Van der Waals Equation of State Coefficient Predictor

Design Report 3
COSC 402
Blake Milstead, Turner Heath, Karim Chmayssani
April 4, 2025

Executive Summary

This report details the design of a machine learning tool aimed at predicting the Van der Waals constants, a and b, for various molecules. These constants, which are incalculable, are essential for understanding real gas behavior, impacting fields such as chemical engineering and pharmaceuticals. Traditional methods for determining these constants are often labor-intensive and time-consuming. This project is heavily research focused, with the goal of finding the best solution to solve our proposed problem. Thus, we have explored a multitude of architectures and models, continually improving our prediction accuracy. These explored models include Graph Neural Networks (GNNs), Convolutional Neural Networks (CNNs), Physics-Informed Neural Networks (PINNs), and Kolmogorov-Arnold Networks (KANs), as well as regression models for baselines. As we continue to test and optimize, this report will explore the results and insight we have gathered through our development process. We will conclude with a comparison of our best performing models, discussing strengths and trade-offs between each one. On the user side, the envisioned software application will allow users to input molecular structures and receive predicted a and b constants, complete with confidence intervals and visualizations. Target users include chemical engineers, pharmaceutical researchers, and academic educators. The design validation plan encompasses usability testing, iterative refinement, and evaluation metrics such as prediction and model accuracy. The final deliverables will comprise of the software application, with multiple predictive models, comprehensive documentation, validation report, source code repository, deployment package, and training materials, providing a comprehensive solution that ensures ease of use, transparency, and adaptability for various user requirements.

Contents

Executive Summary 1

1 Problem Definition & Background **6**
1.1 Problem Statement 6
1.2 Background and Context 6
1.3 Application Areas 7
1.4 Underlying Theory 7
1.5 Prior Work 7
1.6 Unaddressed Needs 8

2 Requirement Specification **9**
2.1 Overview 9
2.2 Functional Requirements 9
2.3 Non-Functional Requirements 10
2.4 Metrics and Targets 10

3 Technical Approach **11**
3.1 Overview 11
3.2 Architecture Overview 11
3.3 Functional Decomposition 12
3.3.1 Data Collection and Preprocessing 12
3.3.2 GUI 13
3.3.3 Encoder Models 13
3.3.4 Decoder Models 14
3.3.5 Model Tuning 14
3.3.6 Output 14

4 Design Concepts, Evaluation & Selection **15**
4.1 User Interface Design 15
4.1.1 Intended User Flows and Workflows 15
4.1.2 Target Use Scenarios 16
4.1.3 Design Validation Plan 16
4.2 Molecular Embedding Techniques 17
4.3 Model Design Decision Analysis 18

5 Product Development and Evaluation Plan **20**
5.1 Architecture Selection . . . 20
5.1.1 Input Selection . . . 20
5.1.2 Model Evaluation Metrics . . . 21
5.2 Architecture Iterations . . . 21
5.2.1 Initial Design . . . 21
5.2.2 GNN-PKAN Pipeline . . . 22
5.2.3 CNN-PKAN Pipeline . . . 24
5.3 GUI Design . . . 26
5.3.1 General Layout . . . 26
5.3.2 Technologies . . . 27
5.4 GUI Evaluation . . . 27
5.4.1 User Experience Survey . . . 27
5.4.2 Error Handling and Stability . . . 27

6 Social Impact Evaluation **28**
6.1 Recognizing and Defining Targeted Needs in a Social Context . . . 28
6.2 Broader Impacts of Engineering Solutions . . . 28
6.3 Ethical and Professional Responsibilities . . . 28
6.4 Communication on Societal Issues . . . 28
6.5 Understanding Professional and Ethical Responsibilities . . . 29

7 Deliverables **30**
7.1 Overview . . . 30
7.2 Final Deliverables . . . 30
7.2.1 Software System . . . 30
7.2.2 Documentation . . . 31
7.2.3 Evaluation Results . . . 31
7.2.4 Source Code and Deployment Tools . . . 31
7.3 Future Extensions . . . 31

8 Project Management **32**
8.1 Overview . . . 32
8.2 Project Timeline . . . 32
8.3 Team Roles and Responsibilities . . . 32
8.4 Future Refinements . . . 33

9 Budget **34**
9.1 Overview . . . 34
9.2 Estimated Expenditures . . . 34
9.3 Engineering Time Allocation . . . 34
9.4 Future Adjustments . . . 35

A **37**

List of Figures

1.1 Real Gas Law Terms Explained 6
1.2 A vs B Regression 8

3.1 Functional Decomposition Block Diagram 12

4.1 GUI Mockup 16
4.2 R2 Score of Initial Models 19
4.3 Individual a/b MSE Score of Initial Models 19

5.1 GNN-PKAN Pipeline 22
5.2 GNN Architecture 23
5.3 PKAN Architecture 23
5.4 PINN Architecture [1] 24
5.5 CNN Architecture [**?**] 25
5.6 GUI upon application startup 26
5.7 GUI after molecule has been generated and model is run 27

A.1 Prediction Results with Initial Embeddings 37
A.2 Equation of State Coefficient Predictor Logo (Van der Waals Coefficient Predictor) 37

List of Tables

8.1 Preliminary Project Timeline 33

9.1 Preliminary Budget Estimates 34
9.2 Estimated Human-Hours Allocation 35

Chapter 1

Problem Definition & Background

1.1 Problem Statement

The accurate prediction of the Real Gas constants, a and b, is critical for understanding and modeling molecular behavior under non-ideal conditions. These constants, which account for intermolecular forces and molecular volume respectively, are essential for engineering, chemistry, and various industrial applications. Currently, determining these constants relies heavily on experimental methods, which can be time-consuming, resource-intensive, and infeasible for large-scale applications. A predictive machine learning-based approach using molecular embeddings could significantly improve efficiency and scalability.

1.2 Background and Context

Accurately modeling the behavior of real gases is a longstanding challenge in physical chemistry and thermodynamics. While the Ideal Gas Law offers a simplified estimate, real-world application necessitates the use of the Van der Waals equation, where the a and b constants provide critical corrections. Industries such as petrochemicals, pharmaceuticals, and material sciences often require precise modeling of gas behavior for applications such as reaction engineering, material synthesis, and environmental control. Current experimental methods

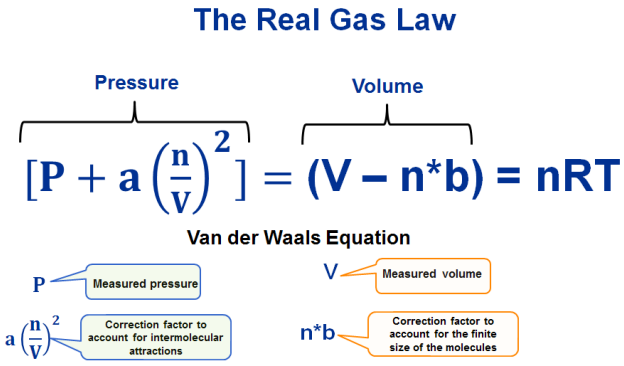

Figure 1.1: Real Gas Law Terms Explained

to determine these constants are laborious and limited in scope, making them unsuitable for rapid prototyping or computational simulations involving large molecular datasets.

