Armaan Shivpuri
4/23/2024

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.

Like traditional neural networks, Convolutional Neural Networks (CNNs) are organized with nodes ordered into strata consisting of an initial layer where information is received, subsequent concealed layers, and a concluding layer where the results are yielded. Nodes within each layer establish connections with specific weights and thresholds, facilitating information flow. However, CNNs implement convolutional layers. Using convolutional layers allows CNNs to efficiently analyze defining features in data such as images. These layers employ filters to extract patterns, enhancing the network's capability to recognize complex structures. The integration of convolutional layers is essential, as it allows the neural network to effectively capture hierarchical representations within the data, making CNNs particularly adept at image recognition tasks. The ability of CNNs to learn and analyze spatial relationships significantly accelerates the processing of large datasets.

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.

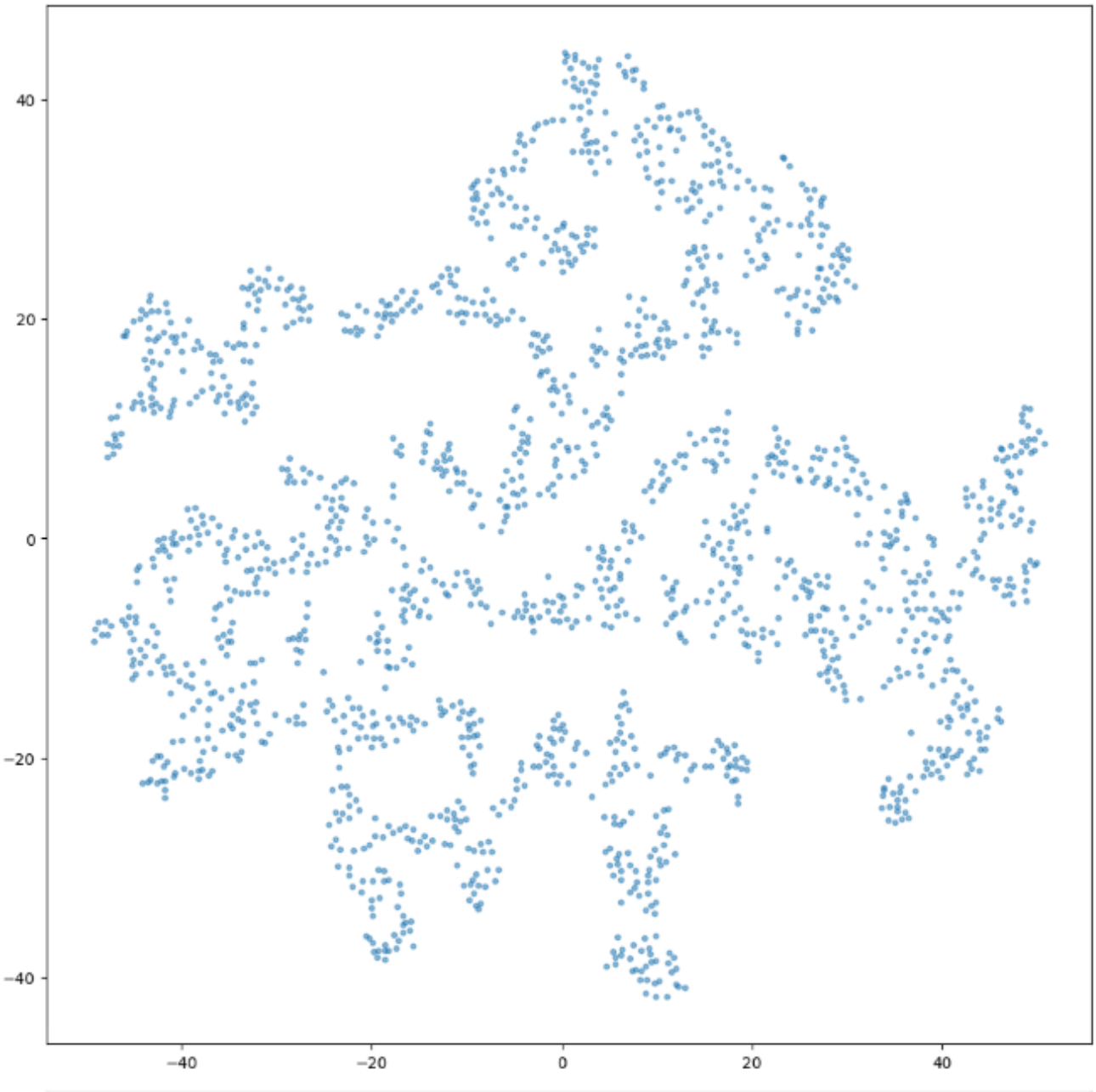
Graphical Autoencoders leverage the synergies between Autoencoders and Graph Neural Networks (GNNs). While Autoencoders excel at learning compact representations of data, GNNs are tailored towards processing graph-structured data. This graph-structured data represents relationships through computational 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 my research, I was tasked with optimizing the performance and reliability of materials utilized in aircraft engines based on 15 distinct characteristics. My goal was to develop an Autoencoder model in Keras to construct one with TensorFlow. I worked directly with images of graphs as a way of working directly with visual representations. I leveraged Convolutional Neural Networks (CNNs) in both the encoder and decoder components of the Autoencoder. By incorporating CNNs into the model, I aimed to capture intricate patterns and features present in the images, providing a more nuanced and context-aware compression and reconstruction of the graphical data. This shift underscored a more image-centric strategy, aligned with the inherent characteristics of the dataset and optimized the Autoencoder for visual information extraction. I also utilized the Metadata gathered in the updated data set to verify if any patterns emerged. The Metadata labeled each value in the dataset of an integer between, 1 and 4 inclusive. It is also separated into 3 different columns, each with varying values for the same value.

