MULTIMODAL MOLECULAR REPRESENTATION LEARNING FOR PROPERTY PREDICTION

by

Tristan Maughan

A senior thesis submitted to the faculty of

Brigham Young University - Idaho

in partial fulfillment of the requirements for the degree of

Bachelor of Science

Department of Physics

Brigham Young University - Idaho

December 2025

BRIGHAM YOUNG UNIVERSITY - IDAHO

DEPARTMENT APPROVAL

of a senior thesis submitted by

Tristan Maughan

This thesis has been reviewed by the research committee, senior thesis coordinator, and department chair and has been found to be satisfactory.

Date	David Oliphant, Advisor
Date	David Oliphant, Senior Thesis Coordinator
Date	Richard Datwyler, Committee Member
Date	Matt Zachreson, Committee Member
Date	Evan Hansen, Chair

ABSTRACT

MULTIMODAL MOLECULAR REPRESENTATION LEARNING FOR PROPERTY PREDICTION

Tristan Maughan

Department of Physics

Bachelor of Science

Artificial intelligence-based molecular representation learning may allow AI to accurately predict the chemical properties of molecules and thereby help facilitate drug discovery by reducing the need for physical experimentation. However, most existing representation learning approaches rely on only a single data-type to represent inputs such as molecular graphs, images, etc. Novel research on the use of multimodal models (using multiple data types) have already shown promise in improving AI's predictive abilities. Of the many data type combinations that exist, notably little research has been done on the unification of molecule-graphs and molecular text-descriptions. The purpose of this study is to investigate the hypothesis of whether an AI model that is trained on unified encodings of graph and text data outperforms AI models trained only on one of such data types. To date, this study remains in progress, and significant progress has been made only towards the devel-

opment of our E(n)-Equivariant Graph Neural Network-based graph encoder. This model is trained to predict molecular properties using a curated subset of the MolTextNet dataset. Evaluation of the encoder showcases its adequate predictive abilities, making for a strong checkpoint towards the completion of this study.

ACKNOWLEDGMENTS

Regarding the underlying research project from which the content of this thesis details, I thank my mentor, Professor Truong–Son Hy, for his example, guidance, and advice as I navigated the process of progressing through the project. From him, I learned many lessons, notably including how to optimally communicate my project to others in both formal and informal contexts. Additionally, I thank Dr. Hy's Ph. D student, Viet Thanh Duy Nguyen (I know him as Duy), for his strong example of how to make effective progress as a graduate student working on an appropriately difficult and relevant research project. Duy offered me significant practical advice from time to time that, undoubtedly, provided me with the resources and perspective I needed to overcome stumbling blocks in my efforts to progress in my research project.

Funding Acknowledgement: Support provided by National Science Foundation, Grant # DMR-244516 – Research Experiences for Undergraduates (REU) awarded to UAB

Contents

List of Figures

Chapter 1

Introduction

The field of drug discovery plays a crucial role in advancing global health. Through the combined efforts of researchers, clinicians, and engineers, vaccines, therapeutic compounds, and life-saving drugs have been successfully developed and deployed, improving the well-being of countless individuals. However, the process of discovering effective drugs is neither fast nor cheap. It typically involves years of iterative testing, with high attrition rates as candidate compounds fail due to poor pharmacological properties. As a result, researchers continue to explore ways to improve the efficiency and accuracy of this process.

In recent years, artificial intelligence (AI) has emerged as a promising tool for accelerating drug discovery. One major application is molecular property prediction, where AI models learn to forecast chemical properties of potential drug-candidates directly from molecular data. These models can significantly reduce development costs and time by filtering out unsuitable candidates early in the pipeline.

To date, most AI models for molecular property prediction have focused on **unimodal** learning, or learning from only a single type of data representation. For reference, examples of data types that AI neural networks are often designed to inter-

pret include human language, images, and many other forms of information. While effective, unimodal models are inherently limited by the scope of the representation provided by the singular data type that is being interpreted. Essentially, for any given subject, there is an unknown amount of useful information that could potentially be "understood" by an AI neural network, but the AI will be limited by how much of that information can be accurately represented by the data provided to the AI. Hypothetically, different forms of data on the same subject will likely excel in conveying different aspects of the nature of the subject, at least to some degree. It stands to reason, then, that an AI model might perform better in "understanding" a given subject if it can interpret and combine multiple differing data representations of that subject.

