SUBSTORM PREDICTION WITH DEEP NEURAL NETWORKS

Greg Starr
Department of Electrical Engineering
Boston University
Boston, MA 02215
`gstarr@bu.edu`

Alan Zhou
Department of Computer Science
Tufts University
Medford, MA 02155
`alan.zhou@tufts.edu`

May 5, 2019

ABSTRACT

Auroral substorms are not fully understood and there is no standard method for predicting their occurrence. In this paper we use convolutional neural networks (CNN) and recurrent neural networks (RNN) to predict the occurrance of substorms based on the ground based magnetic field measurements provided by SuperMAG collaborators. Our results are compared against a logistic regression classifier to establish a concrete baseline for substorm prediction. We analyze the merits and drawbacks of both architecture styles and consider future directions. `https://github.com/alanzhou93/substorm-detection`

***K*eywords** Substorm · Convolutional Neural Network · Recurrent Neural Network

1 Introduction

Auroral substorms are natural phenomena whereby energy is released from the Earth's magnetospheric tail into the ionosphere. During a substorm, a large amount of ions and electrons flow into the ionosphere. This is essentially an electric current which runs through the magnetotail along field lines. This flow of electrons and ions causes an intense and widespread aurora, visible from the ground and from space. Substorms occur near midnight in polar regions of the earth.
Before a substorm, the magnetotail is stretched and builds up a large amount of energy. The mechanism by which substorms are triggered is largely unknown, but when they are, the magnetotail snaps back towards the earth and releases its energy into the ionosphere. The current associated with a substorm flows into the ionosphere in the post midnight region, to the west and returns to the magnetotail along field lines in the pre midnight region. From the Biot-Savart law, this causes a large southward deviation in the magnetic field as measured on the ground. They are determined and measured both by their affect on ground based magnetic field measurements and by the bright auroras they cause. Substorms can release a large amount of enery and therefor can cause damage or disruption to telecommunication, navigation, and spacecraft. Substorms are frequent, occurring multiple times on any given day [1].

Because the mechanism by which substorms occur is not fully understood, it is difficult to accurately predict their occurrence. This presents a perfect opportunity to apply deep learning techniques to attempt to solve this problem.

2 Related Work

Substorms are an active area of research in the space physics community. Many studies have been published to help explain their effects, their causes and their statistics. Several studies offer empirical descriptions of the current system during a substorm [2, 3]. Originally, it was thought that there is a single current "wedge" during a substorm. It runs along magnetic field lines into the ionosphere on "post midnight" side, then turns west and returns along field lines in on the "pre midnight" side. The SuperMAG dataset has allowed more detailed descriptions of the large scale current systems during substorms.

Another study examines several different parameters describing the ionosphere and solar wind, both raw measurements and derived quantities, to see how well they describe substorm probabilities [4]. They found that solar wind speed is the single best predictor of substorm probability. We made use of this discovery and decided to include solar wind parameters as an input to our model.

To our knowledge, there does not exist an established baseline for substorm prediction. With our results we hope to establish such a baseline.

3 Problem Definition

We wish to predict two things: the occurrence of substorms as well as their strength. We frame the substorm occurrence prediction problem as binary classification. The positive class is when a substorm occurs in the interval $[t, t + T_p]$ and the negative class is when no substorm occurs in that interval. We take a set of magnetometer data from S stations in the interval $[t - T_m, t]$. Each station measures the three components of the magnetic field at its location, rotated into a local magnetic coordinate system. These measurements have a baseline removed in order to isolate ionosphere-magnetosphere interactions [5, 6]. For each example, the magnetometer data is a tensor $x_m \in \mathbb{R}^{S \times T_m \times 3}$. We supplement the magnetic field input with two solar wind parameters, B field and solar wind speed, also provided by SuperMAG. Each of these parameters is measured every minute and has three components. This results in a solar wind tensor $x_w \in \mathbb{R}^{T_w \times 6}$. The solar wind parameters are measured by a satellite between the earth and the sun and so there is just one measurement per minute for the whole earth.

The two quantities we are trying to predict are the substorm occurrence $y_L \in \{0, 1\}$ and a proxy for substorm strength $y_E \in \mathbb{R}$ called "SuperMAG Electrojet Index" (SME). Because substorms last more than one minute, we take the largest value of SME over the 20 minute interval starting at the substorm onset as the measure of the substorm's strength. Formally, we want to approximate the following function

$$f : (x_m^i, x_w^i) \mapsto (y_L^i, y_E^i) \tag{1}$$

where x_m^i is the i-th instance of magnetometer data, x_w^i is the the i-th instance of solar wind data, y_L^i is the label of the i-th sequence, y_E^i is the strength of the magnetic storm.

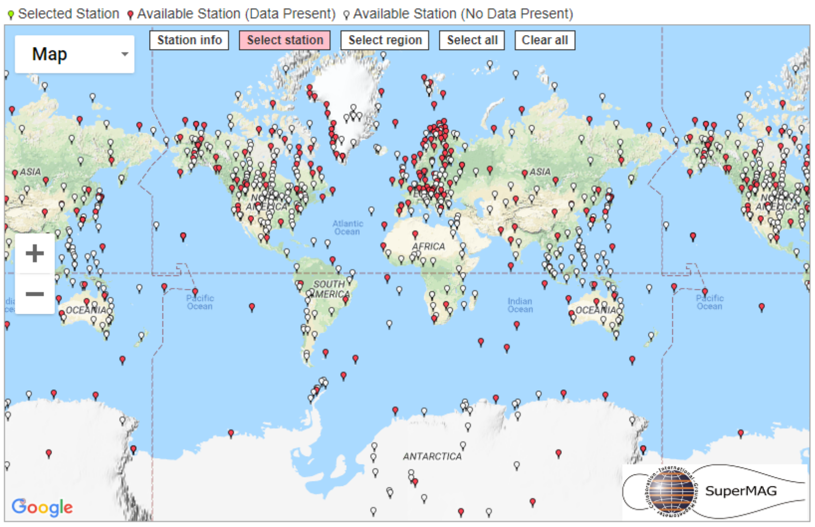


Figure 1: SuperMAG stations

3.1 Dataset

SuperMAG is a collaboration between many different labs and research groups studying the ionosphere and magnetosphere. They collect all of their data into a central repository and make it available to the public [5, 6]. The SuperMAG dataset contains one-minute-resolution magnetic field measurements from over 300 stations around the

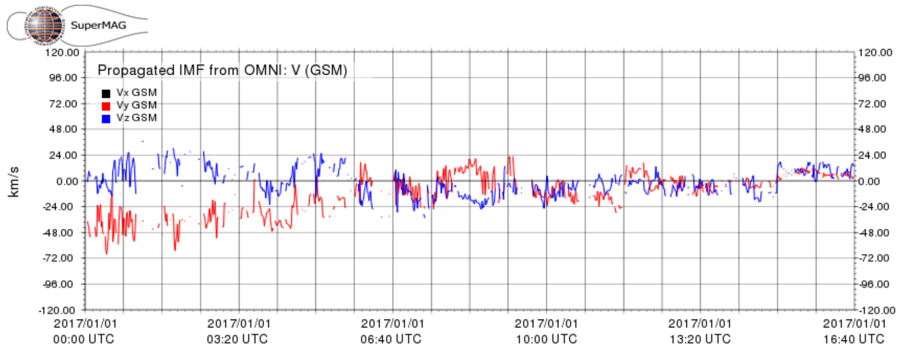


Figure 2: Example of solar wind data. It is patchier than the magnetometer data.

