Analysis of Earthquake Time Series Data to Predict Future Earthquake Events Using Different Approaches

Bellario, M., Elashri, M., Fuad, N.

Department of Mathematics

Department of Physics and Astronomy

University of Minnesota, Duluth

Introduction

In this project, we have two goals. First, we are going to investigate the special type of datasets called time series data. This type of data is specified with time order, so the order of data points represent matters and affect the information content. Usually, time series data measurements occur based on a regular basis in time. Some examples of this type of data could be data about how stocks represent the fluctuating nature of the stock market. We can give this definition of time series data as an ordered sequence of values of a variable at equally spaced time intervals. The area of our interest in this part will be the classification problem of time series data. Using machine learning techniques on this type of data provides us with a powerful machine/technique to deal with the exponential increase in data for better time and efficiency. Time series classification problem is different from regular classification because the data attributes have an ordered sequence. We are going to verify the result obtained on earthquakes datasets that claimed to have the best accuracy performance using rotational random forest algorithm. We are going to implement the algorithm and compare the result with the published result. Also, we are going to use Neural networks on our dataset.

Neural network is a branch of machine learning that has gained importance in the last two decades. Development in algorithms to build a neural network is continuous each day. Last year there was a paper published about replacing the neural network standard residual method by ordinary differential equation method which is called neural ordinary differential equation. [4] This new algorithm is not yet well tested for classification problems and our second goal is to test how well is it in terms of prediction accuracy and the time performance and memory usage which can be a crucial factor for building models for big datasets.

Our hypothesis is that Neural ordinary differential equations are a better algorithm for classification problems and should give us improvements in the accuracy of our prediction on test datasets. It is also faster in terms of computing time and is lighter in memory usage.

Related Work

In a paper in 2016, A[1]. Bagnall et al. applied different machine learning algorithms to find out which one is the best. They found that there is no single best algorithm for every dataset, rather the efficiency of an algorithm depends on the dataset. They used Rotation forest (RotF), Support Vector Machine (SVM), Neural network (NN) etc. to reach to their conclusions. One of the dataset they used was 'Earthquake dataset'[3]. They showed that RotF is the best algorithm for this dataset with an accuracy of 75.92%. In another paper[2] in 2018, they claimed that RotF is probably the best algorithm for classification problem with continuous features. But in a paper[4] in 2018, a group from Toronto came up with a new idea called 'Neural Differential Equation Method' to solve the machine learning problems. Their main idea was that the basic equation for Residual network is similar to the discrete equation of solving differential equation using Euler method, so we can replace a neural network with a differential equation. They showed that in general, this Neural ODE method works around 6 times better than the other algorithms. A. Bagnall et al. did not apply this to the datasets he worked on. We intend to apply this to the Earthquake dataset to verify their claims.

Proposed Work

In this project, we propose to implement machine learning algorithms to predict the future earthquakes using the Earthquake dataset mentioned earlier. We intend to do this using three different algorithms:

- Rotation Forest (RotF)
- Neural Network
- Neural Differential Equation

We want to implement the Rotational Forest and Neural Differential Equation method from scratch and implement

Neural network method using built in function of Keras module in Python. Our first goal is to reproduce the same result as in the paper using RotF and Neural Network method. Then we are going to implement previously unimplemented method of Neural Differential Equation method on the same data. Doing this in three different methods will give us confidence in our result as well as help us to compare the different method and sub-sequentially verify the works of the previous papers mentioned in a small way. While comparing we will focus on three parameters mainly:

- Accuracy: Which model gives the best accuracy on test data.
- Time Complexity: Which model takes lesser time to train on the training dataset.
- Memory Complexity: Which model takes lesser space in memory while training.

So, compactly, objectives of this project are:

- Learning how different Machine Learning algorithms (in this case RotF, Neural Network and Neural Differential Equation method) work.
- Reproducing the result from the author of the datasets that we have already found
- Comparing different ML algorithms in terms of accuracy, time and space complexity.