A machine learning approach leveraging molecular embeddings offers the potential to revolutionize this process. By mapping molecular structures to high-dimensional feature spaces, embeddings can encode complex molecular properties, enabling rapid and accurate predictions of the Real Gas constants.

1.3 Application Areas

- **Chemical Engineers:** Require precise gas behavior modeling for designing processes and reactors.
- **Materials Science:** Predicting properties of new materials, such as thermal stability or conductivity, to optimize performance in applications like electronics or energy storage.
- **Pharmaceutical Researchers:** Use molecular property predictions for drug development.
- **Software Developers in Simulation Tools:** Need efficient algorithms for integrating predictive capabilities into modeling software.

1.4 Underlying Theory

The Van der Waals equation extends the Ideal Gas Law to Real Gases, including:

- a**:** Quantifies intermolecular attractions, critical for understanding liquefaction and condensation phenomena.
- b**:** Represents the volume of a molecule to account for their physical size.

Molecular embeddings encode chemical structures into numerical representations, capturing properties such as bond types, electron distribution, and geometric configurations. Techniques such as RDKit-based descriptors, graph neural networks, and cheminformatics algorithms are foundational to this approach. Figure 1.2 shows the relationship between a and b, giving us proof that a correlation exists and can be learned.

1.5 Prior Work

Previous work to find the Real Gas constants includes:

- Experimental determinations using high-precision instruments [2].
- Thermodynamic models based on molecular theory and statistical mechanics.
- Machine learning models predicting related properties, such as boiling or critical points.

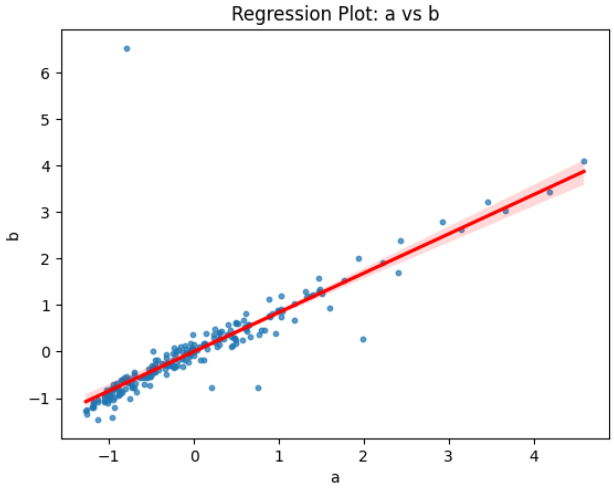

Figure 1.2: A vs B Regression

- Computational chemistry methods such as quantum chemistry and molecular dynamics [3].

1.6 Unaddressed Needs

Current software and tools are either too domain-specific or computationally expensive. This project aims to produce accurate predictions of the Real Gas constants, while remaining lightweight and flexible enough for reapplication.

Chapter 2

Requirement Specification

2.1 Overview

The primary objective of this project is to develop a scalable and efficient prediction pipeline integrated with a user-friendly graphical user interface (GUI). The pipeline will preprocess molecular data, generate embeddings, and use a deep learning model to predict Real Gas properties. The GUI will enable users to input molecular structures, view predictions, and interpret results interactively.

2.2 Functional Requirements

- **Graphical User Interface (GUI):**
  - A user-friendly interface for inputting molecules, visualizing predictions, and exploring properties.
  - Interactive visualization of input molecules and predicted properties (e.g., tables, plots, or 3D molecule views).
  - Intuitive workflows for both novice and expert users, with minimal technical barriers.
- **Data Input and Processing:**
  - Creation and visualization of arbitrary user-designed molecules.
  - Chemically informed validation of generated molecule.
  - Connection and input to encoding model.
- **Encoding Model:**
  - Receive user-generated molecular representation.
  - Provide a fixed-length vector encoding or full prediction of a and b values.
- **Decoding Model:**

  - Receive encoding from encoding model.
  - Provide full prediction of a and b values.

- **Prediction Output and Metrics:**
  - Create prediction statistics, including:
    * Mean
    * Median
    * Mode
    * Standard Deviation
    * 95% Confidence Intervals
  - Display interactive graphical representation of collected statistics.

2.3 Non-Functional Requirements

- **Prediction Accuracy:**
  - For each developed model, we aim for all predictions across the dataset to contain the true a and b values within the predicted confidence interval.
- **Scalability:**
  - Allow for an arbitrary number of models to be stored and accessible for the user.
- **Reliability:**
  - Ensure the system gracefully handles invalid inputs and provides meaningful error messages.
- **Usability:**
  - Design the GUI to be platform-independent, accessible via major operating systems (Windows, macOS, Linux). Must be user-friendly and easy to use.

2.4 Metrics and Targets

- **Prediction Accuracy:** Primarily, predictions aim to capture the true values within the confidence interval. Models will be evaluated on their performance with this task.
- **Learning Metrics:** Mean Squared Error (MSE) and Mean Absolute Error (MAE) are used within model training and validation.
- **Processing Speed:** Complete model training in time proportional to size of dataset. Provide predictions in a reasonable time.
- **GUI Usability:** Receive positive usability feedback in user testing.

Chapter 3

Technical Approach

3.1 Overview

Our design plan aims to develop an advanced molecular property prediction system integrating multiple machine learning approaches with domain-specific molecular representations and encodings. Figure A.1 shows the viability of using embeddings to make accurate predictions. This simple proof of concept allows us to move forward with confidence. The pipeline will include molecular data preprocessing, various encoding methods, and multiple predictive models. Through our design process, we have identified key aspects of the system that our models must incorporate. These include physical constraints, structural information, element identification, and a wide, diverse dataset. With these in mind, we have iteratively designed a pipeline to be compared to baseline predictive models. The system will also incorporate a graphical user interface (GUI) to facilitate user interaction and accessibility.

3.2 Architecture Overview

The pipeline will be fully developed in Python and consists of four main sections:

- **Data Preprocessing:** Using our corpus of molecular data, we generate a representation of the molecule that includes structural information and element identification. This is a background process, not included within the deliverable. The dataset is created once and then reused for all models.
  - Molecule graphs: Node, edge, and edge attribute matrices. Uses the torch.geometric library [4].
  - Molecule images: A 2D image created from the SMILES string for a molecule.
- **Graphical User Interface:** The GUI aims to serve as a landing pad for users, allowing them to create and upload molecules for prediction.
- **Encoder:** Using the generated molecule representation, we create a fixed length embedding through a model trained on a and b. This unique embedding represents the molecule in a vector space in relation to a and b.

