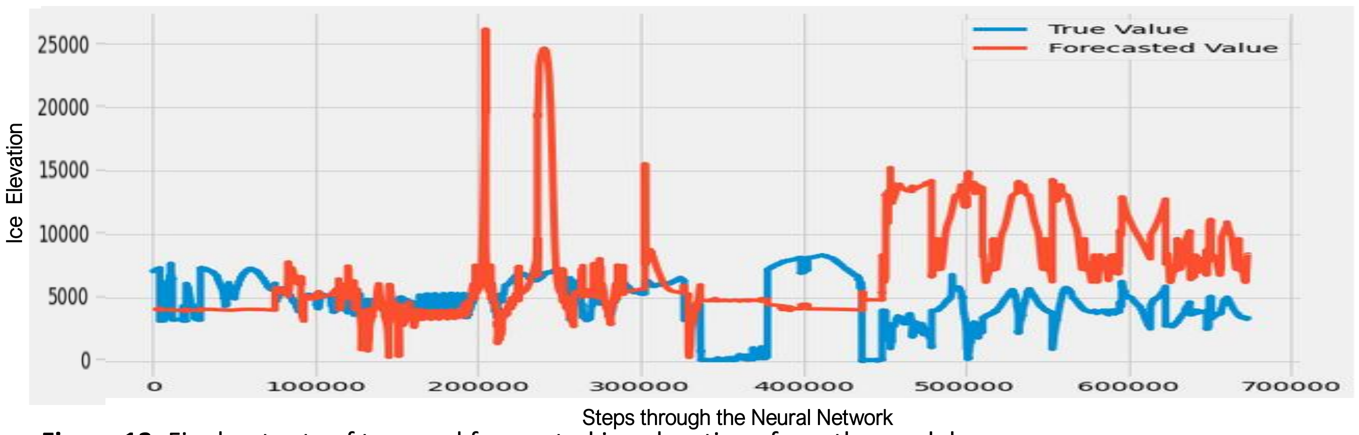


Figure 13. Final outputs of true and forecasted ice elevations from the model.

Something that becomes very apparent when inspecting the predicted values of ice elevation and how they contrast with the true values of the ice elevation, particularly in the second half of the dataset, is that the true values of the ice elevation are consistently lower than the forecasted values – despite the neural network correctly picking up on the pattern of rises, dips and plateus, the network forecasts the second half of the dataset as being a consistently higher than the actual values of the ice.

This could be indicative of a handful of things. This could be a failure of the model to accurately learn the trends of the ice elevation and how they appear to be decreasing; as can be seen in the output, the true values in the second half of the dataset are generally lower than those in the first half of the dataset. This may also explain some of the spikes in the predicted values of the ice elevations in the first half of the dataset, which don't appear to have a clear cause. However, this could also indicate something else – that the ice melt is accelerating through the years. The issue may be resolved by inspecting the error.

The most common way to measure error in a neural network is loss because the loss is factored into the ongoing calculations that allow a neural network to train itself. By inspecting the loss rates throughout the training process, some insight can be gained for how accurately the model performed. Here, the x axis represents epochs through the neural network, as opposed to the number of steps, because loss is only calculated at the end of each epoch.