Experimental Evaluation

Dataset

The dataset we are working on, the Earthquake data, is taken from Northern California Earthquake Data Center and donated by A. Bagnall. This is a time-series dataset where each data is a reading of Richter scale taken each hour since December 3, 1967 to 2003 at Northern California Earthquake Data Centre. Then to transform this dataset into a classification problem, A. Bagnall et al. first defines a major event as any reading of over 5 on the Richter scale. Major events are often followed by aftershocks. The physics of these are well understood. Hence they considered a positive case to be one where a major event is not preceded by another major event for at least 512 hours. To construct a negative case, they considered instances where there is a reading below 4 (to avoid blurring of the boundaries between major and non major events) that is preceded by at least 20 readings in the previous 512 hours that are non-zero (to avoid trivial negative cases). None of the cases overlap in time (i.e. we perform a segmentation rather than use a sliding window). Of the 86,066 hourly readings, they produce 368 negative cases and 93 positive. To have an idea about the dataset, we can have a look at the dataset in the following way:

	Class	***Attr. 1***	***Attr. 2***	***...***	***Attr. 512***
1	1	-0.518	-0.518	...	-0.518
2	0	1.354	-0.353	...	-0.353
3	0	2.639	-0.316	...	-0.316
...	...	...	...	...	...
322	0	-0.484	-0.484	...	-0.484

So we can see that in each row there are 513 columns. First column shows the classification with 1 being the positive cases and 0 being the negative cases. Next 512 columns are the attributes that give that classification. There are 322 of those. And there are also 139 of those data-rows for us to test the model that we build. However, at first, we plotted the dataset to find out if there is any visible trend. In Fig 1 we show the first few data using red/triangle sign as negative case and green/solid circle sign as positive case. As it turns out there is no visible trend and this solidifies the fact that the only way to go on with this data is using machine learning.

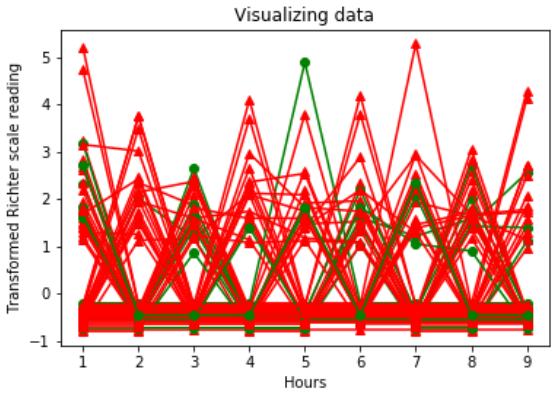

Fig 1: Visualizing earthquake data

Algorithm

We are going to use multiple ML algorithms in this project, namely Rotation Forest (RotF), Neural Network and Neural Differential Equation methods. These are explained very briefly here:

1. Rotation Forest: Rotation forest is a very powerful algorithm for classification problems in ML. This is actually comprised of two different algorithms- PCA and random forest. In this algorithm, PCA is applied first on the data and then that dataset is classified using random forest algorithm.

-PCA: PCA is Principal Component Analysis. It decreases the dimension of data. Its purpose is to prevent over-fitting of data and to improve the accuracy of classification algorithm that will be implemented later.
-Random Forest: Random forest is ensemble type classifier. In this algorithm, some specific number of features and data are chosen randomly and they are classified using decision tree. And after that those decision trees are used to get classification result. Final result is found using taking 'vote' from those trees.

2. Neural Network: Neural network is a very powerful tool in machine learning. The basic idea of this is that this algorithm assigns some weight to the attributes by minimizing a loss function and finally use an activation function (sigmoid, relu, tanh etc) to make the final decision. In addition to that, it may contain some hidden layers too before making that final decision.

3. Neural Differential Equation Method: Neural Differential equation method is a brand new approach to solve ML problems. It is observed that the residual network equation $h_{t+1}=h_t+F(h_t)$ can be represented as special case of discrete Euler ordinary differential equation method formula: $h_{t+1}=h_t+eF(h_t)$ with e=1. So, if we consider a layer of our neural network to be doing a step of Euler's method, then we can model our system by the differential equation: dh(t)/dt = F(h(t),x,t), where x is our training parameter and t is the time dependence.