  - GNN
  - CNN

- **Decoder:** With a fixed length input vector, we have a large amount of freedom in designing our predictive model. This model must account for physical constraints unaddressed by the encoder, and give us our final a and b values.
  - PINN
  - PKAN
  - LR (Linear Regressor)

3.3 Functional Decomposition

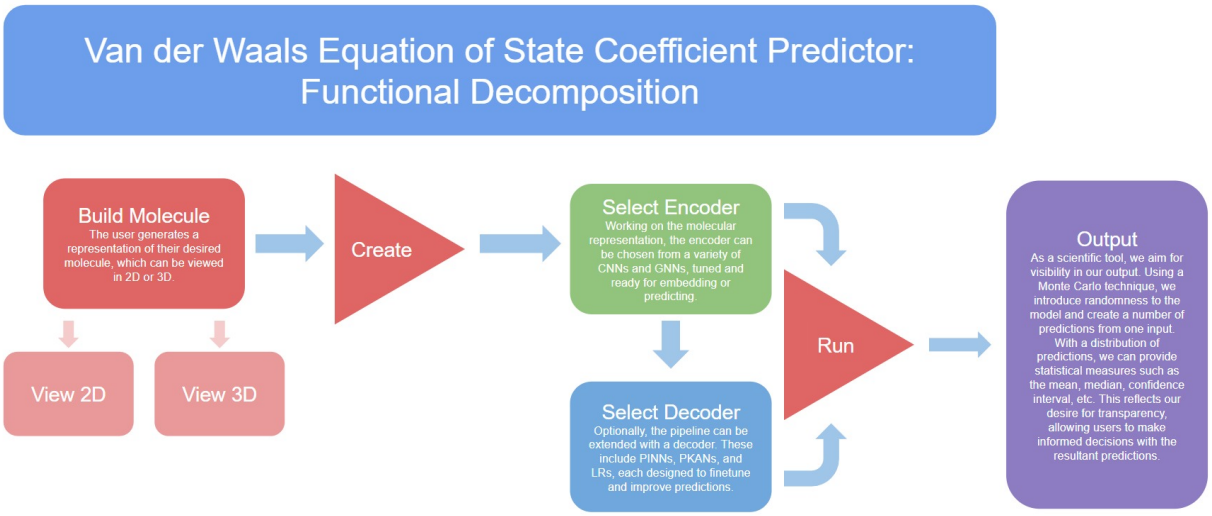

Figure 3.1: Functional Decomposition Block Diagram

To design the pipeline as presented above, we have decomposed each of the four sections of our model.

3.3.1 Data Collection and Preprocessing

The data collection and processing section of the pipeline carries many responsibilities, namely:

- **Molecular Data Acquisition**: Gather molecular structures and their corresponding a and b constants from the dataset.
- **Data Cleaning**: Address missing values, remove duplicates, and standardize molecular representations to ensure data quality.

- **Feature Extraction**: Generate molecular embeddings using techniques such as molecular fingerprints, graph-based representations, and sequence-based embeddings [5].
- **Train and Test Split**: Most important to the integrity of our design, we must be very intentional regarding data leakage between training and test data. As the models are trained, we must ensure that they never see any of the test data, as this can skew results and decrease validity of the model. We must ensure that the training data is representative of the full space, allowing the most generalization to occur.

3.3.2 GUI

The GUI serves as a simple, streamlined interface that allows users to interact with the model pipeline. It can be broken up into its main components:

- **Molecule Builder**: The molecule builder is the central user experience of the GUI. Users will be able to create arbitrary molecules with any amount of atoms, elements, and bond types.
- **Molecule Viewer**: From the built molecule, the user can view a 2D or 3D representation of their designed molecule. This will allow for assessing, correcting, and visualizing of inputs before prediction.
- **Parameters and Options**: During testing, options for simulating, predicting, or visualization will be incorporated as we identify need.
- **Predictions**: The GUI now imports the created molecule into the prediction pipeline at the click of a button, and once completed will display prediction outputs, confidence intervals, and other relevant information.

3.3.3 Encoder Models

The encoder is a machine learning model which takes specified molecular information and creates a fixed-length representative embedding. These embeddings are unique, capturing the details of the molecule while fitting to a desired size. To create these embeddings, encoders are trained on a and b, relating molecular properties to these values. Additionally the encoders can be used as standalone predictive models (decoder). To provide users with a comprehensive tool, multiple encoder models will be included. Encoders used on their own will be considered baseline predicting models. These models will be tuned and validated to their fullest extent, giving us a fair comparison. These include:

- LR
- CNN
- GNN
- PINN

3.3.4 Decoder Models

The decoder is a machine learning model which takes a fixed-length molecular embedding, output from the encoder, and produces a final a and b prediction. Once encoder models have been finetuned, decoder models can be trained and tuned to enhance predictive capabilities. By providing a variety of decoders, users can identify a combination which suits their needs and use case. Decoder models include:

- PINN
- PKAN
- LR
- KAN

3.3.5 Model Tuning

GA STUFF

3.3.6 Output

As a scientific tool, we aim for visibility in our output. Enabling dropout layers within predictive models introduces randomness to predictions. By generating a number of predictions, we can study the distribution of model outputs. With a distribution of predictions, we can provide statistical measures such as the mean, median, confidence interval, etc. This reflects our desire for transparency, allowing users to make informed decisions with the resultant predictions.

Chapter 4

Design Concepts, Evaluation & Selection

4.1 User Interface Design

Figure 4.1 shows an initial design of the GUI, featuring the molecule creation window, as well as parameters and prediction results. Figure A.2 shows a logo design for the Python app.

4.1.1 Intended User Flows and Workflows

The proposed user interface is designed to facilitate the following workflow:

1. **Molecule Input**
    - Users can either draw molecular structures using an integrated editor or import existing structures in standard formats (e.g., SMILES, MOL).
2. **Model Selection**
    - Encoder Selection: Users must select a baseline encoder module to process generated molecule.
    - Decoder Selection: Additionally, a decoder module can be selected, expanding the prediction pipeline.
3. **Prediction Execution**
    - Upon submission, the system processes the input through the selected predictive pipeline, with progress indicators informing users of the computation status.
4. **Results Display**
    - The interface presents the predicted a and b constants alongside confidence intervals. Visual aids, such as graphs comparing predicted values to known data, provide contextual understanding.

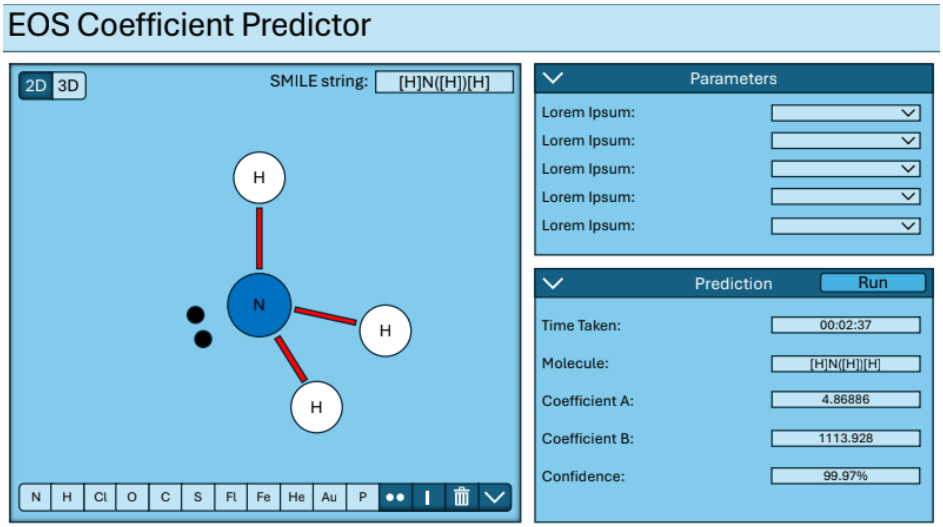

Figure 4.1: GUI Mockup

5. **Data Export**
    - Users have the option to export results in various formats (e.g., CSV, PDF) for integration into external reports or further analysis.