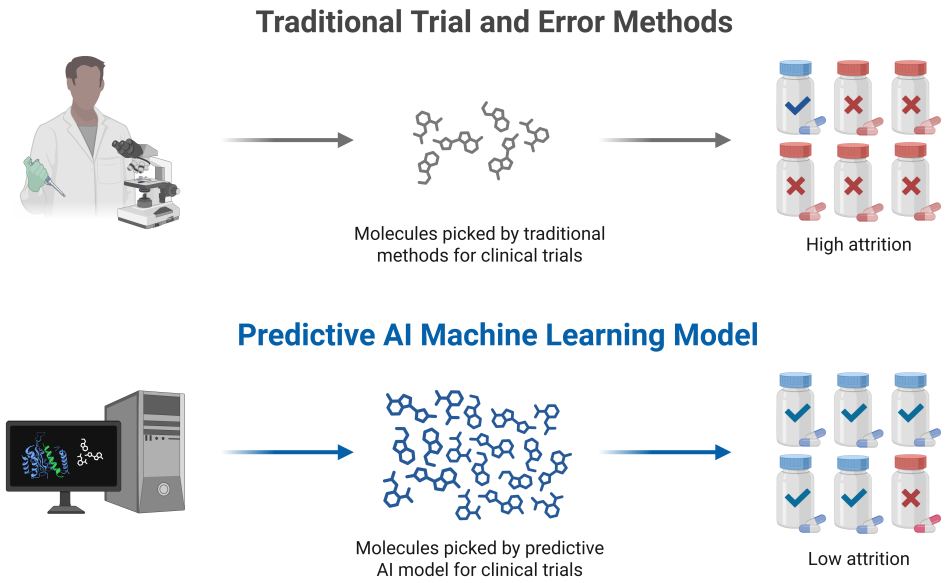

Figure 1.1 While traditional methods lead to high attrition through late-stage failures, predictive AI could enable early filtering and higher clinical success rates.

In recent years, researchers have begun developing **multimodal** models to capi-

talize on the representational complementarity among differing data types. These AI models are designed with the capacity to interpret multiple differing representations of the same subject and unify those interpretations in a manner that forms a richer, more expressive "understanding" of the subject. While early results have demonstrated the promise of multimodal learning for drug discovery, many fundamental questions remain. For instance, it is still unclear which combinations of data representations tend to offer the greatest performance gains, and how best to integrate and unify them within unified architectures.

This ongoing study aims to contribute to this area of research by developing and evaluating a multimodal deep learning software framework that integrates two distinct molecular data representations based on two independent data modalities. These modalities are (1) molecular graphs with 3D atomic positions, and (2) human language descriptions. The first modality involves representing molecules in the form of graphs where the atoms and the atom-to-atom bonds of a given molecule are represented by nodes and edges, respectively. The second modality simply involves representing molecules with thorough text descriptions readable by humans. To our knowledge, relatively little work has explored this particular combination of data inputs, making it a promising direction for novel investigation. The ultimate hypothesis of this study is that an AI framework that integrates the two aforementioned data modalities will perform better than a unimodal AI model based on either modality alone.

As mentioned, this study is ongoing with only partial implementation of the full multimodal framework completed at this time. (Elaboration of what has been completed and what remains for future work will be given shortly, but for context, a description of the architecture will be provided first.) The full software architecture "blueprint", so to speak, consists of three component neural networks (AI models). These three components include a Graph Neural Network (GNN) (Kearnes et al

(2016) [1]), a customized Large Language Model (LLM) (think human-language neural network), and a Multilayer Perceptron (MLP). The GNN and customized LLM components are intended to be fully functional unimodal neural networks when utilized independently of each other and the rest of the framework—both models should be capable of producing reasonably accurate predictions of chemical properties associated with a given molecule. The MLP's role is to combine the interpretations of the GNN and the LLM and compress them into a single interpretation that will produce its own predictions of the same properties. In alignment with our hypothesis, we expect these final predictions to achieve greater accuracy than either of the component neural networks alone.