Implementation of algorithms

1. Rotation Forest

We begin investigating our data using rotation forest with the earthquake data. The rotation forest algorithm was implemented in python. The built in function of *sklearn* package for decision tree was used, but everything else was implemented from scratch. So, the algorithm in brief was: *PCA on training data → Bootstrapping → Making a forest of random decision trees → Making final decision from the 'votes' of the trees → Making prediction using test data.*
With the increase of number of trees, the accuracy of the algorithm did not increase linearly as seen from the Fig 2. Accuracy dropped quite a bit for some number of trees, but the general trend of accuracy was increasing. And after around 20 trees, the accuracy became the highest and stayed the same. And it did not vary even a little with or without PCA. At this point, the confusion matrix was:

	Predicted YES	**Predicted NO**
Actual YES	104	35
Actual NO	0	0

So, the highest accuracy acquired by this rotation forest was 74.82%.

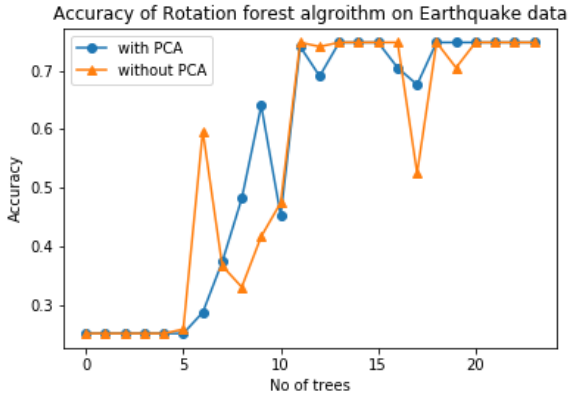

Fig 2: Accuracy of rotation forest algorithm on earthquake data

The aforementioned paper indicates that they found 75.92% accuracy for prediction using this algorithm for the earthquake dataset. In order to compare our implementation, we also applied the built in random forest function from sklearn on our dataset. Fig 3 shows the relation between accuracy and number of trees. We were able to obtain 76.28% accuracy at maximum using this and the average accuracy is 74.32% which is slightly less than the paper claimed.

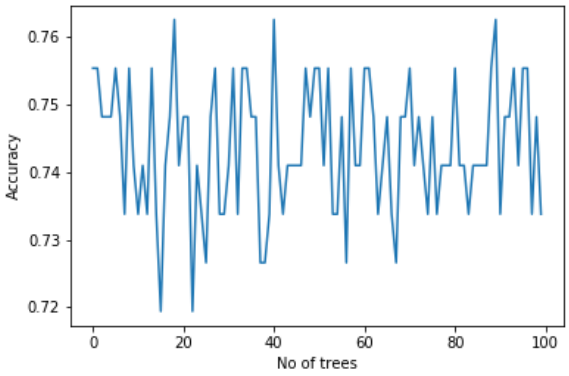

Fig 3: Accuracy of rotation forest using built in function in sklearn

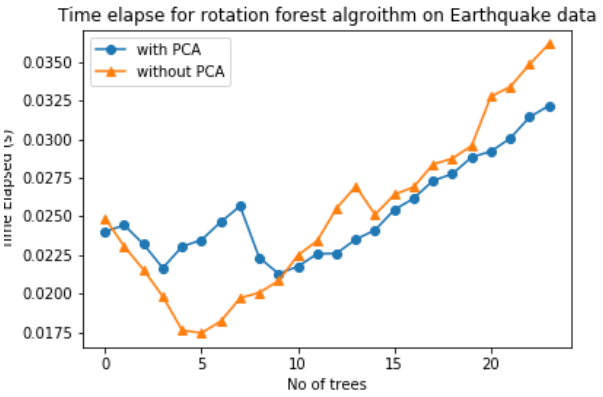

Fig 4: Time elapsed for RotF on Earthquake data

Fig 4 shows the relation between time needed with number of trees. It shows that for both with and without PCA, although elapsed time sometimes increases or decreases, the general trend is that it needs more time with more trees, which was expected. But interestingly, it takes it takes lesser time for relatively larger number of trees if the data is transformed with PCA at first. Fig 3 showed that at least 20 trees are needed to get maximum accuracy in this method. And Fig 4 shows that in the case of using 20 trees, time needed to get to accuracies are 0.028 seconds and 0.032 seconds respectively for with or without PCA. This means that although Random forest and Rotation forest are almost similar in terms of accuracies, but rotation forest is slightly (~12.5%) better than random forest in terms of time consumed.