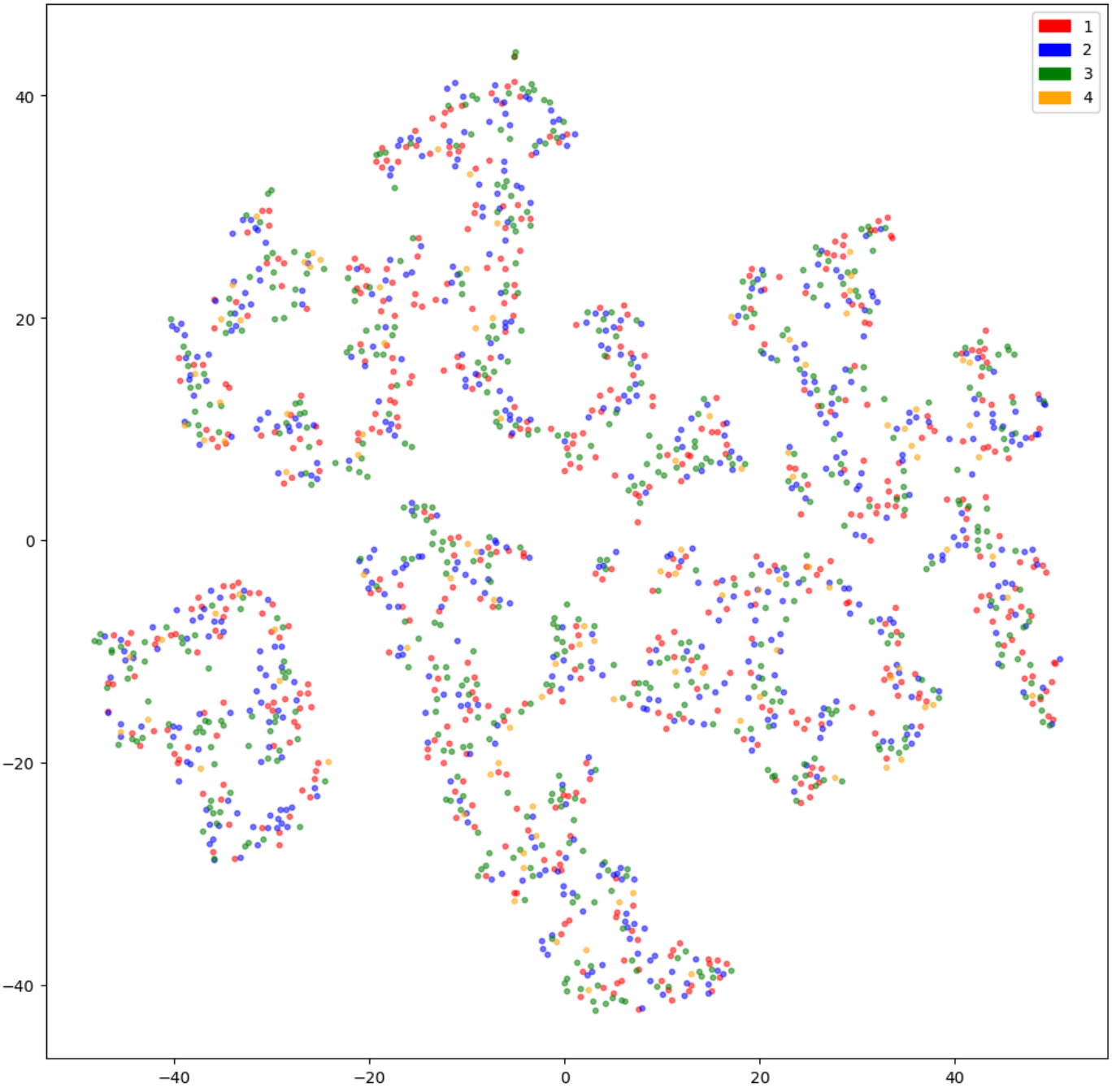
Results

Figure 1:



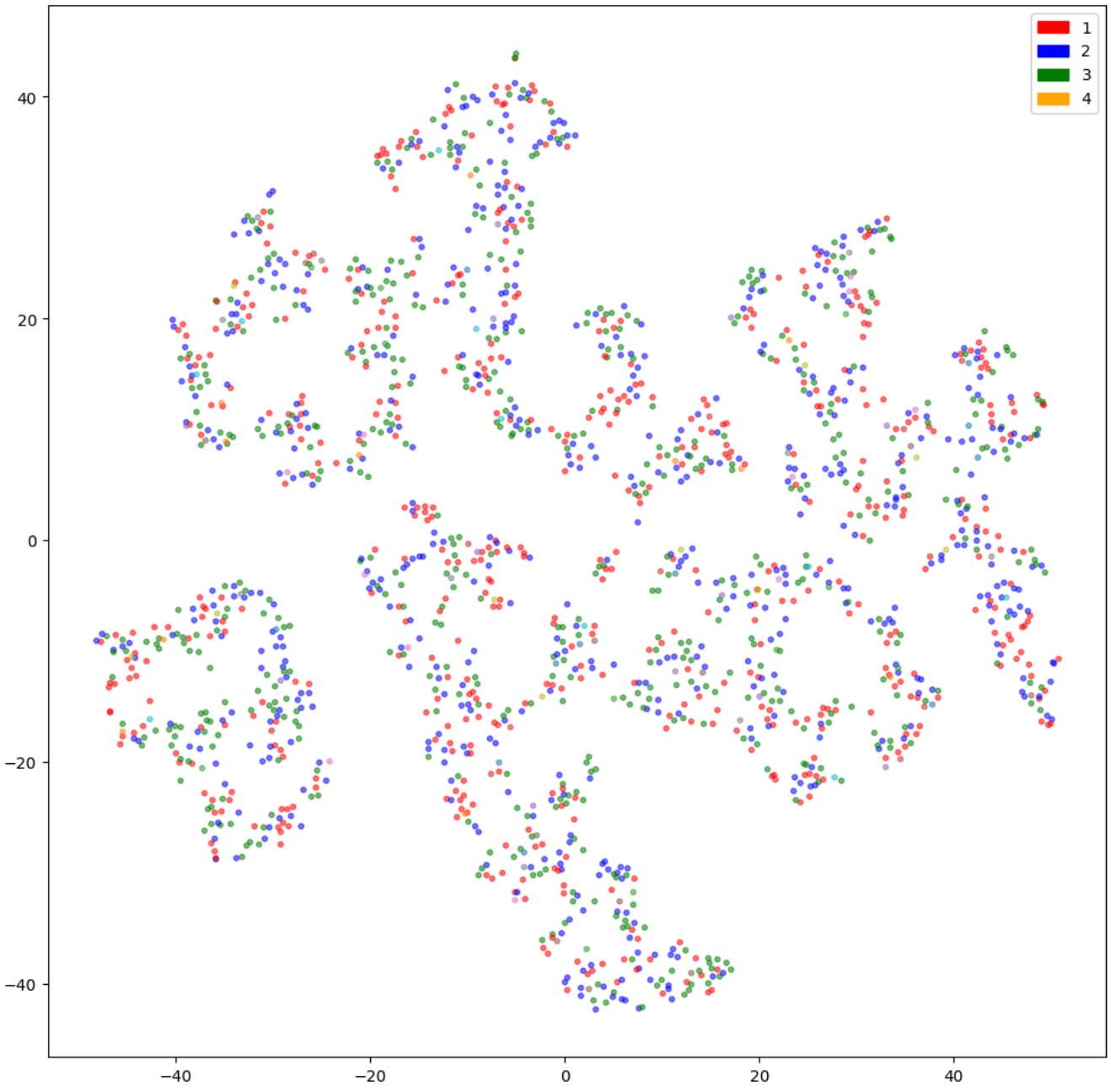
This graph displays the clusters that are created when running the dataset through the Autoencoder. Picking random spots in the gathered groups lets us make new micro-structures, each one showing a different dataset in the learned hidden space. These picked spots act like starting points, helping create lots of unique micro-structures that display the information stored by the autoencoder within the hidden layers.