Thus far, only the GNN component of the full architecture has been successfully implemented. The customized LLM and the MLP components remain in development, and therefore, neither can be appropriately evaluated at this time. Additionally, without these components, the ultimate hypothesis of this study cannot yet be tested. With these things in mind, the intentions for this paper are to detail the methods and experimental results of the GNN component alone, with only mere mention of the LLM and MLP components where helpful.

Thorough evaluation of the performance of AI neural networks often involve analysis of many aspects such as predictive accuracy when the model is optimally trained, the number of training iterations needed to reach optimal accuracy, inference cost and speed (the computation and time required for the model to produce results), model size and memory footprint, and others. As this study remains incomplete, only metrics of two of these aspects are gathered and analyzed—particularly, the accuracy, or correctness, of the model's predictions (after optimal training); and the number of training iterations required to provide the model *with* optimal training. Further detailing of the evaluation process for the GNN model is provided in the **Experiment**

chapter.

For the reader's benefit, a brief, largely qualitative preview of the experiment and results achieved by our GNN model is given: Our process of testing our Graph Neural Network model can be boiled down to a few important steps. First, we trained the model on a handful of arbitrary chemical properties across $100,000+$ arbitrary molecules until the neural network reached a state where further training yielded no further improvement in its predictive ability (in this context, this optimal state coincides with the model's training and validation *convergence*). Second, we tracked the number of training iterations that produced this convergence. Finally, we tested the converged model's ability to predict the same arbitrary chemical properties used in training, but for $2,000+$ molecules that were never revealed to the model during its training. Through this process, we found that our GNN model converges to its optimal state in 23 training iterations. Additionally, the testing of our optimized GNN model's predictive accuracy revealed strong correlation between the values of its predictions and the associated ground-truth (or actual) values. An average R^2 score of 0.954 was achieved across all of the arbitrary chemical properties.

Chapter 2

Method

In this ongoing study, we design the general software structure of a unified neural network model architecture that can be viewed in two layers. The first layer involves the production of two separate chemical-property prediction vectors made by two independent neural networks. The second layer involves, first, concatenation (combining) of those prediction vectors into a single, larger vector, then, inputting that vector into a Multilayer Perceptron (MLP), and finally, obtaining a final prediction vector from the MLP based on the two data representations used in the first layer of the architecture. (Note: An MLP is a basic neural network that processes an input vector of some kind and computes a new output vector dependent on learnable patterns found in a dataset of related input vectors).

The data modalities used as the basis of the representations for the two neural networks comprising the first layer of the architecture are graph-based data and text-based data, respectively. As has already been hinted at, the predictions made by the neural networks are outputted in the form of vectors. Within the context of our architecture, the output vector of a neural network is an n-dimensional vector, where n is equal to the number of regression targets (the number of properties being