2. Neural Network

Neural network is one of the most used ML algorithm. In our case, we did not implement it from scratch, rather we used built-in functions from *keras* to implement this on our data. We used 3 hidden layers with 12,20 and 1 neurons sequentially. Then we ran the earthquake data for 40 epochs with batch size=10. We found that the accuracy and/or loss gets saturated after around 25 epochs as shown in Fig 5. This The average accuracy and loss training was 99.69% and 1.27%. Then we used the network to make prediction on test data and got the confusion matrix:

	Predicted YES	**Predicted NO**
Actual YES	92	12
Actual NO	25	10

This confusion matrix gives us the accuracy of the network to be 73.38%. This accuracy is much lesser compared to the accuracy from Rotf algorithm. It is to be mentioned that Relu has been used as the activation function for the first two layers whereas sigmoid is used the activation function for the last layer. And the loss function used was 'binary cross-entropy'.

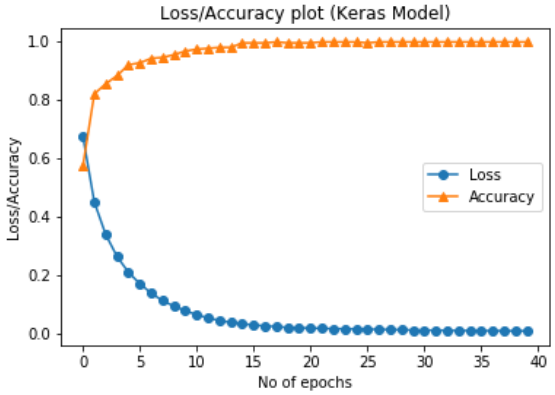

Fig 5: Loss/accuracy in training using Neural network on earthquake data

Fig 6 shows the time needed to find the accuracy (without plotting anything) for different number of epochs used. Elapsed time is increased smoothly as expected. But one thing worth mentioning here is elapsed time is not exactly linear, rather 'slightly quadratic'. From the previous figure, it is seen that we start to get maximum accuracy if we train for 19 epochs and Fig 6 says that time needed for that is 113.69 seconds. Neural network is also applied for other two famous dataset named MNIST and CIFAR10 dataset because of comparison purpose. The result for that is shown in the Appendix A.

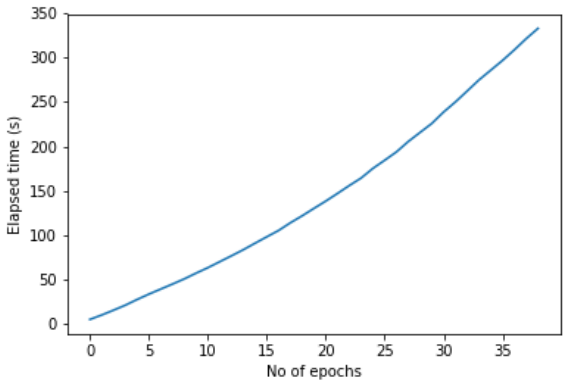

Fig 6: Time needed for different no of epochs in Neural Network implementation on earthquake data

3. Neural Differential Equation

We tried to implement Neural Differential Equation method on our dataset. We used python anaconda framework and used Euler method as our differential equation solving method. But unfortunately we were not able to implement that on our data although we were able to implement the same method on two very famous dataset named MNIST and CIFAR10 dataset. The results from these two are shown in the Appendix B.

Comparison of Algorithms

This comparison is mostly comparison among the codes that run on a specific machine, rather than actual rigorous and theoretical comparison of algorithm. We compare the algorithms in terms of accuracy and runtime.

1. Comparison among random forest, rotation forest and neural network on earthquake data

Random forest, rotation forest and the neural network was applied on the earthquake dataset. Fig 7 shows the comparison of their accuracies and runtime on a specific machine. In case of random/rotation forest, it took 19 trees to get to their highest accuracy. So, time to build 19 trees was taken. In case of Neural network, it took around 20 epochs of training. So time for 20 epochs of training was taken for that. From Fig 7, it is clear that Rotation forest is the slightly better all other algorithms on earthquake dataset in terms of accuracy. And in terms of runtime, Neural network is severely worse than the other two and the rotation forest is again the best algorithm. In fact the random/rotation forest is so much better than neural network (0.032s/0.028s compared to ~113s) that the column for those two cannot even be seen in Fig 7.