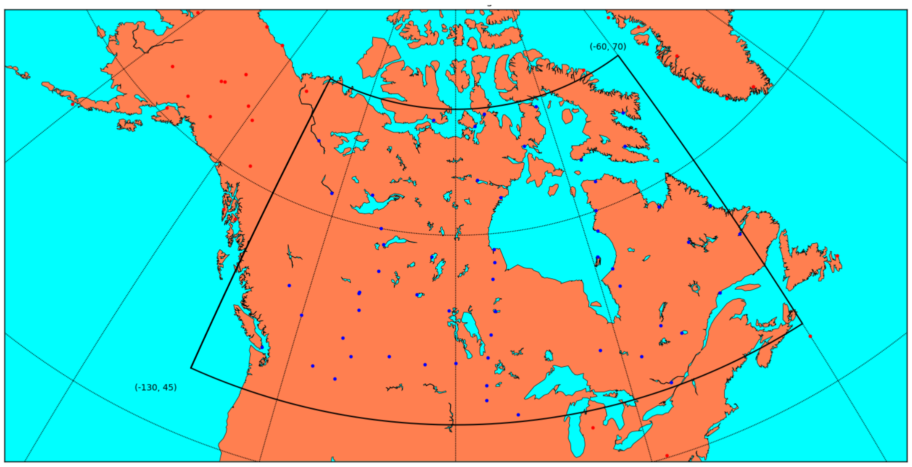


Figure 3: Region which we restricted our problem to. Blue dots are stations included in our dataset, red dots are rejected stations.

globe. The locations of the stations are pictured in figure 1 Data exists as far back as 1975, but the further back in time one looks, the fewer stations there are. Most time intervals contain data from only a subset of the full network of stations. The measurements from each station are rotated into a local coordinate system who's axes point north, east and down respectively (see `http://supermag.jhuapl.edu/mag/?tab=description`). The data can optionally have a baseline removed in order to have seasonal trends removed and to isolate ionosphere-magnetosphere interactions [5]. For use in our neural networks, we replace all missing data with zeros, but perform no additional preprocessing on top of the baseline removal.

The SuperMAG collaborators also provide a list of substorm occurrences including both their time and location. These substorms are determined by a sliding-window algorithm operating on the SME time series (`http://supermag.jhuapl.edu/substorms/?tab=description`). Finally, SuperMAG makes available solar wind data as extracted from NASA/GSFC's OMNI data set through OMNIWeb. Solar wind velocity is known to be a good predictor of substorms [4]. Unfortunately, the solar wind data is particularly patchy, with many intervals missing values. An example can be found in figure 2. We preprocess the solar wind data by performing linear interpolation to fill in missing values. A full description of SME is available at `http://supermag.jhuapl.edu/indices/?tab=description&layers=IMF.BGSM%2CIMF.VGSM`. We remove values we believe to be in error (> 10000) and replace missing entries with zero. The SuperMAG website wouldn't let us download all of the magnetometer data at once (>100GB) so we wrote a python script to request the data in smaller sections.

Large portions of the earth, namely the oceans, don't have any magnetometers. To deal with this problem, we chose to limit our problem to predicting substorms in a particular region of the earth with latitudes in [45, 70] and longitudes in [-130, -60]. This region corresponds to a portion of the US and Canada containing 81 stations. It is pictured in figure 3.

4 Methods

Because there is no known physics-based solution to equation (1), our approach is to use deep neural networks trained on the SuperMAG dataset to approximate this function. In this work, we experiment with both convolutional neural networks (CNN) and recurrent neural networks (RNN). This is because both of these architectures are designed to handle time series data efficiently. We chose fairly simple architectures for both styles and so the key in gaining performance was ensuring that our hyperparameters are properly set, our data is properly formatted, and our data contains the predictive information that we believe it does. In order to make it easier for our networks to learn relationships between different stations, we wanted to order the stations in our input tensors so that stations nearby in the tensor are also nearby on the earth. For this we estimated the shortest path connecting all of the stations using simulated annealing and put the stations in the resulting order. We made use of the Keras deep learning python library in order to implement our networks.

Because there are few if any other attempts at this problem, using deep learning or otherwise, we had little to no intuition about what architecture would work the best. We decided the best approach to overcome this was to find architectures using a random hyperparameter search. Both architectures used a combination of crossentropy loss on the label output and mean squared error loss on the regression output and the weight between the losses was included in the tested hyperparameters.

4.1 CNN Architecture

CNNs were constructed by choosing a random number of stages. Each stage is made up of a random number of blocks, all with the same number of channels. Each stage doubles the number of filters in its convolutional layers. The last block of each stage performs downsampling by increasing the stride of the convolutional layer. One of the hyperparameters was whether to use a residual style CNN or a traditional CNN with no skip connections. For the traditional CNN, each block was made up of a convolutional layer, followed by batch normalization (BN) [7], followed by ReLU activation. For the residual style CNN the blocks were constructed as in [8], i.e. BN - ReLU - Conv - BN - ReLU - Conv with a connection between the input and the output. Downsampling and increasing the number of channels in the residual style CNN was accomplished by adding a convolutional layer along the skip connection path. By varying the number of stages and blocks per stage, the random search could try out different numbers of layers with different amounts of downsampling.

Other important hyperparameters included kernel sizes, whether or not to use the solar wind data, training batch size, amount of downsampling and length of input data (T_m and T_w). Global average pooling was used at the output of the final stage. Global average pooling is a good alternative to fully-connected layers because it uses no parameters whereas often times, the fully connected layers of a deep neural network account for the majority of its parameters.

Overall, we chose to have two separate sub-networks, one for the magnetometer data and one for the solar wind data. Due to the global average pooling, the output from each sub-network is a vector. These vectors were concatenated into one, and this was passed into two separate linear layers, one for classification and one for regression. The best architecture (highest classification accuracy on the test set) that resulted from our search is pictured in figure 4. This model contains 93,010 trainable parameters, and is far from the largest model tested.

The hyperparameter search trained each model for 15 epochs over the training set. We tried training a few of the best models for longer, but they usually performed worse. In the future, a larger search could be done where number of epochs is one of the parameters. We chose 15 because it seemed like a good balance; enough training to fit but not overfit while also not taking too long.

4.2 RNN Architecture

The RNN architecture also uses two seperate sub-networks for the magnetometer and solar wind inputs. Each time step is processed sequentially along with a "context" vector containing information from previous time steps. The overall output of the recurrent layers is the final context vector of the last recurrent layer. This vector is passed through a fully connected layer with sigmoid (softmax for multiple classes) activation. Dropout was employed in the RNN as a regularizer [9]. A hyperparameter search was also run for various RNN architectures. Some hyperparameters included batch size, size of the RNN context vector, number of stacked RNN cells, size of the fully connected layer, dropout rate as well as what type of RNN cell to use (GRU, vanilla, LSTM).