4.1.2 Target Use Scenarios

The tool is intended for the following user scenarios:

- **Chemical Engineers**
    - Predicting gas behavior in new process designs to aid in selecting appropriate materials and conditions.
- **Pharmaceutical Researchers**
    - Assessing molecular interactions under non-ideal conditions to inform drug formulation and stability studies.
- **Academic Educators**
    - Demonstrating real gas behavior concepts in educational settings, providing students with hands-on predictive capabilities.

4.1.3 Design Validation Plan

To ensure the interface meets user requirements, the following evaluation plan is proposed:

- **Usability Testing**

  - Engage a sample group of target users to perform typical tasks, collecting feedback on intuitiveness and efficiency.

- **Iterative Refinement**
  - Based on user feedback, iteratively refine the interface to address identified issues and enhance user satisfaction.

4.2 Molecular Embedding Techniques

Alternatives Considered

1. **Extended-Connectivity Fingerprints (ECFP)**
   - Circular fingerprints capturing molecular substructures.
2. **Graph Neural Network-based Embeddings**
   - Learn representations by processing molecular graphs.
3. **SMILES-based Sequence Embeddings**
   - Utilize sequential representations of molecules for embedding.
4. **Coulomb Matrix**
   - Generates a matrix containing electrostatic forces between atoms in a molecule.

Predicted Performance

- **ECFP**
  - **Advantages**: Simple to compute; captures local structural information.
  - **Disadvantages**: May miss global structural context; fixed size may limit flexibility.
- **Model-based Embeddings**
  - **Advantages**: Learnable embeddings trained specifically on a and b. Fixed-length output.
  - **Disadvantages**: Higher computational cost; requires substantial training data.
- **SMILES-based Sequence Embeddings**
  - **Advantages**: Leverages existing sequence-based models; straightforward implementation.
  - **Disadvantages**: Sensitive to input representation variations; may not fully capture 3D structural information.

- **Coulomb Matrix**
  - **Advantages**: Represents electrostatic forces, highly relevant to molecular properties.
  - **Disadvantages**: Sparse, variable length matrices.

Selected Alternative

Model-based Embeddings are proposed due to their ability to robustly create fixed-length embeddings. Direct training on a and b gives these embeddings predictive power. Exploring model-based embeddings allow for high flexibility in architecture and input information.

4.3 Model Design Decision Analysis

Several critical design decisions will influence the development and effectiveness of the predictive model:

1. **Architecture of the CNN Component**
   - **Options**: Evaluate different CNN architectures, such as VGG16, ResNet, or ConvNeXt, to determine the most effective for feature extraction from molecular structures.
   - **Implications**: The selected architecture will impact the model's capacity to capture spatial hierarchies and local features within molecular data.
2. **Formulation of the PINN Component**
   - **Options**: Define the specific physical laws and constraints to incorporate into the PINN, such as conservation laws or known thermodynamic relationships.
   - **Implications**: Proper formulation ensures that the model's predictions are physically plausible and adhere to established scientific principles.
3. **Configuration of the KAN Component**
   - **Options**: Determine the structure of the KAN, including the number of layers and the types of univariate functions used.
   - **Implications**: This configuration affects the model's ability to capture complex, nonlinear relationships within the data.

Selected Alternatives

Each model has advantages and disadvantages. The final deliverable includes a host of models, so selected alternatives reflect specific model performance.

Each design decision will be guided by specific requirements, such as achieving a target predictive accuracy, ensuring computational efficiency, and meeting user needs for interpretability. Our initial design test results can be seen in Figures 4.2 and 4.3. Figure 4.2 implies that we can accurately capture the patterns of a and b values, while 4.3 shows the results of predicting a and b separately for validation.

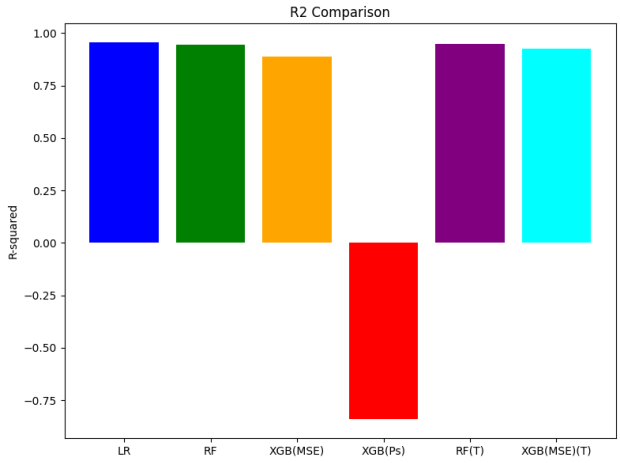

Figure 4.2: R2 Score of Initial Models

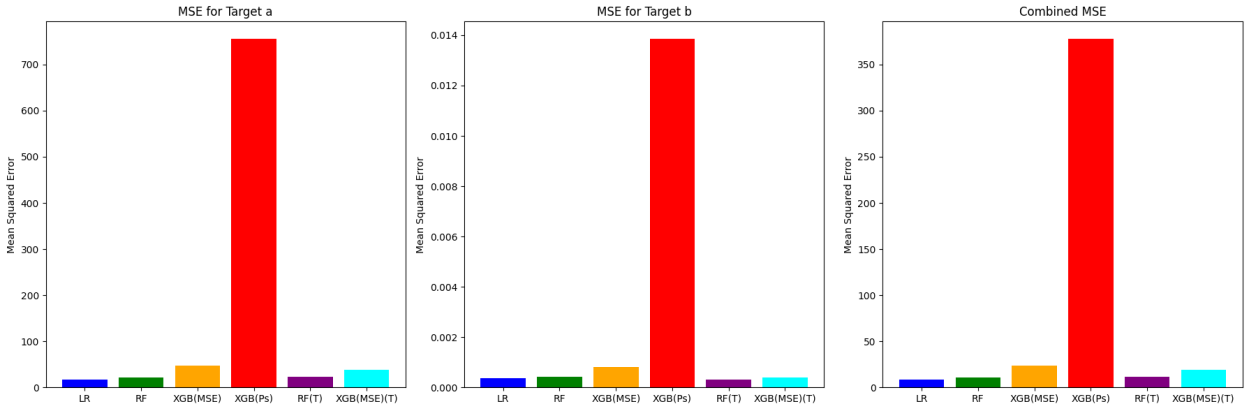

Figure 4.3: Individual a/b MSE Score of Initial Models

Chapter 5

Product Development and Evaluation Plan

5.1 Architecture Selection

5.1.1 Input Selection

One of the biggest choices to make in designing a machine learning pipeline is what the input will be. For our problem, the targets are clear, being a and b, while the input must be some representation of a given molecule. Every molecule can be identified by a unique SMILES string, which, using rdkit [6] we generated a dataset, containing the following features:

- **Molecular Weight** – The sum of the atomic weights of all atoms in a molecule, typically measured in Daltons (Da). It provides an estimate of the molecule's size and mass.
- **LogP** – The logarithm of the octanol-water partition coefficient, which measures a molecule's hydrophobicity. Higher LogP values indicate greater lipophilicity, meaning the molecule is more likely to dissolve in fats rather than water.
- **Topological Polar Surface Area** – A measure of the polar surface area (in $Å^2$) of a molecule, derived from its structure. Higher TPSA values typically indicate greater solubility in water and lower permeability through cell membranes.
- **Rotatable Bonds** – The number of single (non-ring) bonds that allow free rotation. This property affects molecular flexibility, which can influence bioavailability and binding to biological targets.
- **H Bond Donors (Hydrogen Bond Donors)** – The number of hydrogen atoms attached to electronegative atoms (such as oxygen or nitrogen) that can donate hydrogen bonds. This property impacts solubility and intermolecular interactions.
- **H Bond Acceptors** – The number of electronegative atoms (like oxygen and nitrogen) that can accept hydrogen bonds. Similar to donors, this property affects solubility and binding interactions.

- **Aromatic Rings** – The count of benzene-like (aromatic) rings in the molecule. Aromaticity is important in drug design, affecting stability, binding interactions, and electronic properties.
- **Number of Rings** – The total count of ring structures (both aromatic and non-aromatic) in a molecule. This influences structural rigidity and interaction with biological targets.
- **Atom Count** – The total number of atoms in the molecule, providing a basic measure of molecular size.
- **Coulomb Matrix** – A numerical representation of a molecule's atomic structure based on Coulombic (electrostatic) interactions between nuclei. It encodes molecular shape and electronic structure, making it useful for machine learning applications in cheminformatics.
- **Molecular Images** - 2D RGB images showing atoms, bonds, and molecular structure.
- **Molecular Graphs** - A matrix representation of a molecular graph. The molecular graph represents a molecule using atomic number as nodes, bonds as edges, and charge relations as edge weights.

These features provide a large variety of molecular information, giving us options for model architectures and input vectors.

5.1.2 Model Evaluation Metrics

To fairly evaluate developed model architectures, we define a standard validation metric. Our targets are float values, so metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) provide us with meaningful evaluations of our model performance. These metrics capture the distance between predicted and true labels, and are standard use for regression problems. A more abstract metric, the coefficient of determination (R^2), allows us to see if the model can explain the variance within the data, or if it is simply predicting within the mean and standard deviation. A combination of these metrics are used to select and compare model architectures.

5.2 Architecture Iterations

5.2.1 Initial Design

The initial predictive model design of this project consisted of a multi-modal pipeline. Using molecular images and molecular properties, we designed a CNN and PINN which would aggregate results and use a KAN model to create final predictions. The main goal of this model was to capture as much molecular information as possible, with the CNN learning spatial relationships, and the PINN capturing physical system constraints. The KAN model was intended to be used for increased non-linearity, revealing complex molecular relationships.

However, we faced a severe overfitting problem with a pipeline of this scale. With such complexity in the models and a relatively small dataset, the models could not generalize. Therefore, we decided to change the architecture.

5.2.2 GNN-PKAN Pipeline

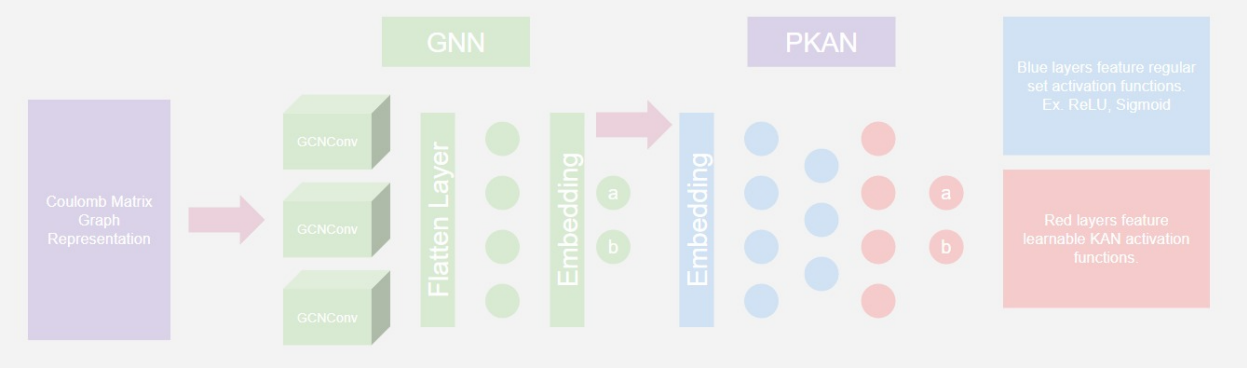

Figure 5.1: GNN-PKAN Pipeline

Our next pipeline design borrows heavily from the main ideas of the initial design, aiming to capture spatial relationships, molecular properties, and physical constraints, while maintaining non-linearity and complexity within the predictions. To combat the clear overfitting issue of the initial design, we reduced the number of models by combining aspects of the KAN and PINN models, which we named PKAN. This model also drastically reduces the number of input features, using only generated molecular graphs.

Graph Neural Network (GNN)

The GNN serves as an encoder for the PKAN model, producing a fixed length embedding which is passed on. GNNs, similar to CNNs, use convolutional layers to extract information from matrices. Figure 5.2 shows these GCNConv layers, as they are called. Mathematically, they can be described by the equation: [7]

$$X' = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}XW \tag{5.1}$$

Where $\hat{A} = A + I_N$ and A is the adjacency matrix to undirected graph $G(V, E)$, which is computed before training. $\hat{D_{ii}} = \sum_{j=-0} \hat{A}_{ij}$, a diagonal matrix of $\hat{A}$. After the GCNConv layers, we flatten and move to FC layers. We train on targets a and b, ensuring that we can get predictive capabilities from the GNN model alone. The box labeled "Embedding" in Figure 5.2 represents the last layer of the FC network. This layer is where we will pull our fixed length encoding for the PKAN to use.

Physics-Informed Kolmogorov-Arnold Network (PKAN)

The final step of the model, we use the fixed length encoding from the GNN to predict a and b. The PKAN architecture follows the design philosophy of a PINN, which is shown