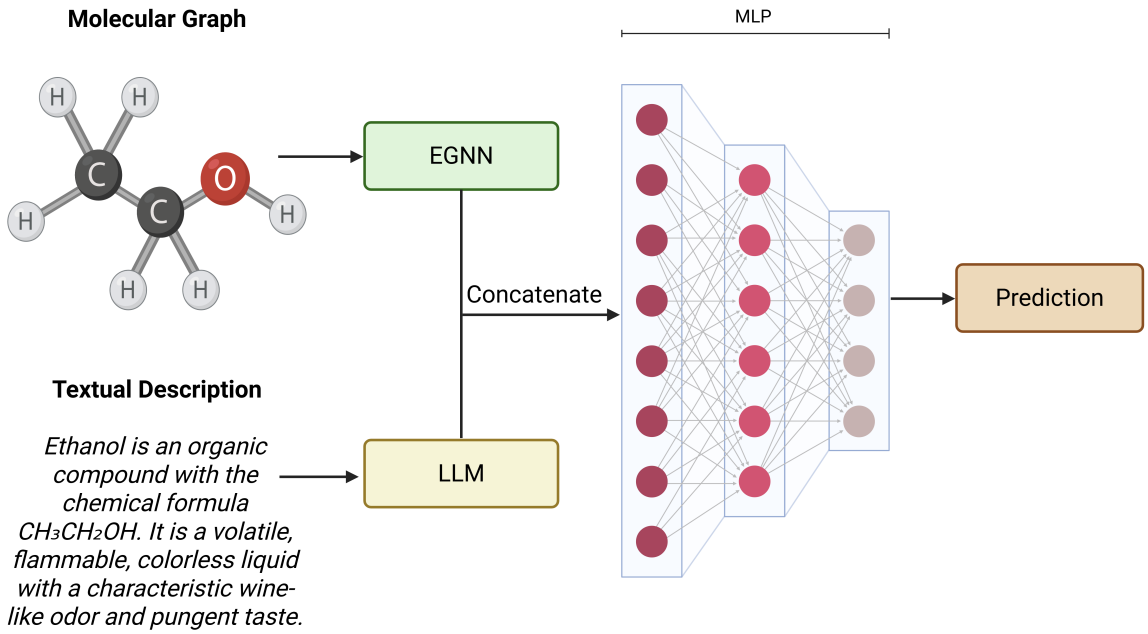

Figure 2.1 Depiction of the full architecture, where graph (EGNN) and text (LLM) interpreters feed concatenated outputs into a unifying neural network (MLP) to produce a multimodally based prediction.

predicted). The scalar values within the vector are the neural network's predictions for the regression targets (each scalar value is associated with one unique target). The graph-based modality represents molecules in the form of graphs, where the atoms of a molecule are treated as nodes and the atom-to-atom bonds are treated as edges. The text-based modality represents molecules in the form of human language descriptions (text descriptions readable by humans) unique to each molecule.

While the full architecture consists of the three mentioned major components—a graph-interpreting AI, a text-interpreting AI, and a unifying MLP—significant progress has only been made on the graph-interpreting AI. Thus, the remainder of this paper will elaborate further on only that component.

To make adequate interpretations of the graph-based data, we employ a specialized version of a GNN called an E(n)–Equivariant Graph Neural Network (EGNN). Typically, a GNN interprets graph-structure in a manner that neglects the specific

physical positioning of nodes for a given graph subject, or, in our case, the physical positioning of atoms relative to one another within a given molecule. In other words, while the GNN can interpret which atoms are chemically bonded to one another, it cannot interpret information regarding the proximity of atoms relative to each other. On the other hand, the E(n)–EGNN can interpret both the atom-bond graph structure of the entire molecule *and* the physical proximity of each of the constituent atoms (relative to each other).

The EGNN architecture used in this study follows the message-passing scheme introduced by Satorras et al. (2021) [2], which extends traditional graph convolutional networks found to handle geometric data while preserving equivariance to E(n) transformations such as translation, rotation, and reflection. In this scheme, node representations are iteratively updated by aggregating information from neighboring nodes. Each node begins with an initial feature vector representing an arbitrary set of attributes (the same attributes must be used for each node), and with each layer, these features are updated based on the representations of neighboring atoms.

As mentioned, the key innovation of the EGNN lies in its treatment of spatial geometry in addition to the typical graphical geometry. Utilizing the same message-passing algorithm for node features, the model also maintains a separate set of 3–D coordinates for each node, representing absolute atomic positions in an x-y-z format. These coordinates are incorporated into the message-passing mechanism in a relative manner, making use of only the difference between node positions, and never absolute positions. This is important because the interpretation of the EGNN can only be consistent (irrespective of any arbitrary absolute atomic coordinates) if relative positions are wholly used in the message-passing mechanism. The node positions, in addition to the node features, are also updated at each layer. This design allows the model to jointly learn from both atomic identities and molecular geometry, enabling

more accurate predictions of structure-dependent properties.