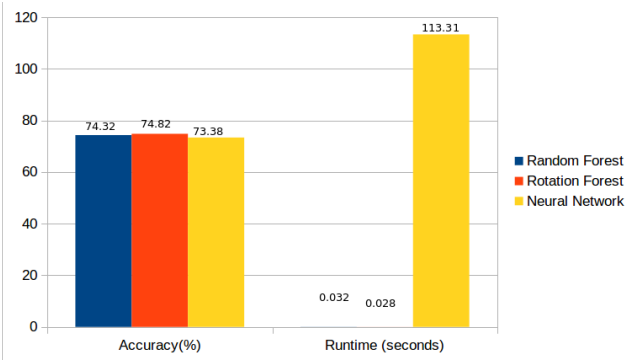

Fig 7: Comparison of random forest, rotation forest and neural network on earthquake data

2. Comparison between Neural network and Neural Differential Equation on MNIST and CIFAR10 Data

We could not get to the point of implementation of neural differential equation method on the earthquake data. So for now we compare that method with neural network using MNIST and CIFAR10 data. Comparison between Neural network and Neural differential equation method for MNSIT and CIFAR10 data for accuracy and time are shown in the Fig 8 and Fig 9.

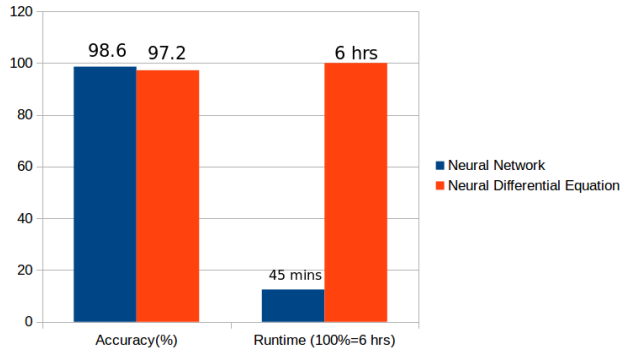

Fig 8: Comparison between Neural network and Neural differential equation for MNIST data

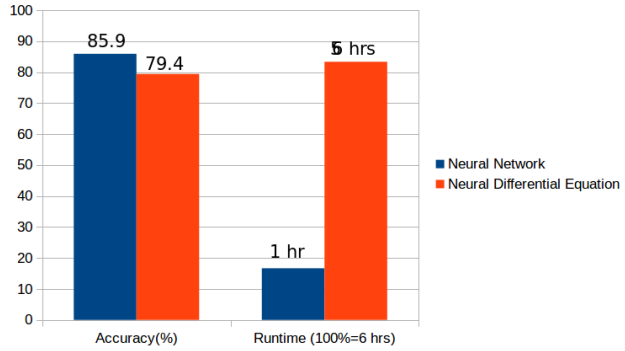

Fig 9: Comparison between Neural network and Neural differential equation for CIFAR10 data

Fig 8 and Fig 9 shows that Neural differential equation method did not work as good as expected either in terms of accuracy of runtime. For example, the former took around 9 times and 5 times more time than the later for MNSIT and CIFAR10 dataset respectively.

Conclusion

The main goal of the project has been to use some earthquake data as a machine learning classification problem to predict the future earthquake events based on recent Richter scale reading. Since this is done using three different algorithms, RotF, Neural network, and neural differential equation method, the methods are also comapred in terms of accuracy and time. We implemented RotF algorithm from scratch and it worked as good as the paper we tried to reproduce having around 75% accuracy instead of 75.92%. Neural network was implemented using built-in functions in keras and found to be 73.38% accurate. So, it is still debatable how reliable the existing machine learning models are to predict future earthquake. The brand new approach in this sector, neural differential equation method, has not been implemented on earthquake dataset yet, but for the sake of comparison it is implemented on two famous dataset: MNIST and CIFAR10. In these two cases, it is surprisingly found that neural network works better than neural differential equation method in terms of both accuracy and time. But the comparison of neither time nor accuracy is done between neural network and Rotf on the earthquake dataset yet. This is one of the future goal that is needed to be fulfilled. Also the neural differential method is also needed to be implemented on the earthquake dataset. This will make it a little bit easier to compare all three algorithms in a more structured way. But from the existing analysis, it is seen that RotF is the best algorithm among RotF, Random forest and Neural network on the earthquake data. And on both MNIST and CIFAR10 data, the Neural network is better than the Neural differential equation method. So it can probably be safely assumed that the Rotation forest is the best algorithm in terms of both accuracy and runtime for the earthquake data.

