# Van der Waals Equation of State Coefficient Predictor

Final Project Report
COSC 402
Blake Milstead, Karim Chmayssani, Turner Heath
May 3, 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 difficult to compute from first principles, are essential for understanding real gas behavior and have applications in fields such as chemical engineering, materials science, and pharmaceuticals. Traditional methods to determine these constants are experimentally intensive and time-consuming, motivating the need for an automated, data-driven solution.

This project is research-focused, with the goal of identifying the most effective pipeline for molecular property prediction. We systematically explored a range of model architectures, including baseline regression models, Convolutional Neural Networks (CNNs), Graph Neural Networks (GNNs), Physics-Informed Neural Networks (PINNs), and Kolmogorov-Arnold Networks (KANs). Our top-performing model, PKAN—a hybrid of PINN and KAN—achieved strong predictive performance when trained on handcrafted molecular descriptors, obtaining an $R^2$ score of 0.905 for $a$ and 0.710 for $b$ on inlier test data.

We also investigated an end-to-end deep learning pipeline, combining RDKit molecular image inputs, a self-supervised U-Net encoder, and a PKAN decoder. Although the U-Net successfully reconstructed masked molecular images, its embeddings lacked diversity and failed to encode chemically meaningful information, resulting in poor predictive performance. This insight shaped our conclusion: visual reconstruction quality does not guarantee chemically relevant representation.

From a user perspective, our envisioned software tool enables users to input molecular structures and receive predicted $a$ and $b$ constants, confidence intervals, and visualizations. Target users include chemical engineers, pharmaceutical scientists, and academic educators. Our validation plan includes usability testing, iterative model evaluation, and performance metrics such as mean squared error and $R^2$ score. Final deliverables include a functional application, trained models, source code, documentation, and deployment scripts—offering a complete, extensible platform for real gas property prediction.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Problem Definition & Background

## 1.1 Problem Statement

Predicting the behavior of real gases is a fundamental aspect of chemical engineering and physical chemistry. The Van der Waals equation introduces two empirically derived constants, $a$ and $b$, to account for intermolecular attractions and the finite volume of gas molecules, respectively. Traditionally, determining these constants requires extensive experimental procedures, which are both time-consuming and resource-intensive [2]. This project aims to develop a machine learning-based system capable of predicting the Van der Waals constants directly from molecular structures using image-based embeddings. Such a system would provide a scalable and efficient alternative to conventional experimental methods.

## 1.2 Affected Stakeholders

The accurate prediction of Van der Waals constants ($a$ and $b$) has significant implications across various scientific and industrial domains. In chemical engineering, precise modeling of gas behavior is essential for designing reactors, separation units, and other process equipment. Deviations from ideal gas behavior can lead to inefficiencies or safety hazards in large-scale operations [2].

In the pharmaceutical industry, understanding molecular interactions is crucial for drug discovery and development. Predicting molecular properties accurately can expedite the identification of viable drug candidates and reduce reliance on costly experimental procedures [3].

Materials scientists also benefit from accurate molecular property predictions. The development of new materials with desired properties often depends on understanding and manipulating molecular interactions. Machine learning approaches have been employed to predict properties of advanced materials, facilitating the design process [4].

Furthermore, educators in chemistry and physics can utilize predictive tools to demonstrate real-gas behaviors, enhancing the learning experience. Simulation software developers can integrate such predictive models to improve the accuracy and efficiency of their tools, benefiting researchers and industry professionals alike.

## 1.3 Design Requirements

The development of a machine learning-based tool for predicting Van der Waals constants necessitates a comprehensive set of design requirements to ensure accuracy, usability, and integration capabilities. The system must accept molecular inputs in standardized formats, such as Simplified Molecular Input Line Entry System (SMILES) strings, which are widely used for representing chemical structures in computational applications [5].

To capture the intricate structural features of molecules, the system should employ image-based molecular embeddings. Recent advancements have demonstrated the efficacy of using molecular images in deep learning frameworks for accurate property prediction [3]. By converting molecular structures into images, convolutional neural networks (CNNs) can extract spatial features that are pertinent to intermolecular interactions influencing the Van der Waals constants.

The predictive model must be trained on a diverse and extensive dataset of molecules with known Van der Waals constants to ensure generalizability and robustness. The training process should incorporate techniques to prevent overfitting and to handle potential data imbalances. Additionally, the model should be designed to output predictions efficiently, facilitating rapid assessments in research and industrial settings.

A user-friendly graphical user interface (GUI) is essential for broad accessibility. The GUI should allow users to input molecular structures, visualize the corresponding images, and receive the predicted Van der Waals constants. Interactive features, such as real-time prediction updates, can enhance user engagement and utility.

Furthermore, the system should be modular and scalable, allowing for future integration with other computational tools and databases. This modularity ensures that the tool can adapt to evolving research needs and technological advancements.

## 1.4 Usage Context

The Van der Waals constant prediction tool is intended to be used by a range of users in academic, industrial, and research settings, each operating under different constraints and computational environments. The primary interface is a cross-platform graphical user interface (GUI) that allows users to input molecular structures in the form of SMILES strings. Once a molecule is submitted, the system processes the structure into a two-dimensional image and feeds it into a trained machine learning model to produce the predicted $a$ and $b$ constants.

Academic users—such as students and instructors—are expected to utilize the tool in educational environments, particularly in physical chemistry, chemical engineering, and thermodynamics courses. In this context, the software can aid in demonstrating real-gas behavior, enhancing conceptual understanding without requiring the overhead of traditional computational chemistry software or laboratory experiments.

In research and industrial laboratories, the tool will serve as an efficient computational aid. Researchers working in materials science, drug design, or chemical process engineering can use the tool to rapidly prototype and compare molecules without needing to experimentally determine their real gas properties. This is particularly useful in early-stage

development, where high-throughput screening and model-driven insight are essential.

The software is designed to run on consumer-grade hardware, supporting Windows, macOS, and Linux environments. It is intended to be deployed either locally on a user's machine or through a cloud-hosted platform, allowing integration into larger computational pipelines or laboratory information management systems (LIMS). Developers and data engineers may also access model outputs via exported files or logs, making the tool extensible beyond the GUI.

Importantly, the design considers more than just the end-user. For instructors, the interface must be easy to demonstrate and interpret in a classroom. For developers, the backend must be modular and well-documented to allow integration with other tools. And for researchers and analysts, the prediction workflow must be reproducible and transparent enough to support publication, regulatory reporting, or further model refinement.