Chapter 3

Experiment

To evaluate the performance of our EGNN-based graph-data interpreter, we considered two major aspects of performance: (1) the progression of training and validation loss over training iterations, and (2) the predictive accuracy of an optimally trained model. Before moving on, several terms warrant for the provision of their definitions. "Loss" is a measure of the average error per prediction for a given iteration. "Training" is the process of "teaching" an AI model the patterns that exist among the samples of the dataset that it trains on. Essentially, the model makes a prediction for each sample in the dataset, and then, depending on the degree of error among its predictions against their associated actual values (called ground-truths), the model has its parameters tuned to prepare it to make a better prediction in the future. "Validation" is the process of testing how well the model generalizes to data that it is not training on. A separate dataset is used for validation, and the model does not tune its parameters according to error against the validation set to ensure that the set continually acts a general representation of non-training data. For reference, training loss and validation loss are both calculated using the Mean-Squared-Error (MSE) algorithm, or the average of the squared errors across all samples in the dataset.

Predictive accuracy of the model is assessed using standard regression metrics, including Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). RMSE is similar to MSE, except it applies a square-root to the MSE result. MAE is the average of the absolute errors across all samples. R^2 is a measure of how well the model explains the variance in the dataset. The equation for calculating R^2 is

$$R^2 = 1 - \frac{sum\,of\,squared\,errors}{total\,variance} = \frac{\sum(\hat{y}_i - y_i)^2}{\sum(y_i - \bar{y})^2}$$

where $\hat{y}_i$ is the predicted value, y_i is the actual value, and $\bar{y}$ is the average of all of the actual values.

The EGNN model is trained, validated, and tested on curated subsets of the MolTextNet dataset [3]. Provided herein are qualitative visualizations of predicted values versus actual (or ground-truth) values for select regression targets.

3.1 MolTextNet Dataset Curation

For training and evaluation of our EGNN model, we selected the MolTextNet (MTN) dataset—a dataset of 2.5 million molecular samples where each sample has an associated SMILES string and a thorough text-description. A SMILES (Simplified Molecular Input Line Entry System) string is a string of characters that communicate the graph layout of atomic bonds within a molecule. For example, carbon monoxide (CO) is written as C=O, where the "=" character represents a double bond. Due to memory and runtime constraints, a subset of MTN was compiled to include only the first $200,000$ samples (about 8% of the 2.5 million total samples). These samples, after undergoing filtering and cleaning, were reduced to a final dataset of $155,241$ samples, representing approximately 6.2% of the full MTN dataset.

From our custom MTN dataset, each sample was processed into a PyTorch-Geometric (PyG) 'Data' object containing the following PyTorch 'tensor' objects:

- `x`: A 10-dimensional vector containing arbitrary atom-level node features [`x` = (atomic number, atomic mass, ...)]
- `pos`: A 3-dimensional vector containing 3-D spatial coordinates of atoms (`pos` $= (x, y, z)$)
- `edge_index`: A 2 x E matrix (where E is the number of atomic bonds) representing bond connectivity between atoms (source atoms in the first row are aligned with target atoms in the second row, using indices)
- `y`: A 5-dimensional regression target vector extracted from the text. Put another way, `y` is a vector that contains the actual property values against which our model's predictions can be compared [`y` $= (y_1, y_2, y_3, y_4, y_5)$]
- `text`: A string of the full molecule description (e.g., "*Ethanol is an organic compound with the chemical formula...*")
- `mask`: A vector of the same shape as, and coinciding with, `y` that indicates which target (actual) values are hidden within the text to prevent the AI model from using the text description to explicitly obtain exact chemical property values. A 1 represents a visible value, and a 0 represents a hidden value. [Example: `mask` $= (0, 1, 1, 0, 1)$]

As samples from the MTN dataset do not directly provide graphs, we used an open-source cheminformatics software toolkit, called RDKit [4], to convert SMILES strings directly into molecular graphs and to generate approximate 3-D atomic positions for all atoms in each molecule. This conversion and generation process allowed

for the creation of the `pos` and `edge_index` tensors. Anytime RDKit determined a SMILES string to be invalid (ie. it did not recognize the string) or it failed to generate 3-D positions, the sample was skipped and excluded from the dataset. For all molecules deemed valid, RDKit featurized their atoms (or provided a list of 10 atomic features associated with each atom. We made the choice of which 10 features to use arbitrarily). This allowed for the creation of the `x` tensor. Note: The algorithms that RDKit employs to generate 3-D positions and perform featurization are beyond the scope of this study.