Figure 2:



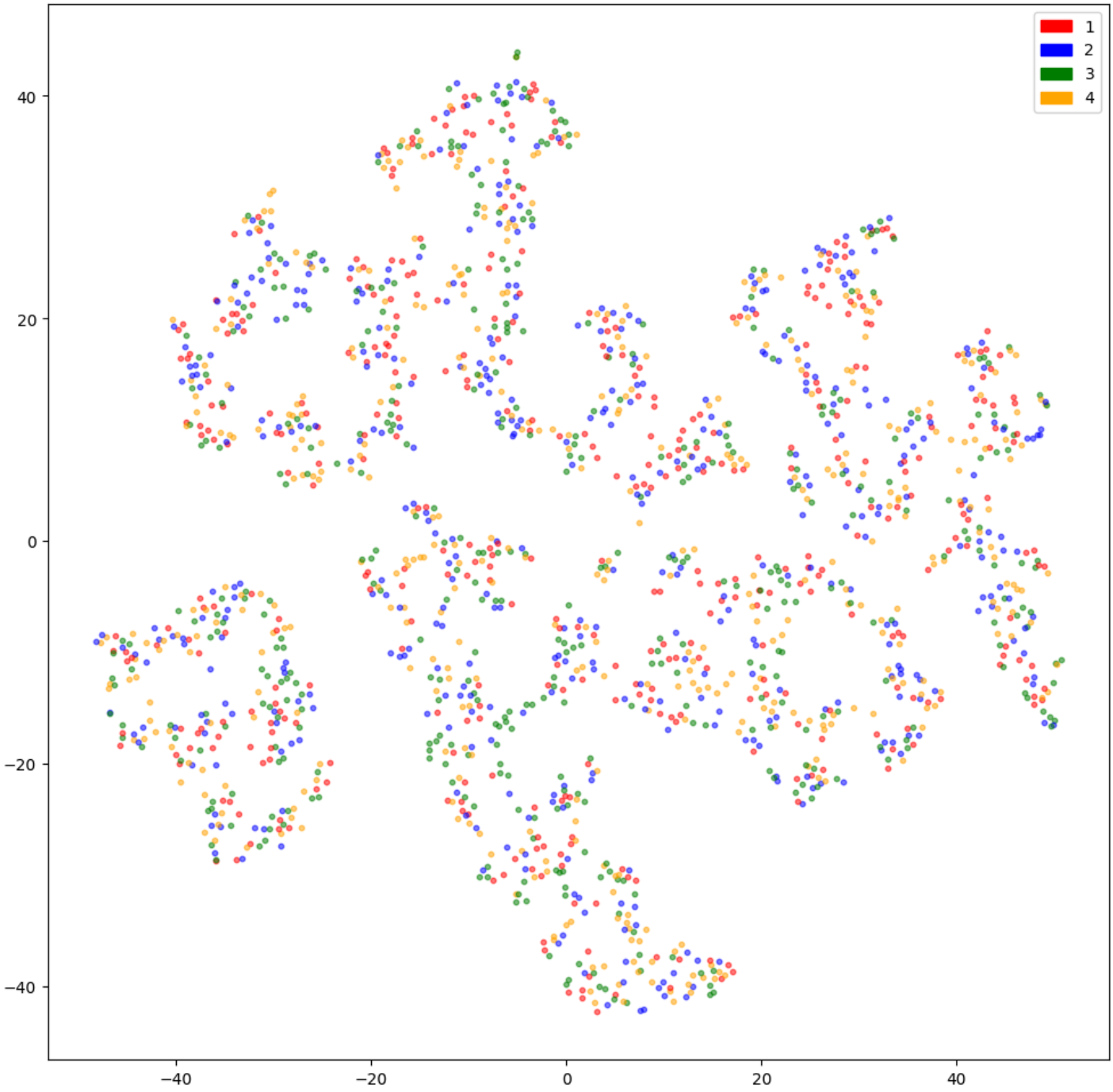
This graph displays the same values as Figure 1, but with the coloring system of Column 1 of the Metadata file.