## 1.5 Theoretical Background

### 1.5.1 Van der Waals Equation and Real Gas Behavior

The ideal gas law, expressed as $PV = nRT$, assumes that gas molecules do not interact and occupy no volume. However, real gases exhibit deviations from this ideal behavior due to intermolecular forces and the finite size of molecules. To account for these factors, Johannes Diderik van der Waals introduced a modified equation:

$$\left(P + a\frac{n^2}{V^2}\right)(V - nb) = nRT$$

In this equation, the constant $a$ corrects for the attractive forces between molecules, while $b$ accounts for the finite volume occupied by the gas molecules themselves. These corrections enable more accurate predictions of gas behavior under non-ideal conditions [6].

### 1.5.2 Machine Learning in Molecular Property Prediction

Advancements in machine learning (ML) have significantly impacted the field of molecular property prediction. ML models can learn complex, non-linear relationships from data, enabling the prediction of various molecular properties such as solubility, toxicity, and reactivity. Central to the success of these models is the representation of molecular structures in a form amenable to computational analysis.

Traditional approaches often utilize graph-based representations, where atoms are nodes and bonds are edges, allowing models to capture the connectivity of molecules. However, alternative representations, such as image-based approaches, have gained attention for their ability to leverage the strengths of convolutional neural networks (CNNs) in processing spatial data [7].

### 1.5.3 Image-Based Molecular Representations

Image-based molecular representations involve converting molecular structures into two-dimensional images, which can then be analyzed using CNNs. This approach allows models

to capture spatial and structural information inherent in molecular diagrams. CNNs are particularly effective in extracting hierarchical features from images, making them suitable for identifying patterns related to molecular properties.

By training CNNs on large datasets of molecular images, models can learn to associate visual features with specific properties, facilitating accurate predictions. This method has been applied in various domains, including drug discovery and materials science, demonstrating its versatility and effectiveness [8].

### 1.5.4 Application to Van der Waals Constant Prediction

In the context of predicting Van der Waals constants, image-based representations offer a novel approach to capturing the structural nuances of molecules that influence intermolecular interactions. By utilizing CNNs trained on molecular images, it is possible to predict the $a$ and $b$ constants directly from the visual representation of a molecule, bypassing the need for extensive experimental measurements or complex quantum chemical calculations.

This approach aligns with the broader trend of integrating machine learning techniques into chemical and physical sciences, offering scalable and efficient alternatives to traditional methods.

## 1.6 Prior Work

The accurate determination of Van der Waals constants ($a$ and $b$) has been a subject of extensive research, employing various methodologies ranging from experimental measurements to advanced computational techniques.

### 1.6.1 Experimental Determination

Traditionally, the constants $a$ and $b$ are derived from experimental data by fitting the Van der Waals equation to observed pressure-volume-temperature (PVT) relationships of gases. This approach involves precise measurements under controlled laboratory conditions to capture the non-ideal behavior of real gases [9].

### 1.6.2 Analytical Thermodynamic Models

Analytical models based on statistical mechanics have been developed to provide theoretical insights into the behavior of real gases. These models aim to account for intermolecular forces and the finite size of molecules, offering corrections to the ideal gas law. However, their applicability is often limited to specific conditions and may not generalize well across different substances [10].

### 1.6.3 Computational Chemistry Approaches

Advancements in computational chemistry have enabled the use of molecular dynamics (MD) and quantum mechanical (QM) simulations to estimate thermodynamic properties, including Van der Waals constants. These methods involve simulating molecular interactions at

the atomic level to predict macroscopic properties. While offering high accuracy, they are computationally intensive and may not be practical for high-throughput applications [11].

### 1.6.4 Machine Learning Models

The emergence of machine learning (ML) has introduced new avenues for predicting molecular properties. ML models, particularly those employing graph-based representations of molecules, have been utilized to predict properties such as boiling points and solubility. However, recent studies have highlighted limitations in the performance and interpretability of graph neural networks (GNNs) in certain contexts [12].

### 1.6.5 Image-Based Representations

To address the limitations of graph-based models, alternative approaches using image-based representations of molecular structures have been explored. By converting molecular structures into two-dimensional images, convolutional neural networks (CNNs) can be employed to capture spatial features relevant to molecular properties. This method has shown promise in various applications, including drug development and material science [8].

## 1.7 Comparable Technologies

Several existing tools and technologies offer partial solutions relevant to the prediction of Van der Waals constants ($a$ and $b$), though none are specifically tailored for this purpose.

### 1.7.1 RDKit

RDKit is an open-source cheminformatics toolkit widely used for molecular feature generation, including the computation of molecular fingerprints and descriptors. Its versatility and integration with Python make it a valuable resource for preprocessing molecular data in machine learning pipelines [13].

### 1.7.2 Quantum Chemistry Suites

Quantum chemistry software packages such as Gaussian and ORCA provide high-accuracy predictions of molecular properties through ab initio and density functional theory (DFT) calculations. Gaussian is a comprehensive suite capable of modeling a wide range of chemical systems [14], while ORCA offers efficient implementations of various quantum chemical methods and is freely available for academic use [15]. Despite their accuracy, these tools are computationally intensive and may not be practical for high-throughput predictions of Van der Waals constants.

### 1.7.3 Cheminformatics Machine Learning Libraries

Several cheminformatics libraries incorporate machine learning techniques to predict molecular properties. The BioChemical Library (BCL) integrates traditional cheminformatics tools

with machine learning-based quantitative structure-activity/property relationship (QSAR/QSPR) modeling [16]. Additionally, ML4Chem is an open-source library designed for developing and deploying machine learning models in chemistry and materials science, offering modules for data preparation, model training, and inference [17]. While these libraries provide general-purpose property prediction capabilities, they lack specialized training on Van der Waals coefficients.

### 1.7.4 Our Approach

Our tool aims to unify the capabilities of these existing technologies into a single, user-centric platform specialized for predicting Van der Waals constants. By leveraging image-based molecular representations and convolutional neural networks, our approach seeks to provide accurate and efficient predictions tailored to the specific task of determining $a$ and $b$ constants.

# Chapter 2

# Requirement Specification

## 2.1 Overview

The goal of our project is to build an accurate, scalable system that predicts Van der Waals gas constants ($a$ and $b$) using image-based deep learning models. The system will feature an interactive GUI and modular architecture that allows researchers to input molecular structures and receive real-time property predictions. Functional and non-functional requirements were derived through internal team meetings, exploratory prototyping, reference to academic literature [3,8,11], and feedback from chemical engineering students and domain researchers.

