Armaan Shivpuri
662044570
shivpa@rpi.edu

Introduction

Neural Networks play a pivotal role in machine learning. They are highly adaptable and capable of accommodating diverse data types. Their ability to handle large datasets and complex problems positions them as essential tools for addressing contemporary data analysis challenges in fields like healthcare, finance, and marketing.

A neural network consists of nodes sequenced into an input layer, hidden layers, and an output layer. Each layer consists of nodes that establish connections with one another, characterized by specific weights and thresholds. When the output of an individual node surpasses a predefined threshold, it becomes activated, forwarding data to the subsequent layer in the network. Although initially training neural networks is time consuming, extensive amounts of training can improve its ability to analyze data. This allows us to perform large amounts of repetitive tasks quickly and efficiently.

Autoencoders are a type of neural network that is implemented in unsupervised learning. An autoencoder consists of an encoder, a latent space, and a decoder. The encoder takes the input data and compresses it into a lower dimensional representation. This representation is often referred to as the latent space or the bottleneck. The latent space captures the most essential features of input data provided by the encoder and disregards the less important features and sends them to the decoder. The decoder takes the reduced representation of the input data and expands it back to the size of the original representation. The primary goal of autoencoders is to minimize the difference between the input data and the data generated by the encoder and decoder. Achieving this allows it to capture significant patterns and features in a dataset.

Graphical Autoencoders leverage the synergies between Autoencoders and Graph Neural Networks (GNNs). While Autoencoders excel at learning compact representations of data, GNNs are tailored towards processing graph-structured data. This graph-structured data represents relationships through computational graphs. By integrating GNNs into the architecture of Autoencoders, Graphical Autoencoders adeptly capture intricate dependencies within graph datasets, which are notoriously challenging for traditional deep learning models. This combination allows for the extraction of meaningful features from complex graphs, facilitating tasks such as node classification, link prediction, and graph generation. Through efficient feature learning and representation, Graphical Autoencoders offer a powerful framework for analyzing and understanding graph-structured data, unlocking insights in various domains ranging from social networks to molecular structures.

Methods

In the research project, my goal is to identify new candidate materials with desired properties to be utilized in hotter sections of aircraft jet engines. We will use graph representations of the material's crystal structure and molecular composition, and pass them through a deep learning-based graph autoencoder (GAE) to extract lower-dimensional features for each material. The lower dimensional representation is expected to contain various clusters, based on the material composition and structure, representing different materials with different desired properties. We will use this low-dimensional space to identify regions representing new materials with desired properties.

I utilized data from the Material Projects Database and the AFlow Database to explore material properties. For each database, I implemented four models: a Graph Convolutional Network (GCN), a Graph Attention Network (GAT), and two Graphical Autoencoders—each with either a GCN or GAT as the encoder. These models were trained using 5-fold cross-validation, and I continually refined them to enhance performance. This approach allowed me to delve into the effectiveness of different models and techniques in predicting material properties.

Towards the end of the research time period, Dr. Subrato and I collaborated on applying these models to the team's custom dataset. Dr. Subrato provided me with the raw data, and I wrote an algorithm that took this raw data and transformed it into the formatt needed to use my Graphical Autoencoder Models. Although these models are yet to be developed and fine tuned, we managed to begin the process of applying the models used on the Materials Project and Aflow Datasets.

Results - Materials Project Dataset

Figure 1:

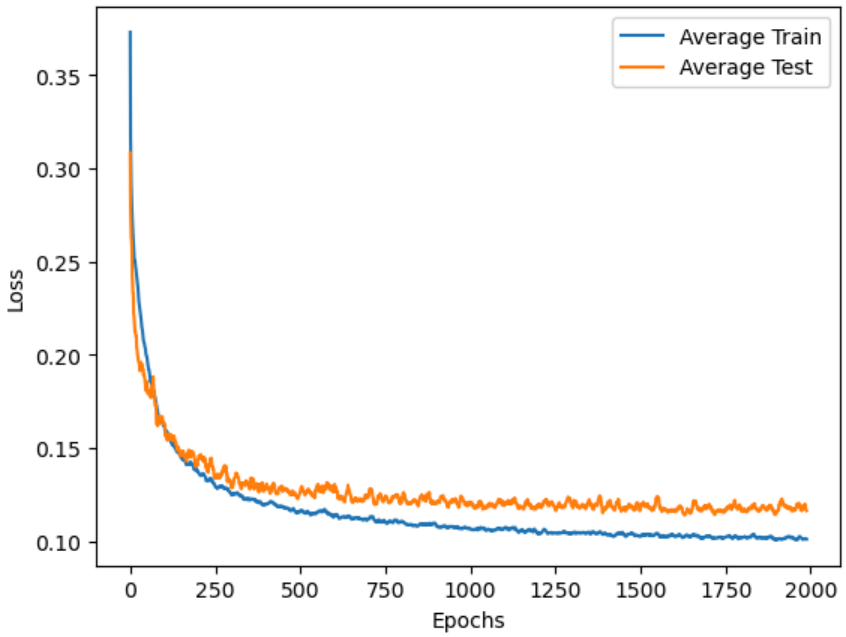


Figure 2:

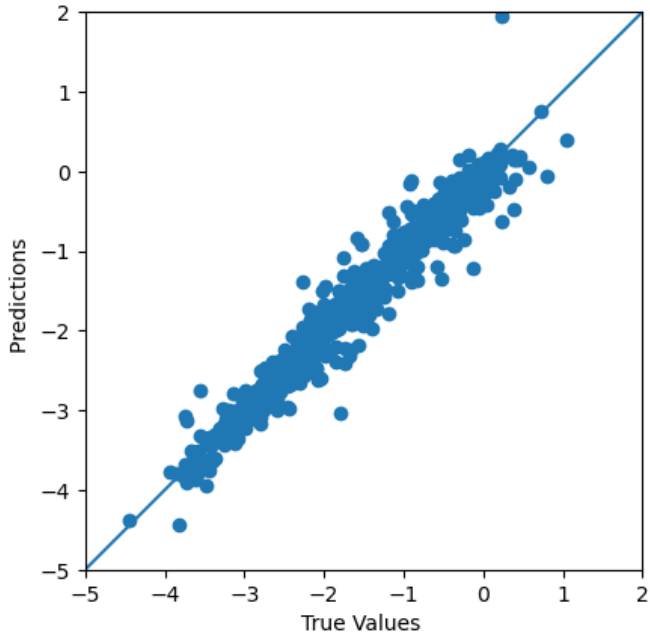


Figure 3:

```
GCN(
  (conv1): GraphConv(11, 64)
  (conv2): GraphConv(64, 64)
  (conv3): GraphConv(64, 64)
  (conv4): GraphConv(64, 64)
  (conv5): GraphConv(64, 64)
  (bn1): BatchNorm(64)
  (bn2): BatchNorm(64)
  (bn3): BatchNorm(64)
  (bn4): BatchNorm(64)
  (lin1): Linear(in_features=64, out_features=4, bias=True)
  (lin2): Linear(in_features=4, out_features=1, bias=True)
)
```

Here are the results of the standard GCN Model for the Material Projects Dataset. The Train Loss was 0.101, the Test Loss was 0.116, and the R^2 was 0.971.

Figure 4:

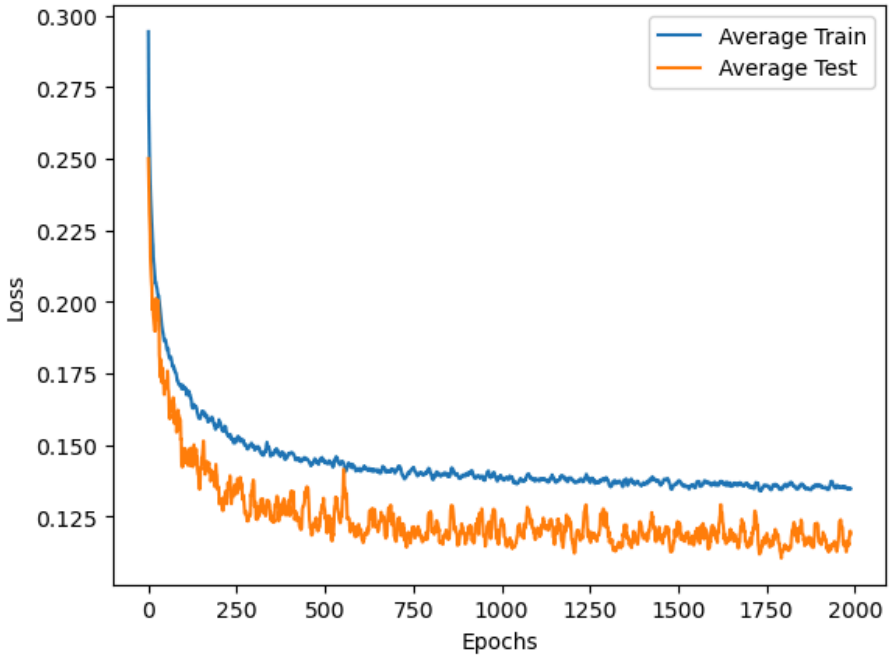


Figure 5:

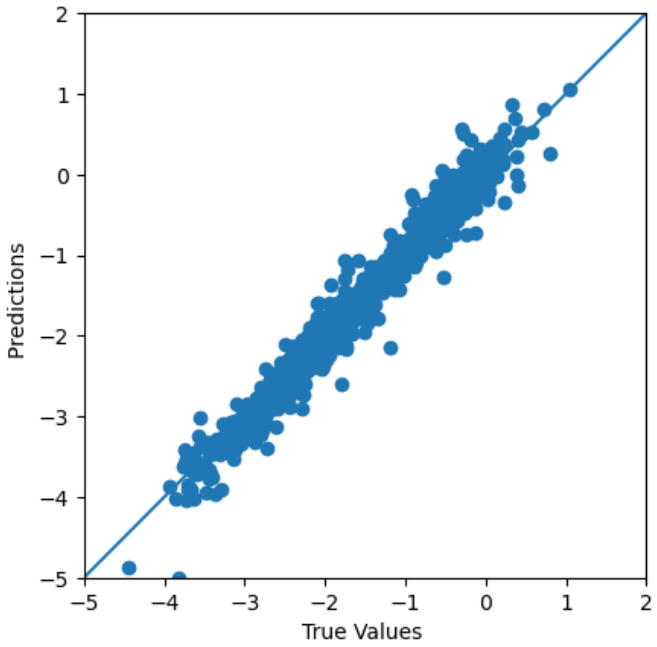


Figure 6:

```
GAE(
  (encoder): GCN(
    (conv1): GraphConv(11, 128)
    (conv2): GraphConv(128, 128)
    (conv3): GraphConv(128, 128)
    (bn1): BatchNorm(128)
    (bn2): BatchNorm(128)
    (lin1): Linear(in_features=128, out_features=16, bias=True)
    (lin2): Linear(in_features=16, out_features=4, bias=True)
    (lin3): Linear(in_features=4, out_features=1, bias=True)
  )
  (decoder): InnerProductDecoder()
)
```

Here are the results for the Graphical Autoencoder with a GCN encoder for the Material Projects Dataset. The Train Loss was 0.135, the Test Loss was 0.119, and the R^2 was 0.962.