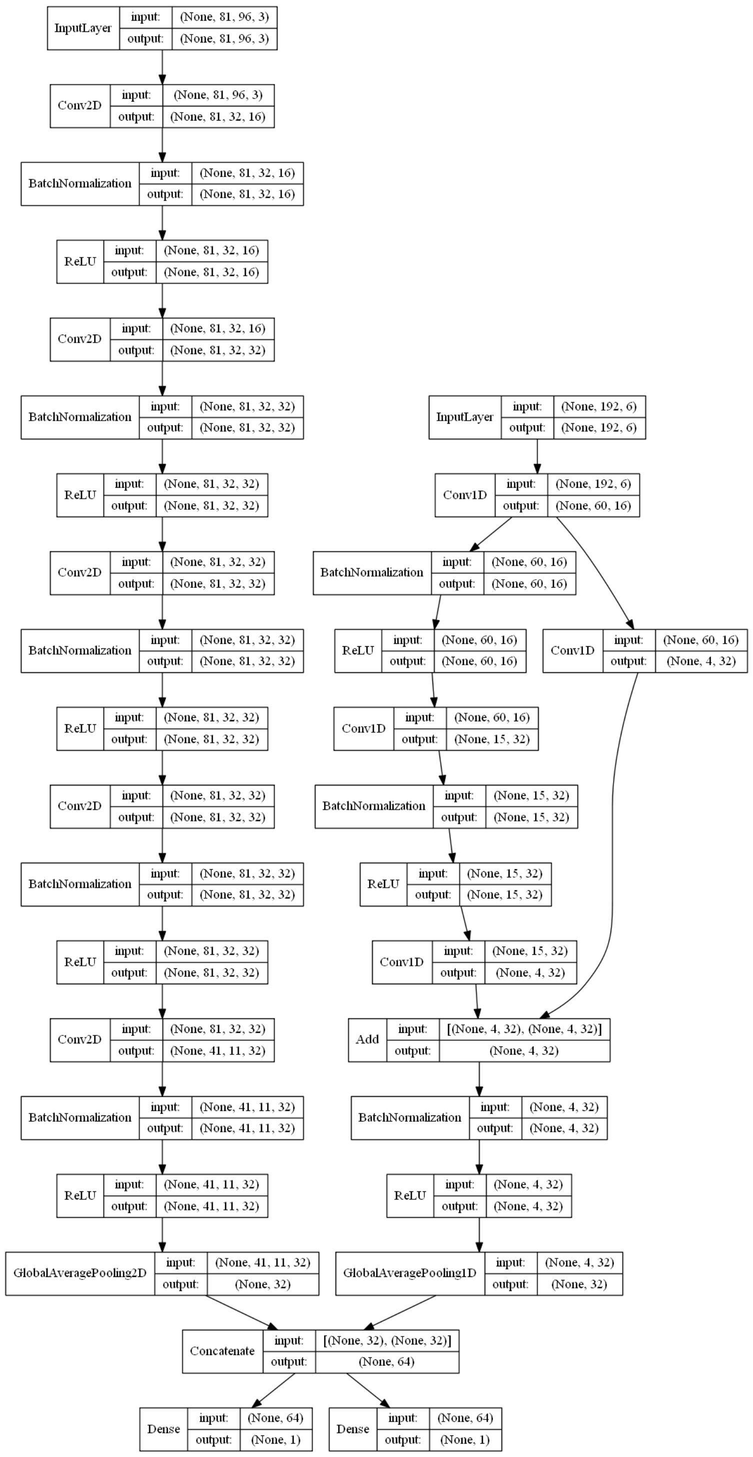

InputLayer | input: (None, 81, 96, 3) | output: (None, 81, 96, 3)
Conv2D | input: (None, 81, 96, 3) | output: (None, 81, 32, 16)
BatchNormalization | input: (None, 81, 32, 16) | output: (None, 81, 32, 16)
ReLU | input: (None, 81, 32, 16) | output: (None, 81, 32, 16)
Conv2D | input: (None, 81, 32, 16) | output: (None, 81, 32, 32)
BatchNormalization | input: (None, 81, 32, 32) | output: (None, 81, 32, 32)
ReLU | input: (None, 81, 32, 32) | output: (None, 81, 32, 32)
Conv2D | input: (None, 81, 32, 32) | output: (None, 81, 32, 32)
BatchNormalization | input: (None, 81, 32, 32) | output: (None, 81, 32, 32)
ReLU | input: (None, 81, 32, 32) | output: (None, 81, 32, 32)
Conv2D | input: (None, 81, 32, 32) | output: (None, 81, 32, 32)
BatchNormalization | input: (None, 81, 32, 32) | output: (None, 81, 32, 32)
ReLU | input: (None, 81, 32, 32) | output: (None, 81, 32, 32)
Conv2D | input: (None, 81, 32, 32) | output: (None, 41, 11, 32)
BatchNormalization | input: (None, 41, 11, 32) | output: (None, 41, 11, 32)
ReLU | input: (None, 41, 11, 32) | output: (None, 41, 11, 32)
GlobalAveragePooling2D | input: (None, 41, 11, 32) | output: (None, 32)
InputLayer | input: (None, 192, 6) | output: (None, 192, 6)
Conv1D | input: (None, 192, 6) | output: (None, 60, 16)
BatchNormalization | input: (None, 60, 16) | output: (None, 60, 16)
ReLU | input: (None, 60, 16) | output: (None, 60, 16)
Conv1D | input: (None, 60, 16) | output: (None, 4, 32)
Conv1D | input: (None, 60, 16) | output: (None, 15, 32)
BatchNormalization | input: (None, 15, 32) | output: (None, 15, 32)
ReLU | input: (None, 15, 32) | output: (None, 15, 32)
Conv1D | input: (None, 15, 32) | output: (None, 4, 32)
Add | input: [(None, 4, 32), (None, 4, 32)] | output: (None, 4, 32)
BatchNormalization | input: (None, 4, 32) | output: (None, 4, 32)
ReLU | input: (None, 4, 32) | output: (None, 4, 32)
GlobalAveragePooling1D | input: (None, 4, 32) | output: (None, 32)
Concatenate | input: [(None, 32), (None, 32)] | output: (None, 64)
Dense | input: (None, 64) | output: (None, 1)
Dense | input: (None, 64) | output: (None, 1)

Figure 4: CNN architecture. Left network processes magnetometer data, right network processes solar wind data. outputs of both networks were fed into linear layers for classification (with sigmoid activation) and regression (no / linear activation)

The RNN architecture for classification is given by figure 6. Gated recurrent units [10] are used because they are able recognize long term relationships better than vanilla RNN cells. The same architecture was used to predict strength magnitude, but instead of the last softmax layer, the last layer is a dense layer of size 1, which predicts the strength output. Additionally a third architecture was trained where the fully connected layer feeds into both a softmax layer and a dense layer of size 1 such that the classification and strength prediction were trained simultaneously on the same network.