References

[1] Bagnall, A., Lines, J., Bostrom, A., Large, J. & Keogh, E (2016), The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining And Knowledge Discovery, 31(3), 606-660*

[2] Bagnall, A., et al., Is rotation forest the best classifier for the problems with continuous features? , (2018) arxiv.org/abs/1809.06705

[3] Time Series Classification Website. (2019). Retrieved from http://www.timeseriesclassification.com/description.php?Dataset=Earthquakes

[4] Chen, T.Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D.K. (2018). Nueral Ordinary Differential Equations, NeuralPS.

Appendix

A. Implementation of Neural network on MNIST and CIFAR10 dataset

We also used neural network on two different dataset for later comparison. The first dataset is called MNIST (handwriting recognition dataset) which is one of the most basic data used in machine learning. The other one is CIFAR10 which is a dataset consisted of images that need to be classified into 10 different categories. In Fig 10 and Fig 11 we see the Accuracy and Loss versus the number of epochs using residual neural networks for MNIST dataset.

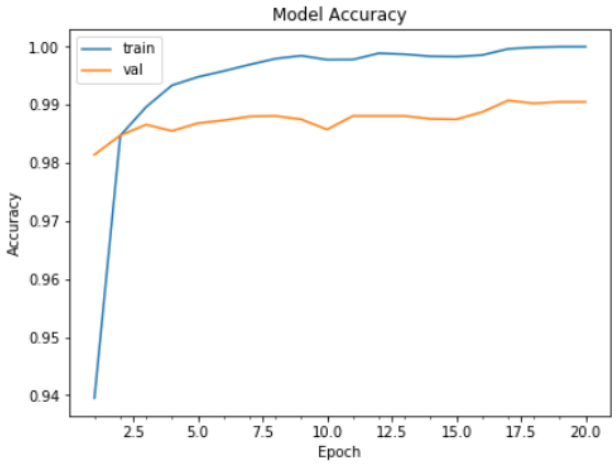

Fig 10: Training accuracy for MNIST handwriting data

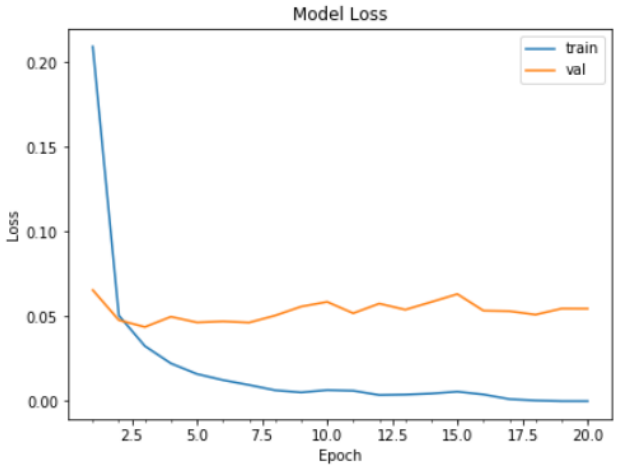

Fig 11: Training loss for MNIST handwriting data

Fig 12 and Fig 13 also give the same information but for CIFAR10 dataset. MNSIT dataset takes about 2348 seconds on average to be trained and CIFAR10 takes about 3748 seconds.