Figure 7:

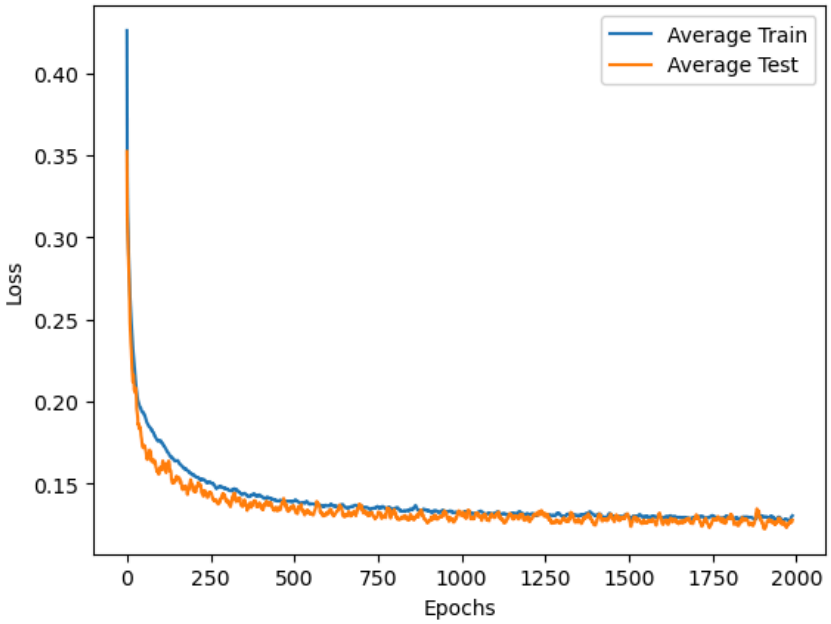


Figure 8:

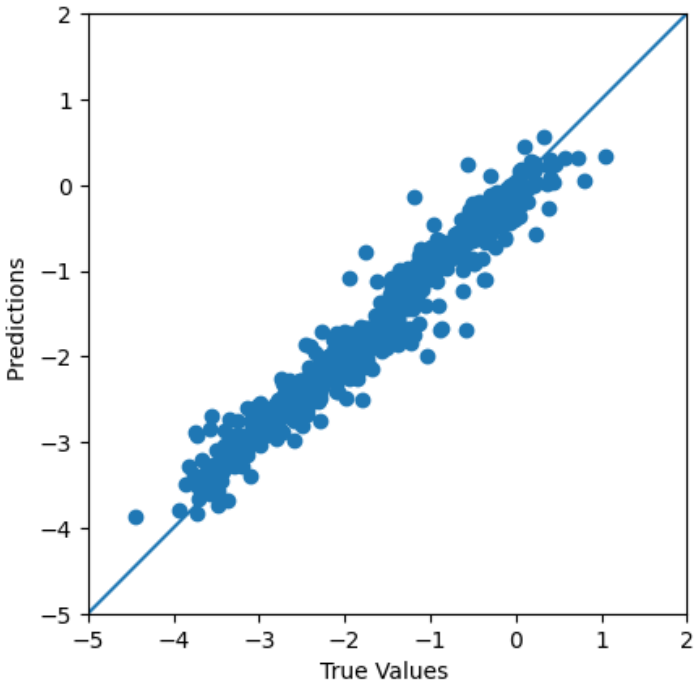


Figure 9:

```
GAT(
  (gat1): GATv2Conv(11, 64, heads=4)
  (gat2): GATv2Conv(256, 64, heads=4)
  (gat3): GATv2Conv(256, 64, heads=1)
  (bn1): BatchNorm(256)
  (bn2): BatchNorm(256)
  (lin1): Linear(in_features=64, out_features=2, bias=True)
  (lin2): Linear(in_features=2, out_features=1, bias=True)
)
```

Here are the results of the Graph Attention Network for the Material Projects Dataset. The Train Loss was 0.130, the Test loss was 0.128, and the R^2 was 0.968.

Figure 10:

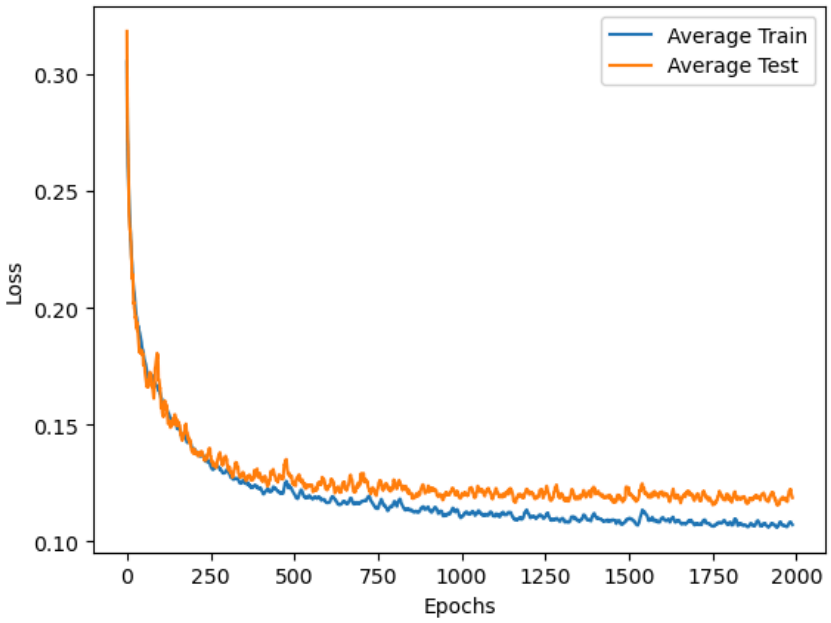


Figure 11:

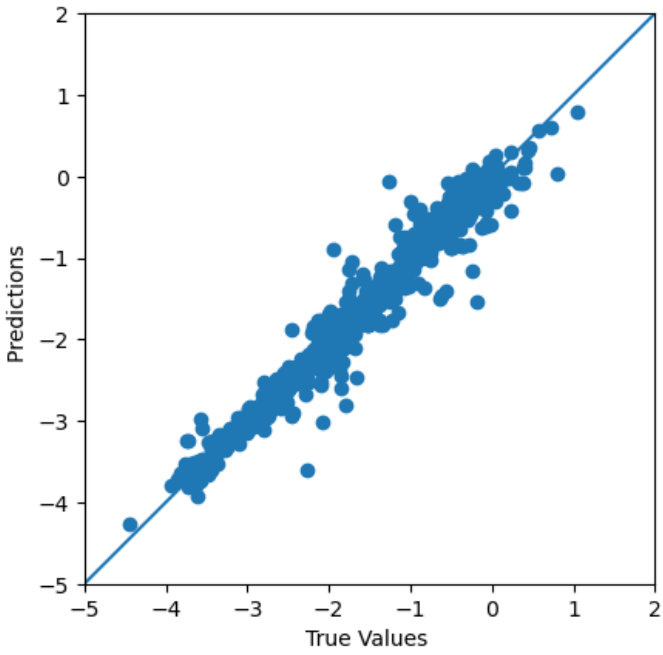


Figure 12:

```
GAE(
  (encoder): GAT(
    (gat1): GATv2Conv(11, 64, heads=4)
    (gat2): GATv2Conv(256, 64, heads=4)
    (gat3): GATv2Conv(256, 64, heads=4)
    (gat4): GATv2Conv(256, 64, heads=4)
    (gat5): GATv2Conv(256, 64, heads=4)
    (gat6): GATv2Conv(256, 64, heads=1)
    (bn1): BatchNorm(256)
    (bn2): BatchNorm(256)
    (bn3): BatchNorm(256)
    (bn4): BatchNorm(256)
    (bn5): BatchNorm(256)
    (lin1): Linear(in_features=64, out_features=2, bias=True)
    (lin2): Linear(in_features=2, out_features=1, bias=True)
  )
  (decoder): InnerProductDecoder()
)
```

Here are the results of the Graphical Autoencoder with a Graph Attention Network as the Encoder for the Material Projects Dataset. The Train Loss was 0.107, the Test loss was 0.0.119, and the R^2 was 0.968.

Results - AFlow Dataset

Figure 13:

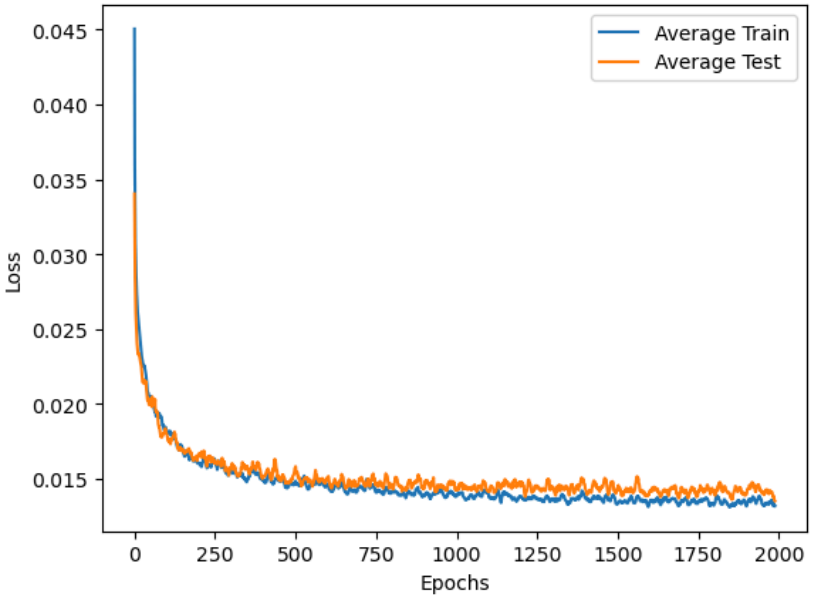


Figure 14:

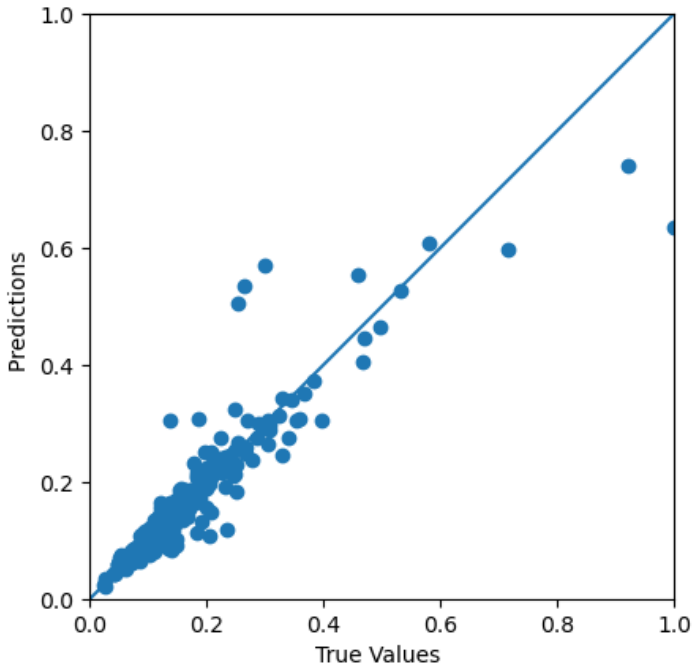


Figure 15:

```
GCN(
  (conv1): GraphConv(11, 64)
  (conv2): GraphConv(64, 64)
  (conv3): GraphConv(64, 64)
  (bn1): BatchNorm(64)
  (bn2): BatchNorm(64)
  (bn3): BatchNorm(64)
  (lin1): Linear(in_features=64, out_features=4, bias=True)
  (lin2): Linear(in_features=4, out_features=1, bias=True)
)
```

Here are the results of the GCN Model for the AFlow Dataset. The Train Loss was 0.013, the Test Loss was 0.014, and the R^2 was 0.881.

Figure 16:

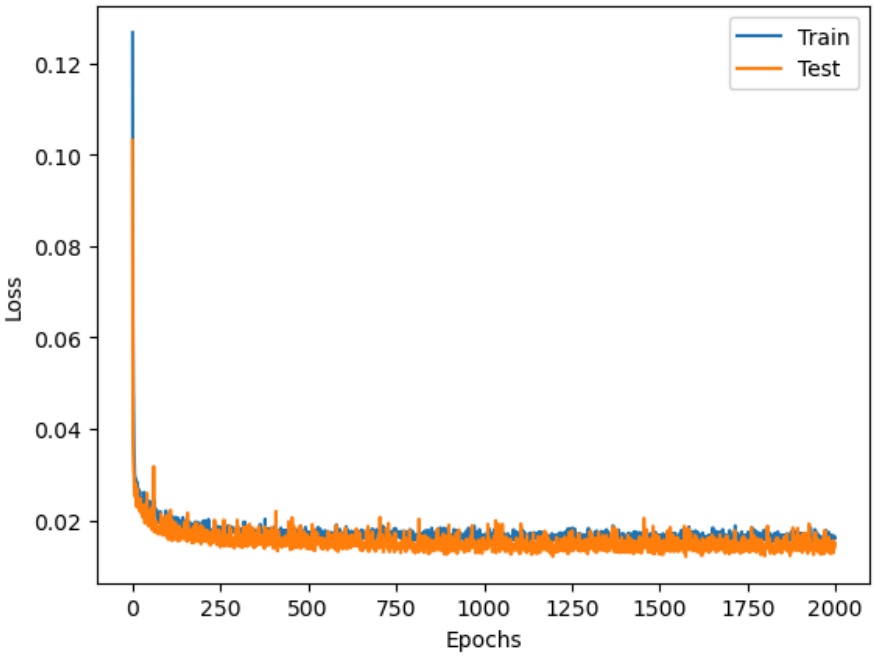


Figure 17:

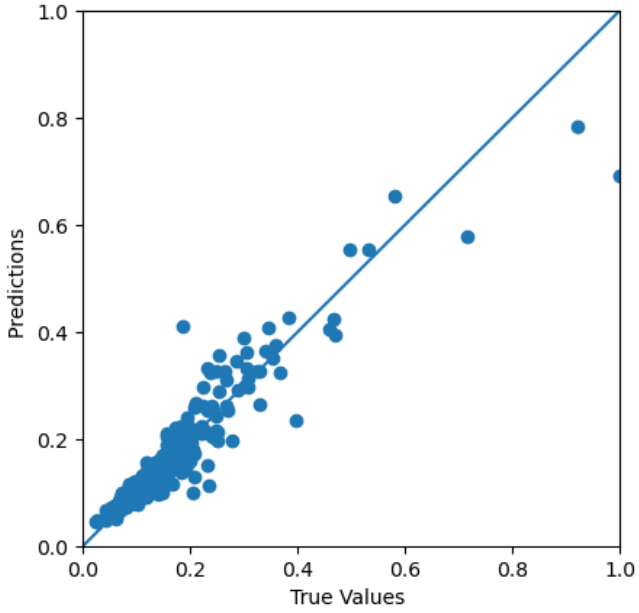


Figure 18:

```
GAE(
  (encoder): GCN(
    (conv1): GraphConv(11, 64)
    (conv2): GraphConv(64, 64)
    (bn1): BatchNorm(64)
    (lin1): Linear(in_features=64, out_features=4, bias=True)
    (lin2): Linear(in_features=4, out_features=1, bias=True)
  )
  (decoder): InnerProductDecoder()
)
```

Here are the results for the Graphical Autoencoder with a GCN encoder for the AFlow Dataset. The Train Loss was 0.013, the Test Loss was 0.014, and the R^2 was 0.881.