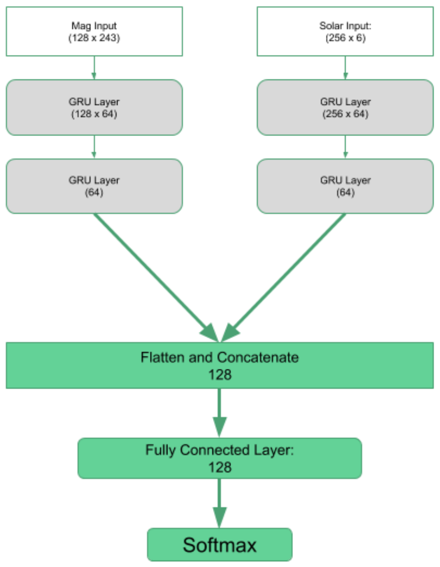


Figure 5: RNN Architecture for classification. For strength prediction, the last softmax layer is replaced with a dense layer of sizes 1. For simultaneous classification and substorm prediction there are two last layers, one sotfmax and one dense layer. This architecture has approximately 100,000 parameters. We also trained a network similar network that returned the entire sequences from each RNN model but this resulted in similar results and training time (figure 6.

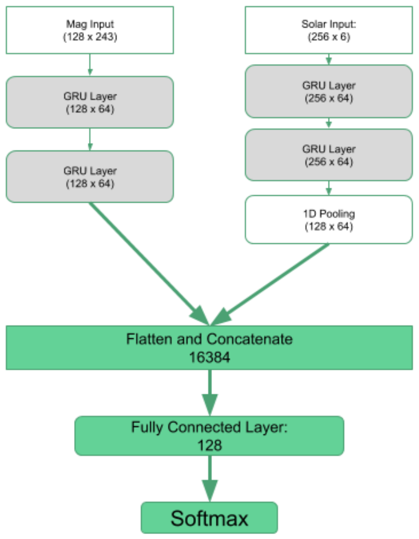
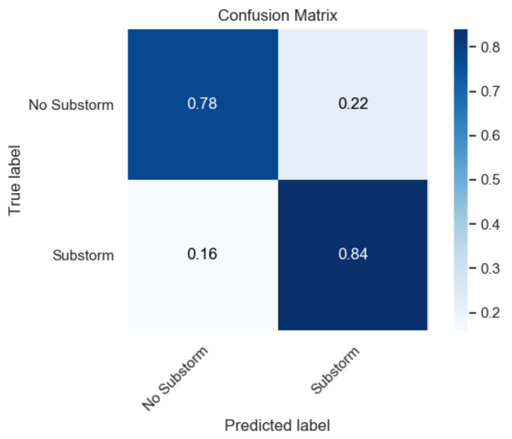


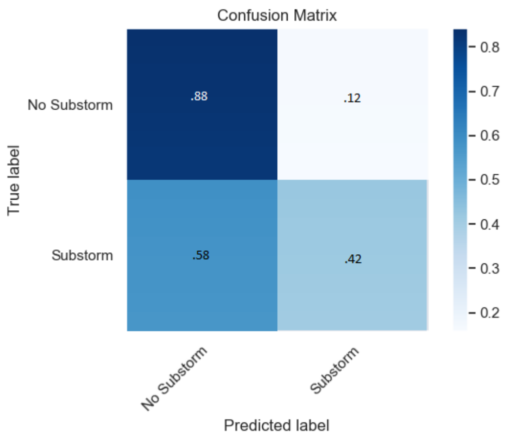
Figure 6: RNN architecture with sequences returned from each RNN network. Despite the larger number of parameters, the performance of this network did not differ greatly from the the architecture from figure 5

Method	Classification Accuracy	Strength Mean Squared Error
CNN - joint	.789 $\pm$ .0164	48,831 $\pm$ 6,397
RNN - joint	.761 $\pm$ .008	70,362 $\pm$ 1,068
RNN - mag data only	.692	N/A
RNN - solar wind data only	.701	N/A
Logistic Regression	0.660	N/A

Table 1: Performance comparison between our models and the baseline



(a) CNN confusion matrix



(b) RNN confusion matrix

Figure 7: Confusion matrices

5 Results

Our networks were trained on a dataset containing 33424 examples from the years 1990 - 2015. Each example contains an array of magnetometer data of size $(T_m \times 81 \times 3)$ (input length in minutes depending on architecture, 81 stations, 3 axes) and an array of solar wind data of size $(T_w \times 6)$ (input length in minutes, 6 axes). Each example is associate with a 30 minute prediction period that occurs after the example, for which there is binary label of 1 (a substorm occurs in that period) or 0 (no substorm occurs in that period). Our dataset is a small subset of the overall larger dataset, as we curated our dataset to have a 50-50 split of both classes. In the true data, the amount of examples labeled "no substorm" vastly outnumbers the number labeled as "yes substorm". As such, in order to train an accurate network we need to select our data in such a way that it was not biased towards one of our two classes. With significant class imbalance, a machine learning algorithm could achieve decent score just by always choosing the more likely class. By artificially balancing our classes, we ensure that any predictions the neural networks make are based on features in the input data.

We split our initial dataset into training, validation and test sets. The training and validation sets are selected from the years 1990 - 2015, and the test set is from 2015 - 2018. The purpose of the validation set is to make sure our networks aren't overfitting and the test set is to evaluate our trained networks on data actually from the future. In a real world setting, we would want to train the network on all the data we have up until now, but would want to be confident that it would still work on next year's data. This resulted in a .7565 train - .1355 validation - .11 test split. While these split proportions are somewhat arbitrary, they provide a good balance between having a large training dataset and having enough validation and test examples for meaningful evaluation of performance. For the RNN models we trained each model for 20 epochs and for the CNN models we trained for 15 epochs. We retrained the CNN 10 times and the RNN 5 times (because it took much longer to train) to estimate the variance across different trainings.

For our baseline we used a logistic regression model on the SME data. SME is a scalar directly derived from the magnetic field measurements every minute and we found that reducing the dimension of the input to the logistic regression model improved its accuracy. We are unaware of any physics-based attempt to predict substorms and so given its popularity, we decided logistic regression would make for a good baseline.

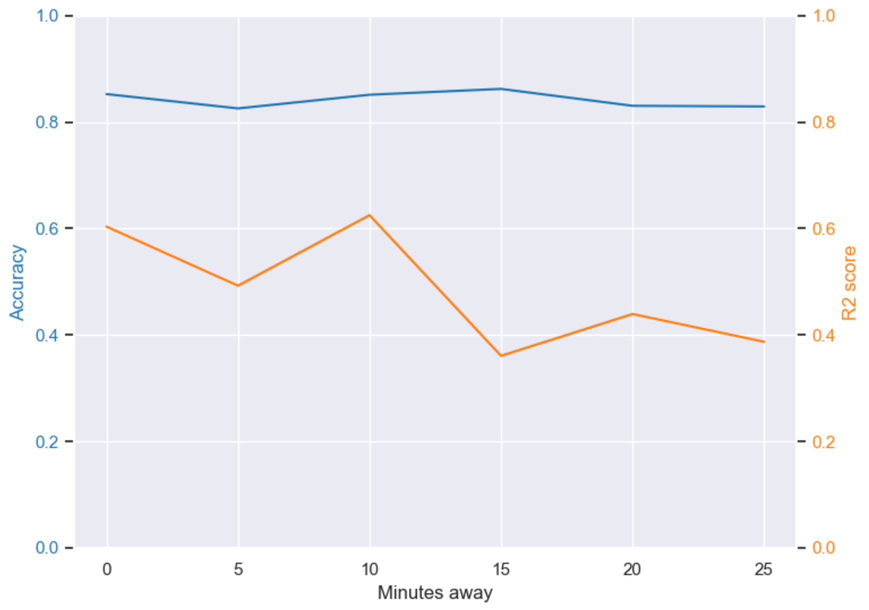


Figure 8: Performance over time

Our results, shown in Table 1 are based on the networks' evaluation of our test data. Figure 7 shows the confusion matrices for our models. The CNN and RNN performed similarly, both reaching almost 80% classification accuracy, with the CNN having a high true positive rate and the RNN having a low false positive rate. Both significantly outperformed the baseline. Figure 8 shows how the true positive classification rate and r2 score for regression of the CNN is affected by how far away the substorm occurs in time. The classification accuracy doesn't seem to degrade over the 30 minute prediction window, whereas the r2 score does.