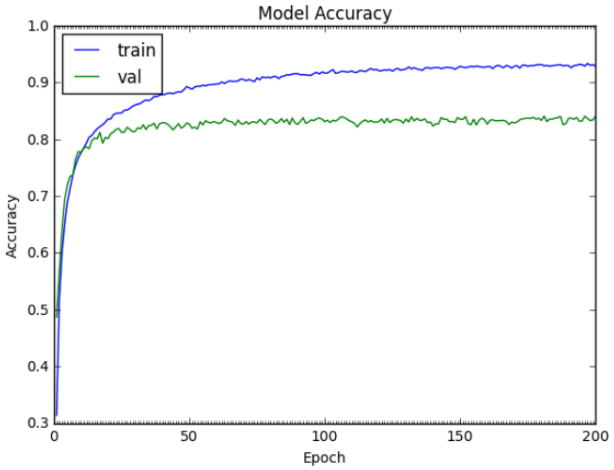

Fig 12: Training accuracy for CIFAR10 data

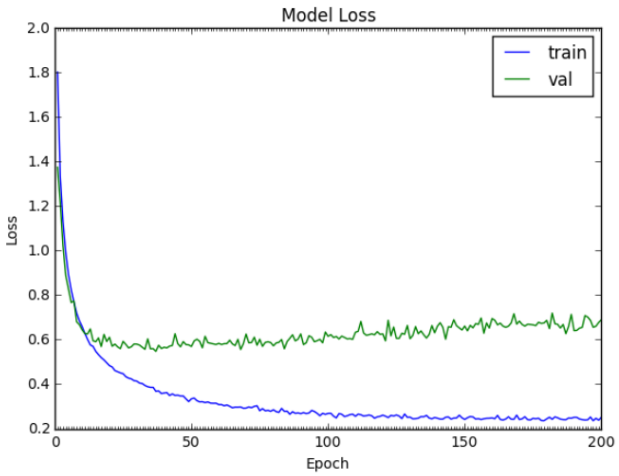

Fig 13: Training loss of CIFAR10 data

We can observe the pattern that after a small number of epochs the loss and accuracy of our models don't improve much and this is good in terms of the time and computing power. This behavior is the same for both of the datasets and agreed with the recorded time to train the model. The accuracy of neural network on MNSIT and CIFAR10 data were 98.6% and 85.9% respectively (Number of graphs could probably be reduced in this section with combining few. This will be taken care of in the final submission.)

B. Implementation of Neural ordinary differential equation on MNIST and CIFAR10 dataset

We have implemented Neural ordinary differential equations using python anaconda framework. We have used the Euler method as our differential equation solving method. We still have not been able to implement this method on our dataset, but we have tested our implementation using two of the standard datasets that are suitable for such comparison tests, as mentioned in the previous section. The loss and accuracy plot are also extracted using Neural Ordinary differential equation and Fig 14 and Fig 15 give the loss versus the number of epochs for MNIST and CIFAR10 data respectively.

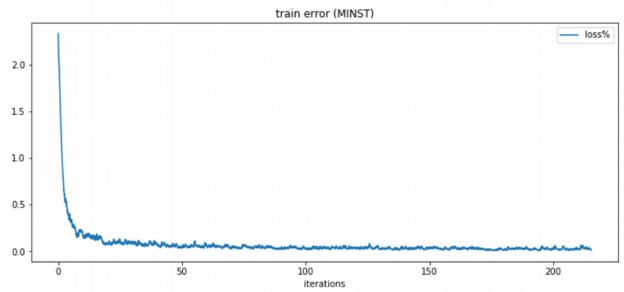

Fig 14: Training error using Neural Differential equation method on MNIST data

This neural differential equation method gave accuracy of 97.2% and 79.4% for MNIST and CIFAR10 data respectively, which is, quite shockingly, less than the accuracy in neural network method! Moreover this method appeared to be quite slow, since 10 epochs of MNIST and CIFAR10 data took 6 hrs and 5 hours to train, respectively, in our machine.

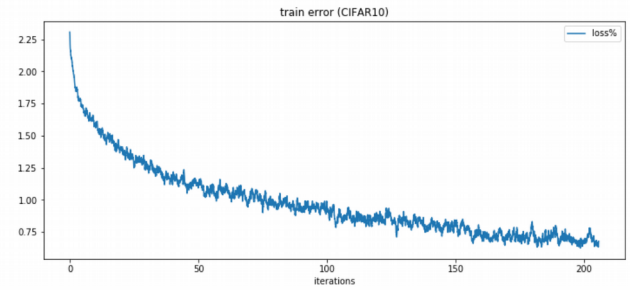

Fig 15: Training error using Neural Differential equation method on CIFAR10 data