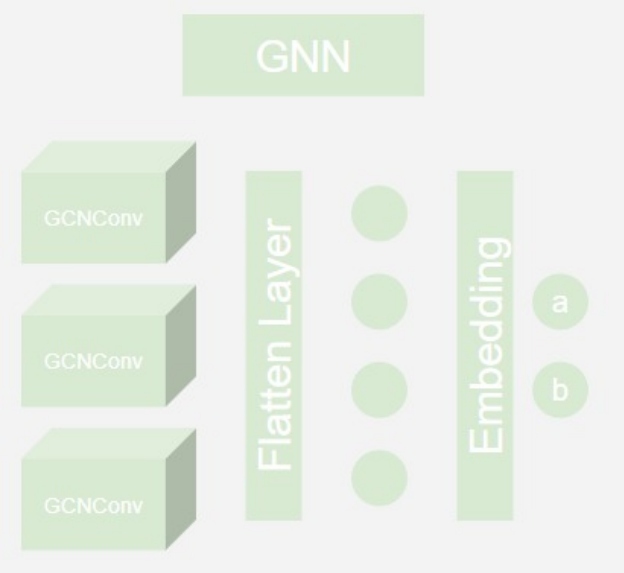

Figure 5.2: GNN Architecture

Figure 5.3: PKAN Architecture

in Figure 5.4. We differ in our output layers, borrowing the KAN activation layer [8] for increased non-linearity. The KAN layer can be described by the equation:

$$\phi(z) = \sum_{i=1}^{K} w_i k(z, c_i) \tag{5.2}$$

Where $k(z, c_i)$ is a kernel function which acts on input vector z. It features a learnable center, c, and width, σ. We also have the learnable weight w_i, which acts to weigh the outputs before summation. The model is also trained on a and b targets, as the GNN was, but we will take the output as our final predictions this time. Figure 5.4 shows the central idea of a PINN architecture, featuring the physical constraints that bind our system. We have identified constraints:

- $a > 0$
- $b > 0$

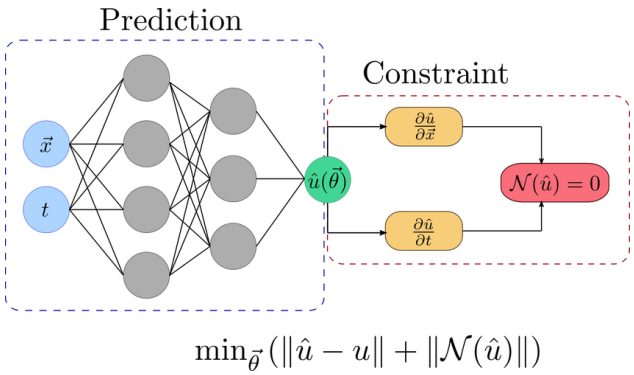

Figure 5.4: PINN Architecture [1]

- When ordered by molecular weight: $a_i \leq a_{i+1}$, $b_i \leq b_{i+1}$, $\forall i \in \{1, ..., N-1\}$

These constraints are incorporated into the Physics Loss, which is linearly combined with the Model Loss to produce our Total Loss. The Physics Loss is defined as:

$$\begin{aligned} L_{pinn} &= L_{physics} + L_{trend} \\ L_{physics} &= \frac{1}{N}\sum_{i=1}^{N}(max(0, -a_i) + max(0, -b_i)) \\ L_{trend} &= \frac{1}{N}\sum_{i=1}^{N}(max(0, a_i - a_{i+1}) + max(0, b_i - b_{i+1})) \end{aligned} \tag{5.3}$$

In tandem, we calculate the Model Loss using the Mean Squared Error from our predicted (a, b) and the actual targets. We calculate the total loss as:

$$L = A(L_{model}) + B(L_{pinn}) \tag{5.4}$$

GNN-PKAN Evaluation

The biggest factor in abandoning this pipeline was the inability to represent all molecules in our dataset with a molecular graph. This would reduce the diversity of our dataset and limit the model in terms of allowed molecule inputs. For these reasons, we continued to iterate on this design concept.

5.2.3 CNN-PKAN Pipeline

To address the GNNs inability to accept all molecule inputs, we explored a CNN model to provide encodings for the PKAN. Using our generated molecular images, we can train on all possible molecule SMILES strings.

CNN

In this pipeline, the CNN performs the same function as the GNN of the previous. Figure 5.5 shows the architecture of a CNN. The convolutional layers act as filters on the input images, extracting relevant features such as structural and RGB information. These can be interpreted as the model learning molecular structures and representations of elements based

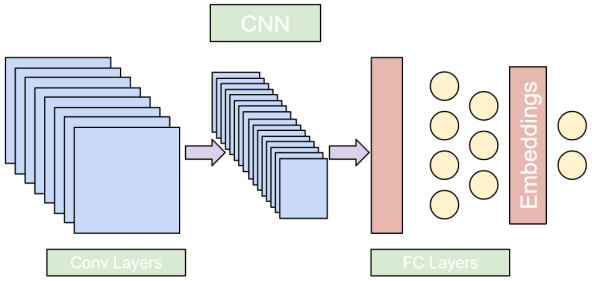

Figure 5.5: CNN Architecture [**?**]

on the given image. These extracted features are flattened and reduced to a fixed length vector for interpretation by the PKAN model.

A benefit of working with images is the ability to use data augmentation to increase dimensionality within the dataset. Randomly applying transformations, color jitter, and erasing to the molecule images allows the model to generalize better. If we define X as an input image matrix, the transformations applied to X are defined as follows:

$$X' = N \circ E \circ J \circ A \circ R \circ T(X)$$

where:

$$T(X) = \text{ToTensor}(X)$$

$$R(X) = \begin{cases} \text{RandomRotation}(10°, X), & \text{P}(0.5) \\ X, & \text{otherwise} \end{cases}$$

$$A(X) = \begin{cases} \text{RandomAffine}(X, \theta = 0, t_x \sim U(-0.1, 0.1), t_y \sim U(-0.1, 0.1)), & \text{P}(0.5) \\ X, & \text{otherwise} \end{cases}$$

$$J(X) = \text{ColorJitter}(X, \text{brightness} = 0.2, \text{contrast} = 0.2)$$

$$E(X) = \begin{cases} \text{RandomErasing}(X, p = 0.5, s \sim U(0.02, 0.1)), & \text{P}(0.3) \\ X, & \text{otherwise} \end{cases}$$

$$N(X) = \frac{X - 0.5}{0.5}$$

where:
- $U(a, b)$ denotes a uniform distribution over the interval $[a, b]$.
- t_x and t_y are the translation values along the x and y axes.
- s represents the erasing scale.

5.3 GUI Design

5.3.1 General Layout

The GUI, shown in Figures 5.6 and 5.7, has three main sections for interacting with the model pipeline

- **Molecule Input** - A SMILES string can be input in the text-box at the top of the section, upon hitting "Generate" a 2D representation of the molecule is displayed and the inputs to the model are saved to files. The molecule representation can be toggled between a 3D and 2D view. In addition to SMILES string input, a molecule builder popup will allow users to place atoms and bonds to create molecules without memorizing SMILES strings.
- **Settings** - This is where the user can switch which models they want running the prediction (Baseline or novel predictive models), or choose to run the full model pipeline. The Settings will also determine how the models output, such as outputting a calculated confidence interval, MSE, MAE, and R^2.
- **Output** - Upon hitting "Run", the model's output will be displayed in the box, containing the final a and b predictions. The output, as well as the model being run, is determined by the settings chosen in the Settings section.