6 Discussion

6.1 Hyperparameter Search

There were a number of interesting results from the hyperparameter search. As expected, smaller batch sizes resulted in higher test accuracy, both for the RNN and CNN architectures. One drawback of the RNN style architecture was that it took a lot longer to train. This prevented us from using smaller batch sizes, which could in part explain the discrepancy in accuracy between the styles. Figure 9 shows for CNN that on average, the accuracy can increase by about .03 by decreasing the batch size from 64 to 8. One explanation for why batch size affects performance is that smaller batches result in more training steps per epoch. We wonder if batch size would have the same effect if networks were trained for the equal number of steps as opposed to equal number of epochs. Smaller batches also provide a noisier estimate of the gradient which could act as a regularizer.

Another observation, seen in Figure 10, is that the accuracy improves when the strides over the magnetometer data are lengthened (more downsampling). For downsampling over stations, this could imply that the features relevant to predicting substorms mostly appear at individual stations. Another explanation is that a lot of the input data is missing and so the downsampling often occurs over zeros (missing data). Increasing downsampling over time also has a positive impact on performance, implying that there is a lot of overlap in information between two measurements from the same station. Overall downsampling results in a lower dimension deeper into the network which makes the global average pooling layer throw away less information.

Finally the hyperparameter search proved fairly definitively that solar wind data is predictive of substorms. Seen in Figure 11, there is a .02 increase in accuracy on average when including solar wind data as an input and the error bars hardly overlap. Interestingly, increasing the solar wind data input length T_w doesn't improve accuracy on average.

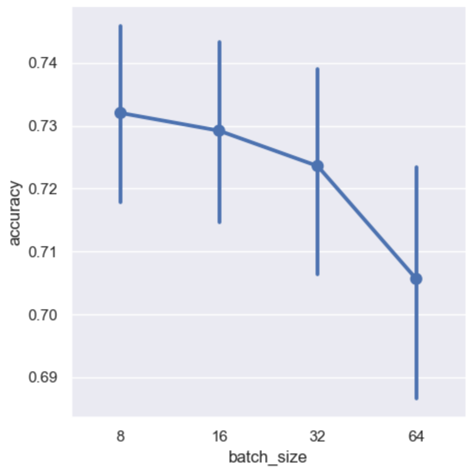
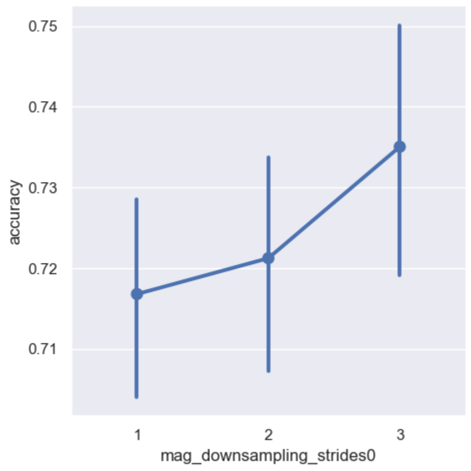
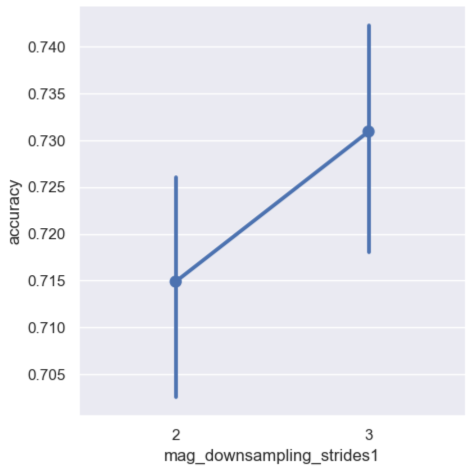


Figure 9: Effect of batch size on test accuracy



(a) Downsampling over stations



(b) Downsampling over time

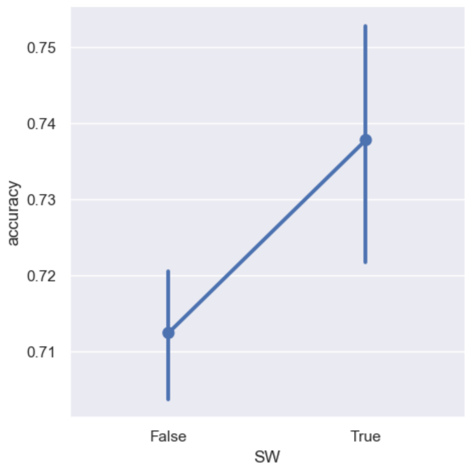
Figure 10: Effect of downsampling strides for the magnetometer data on accuracy

There appears to be a length that balances amount of information gained with dimensionality of input. This could be because the networks benefit from lower dimensional input data or because the solar wind data hold less information about the future the farther back one looks.

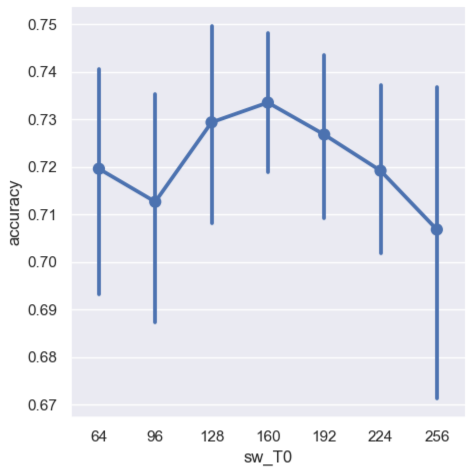
6.2 General Discussion

Overall we are happy with our results. Achieving almost 80% test accuracy seems like a very good first attempt at this task which significantly outperforms a reasonable baseline. Our results show that ground based magnetometer measurements definitely hold information about whether a substorm will soon occur. Adding in the solar wind data noticeably improved the accuracy of the model. Because substorms are a natural process, there is bound to be a stochastic element to their occurrence. This raises the question of what the limit is for prediction accuracy. We wonder if denser measurements of the magnetic field would help. One parameter we didn't get a chance to play with is the number of stations. It would be interesting to see how much the performance degrades as we remove stations.