## 2.2 Functional Requirements

**FR1 Accept SMILES Input:** Users must be able to input molecules as SMILES strings [5]. This is a widely adopted standard in cheminformatics, making the system compatible with most open-source and proprietary molecular databases.

**FR2 Image-Based Embedding:** The backend must transform molecular SMILES into 2D image representations for processing by a convolutional neural network (CNN). This model will then generate a fixed-length embedding, trained to encode information relevant to $a$ and $b$ [8].

**FR3 Predict $a$ and $b$ Coefficients:** The system must return floating-point predictions of the Van der Waals coefficients. The pipeline should handle both high and low molecular complexity within the training distribution.

**FR4 Graphical User Interface (GUI):** The GUI should support SMILES input, model selection, and result visualization in tabular and graphical form. It should operate across platforms (Windows, macOS, Linux).

**FR5 Model Management System:** Users must be able to choose between at least three trained models, enabling comparison between architectures such as CNNs, PINNs, and hybrid methods like PKANs [18, 19].

## 2.3 Non-Functional Requirements

**NFR1 Prediction Accuracy:** On benchmark data, models should achieve an $R^2$ score of at least 0.75 and MSE below 30 for $a$ and a score of at least 0.75 and MSE below 0.05 for $b$ on test sets. These thresholds are supported by state-of-the-art results on similar molecular prediction tasks [7, 11].

**NFR2 Runtime Performance:** A trained model must return predictions in under 3 seconds for a single molecule, ensuring responsiveness in GUI usage scenarios.

**NFR3 Modularity and Reproducibility:** The system must allow for modular model replacement and reproducible experiments by packaging all dependencies and saving metadata (random seed, timestamp, hyperparameters) for each run.

**NFR4 Cross-Platform Usability:** The GUI must function identically across at least two operating systems and must not require GPU acceleration to run.

## 2.4 Metrics and Targets

| **Metric** | **Target** | **Requirement ID** | **Justification** |
|---|---|---|---|
| $R^2$ Score (test set) | $\geq 0.75$ | NFR1 | Reflects variance explained by the model; minimum threshold for scientific usability. |
| MSE (test set) | $\leq 30/0.05$ | NFR1 | Ensures tight error bound; validated via domain expert feedback. |
| Prediction Time | $\leq 3$ sec | NFR2 | Required for interactive GUI use; benchmarked on CPU hardware. |
| GUI Compatibility | Windows + macOS | NFR4 | Broadest user accessibility among researchers and educators. |
| Model Options | $\geq 3$ models | FR5 | Allows comparative evaluation and user-driven selection. |

Table 2.1: Project Metrics and Target Thresholds

# Chapter 3

# Technical Approach

## 3.1 Overview

To begin a design plan, we need a clear view of the project space, and what it defines and requires of us. We began with an initial problem statement: *Given a molecule, predict a and b.*

Investigating the terms of our problem statement shows a clear flow and provides an operative term—predict. To predict, we need a *predictive model.* Predictive models encapsulate a wide and varied range of models, techniques, and algorithms which can learn relationships within a data space and perform predictions on new, unseen data with high accuracy. Predictive models demand some input and, optionally, targets for those inputs. Our problem statement makes our inputs and targets clear, being molecules and their Van der Waals constants, respectively. With inputs and targets defined, *supervised machine learning* becomes a clear option for the development of our predictive model.

Supervised machine learning models operate on *feature vectors* and output some prediction, which will be $a$ and $b$ in our case. Therefore, we must map our input, a molecule, to one a machine learning model can understand and work with, a feature vector.

A foundation has been laid: *Given a molecule, map to a feature vector and, using supervised machine learning, predict a and b.* To encapsulate our solution and facilitate input, interaction, and output, a user interface is required as well. This overview gives us the base to build our solution from.

As the practice of predicting Van der Waals constants using machine learning techniques is largely unexplored, this project takes on a more research-focused nature. Therefore, an iterative design approach will be followed in the development of our final predictor pipeline. To begin the exploration of this field, we will create a repository of baseline models of increasing complexity, culminating in the design of our fully novel pipeline. This approach provides us with motivation regarding specific design components and multiple predictive options for the user.

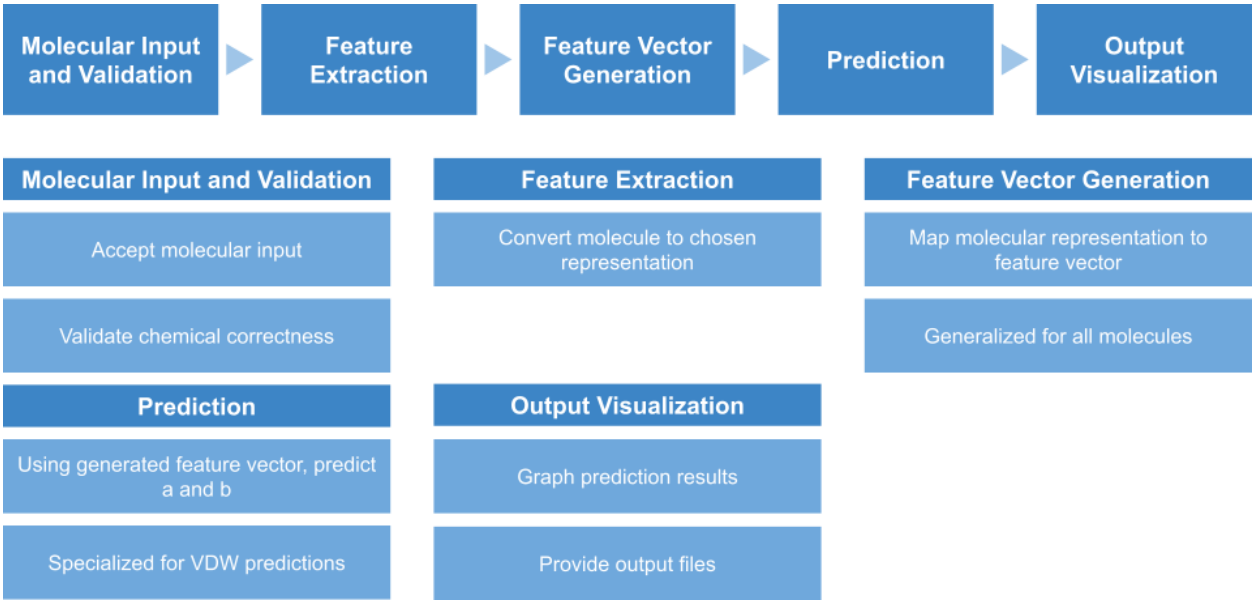


Figure 3.1: Functional Decomposition Block Diagram

## 3.2 Functional Decomposition

Our exploration of the problem space provides us with a framework of five functional blocks. While each block varies greatly in function and purpose, they are deeply connected, with changes in one requiring refactoring of the following. We explore the purposes, requirements and relations of the functional blocks in this section.