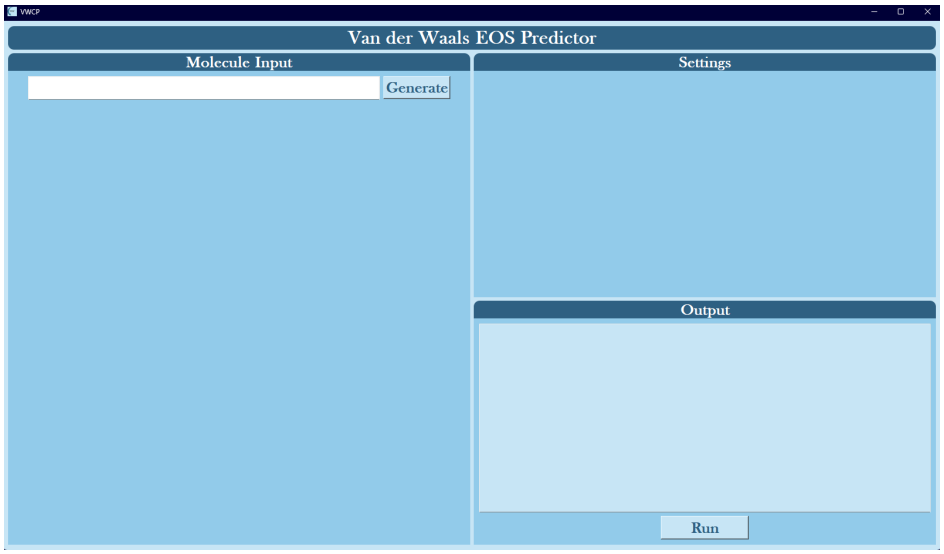

Figure 5.6: GUI upon application startup

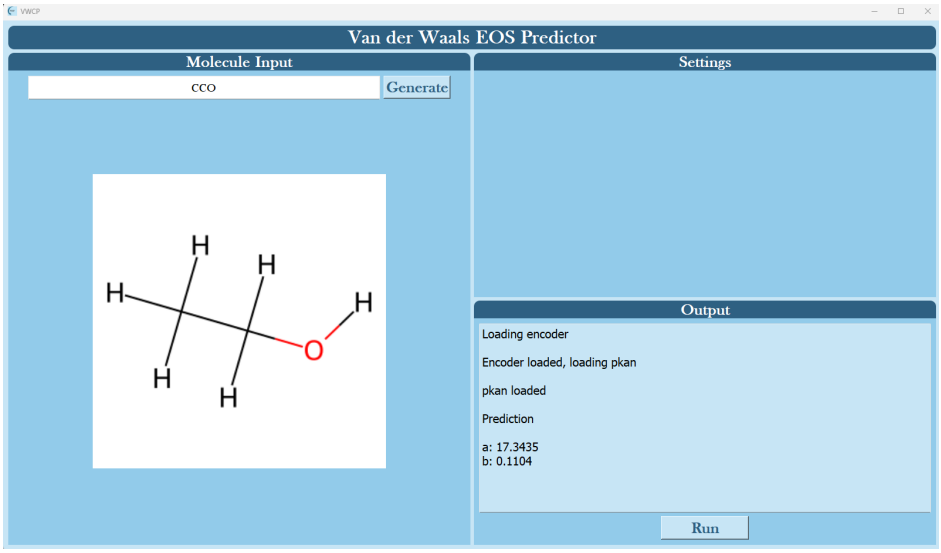

Figure 5.7: GUI after molecule has been generated and model is run

5.3.2 Technologies

The GUI is written in Python for easy integration with the model pipeline. It uses following Python packages to design the look and functionality of the GUI:

- **PyQt5** - The GUI package used to create the look and basic functionality of the program, it also handles text input and "Generate"/"Run" button functionality.
- **RdKit** - A general-purpose chemistry package, used in the GUI to read SMILES input and generate the 2D molecule representations.

5.4 GUI Evaluation

5.4.1 User Experience Survey

Multiple experienced and unexperienced users will be surveyed to evaluate the quality of the GUI. The users will provide feedback on what problems they found, as well as comment on the positive aspects of the GUI. Upon evaluation, the GUI will be updated to address problems and be given back to the users for more feedback.

5.4.2 Error Handling and Stability

The GUI will be rigorously tested to ensure that it can handle all inputs and will provide helpful error messages should an invalid input be entered. The model output will also be able to receive and display error messages from the model being run. The GUI will also be tested for stability and needs to properly handle any size of molecule input.

Chapter 6

Social Impact Evaluation

6.1 Recognizing and Defining Targeted Needs in a Social Context

This project addresses the need for accurate molecular property predictions, benefiting fields such as drug discovery, materials science, and environmental safety. By improving computational efficiency and predictive accuracy, we enhance scientific research and industrial applications.

6.2 Broader Impacts of Engineering Solutions

The integration of machine learning and physics-based modeling ensures that our system produces reliable and interpretable predictions. This contributes to advancements in cheminformatics, supports sustainable research practices by reducing reliance on costly laboratory experiments, and accelerates innovation in material and pharmaceutical development.

6.3 Ethical and Professional Responsibilities

Ensuring data integrity and model transparency is critical. Our approach prioritizes ethical AI development by:

- Using well-documented and publicly available molecular datasets.
- Clearly stating model limitations and potential biases.
- Ensuring reproducibility by providing open-source implementation details.

6.4 Communication on Societal Issues

Our project highlights the importance of computational methods in addressing global challenges such as efficient drug design and sustainable material development. We will commu-

nicate these impacts through presentations, publications, and collaborations with industry professionals and researchers.

6.5 Understanding Professional and Ethical Responsibilities

As engineering practitioners, we are committed to:

- Upholding rigorous scientific standards in data handling and analysis.
- Promoting accessibility and fairness in AI-driven predictions.
- Maintaining transparency in reporting findings and model performance.

Chapter 7

Deliverables

7.1 Overview

At the conclusion of this project, we will deliver a comprehensive molecular property prediction system, consisting of a user-friendly graphical interface, robust predictive models, and detailed documentation. This system is designed to enable researchers and industry professionals to efficiently analyze molecular properties and leverage machine learning in their workflows.

7.2 Final Deliverables

The project will produce the following key deliverables:

7.2.1 Software System

- **Molecular Property Prediction Pipeline:**
  - Preprocessing tools for generating molecular encodings (e.g., model-based Embeddings)
  - Full predictive model pipeline, including:
    - Multiple baseline predictive models that can produce high accuracy on the a and b constants.
    - Novel architectures that can perform equal or better than the baseline.
- **Graphical User Interface (GUI):**
  - Fully functional Python application with model-pipeline backend.
  - Interactive molecule input via drawing, file uploads, or SMILES entry.
  - Visualization tools for molecular graphs and predicted properties.

7.2.2 Documentation

- **User Guide:**
  - Instructions on how to install, configure, and use the system.
  - Examples of workflows for different use cases (e.g., drug discovery, materials optimization).
- **Technical Documentation:**
  - Detailed explanation of the system architecture, model training process, and hyperparameter optimization.
  - Guidelines for extending the system (e.g., adding new encodings or models).

7.2.3 Evaluation Results

- **Performance Metrics:**
  - Full model validation with training and test sets will be performed. Using predicted confidence intervals, we aim to capture the true values within our predicted range.
  - Validation results for prediction accuracy (R^2, mean squared error, and mean absolute error) across benchmark datasets.
- **User Feedback:**
  - Summarized findings from usability testing of the GUI.
  - Recommendations for future iterations based on user feedback.

7.2.4 Source Code and Deployment Tools

- Fully commented source code for the pipeline and GUI.
- Deployment scripts for running the system locally or on a cloud platform.
- Pre-trained models for immediate use without retraining.