One drawback with our approach is that an input sequence during which a substorm occurs is typically in the negative class. This means that the network could learn to associate substorm features and features immediately prior to



(a) including solar wind data vs not including



(b) Solar wind input length

Figure 11: Effect of solar wind on test accuracy

substorms with the negative class instead of the positive class. We could potentially deal with this by removing all negative examples which contain a substorm, but the more we modify the dataset, the less it resembles real measurements. In the future, a more flexible approach which can deal with these situations may be necessary.

Another issue that we note will be more difficult to solve is the existence of long scale time effects in the magnetometer data. Since the dataset occurs over many decades, it is likely that long time scale effects are not accounted for by our model. It could be that substorm frequency is modulated by the solar cycle, an 11 year variation in solar activity. Although we train on data from a long enough time period to see more than one solar cycle, our model has no capacity to explicitly take into account the phase of the solar cycle. Furthermore, there are many other types of events that can occur which affect the magnetic field including other types of geomagnetic storms. Our network is only trained to recognize substorm or non-substorm conditions, which is an oversimplification of the real world. Finally, the datasets we used were very large. Despite training our networks on a subsection of the stations as well as a small subsection of the total time of collection, our final dataset ended up being on the order of 10's of gigabytes in size. This resulted in long training times for our networks (on the order of hours in some cases). With more and time we might be able to achieve even better results (e.g., through more extensive hyperparameter tuning).

6.3 Future Direction

Since our work represents an initial establishment of a baseline, there are many different directions we could continue our work, which we will discuss here. One obvious thing we didn't get a chance to do is train on the full dataset. In a real-world scenario, there will be a significant class imbalance and a good substorm prediction model should be able to deal with that. Nonetheless, considering our high accuracy for the 30 minute prediction window, we suspect we could extend our predictions out further in time, as well as with more precision. That is we might consider a multiclass classification problem where, for example we predict substorms for minutes 0-10, 10-20, 20-30, etc. Alternatively we could consider changing the problem into a regression problem for the highest possible precision.
We made a naive approach at predicting the location of substorms, but did not get good enough results to include in our report. Basically we looked at which station contributed most to the prediction and estimated that the substorm would happen near that station. This turned out to be a mediocre predictor of location, but we suspect that we could do better if the networks were explicitly trained to predict location.

Although our hyperparameter search yielded pretty good results, it's possible that it was not sufficiently complex. Skip connections in neural networks allow for easier back propagation of gradients and our model could benefit if we allowed for all possible shortcuts as opposed to just simple Resnet-syle skip connections. In this study we considered CNN and RNN style architectures separately, but perhaps we could make a more accurate model by combining the two architectures. Additionally, we would like to try new state of the art architectures. In particular we suspect that attention [11] would be well suited for our time series analysis problem. Graph CNN architectures could provide a flexible way to deal with stations with missing data. Our networks establish relationships between stations using either multi-station convolution kernels or fully connected layers. These are rigid structures which don't allow for varying input shapes. Graph CNNs on the other hand are able to deal with nodes which have many close neighbors as well as nodes without any neighbors. This would also give us a way to have our network explicitly ignore missing stations.

We believe that the most interesting part of this project is its implications for physics. One of our hopes is that with high accuracy predictions we might be able to help inform physics on the nature of substorms. Since substorms are still not fully understood from a first principles standpoint, perhaps high fidelity statistical analysis methods such as the one we present here could contribute to our understanding of this phenomena. Dissecting neural networks is an actively researched topic and there are many methods which we could use to visualize the features learned by our networks. One promising method is called class activation maps [12]. We made an initial attempt at using this method but ran out of time before finding any interesting results. Perhaps by considering from a physics perspective what features are predictive of substorms, it could help better the community's understanding of these events.

7 Conclusion

In this paper we present a method for substorm prediction with the goal of establishing a baseline mark for accurate prediction. Our model is relatively simple, allowing classification for substorm incidence within the next 30 minutes. While we have shown relative success in our task defined above, we recognize that there are a number of ways we can continue this work, including accurate predictions of substorm strength and location. Eventually for accurate enough models, we might even eventually help elucidate some features of the underlying physics of substorm dynamics.

8 Workload Divide

The workload was largely evenly divided - with report writing and presentation creating divided evenly between Greg and Alan. Greg worked more on the CNN models as well as data downloading while Alan was more responsible for RNN networks and the baseline model. It is our opinion that the work was divided evenly - in large part most commits to the repository were a joint collaboration between the two authors.

9 Acknowledgements

We acknowledge use of NASA/GSFC's Space Physics Data Facility's OMNIWeb service, and OMNI data. We gratefully acknowledge the SuperMAG collaborators (`http://supermag.jhuapl.edu/info/?page=acknowledgement`). We would also like to thank the members of Professor Semeter's lab at Boston University's Center for Space Physics for their help in understanding the problem and the physics behind it as well as feedback on our approach.

We would also like to thank Liping Liu and the TFs of 150-09 for a great semeseter!

References

[1] David P. Stern and Mauricio Peredo. Substorms. `https://www-spof.gsfc.nasa.gov/Education/wsubstrm.html`, 2006.

[2] J. W. Gjerloev and R. A. Hoffman. The large-scale current system during auroral substorms. *J. Geophys. Res. Space Physics, 119, 4591–4606*, 2014.

[3] R. L. McPherron, C. T. Russell, M. G. Kivelson, , and Jr. P. J. Coleman. Substorms in space: The correlation between ground and satellite observations of the magnetic field. *Radio Science, Volume 8, Number 11, pages 1059-1076*, 1973.

[4] P.T. Newell, K. Liou, J.W. Gjerloev, T. Sotirelis, S. Wing, and E.J. Mitchell. Substorm probabilities are best predicted from solar wind speed. *Journal of Atmospheric and Solar-Terrestrial Physics*, 2015.

[5] J. W. Gjerloev. The supermag data processing technique. *J. Geophys. Res., 117, A09213*, 2012.

[6] P. T. Newell and J. W. Gjerloev. Evaluation of supermag auroral electrojet indices as indicators of substorms and auroral power. *J. Geophys. Res., 116, A12211*, 2011.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arxiv*, 2015.

[8] K. He, X. Zhang, S. Ren, , and J. Sun. Identity mappings in deep residual networks. *ECCV 2, 3, 5, 7*, 2016.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15 1929-1958*, 2014.

[10] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *arxiv*, 2014.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *31st Conference on Neural Information Processing Systems*, 2017.

[12] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *CVPR 2, 3, 4, 5*, 2016.