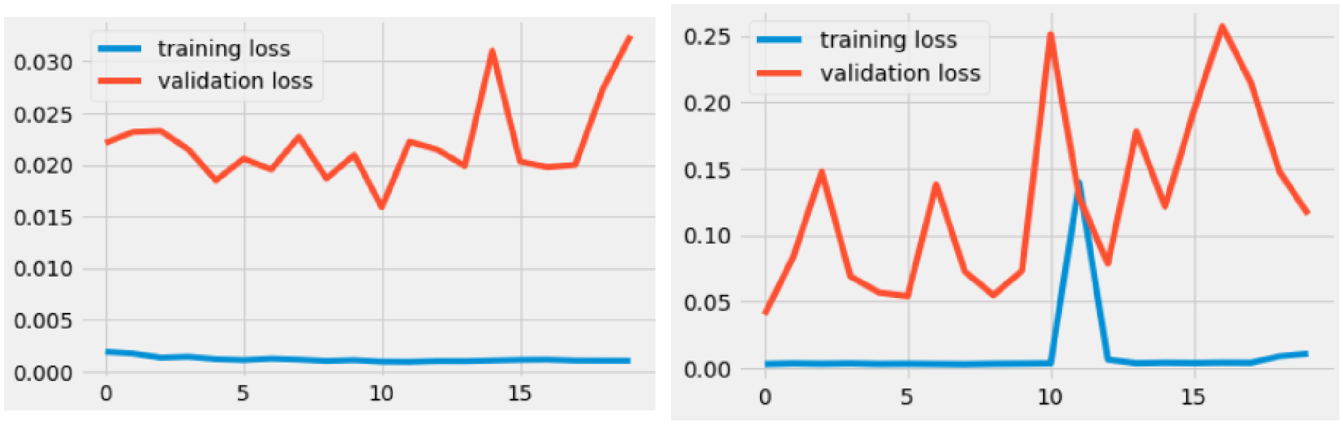


Figure 14. Loss calculations during the training and testing of the first half of the dataset in the model (left), and the second half (right).

By checking the location of the spike in training loss in Figure 13 against the values in Figure 12, we can see that this most likely correlates to the unexpected plateau present around 400,000. What might be happening, in that case, is that the neural network recognized that the values it was generating for that piece of the network were lower than expected and compensated by increasing all of its expected outputs. However, if that were the case, it would be expected for the network to then realize that its prediction values were higher than expected and lower them all to compensate. Why the network did not do this is curious. Unfortunately, this question isn't something that can be answered easily due to the black box nature of neural networks as a whole. This curiosity could be due to ice elevations lowering with time, or it could be due to a fluke in the network.

The model is not incredibly precise in its prediction of ice elevations, but it does seem capable of predicting the pattern of height increases and decreases displayed in the dataset. The lack of precision is likely due to the fact that other factors heavily influence the height of ice around Antarctica – such as ambient temperatures, precipitation and other weather factors not included in this model. If they were included, it is possible that the model would show an increase in accuracy.

6: Conclusions and Future Works

In conclusion, the final form of the model was able to train itself on the Antarctic ice data provided to it and provide estimates of future ice elevations, based on the year, day of the year, and location of the data in question. The provided data and the model's predictions were then able to be mapped onto a projection of Antarctica to show where the ice elevation predictions would fall on the map.

While functional, the model lacks some accuracy, and the loss tends to fluctuate. Future work on the subject could be dedicated to decreasing the error of the model, in order to provide better predictions of ice elevation in the future. This would likely involve including factors that were not included in the base IceBridge dataset, such as temperatures of the Antarctic by year and location, and weather conditions by the same variables. Projection of the model farther into the future, beyond the bounds of the dataset, would also be very interesting. One goal of the original project that had to be postponed was to project the outputs onto a map of Antarctica colour coded in accordance with the height of ice elevation predicted by the model; this expansion on the project would surely be an interesting addition. Other expansions on the project could include testing other forms of machine learning, to see if other models may be able to better encapsulate the patterns being shown in the data.