Figure 3:



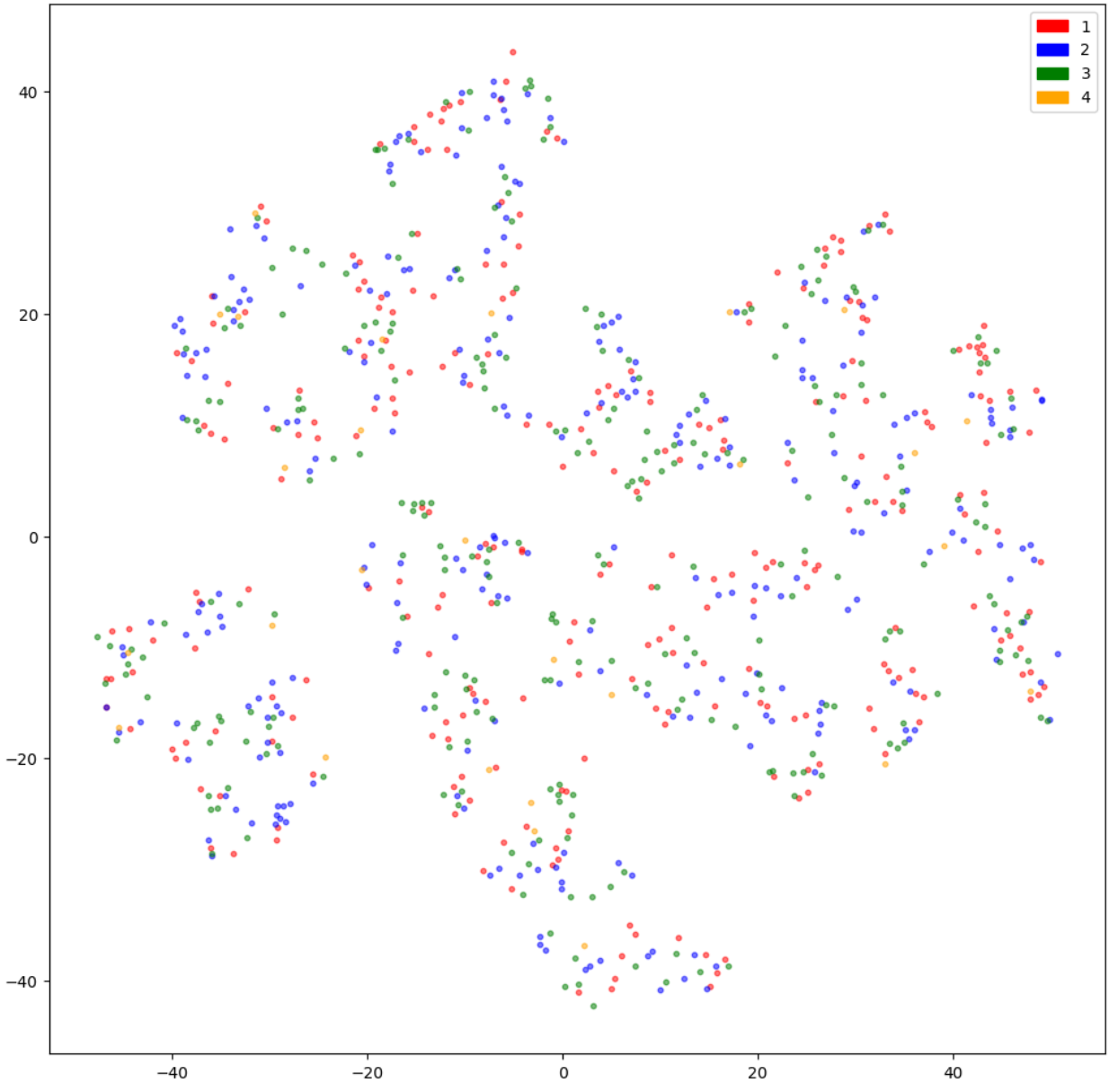
This graph displays the same values as Figure 1, but with the coloring system of Column 2 of the Metadata file.