Actual chemical property values for making predictions against were not explicitly provided in the dataset. Fortunately, there were multiple property values consistently embedded within the molecule descriptions. To create the `y` tensor, the vector of regression targets, we extracted the actual values within each text using a keyword-based parser algorithm. The algorithm searched each description for target-specific keywords and phrases (such as "molecular weight" and "logp", etc.) and extracted the next numerical value enclosed in `<number>...</number>` tags (provided by the authors of MolTextNet). If one or more target values were missing or improperly formatted (ie. the algorithm failed to identify the value), the sample was excluded. A probabilistic masking scheme was then applied to obscure certain values in the text, and a corresponding binary mask vector was generated to indicate which targets were visible in the description.

Neither our algorithm nor each description were perfect, so incorrect values were occasionally extracted. To help ensure that these values did not interfere with the rest of the dataset, all entries where any two or more target values were identical were excluded, as this was very likely an indication that the text displayed a duplicate numerical value for two unrelated chemical properties. Next, to identify samples with at least one target value that would be considered an extreme outlier, distribution

analysis was performed on each target. Specifically, only samples with all 5 regression target values between the 1st and 99th percentiles of each target distribution were retained. After completion of the entire curation process described throughout this section, the final filtered dataset consisted of $155,241$ valid entries, or about $45,000$ less samples than the original set of $200,000$.

3.2 Implementation of the Graph-based Interpreter

We implemented a custom $E(n)$-Equivariant Graph Neural Network (EGNN) model using the publicly available egnn_pytorch module developed by Lucidrains [5], which provides a PyTorch-based $E(n)$-equivariant message-passing scheme. The model architecture consists of an input projection layer, a configurable sequence of EGNN layers, a dropout layer, and a final output projection layer. The input layer maps initial atomic feature vectors to a fixed hidden dimension, and each EGNN layer jointly updates both node features and 3D spatial coordinates through equivariant message passing. A non-linear activation is applied after each EGNN layer (without going into unnecessary detail, non-linearity within the parameters of the neural network is necessary to allow it to detect meaningful patterns). The final learned node representations are masked to exclude padding nodes, normalized based on the number of active nodes, and passed through a dropout layer before being projected to a vector of scalar outputs via the final linear layer. The output dimension corresponds to the number of chemical properties being predicted. In this study, we arbitrarily chose five properties to evaluate our model with (the common availability of these properties throughout MTN text-descriptions had influence on this choice). These properties are molecular weight, Log P, Polar Surface Area, Synthetic Complexity Score (SCS), and Synthetic Accessibility Score (SAS). Per the scope of this study, let it be known

that the reader need not know the exact meanings of these terms, but only that each is a chemical property commonly associated with molecules.

Due to implementation constraints, input batches were transformed from PyG's default sparse format (ΣN, F) to a dense batch format of shape (B, N, F), where B is the batch size, N is the maximum number of atoms in any molecule within the batch, and F is the number of features per atom. Molecules with fewer atoms than N were padded accordingly, and a binary mask of shape (B, N) was used to track valid (non-padding) atoms. This batching transformation was applied consistently to both the node feature tensor and the 3D positional coordinates (B, N, 3), as well as the adjacency representation, which was converted into dense edge matrices of shape (B, N, N). (Note: we arbitrarily chose 10 atomic features to act as node features for each atom. These features are atomic number, ring membership, aromaticity flag, atom degree, atomic mass, explicit hydrogens count, implicit hydrogens count, total degree, total hydrogen count, and valence electron count. Similar to the chemical properties, the scope of this study does not demand that the reader understand the specific meanings of these terms, but only that they are each a valid atomic feature).