Figure 19:

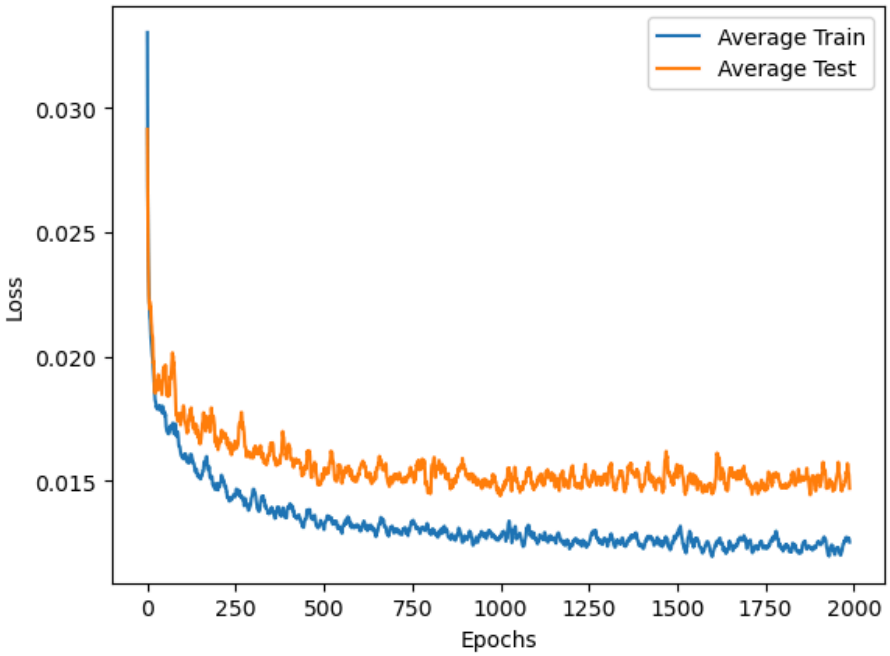


Figure 20:

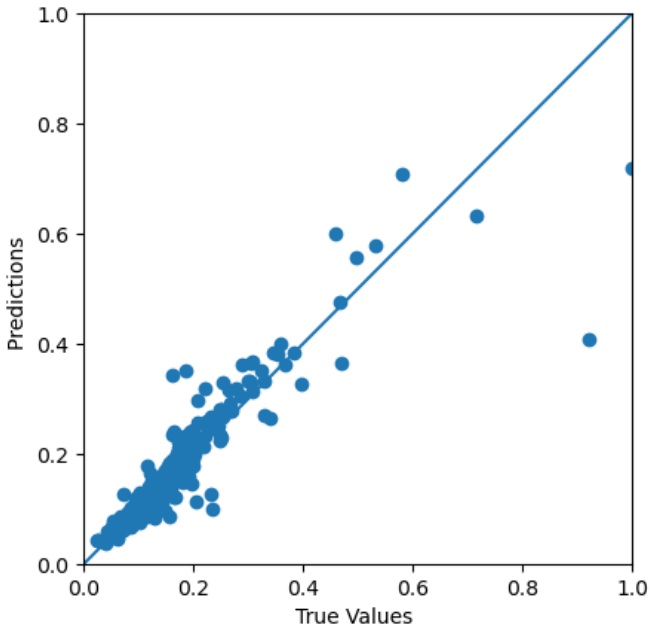


Figure 21:

```
GAT(
  (gat1): GATv2Conv(11, 64, heads=4)
  (gat2): GATv2Conv(256, 64, heads=4)
  (gat3): GATv2Conv(256, 64, heads=4)
  (gat4): GATv2Conv(256, 64, heads=4)
  (gat5): GATv2Conv(256, 64, heads=4)
  (gat6): GATv2Conv(256, 64, heads=4)
  (gat7): GATv2Conv(256, 64, heads=1)
  (bn1): BatchNorm(256)
  (bn2): BatchNorm(256)
  (bn3): BatchNorm(256)
  (bn4): BatchNorm(256)
  (bn5): BatchNorm(256)
  (bn6): BatchNorm(256)
  (lin1): Linear(in_features=64, out_features=2, bias=True)
  (lin2): Linear(in_features=2, out_features=1, bias=True)
)
```

Here are the results of the Graph Attention Network for the AFlow Dataset. The Train Loss was 0.013, the Test loss was 0.015, and the R^2 was 0.896.

Figure 22:

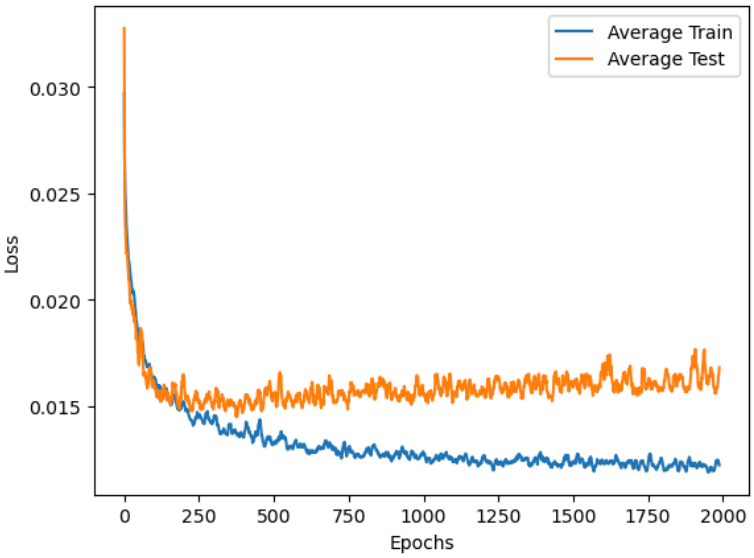


Figure 23:

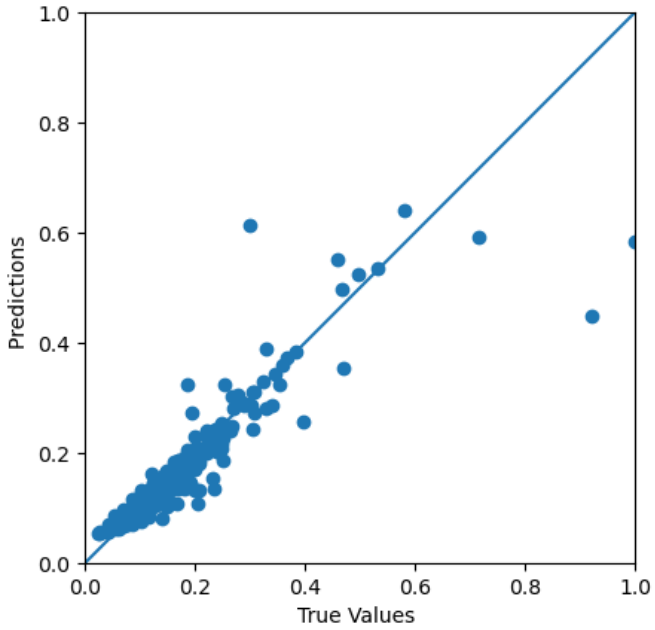


Figure 24:

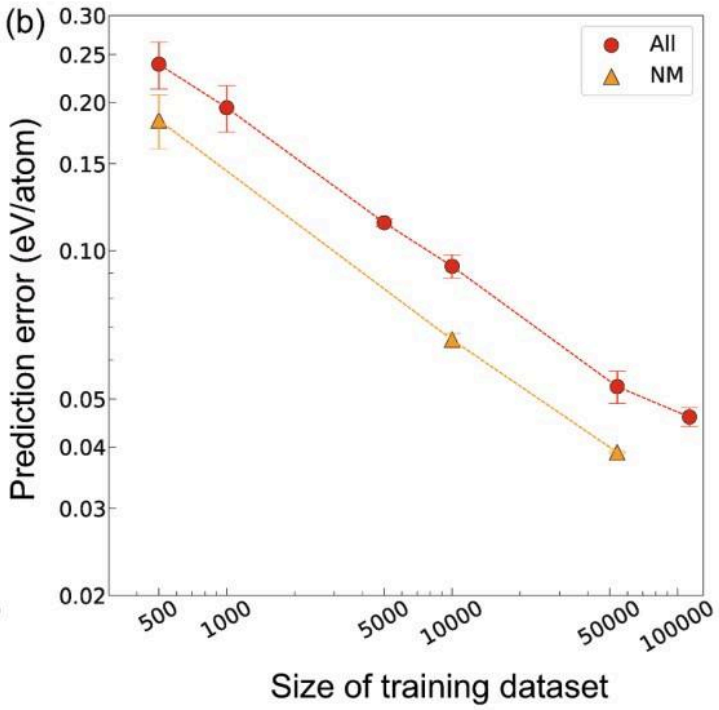
```
GAE(
  (encoder): GAT(
    (gat1): GATv2Conv(11, 128, heads=2)
    (gat2): GATv2Conv(256, 128, heads=1)
    (bn1): BatchNorm(256)
    (lin1): Linear(in_features=128, out_features=32, bias=True)
    (lin2): Linear(in_features=32, out_features=8, bias=True)
    (lin3): Linear(in_features=8, out_features=1, bias=True)
  )
  (decoder): InnerProductDecoder()
)
```

Here are the results of the Graphical Autoencoder with a Graph Attention Network as the Encoder for the AFlow Dataset. The Train Loss was 0.011, the Test loss was 0.013, and the R^2 was 0.910.

Analysis - Materials Project Dataset

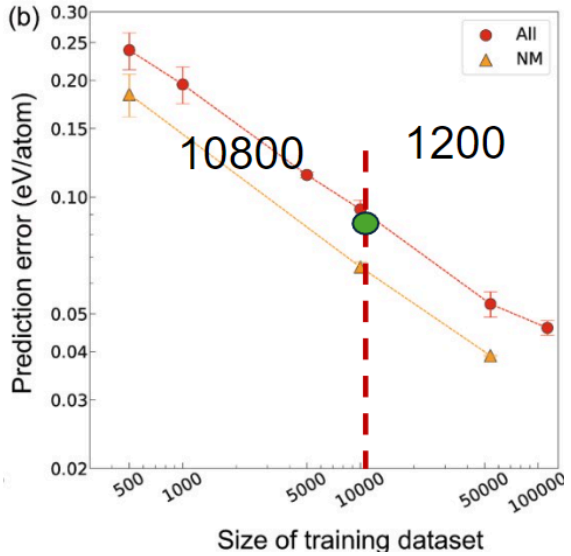
For the Materials Project Dataset, we used the following reference from a research paper titled Transfer Learning for materials informatics using crystal graph convolutional neural network by Joowi Lee and Ryoji Asahi. In this graph, the red line represents the expected Train Loss for a dataset of a certain size.