Figure 4



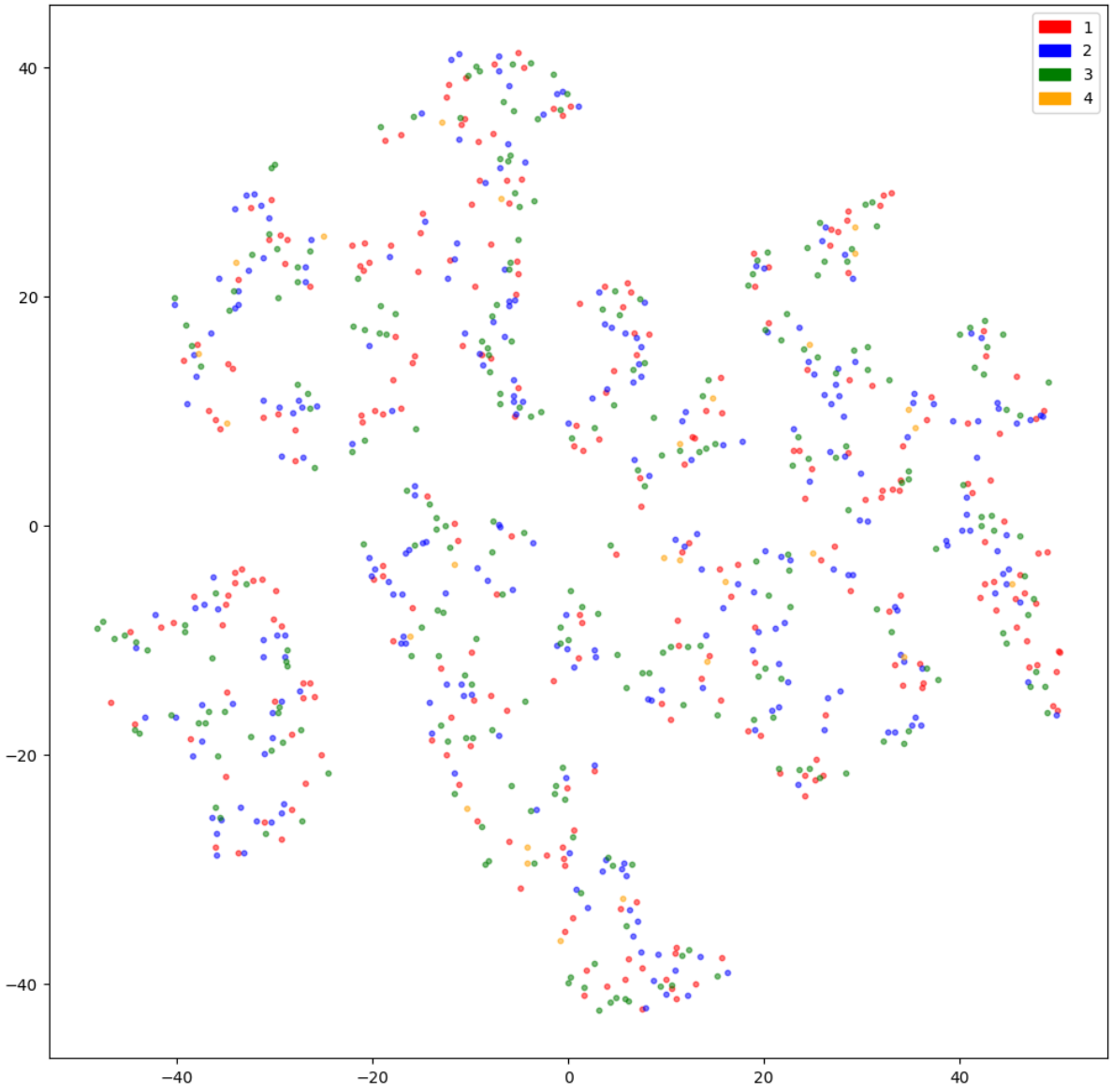
This graph displays the same values as Figure 1, but with the coloring system of Column 3 of the Metadata file.