3.3 Graph Encoder Experimental Results

We evaluated the performance of the EGNN graph encoder as a unimodal baseline on the filtered MolTextNet dataset. The model was trained using a batch size of 64 and optimized with the Adam, or Adaptive Moment Estimator, optimizer (an optimizer that uses stochastic gradient descent with an adaptive learning rate) using an initial learning rate of $1 \cdot 10^{-3}$ (widely considered the standard for the Adam Optimizer). The training procedure used early stopping based on validation loss, with a patience of 5 iterations. In other words, a relative optimal model was determined during the

training procedure after 5 iterations passed with no decrease in average error for model against the validation set. Training halted after 28 iterations, meaning the relative optimal model finishes training after only 23 iterations. Prior to training, all target values were standardized by subtracting the respective mean from each target and dividing the result by the respective standard deviation. This caused each target variable to have a mean of 0 and a standard deviation of 1. (Standardizing is a common machine learning practice to help stabilize training and prevent variables with inherently larger magnitudes from dominating the tuning of the neural network's parameters.) To assess the predictive accuracy of the model, a separate held-out test set was used to prevent any possible adaptation of the model directly to the test set, keeping the set as an adequate general representation of non-training and non-validation data. Reiterating which regression metrics are used, the three metrics are: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). (All training was conducted in Google Colab using an A100 GPU instance.)

Figure 3.1 illustrates the progression of training and validation loss over the 23 training epochs. Table 3.1 summarizes the model's predictive performance across all five molecular property targets. Finally, figure 3.2 shows scatter plots comparing predicted and ground-truth values for four of the five regression targets. The fifth plot was omitted for layout clarity but followed similar predictive trends.

The model achieved high accuracy on most targets, with R^2 scores exceeding 0.95 for Molecular Weight, Log P, Polar Surface Area, and Synthetic Accessibility Score (SAS). The Synthetic Complexity Score (SCS) target showed comparatively lower predictive performance ($R^2 = 0.880$), likely reflecting greater variability or complexity in its underlying signal. Overall, the average MAE across all targets was 0.158, the average RMSE was 0.206, and the average R^2 score was 0.954. Keep in

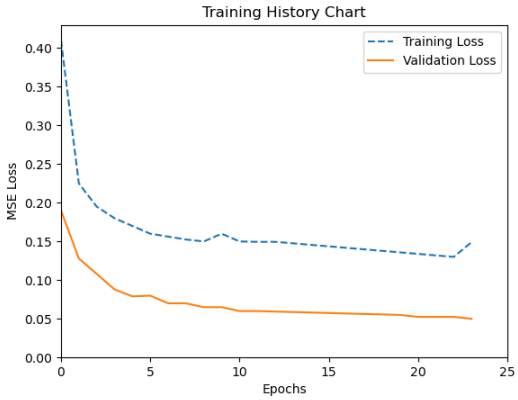

Figure 3.1 Training and validation loss plotted over the course of 23 training epochs. The plot reflects the model's learning behavior during optimization, with early stopping triggered after **validation loss** ceased improving. Loss values are computed using Mean Squared Error on standardized target values.

mind that all target values were standardized, so all targets share comparable scales.

(Note: R^2 scores range from $-\infty$ to 1, with a higher value indicating better alignment of the model with variance in the dataset. An R^2 score of $R^2 > \approx 0.9$ is generally viewed as indicating a strong fit of the model.)

Property	**RMSE**	**MAE**	$\boldsymbol{R^2}$
Molecular Weight	0.134	0.103	0.982
Log P	0.172	0.126	0.969
Polar Surface Area	0.143	0.114	0.981
SCS	0.338	0.266	0.880
SAS	0.243	0.182	0.955
Average	**0.206**	**0.158**	**0.954**

Table 3.1 Standardized EGNN test set results across five molecular property targets.