### 3.2.1 Molecular Input and Validation

This block defines the entry point to the system. It is responsible for receiving molecular input and ensuring that it is chemically and structurally valid before processing. The input format may vary—users might supply molecular formulas, SMILES strings, or draw molecular images through an interface. Regardless of the form, the system must validate the molecule to confirm it meets fundamental chemical rules (e.g., proper bonding and valency). This validation ensures downstream modules receive well-formed data and reduces the likelihood of ambiguous behavior later in the pipeline.

### 3.2.2 Feature Extraction

Once a molecule has been validated, the next block is responsible for extracting meaningful information from its structure. This may involve identifying atom types, bond types, spatial configuration, physicochemical properties, or other molecular characteristics. At this stage of the design process, we remain agnostic about the exact representation of the molecule—whether it is image-based, graph-based, or purely numerical—but we acknowledge that the system must translate raw molecular form into some structured internal representation.

This block must support flexible mappings depending on upstream input formats and downstream learning algorithms. It also plays a critical role in generalizing to a broad range of molecules and maintaining chemical relevance in the derived features.

### 3.2.3 Feature Vector Generator

This block bridges the gap between raw extracted features and machine learning inputs. It is responsible for encoding the molecule into a fixed-length numerical representation, often referred to as a feature vector or embedding. The chosen method for this step can have a significant impact on both predictive performance and interpretability. A prominent requirement for this block is the ability to process any molecule that it encounters.

Depending on the final approach, this transformation may be learned (via neural networks) or predefined (via hand-crafted descriptors). Regardless of the approach, the output of this block must be compatible with the learning models that follow and sufficiently expressive to encode the relevant relationships between molecular structure and target properties.

### 3.2.4 Prediction

The prediction block accepts the feature vector and returns numerical estimates of the Van der Waals constants. This is the core analytical component of the system and must be designed to maximize accuracy, generalization, and robustness. It may optionally incorporate mechanisms for estimating uncertainty or enforcing physical plausibility (e.g., monotonicity, non-negativity), depending on customer needs and application context.

### 3.2.5 Output Visualization

The final block handles user interaction and result presentation. It takes the predicted outputs and delivers them in a usable, interpretable form. This could include numerical output, confidence intervals, visual plots, molecule renderings, or downloadable reports.

Because this block is responsible for bridging technical output and user understanding, its design will be tightly linked to usability goals and user experience requirements. It must clearly convey both the prediction results and any relevant indicators of reliability or error, without overwhelming or misleading the user.

## 3.3 Design Choices

The functional decomposition outlined in the previous section provides a high-level view of the system's core components. Each of these blocks introduces opportunities—and in some cases, obligations—for engineering decision-making. In this section, we identify the most significant design decisions we will encounter as we transition from system architecture to implementation. While specific solutions are deferred to later sections, the following choices represent the key axes along which our design space varies.

### 3.3.1 Molecular Representation and Input Format

One of the earliest and most impactful decisions involves how molecular structures will be represented internally. Options include SMILES strings, molecular graphs, numerical descriptor vectors, and 2D or 3D images. This choice directly influences compatibility with downstream feature extraction methods and machine learning models, as well as the flexibility of user input methods. It also impacts the system's ability to generalize across diverse molecular structures and maintain a consistent interface for different molecule types.

### 3.3.2 Feature Extraction Strategy

Once a molecule is represented, we must decide how to extract meaningful information from it. This could involve predefined descriptors from cheminformatics toolkits, learned representations from neural networks, or generated visual representations. This decision affects prediction accuracy, computational efficiency, and how well the system scales to user-defined or uncommon molecules. The extraction strategy must preserve the essential chemical and structural information required for reliable prediction.

### 3.3.3 Feature Vector Dimensionality and Embedding Method

The system must convert extracted features into fixed-length numerical vectors to be used as model inputs. This mapping could be manual, algorithmic, or learned during training. The dimensionality and construction of these vectors affect both the expressiveness of the model and the ease of training. The choice here also affects compatibility with different learning architectures and has implications for interpretability, especially if users wish to understand how features contribute to predictions.

### 3.3.4 Model Architecture and Training Objective

A central decision lies in selecting the predictive model architecture. Options include regression models, deep neural networks, graph-based architectures, or models augmented with physical constraints. We must also determine how the model will be trained, with decisions regarding loss functions, optimizers, and many smaller choices made within this block. This design component will strongly affect predictive accuracy, robustness, and the model's ability to reflect real-world physical relationships.

### 3.3.5 User Interface and Interaction Design

Finally, we must decide how users will interact with the system, from molecule input to prediction and result visualization. The interface must accommodate a range of user types, from domain experts to students, while maintaining clarity and accessibility. These decisions will influence user satisfaction, ease of use, and how readily the tool can be adopted in different contexts, whether academic, industrial, or educational.