Figure 5:



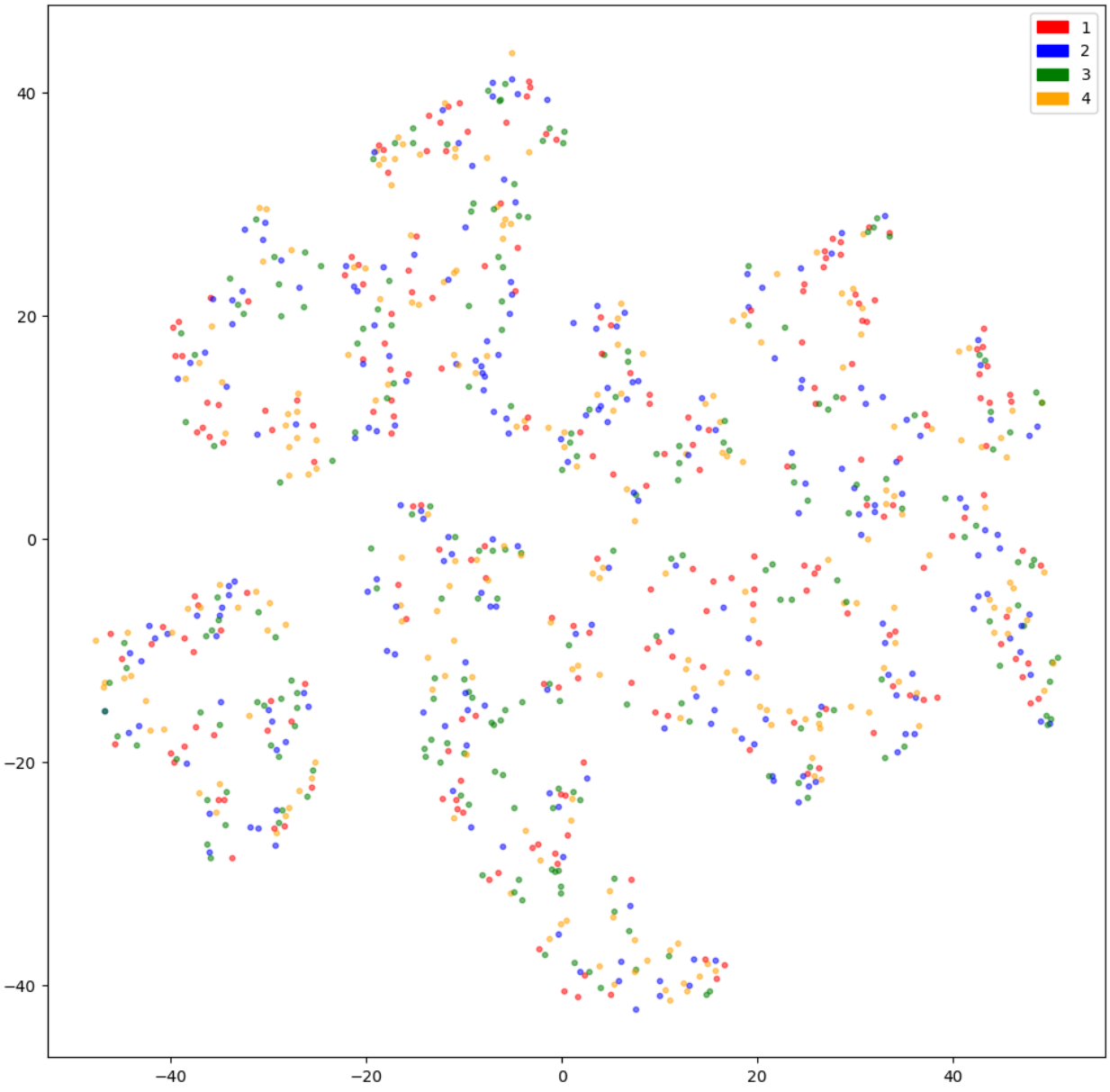
This graph displays the same values as Figure 1, but filtered by all of the values in Columns 2 and Column 3 marked with a 1 and the coloring of Column 1.