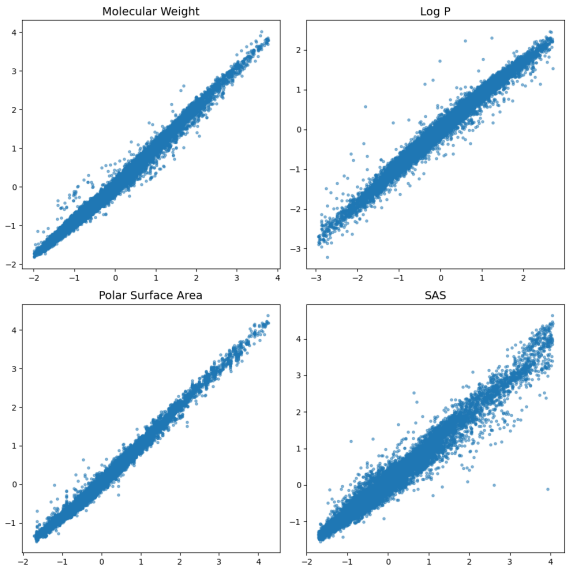

Figure 3.2 Scatter plots comparing predicted versus ground-truth values for four of the five molecular property targets in the test set. Each point represents a single molecule, plotted using standardized target values. A diagonal line would ($y = x$) indicate a perfectly accurate model and a straight horizontal line ($y =$ constant) would indicate a very poor model.

Chapter 4

Discussion

Ultimately, the goal of this study is to evaluate whether a multimodal model combining interpretations based on molecular graphs and human language descriptions can outperform the component unimodal models in molecular property prediction. Developing a working, effective EGNN-based graph interpreting AI is the first step toward that goal. Our EGNN model achieved high predictive accuracy, notably, with average R^2 exceeding 0.95 on four out of five target properties. These results confirm that a spatially-aware (ie. atomic positions) graph model can capture significant structure–property relationships pertaining to molecules and, thus, provide a robust baseline for subsequent comparisons of the full architecture.

An explanation for the relatively lower performance on the SCS target is not yet known, but may include a reflection of greater variability in how this property is expressed structurally, and/or limitations in how it was extracted from the dataset. Additionally, the use of standardized target values improves numerical stability but complicates interpretation in physical units. These limitations, along with the exclusion of text-based and fusion model results at this stage, frame this study in its current form as an in-progress contribution toward the development and evaluation

of a multimodal framework.

Future evaluations will assess the performance of the text interpreter and the full architecture. It is anticipated that the text descriptions may encode complementary semantic (language-based) signals that are not explicitly captured in the graph structure. If so, it is expected that the multimodal framework will demonstrate improved generalization and performance across difficult-to-predict properties. While the full hypothesis has yet to be tested, the current findings establish a solid foundation and justify further development of multimodal approaches in molecular representation learning.

Chapter 5

Conclusion

In this study, we proposed a multimodal deep learning software framework for chemical property prediction that integrates graphical and textual data modalities. This framework includes a unified architecture composed of an E(n)-Equivariant Graph Neural Network (EGNN) for molecular graph encoding, a customized Large Language Model for processing human language descriptions, and a final Multilayer Perceptron (MLP) for prediction.

As a first step, we curated and preprocessed a large subset of the MolTextNet dataset into a PyTorch Geometric-compatible format and implemented the EGNN-based graph interpreter. Our experimental results on this model demonstrate strong predictive performance across five regression targets, with high R^2 scores and low error metrics, confirming the EGNN's capability to model molecular structure effectively.

Future work will involve completing the implementations of the text interpreter and the unifying MLP, followed by a thorough comparison of unimodal versus multimodal performance to assess the added value of combining diverse molecular representations.

Bibliography

[1] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of computer-aided molecular design*, vol. 30, no. 8, pp. 595–608, 2016.

[2] V. G. Satorras, E. Hoogeboom, and M. Welling, "E (n) equivariant graph neural networks," in *International conference on machine learning*, pp. 9323–9332, PMLR, 2021.

[3] Y. Zhu, G. Liu, E. Inae, and M. Jiang, "Moltextnet: A two-million molecule-text dataset for multimodal molecular learning," *arXiv preprint arXiv:2506.00009*, 2025.

[4] G. Landrum, "RDKit: Open-source cheminformatics," 2006. http://www.rdkit.org.

[5] Lucidrains, "egnn-pytorch: Pytorch implementation of e(n)-equivariant graph neural networks." https://github.com/lucidrains/egnn-pytorch, 2024. Version 0.2.8.