# Chapter 4

# Design Concepts, Evaluation & Selection

## 4.1 User Considerations and Evaluation

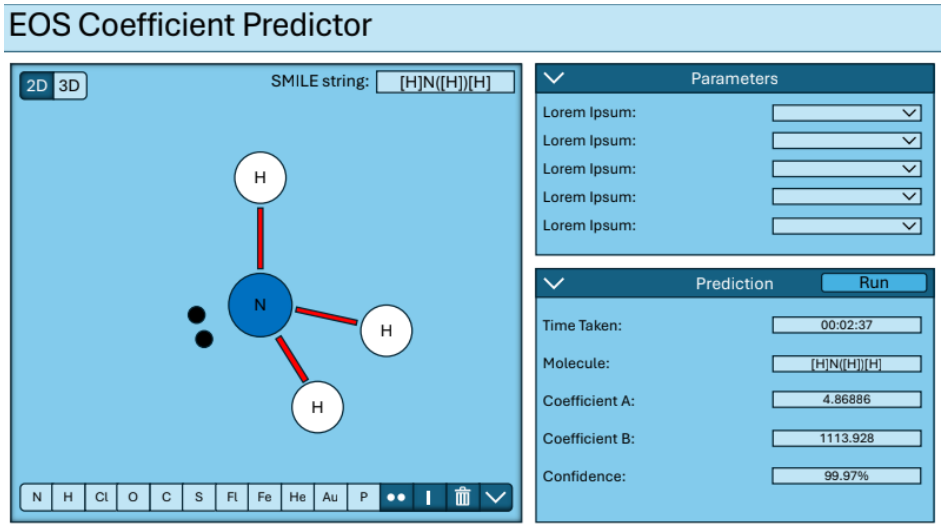


Figure 4.1: GUI Mockup

Before diving into individual technical design alternatives, it is essential to contextualize the system in terms of its human-facing goals. Our project is a usable tool designed for scientists, students, and engineers who need to quickly and reliably predict Van der Waals constants from molecular structures. As such, human-centered design principles played a major role throughout the architecture development process.

To ensure usability and accessibility, we began by outlining several key workflows from the perspective of the end user. The most common scenario involves a researcher or student entering a molecule—typically in SMILES format—and requesting a prediction of its Van der Waals constants. The user then reviews the results, including predicted values, confidence intervals, and supporting information such as molecular structure or error bounds. In more

advanced use cases, the user may wish to compare multiple models, adjust settings, or export the results for use in a lab notebook, spreadsheet, or simulation environment.

A sketch of the GUI design can be seen in Figure 4.1. The GUI allows the user to paste or type a SMILES string, optionally visualize the molecule, select a prediction model, and click a single button to compute and view results. Confidence intervals are displayed alongside the predictions to communicate uncertainty, and results can be exported as text or CSV files. The GUI was developed using PyQt5, which provided a stable, cross-platform framework and the flexibility to integrate cheminformatics components such as RDKit for visualization.

To validate the user experience, we developed a lightweight evaluation plan. Users were asked to perform typical tasks such as submitting a molecule and reading the output, and we recorded the time taken, number of input errors, and overall task success. We also collected feedback through a short post-test survey. These informal tests guided the final iteration of our interface, ensuring it was intuitive and aligned with the needs of our target audience.

This user-focused design foundation shaped the technical decisions described in the following sections. Each design choice was evaluated not only for performance or implementation complexity but also for how well it supported the overall user experience.

## 4.2 Molecular Representation and Input Format

A critical design decision involved determining how users would input molecular structures into the system. This representation format directly affects usability, interface design, and the ease with which we can transform user-provided molecules into structured data for model prediction. We explored three options: SMILES strings, molecular symbols, and constructed molecule images.

SMILES strings are a compact, text-based format widely used in chemical databases and research tools. They offer a highly scalable input method that can be easily entered into a text box, copied from online resources, or batch-loaded from files. SMILES strings are directly convertible into detailed molecular structures using cheminformatics libraries, allowing seamless integration with downstream feature extraction methods. Their consistency, portability, and ease of validation make them especially suitable for users who are comfortable working with chemical software or digital workflows. However, they may be unintuitive for users without prior exposure to chemical encodings.

Molecular symbols, such as chemical formulas or fragment shorthand (e.g., “COOH”, “benzene ring”), were considered for their accessibility and readability. This approach emphasizes simplicity and aligns with traditional classroom or textbook chemistry. However, symbols are inherently ambiguous and context-dependent, lacking the formal structure needed for automatic parsing or model-ready feature extraction. While they may be easy to type or recognize, their lack of standardization limits their utility for automated workflows or scalable deployment.

Constructed molecule images, drawn using a graphical molecule editor, provide a highly visual and intuitive input method. Users can draw molecules directly, mimicking the process of sketching on paper or using software like ChemDraw. This method is particularly accessible to students or researchers unfamiliar with SMILES. Once drawn, the molecules can be converted into internal representations using cheminformatics libraries. However,

this process adds interface complexity and introduces potential inconsistencies in structure interpretation, especially if stereochemistry or connectivity is unclear. It is also slower and less scalable for users who wish to test large numbers of molecules.

After evaluating these alternatives, we selected SMILES strings as the primary input format. This decision was driven by their compatibility with automated feature extraction, ease of integration into both GUI and CLI workflows, and their alignment with user needs in research, education, and high-throughput experimentation. The system remains modular enough to support visual input extensions in the future, but SMILES provides a robust and standardized foundation.

## 4.3 Feature Extraction Strategy

Once a molecule is input into the system, it must be transformed into a format suitable for machine learning models. This step—feature extraction—defines how chemical structure is translated into numerical information that a model can interpret. We evaluated several approaches to this transformation, each with distinct strengths and limitations in terms of expressiveness, standardization, and compatibility with different modeling strategies.

One option we considered was the Coulomb matrix, a physics-inspired representation that encodes atomic numbers and interatomic distances into a symmetric matrix. This method preserves spatial relationships and has proven effective in some chemical property prediction tasks. However, Coulomb matrices are sensitive to molecular orientation and atom ordering, requiring careful sorting or padding strategies to ensure consistency. Their variable size and dependency on 3D geometry also posed integration challenges within our fixed-input architecture.

We also explored molecular images generated through RDKit. These 2D visualizations of molecular structures capture connectivity, atom identity, and bond type in a spatially organized format. From a machine learning perspective, these images provide a fixed-size, grid-like representation ideal for use with convolutional architectures. Moreover, because they are generated consistently from SMILES strings using a standardized toolchain, they are highly reproducible and format-agnostic—suitable for a wide range of molecules without requiring additional structural information or conformer generation. Their visual nature also aligns well with how chemists interpret molecules, providing an intuitive link between structure and output.

A third approach was graph-based embeddings, where molecules are modeled as graphs with atoms as nodes and bonds as edges. These embeddings, typically derived using graph neural networks, are powerful in that they retain explicit structural relationships and can capture intricate patterns in molecular topology. However, they are more complex to implement, often require dataset-specific tuning, and produce variable-size structures that complicate integration with standardized downstream architectures.

Lastly, we considered handcrafted molecular features such as molecular weight, LogP, number of atoms, and other descriptor-based inputs. These are simple to compute, interpretable, and work well in traditional regression pipelines. However, they rely heavily on domain-specific assumptions and may fail to capture the nuanced structural or electronic characteristics needed for fine-grained prediction tasks.