Figure 6:



This graph displays the same values as Figure 1, but filtered by all of the values in Columns 2 and Column 3 marked with a 2 and the coloring of Column 1.

Figure 7:



This graph displays the same values as Figure 1, but filtered by all of the values in Columns 1 and Column 2 marked with a 1 and the coloring of Column 2.

Analysis

After reviewing figures 1-4 and creating figures 5-7 as attempts to find any level of clustering, I was unable to find any observable clustering. This lack of clustering indicates the need for further exploration and refinement of analytical techniques. This includes potentially finding new Neural Network Models to implement when analyzing the data.

GNN Research

In my recent research, I've been diving deep into Graph Neural Networks (GNNs). My focus has been on how these powerful models can be used to predict material properties based on crystal structure and compositions. This field has the potential to completely transform materials science by making predictions about material behavior and properties faster and more accurately.

At the core of my research is the concept of Graph Autoencoders (GAE), which are a specific type of GNN designed for encoding graph-structured data into a lower-dimensional latent space and then reconstructing the original graph from this representation. This idea is especially relevant in materials science, where we can represent crystal structures and compositions as graphs, with atoms or elements as nodes and bonds or interactions as edges.

To put this idea into practice, I've been using PyTorch, a popular deep learning framework in Python known for its flexibility and efficiency in building neural network architectures. PyTorch offers a wide range of tools that make it easy to create and train complex models, making it the perfect choice for developing GAEs for material property prediction.

In addition to PyTorch, I'm also utilizing other Python libraries like scikit-learn for tasks such as preprocessing data and evaluating model performance. These libraries work alongside PyTorch and support different parts of the research process, from preparing data to training models and analyzing results.

By combining GAEs with PyTorch and integrating these Python libraries, my goal is to create a concise yet meaningful representation of crystal structures and compositions. Ultimately, this representation will help us accurately predict material properties like never before. This approach has the potential to greatly advance materials science research by making materials discovery and design processes more efficient.

Next Steps

In our research, we have made progress by incorporating Graph Neural Networks (GNNs) into our material property prediction framework. Building on Graph Autoencoders (GAEs), we can now use GNNs to improve our understanding and prediction abilities even further. We represent the crystal structures and molecular compositions of materials as graphs. These graphs are then passed through a Graphical Autoencoder to extract important structural and compositional information in a more concise form. The extracted features are expected to form distinct clusters, with each cluster representing materials that share similar characteristics. By leveraging this compressed representation, we hope to identify specific areas in the feature space where new materials with desired properties may exist. This is a crucial step towards speeding up the process of discovering and designing new materials. This integration of GNNs into our framework is a significant advancement, highlighting our dedication to using machine learning and graph-based methods to transform the field of materials science.