10 Additional Figures

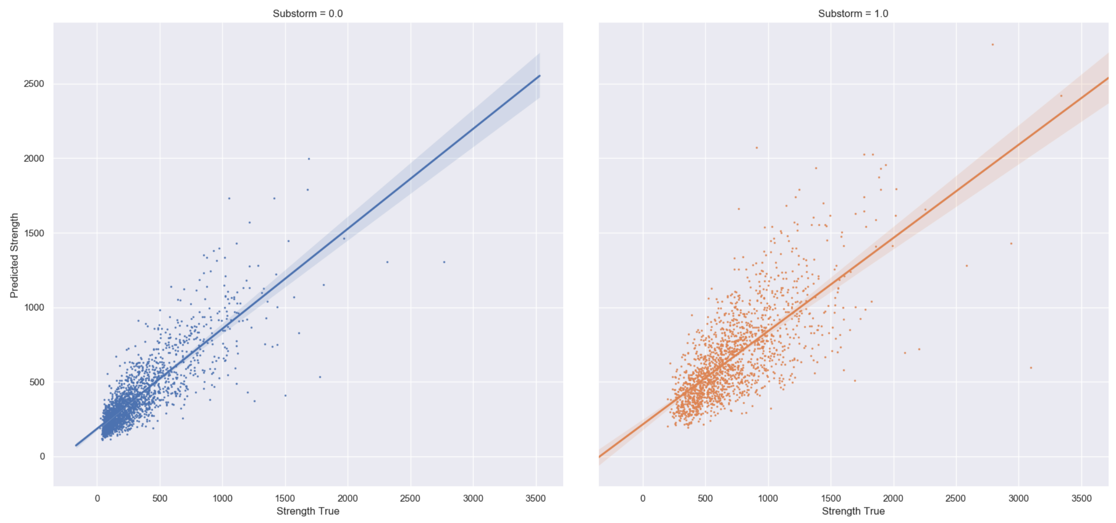


Figure 12: Strength estimate for positive and negative examples

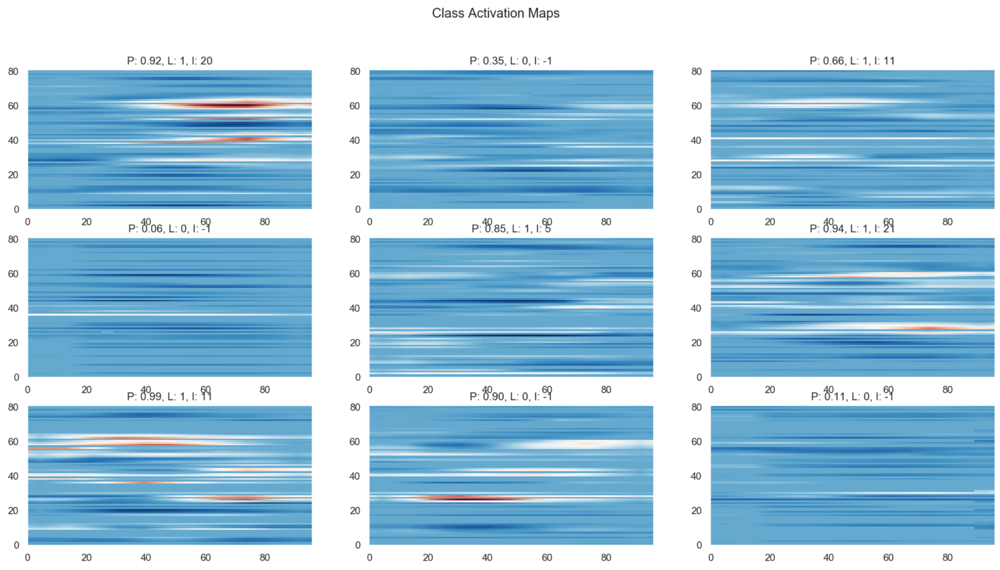
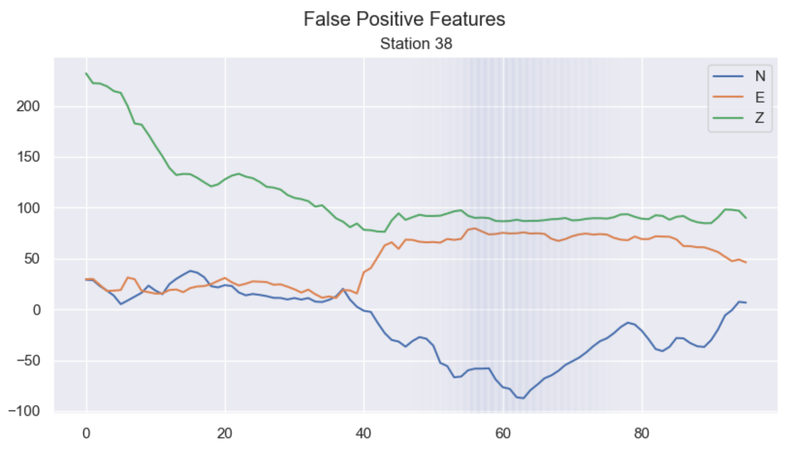
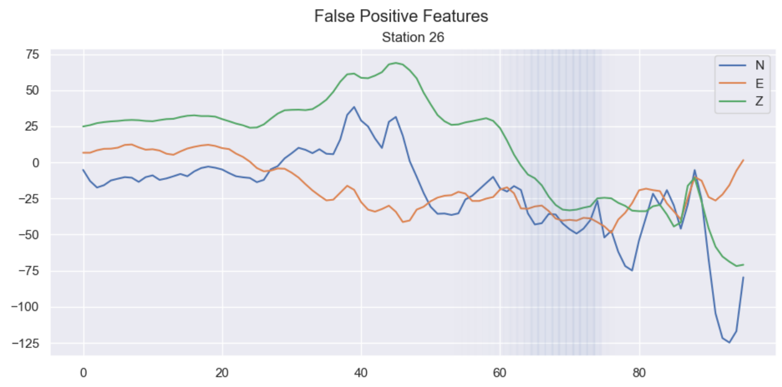
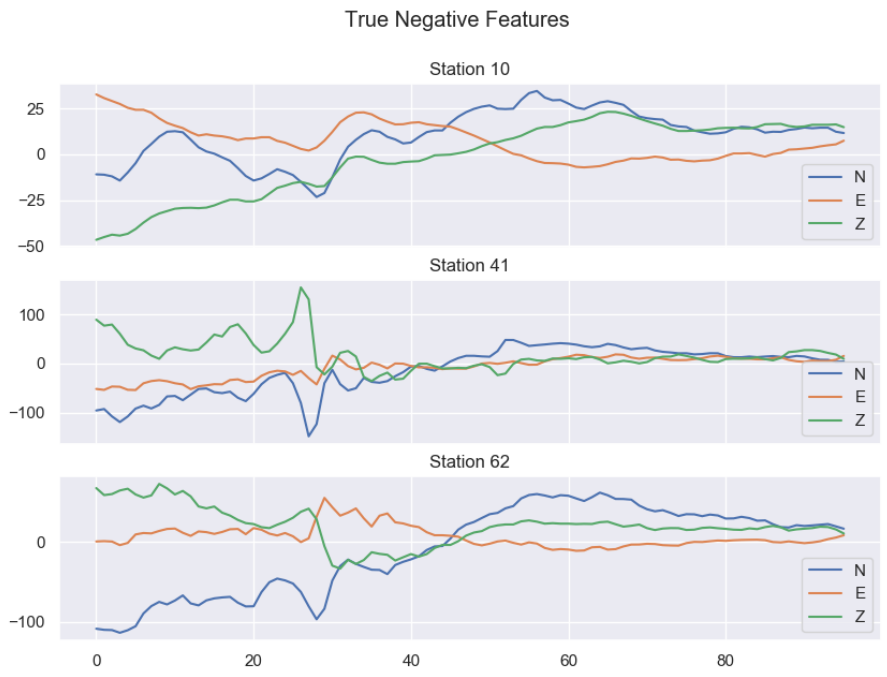