After assessing all four approaches, we selected RDKit molecular images as our feature extraction method. Their universality, consistency, and fixed input size aligned best with our goals of building a scalable, modular, and visually interpretable system. The chosen format enables smooth integration with our feature vector generator architecture and ensures compatibility with a wide range of molecular inputs.

## 4.4 Feature Vector Dimensionality and Embedding Method

After feature extraction, the system must compress the resulting data into a fixed-length vector representation that can be used by predictive models. This stage is crucial, as the embedding must retain the molecular information necessary to accurately estimate Van der Waals constants while remaining computationally efficient and model-compatible. We explored three primary approaches to generating these embeddings: standard convolutional neural networks (CNNs), graph neural networks (GNNs), and a modified U-Net architecture trained using masked autoencoding.

CNNs were a natural first choice given our use of molecular images. These models are well-established for visual feature extraction and have been successfully applied to chemical imaging in previous literature. CNNs are relatively lightweight, easy to train, and can learn local structural patterns such as rings or functional groups. However, their fixed receptive fields and lack of long-range relational awareness can limit their ability to capture global molecular context, especially when molecules vary in size, shape, or complexity.

GNNs provide a structure-aware alternative, capable of directly processing molecular graphs where atoms are nodes and bonds are edges. This representation naturally reflects chemical bonding and topology. GNNs can learn relational and context-dependent features, making them powerful tools for molecular property prediction. However, they require substantial preprocessing, are less compatible with image-based inputs, and tend to be more sensitive to dataset size and graph connectivity assumptions. Their implementation also adds complexity to the system and introduces challenges with standardization and batching.

To address the limitations of these approaches, we developed a U-Net architecture trained with a masked autoencoding (MAE) objective. This design combines the spatial resolution of CNNs with a more expressive, self-supervised training strategy. The model is trained to reconstruct partially masked molecular images, forcing it to learn rich, generalizable representations of molecular structure. By removing skip connections—traditionally used in U-Nets—we ensured that the encoder had to extract the most salient features without relying on shallow visual cues. The resulting embeddings are fixed-size, dense, and task-informed, making them well suited for downstream regression models.

After comparative testing, we selected the U-Net with masked autoencoding as our embedding generator. It offered a strong balance of structural expressiveness, robustness to overfitting, and compatibility with our RDKit-based image pipeline. This choice also kept our architecture modular, allowing us to use the same encoder across different predictive models while maintaining consistent input structure.

## 4.5 Model Architecture Design and Iterations

### 4.5.1 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.

### 4.5.2 GNN - PKAN Pipeline

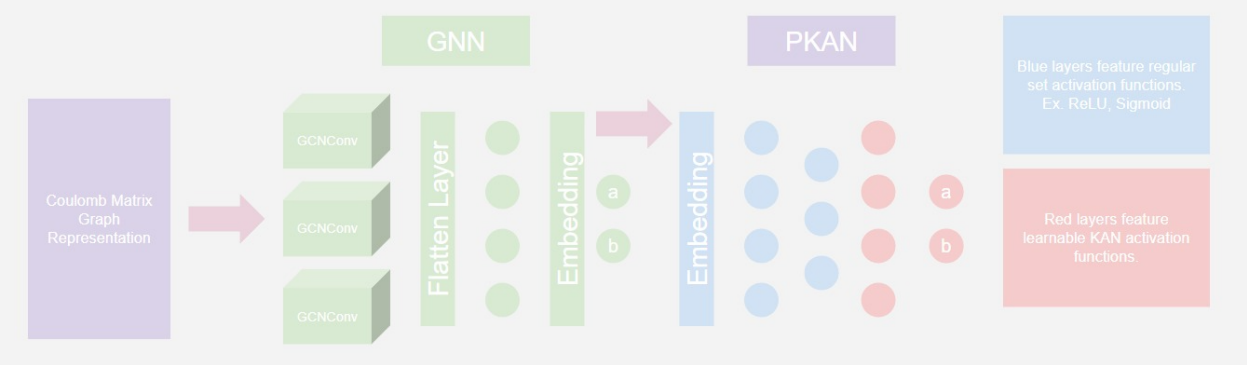


Figure 4.2: GNN - PKAN Pipeline

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 4.3 shows these GCNConv layers, as they are called. Mathematically, they can be described by the equation: [20]

$$X' = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}XW \tag{4.1}$$

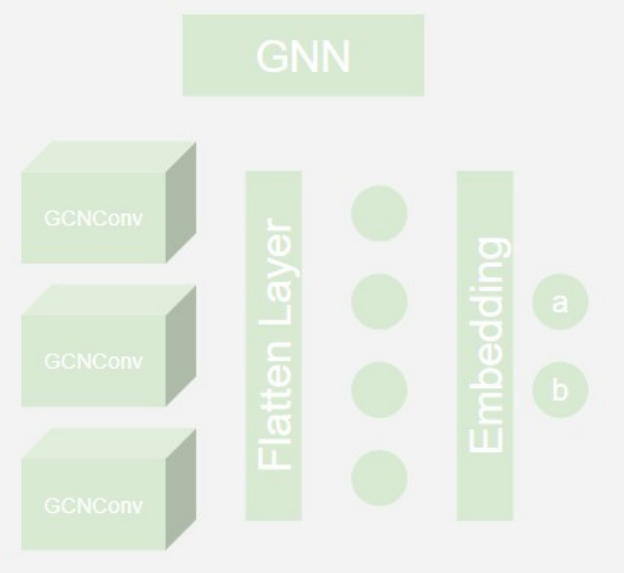


Figure 4.3: GNN Architecture

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 4.3 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)**




Figure 4.4: PKAN Architecture

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 in Figure 4.5. We differ in our output layers, borrowing the KAN activation layer [19] for increased non-linearity. The KAN layer can be described by the equation:

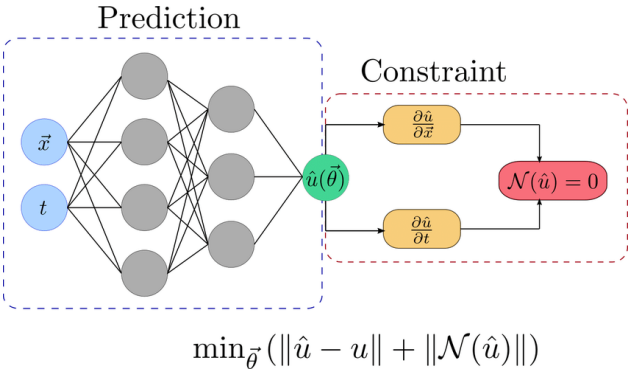


Figure 4.5: PINN Architecture [1]

$$\phi(z) = \sum_{i=1}^{K} w_i k(z, c_i) \tag{4.2}$$

Where $k(z, c_i)$ is a kernel function which acts on input vector $z$. It features a learnable center, $c$, and width, $\sigma$. 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 4.5 shows the central idea of a PINN architecture, featuring the physical constraints that bind our system. We have identified constraints:

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

$$\begin{gathered} L_{pinn} = L_{physics} + L_{trend} \\ L_{physics} = \frac{1}{N} \sum_{i=1}^{N} (max(0, -a_i) + max(0, -b_i)) \\ L_{trend} = \frac{1}{N} \sum_{i=1}^{N} (max(0, a_i - a_{i+1}) + max(0, b_i - b_{i+1})) \end{gathered} \tag{4.3}$$

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:

$$L = A(L_{model}) + B(L_{pinn}) \tag{4.4}$$

### 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.

### 4.5.3 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**

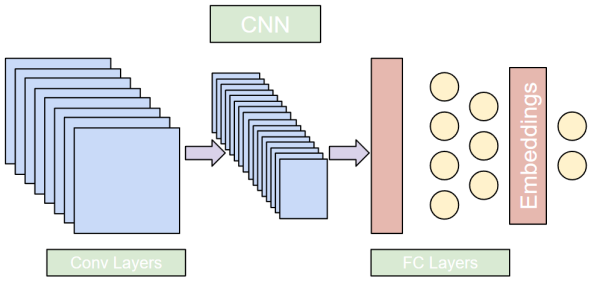


Figure 4.6: CNN Architecture [**?**]

In this pipeline, the CNN performs the same function as the GNN of the previous. Figure 4.6 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 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.

### 4.5.4 Masked Autoencoding U-Net with Attention - PKAN Pipeline

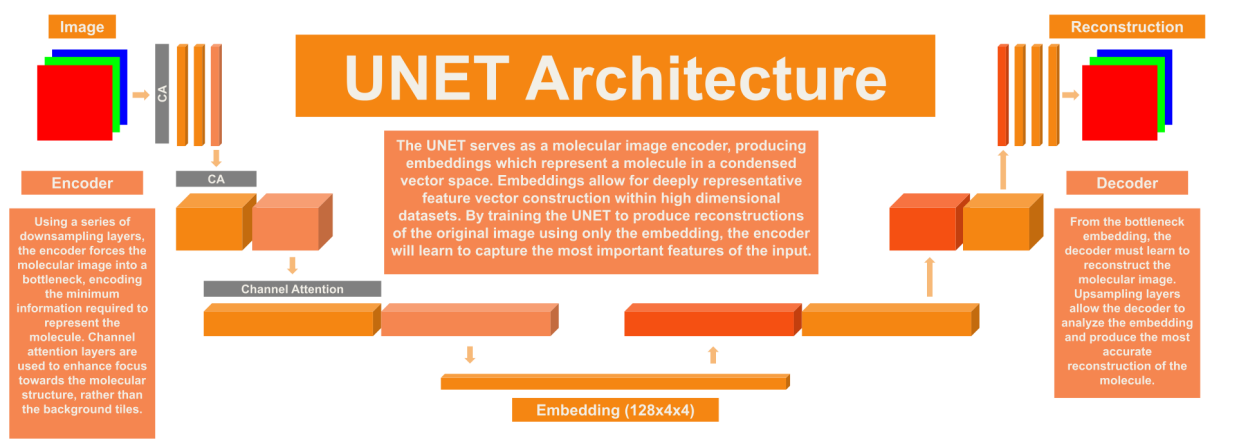


Figure 4.7: Masked Autoencoding U-Net Architecture

To generate molecular embeddings from RDKit-generated images, we developed a convolutional encoder based on a U-Net architecture. This model, illustrated in Figure 4.7, is trained in a self-supervised manner using masked autoencoding (MAE). Unlike conventional U-Nets, we removed skip connections to encourage the encoder to learn deeper, global representations rather than relying on shallow spatial cues.

The network is trained by randomly masking input patches and requiring the decoder to reconstruct the full image. This forces the encoder to produce a semantically rich, fixed-length embedding that captures meaningful structure in the molecular input. The masking strategy randomly selects 75% of patches, applied independently to each image in a batch. Masked images are then passed through the network to compute a reconstruction loss.

To enhance the expressiveness of the model, we introduced channel attention blocks. Channel attention refines feature maps by emphasizing informative channels through global average pooling and a learnable gating mechanism.

The model architecture includes a downsampling encoder path, a bottleneck, and an upsampling decoder path. In the encoder, each stage is followed by channel attention and optional dropout. The decoder mirrors the encoder, using transposed convolutions to restore spatial resolution.

The model is trained using a composite loss that includes:

- **Foreground-Weighted MSE:** Penalizes reconstruction errors more heavily in chemically relevant regions, identified by thresholding pixel values.

- **Foreground L1 Penalty:** Adds a spatially focused L1 loss to further sharpen boundaries and reinforce structural fidelity.

Let $x$ be the original image and $\hat{x}$ the reconstruction. The masked image $x_m$ is computed as:

$$x_m = x \odot M$$

where $M$ is a binary patch mask. The total loss is calculated as:

$$L = \lambda_1 \cdot \mathrm{MSE}_{\mathrm{fg}}(x, \hat{x}) + \lambda_2 \cdot \mathrm{L1}_{\mathrm{fg}}(x, \hat{x})$$

Similar to the CNN, we have the ability to apply data augmentation to our inputs. If we define $X$ as an input molecular image, our transformation pipeline can be expressed as:

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

where:

$$\begin{aligned}
T(X) &= \mathrm{ToTensor}(X) \\
I(X) &= \mathrm{Invert}(X) \\
R(X) &= \begin{cases} \mathrm{RandomRotation}(X, \theta \sim U(-10^\circ, 10^\circ)), & \mathrm{P}(0.5) \\ X, & \text{otherwise} \end{cases} \\
A(X) &= \begin{cases} \mathrm{RandomAffine}(X, \theta = 0, t_x \sim U(-0.1, 0.1), t_y \sim U(-0.1, 0.1)), & \mathrm{P}(0.5) \\ X, & \text{otherwise} \end{cases} \\
N(X) &= X + \varepsilon, \quad \varepsilon \sim U(-0.25, 0.25)
\end{aligned}$$

Here:

- $T$ converts the image from PIL format to a normalized tensor.
- $I$ inverts the color scheme to aid model convergence, as RDKit renders molecules on a white background.
- $R$ applies a random rotation up to 10 degrees with 50% probability.
- $A$ performs a small random translation up to 10% of the image width or height, again with 50% probability.
- $N$ introduces uniform random noise to simulate imaging imperfections.