References

1. Donald Blankenship. (2013). IceBridge Riegle Laser Altimeter L2 Geolocated Surface Elevation Triplets V001. NASA NSIDC DAAC. Accessed February 2023 at: https://catalog.data.gov/dataset/icebridge-riegl-laser-altimeter-l2-geolocated-surface-elevation-triplets-v001

2. National Snow & Ice Data Center (2013). Data Download. Accessed 2023 at: https://n5eil01u.ecs.nsidc.org/ICEBRIDGE/ILUTP2.001/

3. NASA Scatterometer Climate Record Pathfinder, European Space Agency (2021). Day of Year Chart. Accessed 2023 at: https://www.scp.byu.edu/docs/doychart.html

4. Wei, X., Zhang, L., Yang, H-Q., Zhang, L., Yao, Y-P. (2021) Machine learning for pore-water pressure time-series prediction: Application of recurrent neural networks. *Geoscience Frontiers, 12(1),* 453-467. Accessed March 2023 from https://www.sciencedirect.com/science/article/pii/S1674987120301134

5. Sezer, O. B., Ozbayoglu, A. M. (2018) Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing, 70,* 525-538. Accessed March 2023 from https://www-sciencedirect-com.ezproxy.rit.edu/science/article/pii/S1568494618302151?via%3Dihub

6. Pao, Spencer (2022) Demo of Gated Recurrent Unit. Accessed April 2023 from https://github.com/SpencerPao/Data_Science/blob/main/GRU/GRU.ipynb

7. Loye, Gabriel (2019) Gated Recurrent Unit (GRU) with PyTorch. *LoydHub.* Accessed April 2023 from https://blog.floydhub.com/gru-with-pytorch/

8. Yadav, Siddharth (2019) Intro to Recurrent Neural Networks LSTM | GRU. *Kaggle.* Accessed April 2023 from https://www.kaggle.com/code/thebrownviking20/intro-to-recurrent-neural-networks-lstm-gru

9. Owari, Y., Miyatake, N. (2019) Prediction of Chronic Lower Back Pain Using the Hierarchichal Neural Network: Comparison with Logistic Regression – A Pilot Study. *Medicina (Kaunas), 55(6),* 259. Accessed March 2023 from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6630563/

10. Long, L., Zeng, X. (2022) Beginning Deep Learning with TensorFlow, Work with Keras, MNIST Data Sets, and Advanced Neural Networks. *Apress.* Accessed March 2023 from https://link-springer-com.ezproxy.rit.edu/book/10.1007/978-1-4842-7915-1

11. Pang, Z., Niu, F., O'Niell, Z. (2020) Solar radiation prediction using recurrent neural network and artificial neural network: A case study with comparisons. *Renewable Energy, 156,* 279-289. Accessed March 2023 from https://www-sciencedirect-com.ezproxy.rit.edu/science/article/pii/S0960148120305747?via%3Dihub

12. Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986) Learning representations by back-propagating errors. *Nature, 323,* 533-536. Accessed March 2023 from https://www.nature.com/articles/323533a0

13. Hewamalage, H., Bergmier, C., Bandara, K. (2021) Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *International Journal of Forecasting, 37(1),* 388-427. Accessed March 2023 from https://www-sciencedirect-com.ezproxy.rit.edu/science/article/pii/S0169207020300996?via%3Dihub

14. Cun, Y. L., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D. (1992) Handwritten Digit Recognition with a Back-Propagation Network. *Neural Networks, current applications.* Chapman Hall/CRC Publishers. Accessed March 2023 from https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf

15. Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., Inman, D. (2021) 1D convolutiona neural networks and applications: A survey. *Mechanical Systems and Signal Processing, 151,* 107398. Accessed March 2023 from https://www-sciencedirect-com.ezproxy.rit.edu/science/article/pii/S0888327020307846?via%3Dihub

16. Sezer, O. B., Ozbayoglu, A. M. (2018) Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing, 70,* 525-538. Accessed March 2023 from https://www-sciencedirect-com.ezproxy.rit.edu/science/article/pii/S1568494618302151?via%3Dihub

17. Donald Blankenship. (2013). IceBridge Riegle Laser Altimeter L2 Geolocated Surface Elevation Triplets V001. NASA NSIDC DAAC. Accessed February 2023 at: https://catalog.data.gov/dataset/icebridge-riegllaser-altimeter-l2-geolocated-surface-elevation-triplets-v001