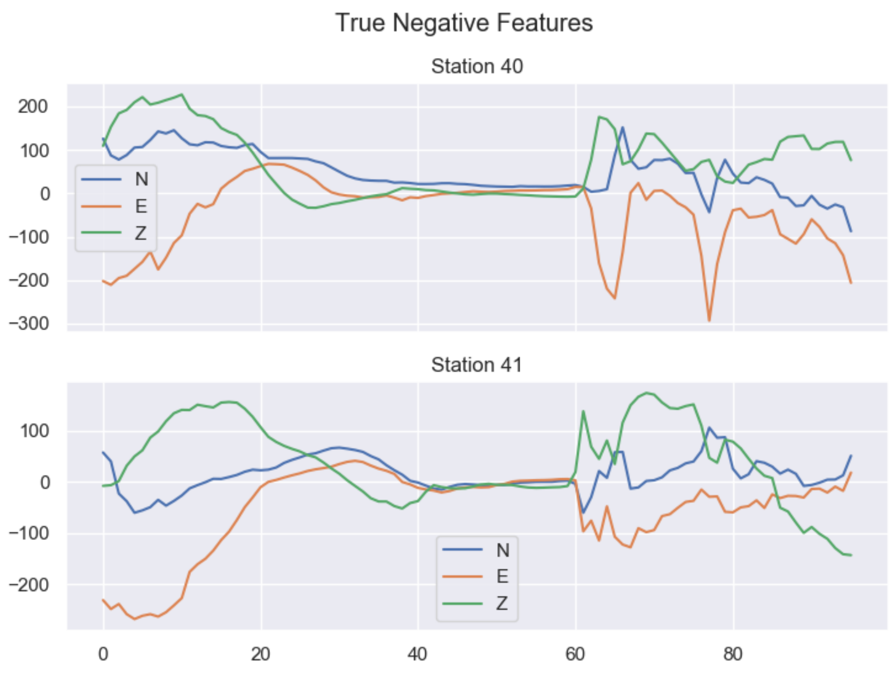
Figure 13: Class activation maps

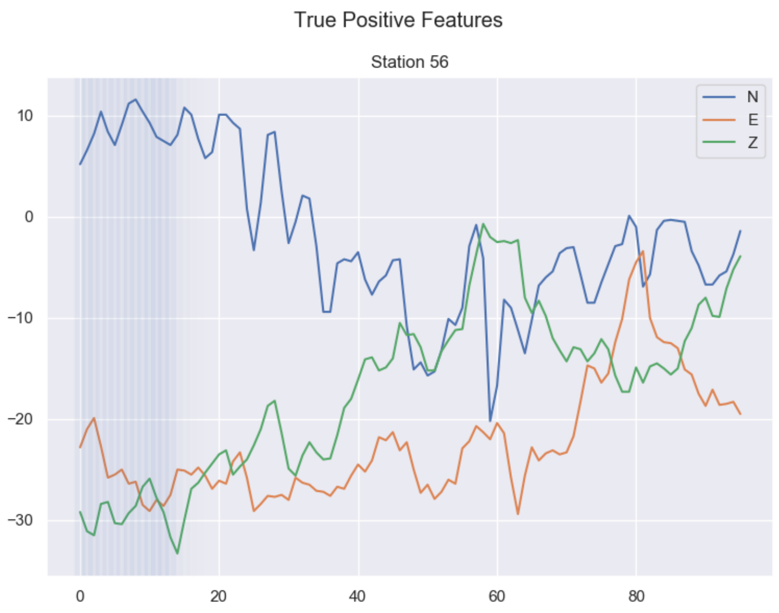
10.1 examples of magnetometer measurements



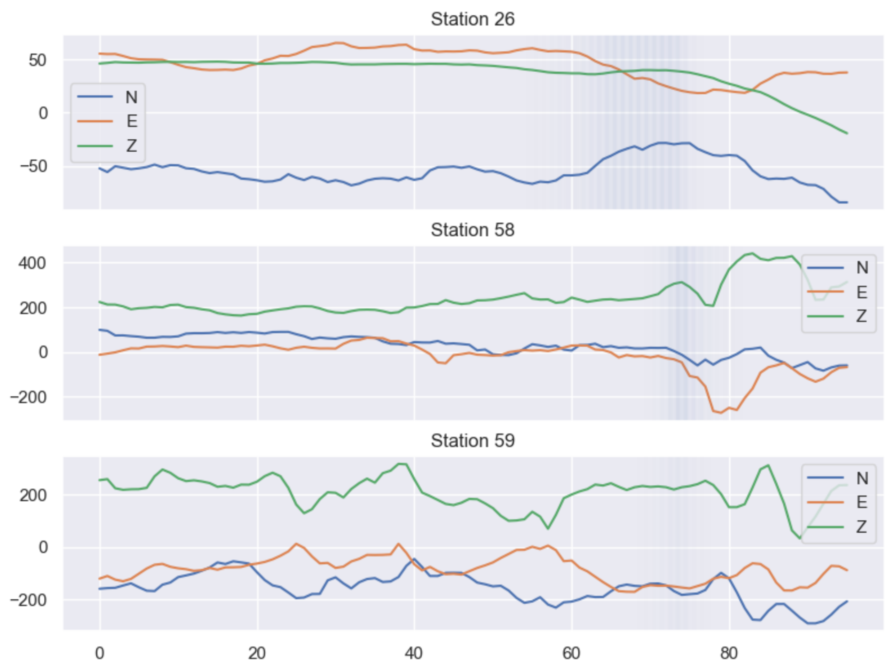




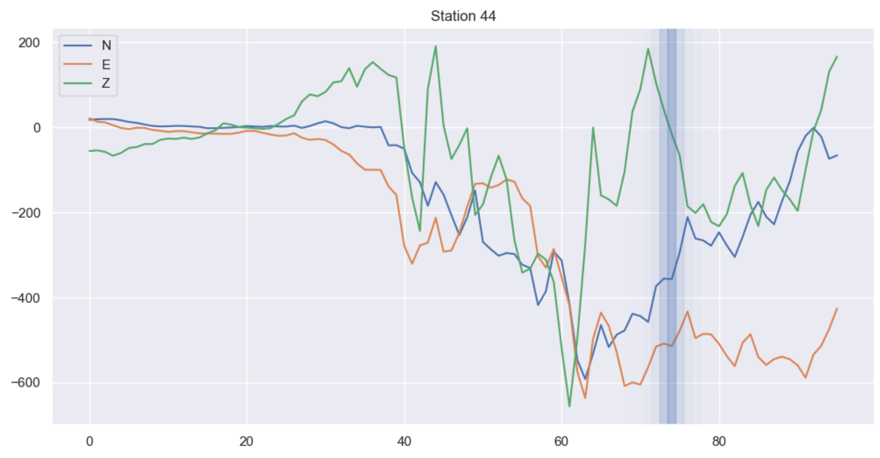

True Negative Features
Station 40
Station 41
N
E
Z
200
100
0
−100
−200
−300
0
20
40
60
80


True Positive Features
Station 56
N
E
Z
10
0
−10
−20
−30
0
20
40
60
80

True Positive Features



True Positive Features


Station 44
N
E
Z