Figure 25:



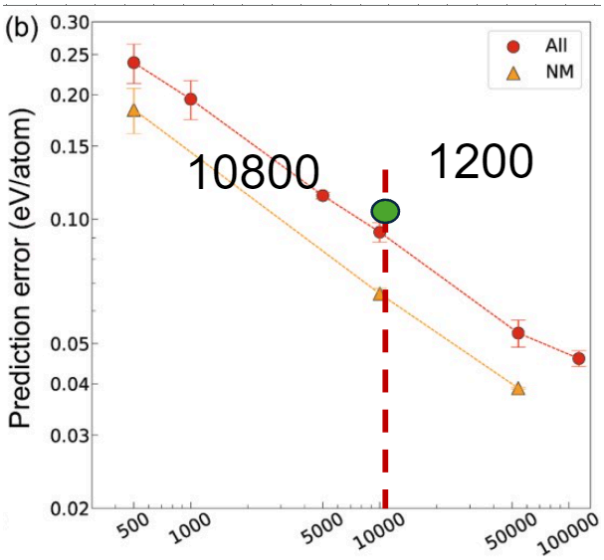
Below are the 4 comparisons using Figure 25 above for each Model. In each graph below, the green dot represents the value of the Train Loss of the corresponding models. The expected value for our models is represented by the interaction between the vertical dotted red line and the solid red diagonal line.

Figure 26:



This is the comparison for the GCN Model for the Materials Project Dataset. As you can see, the Train Loss of 0.101 is lower than the expected value.

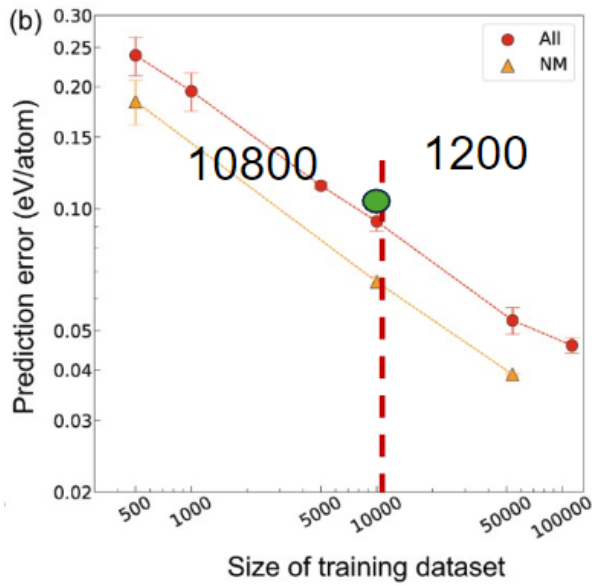
Figure 27:



This is the comparison of the GAE with the GCN encoder for the Materials Project Dataset. As you can see, the train loss of 0.136 is slightly higher than expected.

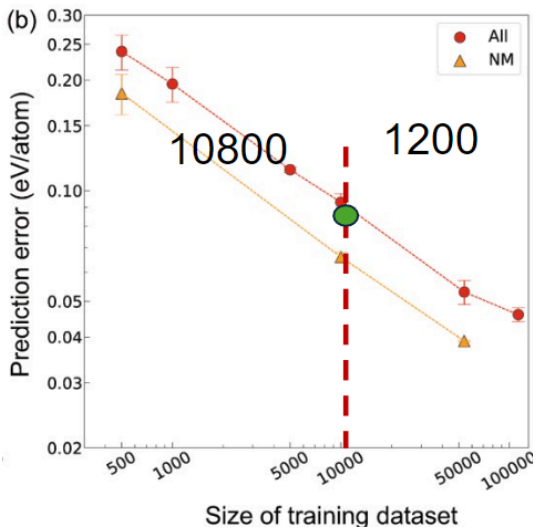
When comparing the GCN and GAE with the GCN encoder models, we notice that there is an overall decrease in performance. This was unexpected, as Autoencoders are expected to improve the performance.

Figure 28:



This is the comparison for the GAT Model for the Materials Project Dataset. As you can see, the Train Loss of 0.130 is slightly higher than the expected value.

Figure 29:



This is the comparison for the GAE with the GAT encoder for the Materials Project Dataset. As you can see, the Train Loss of 0.130 is slightly higher than the expected value.

When comparing the GAT and GAE with the GAT encoder models, we observe that there is a slight decrease in the Train Loss, which implies that there is a slight increase in performance. Although there is some improvement made when using the GAE with the GAT Encoder Model, we expected a more noticeable difference in performance.

Analysis - AFlow Dataset

Unlike the Materials Dataset, the AFlow Dataset does not have any references available. However, we can still compare the GCN model with the GAE with the GCN encoder model and the GAT model with the GAE with the GAT encoder model. When comparing the GCN model with the GAE with the GCN encoder model, we notice that there is no difference in the Train and Test Losses. This was unexpected, as Autoencoders are expected to improve the performance. When comparing the GAT model with the GAE with the GAT encoder model, there is a slight increase in performance. Although there is some improvement made when using the GAE with the GAT Encoder Model, we expected a more noticeable difference in performance.

Next Steps

As mentioned before, I still have to fine tune and develop the 4 models for the custom dataset provided by Dr. Subrato and his team. My goal for future semesters is to continue working with this team and continue this project. I also plan on further assessing why there was little to no performance for the Materials Project Dataset and the AFlow Dataset and use these findings to assess the performance of the custom dataset.