7.3 Future Extensions

While not part of the primary deliverables, the system will be designed to facilitate future enhancements, such as:

- Integration with additional molecular encoding methods.
- Support for predicting more advanced molecular properties.
- Scalability to handle larger datasets and distributed computing environments.

Chapter 8

Project Management

8.1 Overview

Effective project management is critical to ensuring the successful and timely delivery of this molecular property prediction system. While the specifics of certain components may evolve, we have developed a preliminary timeline that outlines the major milestones and their corresponding deadlines. This schedule will be refined as the project progresses and new challenges or opportunities arise.

8.2 Project Timeline

Table 8.1 summarizes the key milestones and deliverables for the project:

8.3 Team Roles and Responsibilities

To ensure efficient progress, the following team roles have been defined:

- **Project Manager - Karim Chmayssani:** Oversees the project schedule, ensures deadlines are met, and coordinates between team members.
- **Data Scientist - Blake Milstead:** Develops preprocessing tools and ensures the accuracy of molecular encodings.
- **Model Developer - Karim Chmayssani, Blake Milstead:** Focuses on designing, training, and optimizing predictive models (GNNs, PINNs, KANs).
- **UI/UX Designer - Turner Heath:** Leads the design and implementation of the graphical user interface.
- **Tester - Turner Heath:** Conducts usability and performance testing, ensuring system reliability and user satisfaction.

Milestone	Deadline	Description
Initial Planning and Research	Week 1	Finalize project objectives, gather requirements, and review relevant literature.
Data Preprocessing Pipeline	Week 3	Implement and test tools for generating molecular encodings (Coulomb matrices, SMILES embeddings).
GUI Prototype	Week 5	Build a basic GUI prototype to input molecular data and display results.
GNN Model Development	Week 7	Train and evaluate the GNN for molecular graph analysis.
PINN Model Development	Week 10	Incorporate physics-informed constraints into a predictive model.
KAN Aggregation Implementation	Week 11	Develop and integrate the kernel attention network for prediction aggregation.
System Integration	Week 12	Combine all components (models, preprocessing, GUI) into a cohesive pipeline.
Performance Optimization	Week 15	Optimize models and preprocessing for speed and accuracy.
Usability Testing	Week 17	Conduct testing with end-users to gather feedback and refine the GUI.
Final Deliverables Submission	Week 18	Submit the completed system, documentation, and evaluation results.

Table 8.1: Preliminary Project Timeline

8.4 Future Refinements

This timeline and role allocation will be revisited at regular intervals to ensure alignment with project goals and to address unforeseen challenges. Updates will be documented and incorporated into subsequent phases of the project.

Chapter 9

Budget

9.1 Overview

This section outlines the preliminary budget estimates for the molecular property prediction project. The budget considers software, hardware, and personnel costs, with engineering time being the most significant expenditure. The estimates will be refined as the project progresses and specific requirements are finalized.

9.2 Estimated Expenditures

The anticipated costs are categorized as follows:

Category	**Estimated Cost (USD)**	**Description**
Software Licenses	0	All used software is free
Cloud Services	300	Google Colab computing hours, GPU computing credits ($10/100hrs)
Miscellaneous	100	Expenses for documentation, presentations, and potential third-party APIs for molecular data.
Total Estimated Cost	400	

Table 9.1: Preliminary Budget Estimates

9.3 Engineering Time Allocation

The project timeline is broken into milestones, with estimated human-hours allocated for each phase:

Milestone	**Estimated Hours**	**Description**
Initial Planning and Research	10	Literature review, requirements gathering, and project planning.
Data Preprocessing Pipeline	10	Implementing tools for molecular encodings and data validation.
GNN Development	30	Designing, training, and testing graph neural networks.
PINN Development	30	Incorporating physics-based equations into a predictive model.
KAN Implementation	30	Developing kernel attention networks.
GUI Development	20	Creating a user-friendly interface for input, prediction, and visualization.
System Integration and Optimization	10	Combining all components and optimizing performance.
Testing and Documentation	30	Usability testing, system evaluation, and preparing project documentation.
Total Estimated Hours	**170**	

Table 9.2: Estimated Human-Hours Allocation

9.4 Future Adjustments

This budget represents an initial estimate and will be updated as the project evolves. Actual expenditures and time allocations will be recorded and compared to these estimates to ensure accountability and adaptability.

Bibliography

[1] Unknown, "A sample schematic of a pinn architecture for the linear transport equation," https://www.researchgate.net/figure/A-sample-schematic-of-a-PINN-architecture-for-the-linear-transport-equation-u-t-u-x_fig5_344783581, Accessed 2024, accessed on 2024-12-03.

[2] S. M. College, "10: Experimental determination of the gas constant (experiment)," https://chem.libretexts.org/Ancillary_Materials/Laboratory_Experiments/Wet_Lab_Experiments/General_Chemistry_Labs/Online_Chemistry_Lab_Manual/Chem_10_Experiments/10%3A_Experimental_Determination_of_the_Gas_Constant_%28Experiment%29, 2020, accessed on 2024-12-03.

[3] S. Genheden, A. Reymer, P. Saenz-Méndez, and L. A. Eriksson, "Computational chemistry and molecular modelling basics," in *Computational Tools for Chemical Biology*, S. Martín-Santamaría, Ed. Royal Society of Chemistry, 2017, pp. 1–38. [Online]. Available: https://books.rsc.org/books/edited-volume/630/chapter/312482/Computational-Chemistry-and-Molecular-Modelling

[4] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," 2019. [Online]. Available: https://arxiv.org/abs/1903.02428

[5] Microsoft_Research, "Visnet: A general molecular geometry modeling framework for predicting molecular properties and simulating molecular dynamics."

[6] T. R. community, "Rdkit documentation," 2025, accessed: 2025-03-06. [Online]. Available: https://www.rdkit.org/docs/source/rdkit.Chem.html

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[8] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "Kan: Kolmogorov-arnold networks," *arXiv preprint arXiv:2404.19756*, 2024.

Appendix A

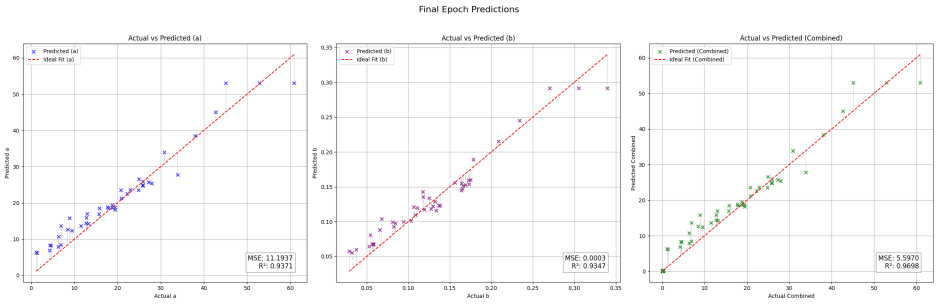

Figure A.1: Prediction Results with Initial Embeddings

Figure A.2: Equation of State Coefficient Predictor Logo (Van der Waals Coefficient Predictor)