Applied Deep Learning

Final Report

Muhamad Alawad 12402140

The Problem attempted to be solved and its significance:

The project scope or problem concerns scientific machine learning, particularly the classification of chemical compounds based on magnetic order. The data is obtained from the "Material Project" database, which tabulates chemical formulae and magnetic orders. There was a research gap as there are not any published models to classify chemicals by their magnetic ordering.

The input features chosen are the number of atoms for each element represented as one hot-coded vector, as well as the density of the material, so it is a 119-dimensional vector (118 for chemical elements + 1 for density). The output was the magnetic order represented by a four-dimensional vector for each class in one hot-coded vector. The classes are ferromagnetic, Non-magnetic, ferrimagnetic, and Antiferromagnetic.

If we can infer the chemical properties of materials by using their known features, then we will save a lot of money, time, and resources on experiments to find out such properties as magnetic ordering, hence the significance for such models.

The attempted solution and the rationale for choosing it:

A deep learning model was used to take the dataset and use the input features to predict the magnetic ordering. The desired NN architecture used was the traditional Feedforward Neural Networks and the data was encoded in a way to support this architecture. However, at the beginning of the project, I decided that the best way to model chemical compounds is by using Networks or graphs where atoms are nodes and bonds are edges and the NN architecture that can work are GNNs or Graph Neural Networks as well as CGNs or convolutional Graph Networks which are generalizations of CNNs or convolutional NNs as grids in the picture are subset of graphs.

Of course, I started with this approach but I encountered many problems.

Firstly, I am not familiar with CGNs or convolutional graph networks and the mathematics behind them and how to analyze the error or construct a cost function, so learning this and then applying it will take a very long time. Secondly, I tried to pre-process my data set by encoding each chemical compound with a graph using some library algorithms used by chemists, however, even by using Google Colab Pro, a paid cloud computing service, it took a few hours to transform the chemical formulae into a set of graphs and that is only pre-processing, and I thought training and training multiple times to fine-tune the hyperparameters will take lots of computation and more time.

So because of those two reasons above and being short of time, i switched to traditional NNs and used one hot encoding as vectors for the feature inputs instead of graphs.

The model details are as follows:

1. Parameters number: ~ 2,000,000
2. Number of layers 40 layers layers with SiLU activation
3. Number of nodes 256 for hidden layers
4. The number of nodes in the output layer is 4 with softmax activation
5. Google Colab Pro TPUs were used to train the model

It should work in theory for general deep learning solutions as we have an abundance of data and computing and good neural network algorithms and frameworks like Tensorflow. However, this specific solution had some issues which render it not functional or useful compared to other types of solutions.

It seems working with Traditional NNs is computationally expensive and does not actually learn the underlying patterns, and I do suspect (as I do not have direct evidence) that CGNs will work yet they seem to require too much computation during the training process, hence this explains why in the literature researchers are using:

1. Machine learning (Not NNs) like trees and so on.
2. They are trying to limit the domain and scope of their model in order to increase its utility by specifying the type of materials they want to classify like being made from specific atoms being organic or inorganic and so on.

I did not do the above as the scope of my model includes all materials and I used feedforward NNs which would require way more data and input features (more than just formula + density) and would not use locality in understanding molecular properties as opposed to CGNs.

The mistakes, insights learned and how the project could be approached differently:

The mistake of this NN solution is that the model is so huge with 2 million parameters and 40 layers with 256 nodes yet because the dataset was skewed most materials were Non-magnetic so it made the model have an incentive to predict all materials as non-magnetic and it got high categorical accuracy. So the model was reduced to a simple rule that assumes that most chemicals are non-magnetic which is kind of true but it does not do what it was designed for which is to predict the materials that are actually magnetic and predict what kind of magnetism they possess. So the algorithm rendered itself equivalent to a simple print statement with "non-magnetic" as an argument. This of course is a huge waste of computation and effort and a very basic understanding of the data before building the model will prevent such issues easily by choosing a target that penalizes the strategy used which is to assume most materials are not magnetic.

The dataset should be analyzed and understood before starting the Deep Learning and model-building part. If the dataset was known to be skewed then a different target would be used instead of categorical accuracy. Instead, something like precision and recall or a combination of both like the F1 score will be used to penalize this behaviour. Another thing that will be done if a more rigorous error analysis. Instead of relying on bias and variance errors, it is also advisable to use manual error analysis where 100 random data points will be chosen to see why they were misclassified. If I had done that and realized that all predictions were non-magnetic, then I would have changed my approach instead of increasing the number of layers and nodes and compute time.

A mistake made during fine-tuning is that to eliminate bias I tried to increase the complexity of the model as in the number of layers, nodes, and epochs, however, a way to increase the complexity to eliminate error is to increase the number of features in the input data as the features were available. Sometimes improving the data is way more effective than improving the algorithm.

Data-centric not Algorithm centric: How the time was allocated to each task:

It seems that more time was spent on Data cleaning and preprocessing than it was anticipated. It should not be surprising in retrospect as deep learning is data-centric, not algorithm-centric.

More than 60% of the time was used to pre-process the data and encode the molecular fingerprints into a format that can be used by the NNs. I planned that it would take less than 10% of the time and in retrospect and in other future projects I will anticipate that data analysis and preprocessing will take the majority of the time.

Conclusion and Insights:

In conclusion, this project underscored that simply using a large deep neural network for an imbalanced classification task can lead to poor outcomes. More specialized architectures, like Graph Neural Networks, may better capture local structure but demand deeper expertise and higher computational resources. Crucially, early data inspection would have revealed the significant “Non-magnetic” bias in the dataset and emphasized the need for more suitable metrics than just accuracy. Examining misclassified examples and enhancing input features—instead of just adding layers—are more productive strategies. Overall, the process highlighted that machine learning is fundamentally guided by human decisions around data quality, feature engineering, and metric selection. Most of the project’s time was spent on data-related tasks, reinforcing how vital thorough preprocessing and analysis are for success.