During testing, only $T$, $I$, and $N$ are applied to maintain consistency while still regularizing the model against noise. These augmentations serve to diversify the training data, encouraging the U-Net to learn invariant and generalizable representations that reflect chemical structure rather than superficial pixel arrangements.

The fixed-length encodings extracted from the bottleneck layer are passed directly to our defined predictive model, the PKAN.

## 4.6 User Interface and Interaction Design

Because our system is intended for human users, including students, researchers, and engineers, usability was a core consideration from the beginning. The user interface not only facilitates input and output, but also plays a critical role in trust, accessibility, and workflow efficiency. We explored two primary paradigms for user interaction: a command-line interface (CLI) and a graphical user interface (GUI).

A command-line interface offers simplicity and speed. It is easy to implement, highly scriptable, and aligns well with research workflows that involve automation or batch processing. It also appeals to users familiar with Python-based tools or shell environments. However, for less technical users, a CLI introduces barriers. It lacks discoverability, requires precise syntax, and provides limited support for real-time feedback or visualization. This makes it less ideal for educational use or for scenarios where interpretability and guided interaction are essential.

In contrast, a graphical user interface presents a more intuitive and accessible experience. We envisioned an interface where users could input SMILES strings, draw molecules interactively, select from multiple models, and view results—both numerically and visually—on the same screen. This approach aligns with how chemists often work and allows the system to provide immediate visual confirmation, confidence intervals, and export options. The GUI also enables future extensibility, such as embedding molecule libraries, batch evaluation tools, or help prompts for beginners.

To guide the design of the interface, we developed several target user flows: entering a known molecule to evaluate its Van der Waals constants, comparing outputs across different models, and exporting results for use in a lab notebook or paper. These use cases informed the interface layout and helped prioritize clarity, minimalism, and responsiveness.

Based on these considerations, we selected a graphical user interface as our interaction model. Implemented using PyQt5, the GUI supports real-time molecule input, model selection, and output visualization in a single cohesive workflow. While the system architecture remains modular enough to support CLI extensions in the future, the GUI prioritizes approachability, scientific transparency, and a more modern user experience.

# Chapter 5

# Product Development and Evaluation Plan

## Technologies Used

This project integrates a comprehensive suite of modern machine learning and data processing libraries to support the end-to-end pipeline, from data loading to model training and evaluation:

- **Python** – The primary programming language used for all code development [21].
- **PyQt5** – Used to develop the graphical user interface, enabling structured layout, text input handling, and functional buttons for executing model generation and prediction tasks [22].
- **PyTorch** – A deep learning framework used to implement and train neural network models, including MLP, CNN, PINN, PKAN, and U-Net architectures [23].
- **scikit-learn** – Utilized for baseline machine learning models (e.g., Random Forest), feature scaling with StandardScaler, and hyperparameter tuning with GridSearchCV [24].
- **RDKit** – A cheminformatics toolkit used to parse SMILES strings and generate consistent 2D molecular images for training the U-Net encoder [13].
- **Torchvision** – Provided image preprocessing and augmentation routines such as rotations and affine transformations for molecular image inputs [25].
- **NumPy & pandas** – Core libraries for numerical computations and data manipulation, used to preprocess both feature and target variables [26, 27].
- **Matplotlib & Seaborn** – Used for data visualization, including plotting training loss curves, prediction distributions, and feature importances [28, 29].
- **TensorBoard** – Employed to visualize model training progress and qualitative outputs during the U-Net training [30].

- **YAML** – Configuration files were written in YAML format to allow dynamic specification of neural network layers in the PKAN-based predictor [31].
- **Joblib** – Used for model serialization and deserialization of trained scikit-learn models [32].

This robust technology stack enabled efficient experimentation, rapid prototyping, and integration of both traditional feature-based and image-based molecular representations.

## 5.1 Van der Waals Dataset

The dataset used in this study consists of 226 unique non-ionic gaseous molecules, each annotated with experimentally determined Van der Waals constants $a$ and $b$ [33]. These values were compiled from publicly available chemical databases and literature sources. Each molecule was represented in multiple formats to support both feature-based and image-based learning approaches.

### 5.1.1 Molecular Representations

To accommodate diverse modeling strategies, we extracted two forms of input representation:

- **Numerical Descriptors:** Each molecule was encoded using nine numerical features derived from its structure. The specific features are described in the following section.
- **2D Molecular Images:** For models employing convolutional neural networks, we generated 256×256 pixel RGB images of each molecule using RDKit's coordinate generation and rendering tools. These images capture the 2D topological structure of the molecules, including atom types and bond connectivity, and were used without additional chemical annotation.

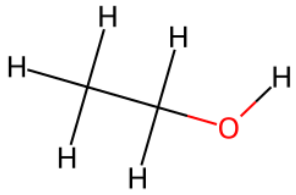


Figure 5.1: RDKit 2D Molecular Image of CCO (Dicarbon Monoxide)

**Molecular Numerical Descriptors**

- **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.

### 5.1.2 Target Variables

The regression targets were the Van der Waals constants $a$ and $b$. These values span multiple orders of magnitude and were standardized using StandardScaler to ensure numerical stability during training. The inverse transform was applied to all predictions during evaluation to report performance in physical units.

### 5.1.3 Preprocessing and Splitting

Input features and targets were scaled separately for the training and test sets using standard normalization. The dataset was randomly shuffled and split into training (80%) and test (20%) subsets, ensuring that both partitions maintained similar distributions of $a$ and $b$. For CNN-based models, additional preprocessing included random rotations and affine transformations to augment image diversity and improve generalization.

### 5.1.4 Limitations

While the dataset size is sufficient for proof-of-concept experiments, it remains relatively small by deep learning standards. As a result, careful regularization, early stopping, data augmentation, and incorporation of domain knowledge (e.g., physics-informed constraints) were critical to achieving reliable model performance. In future work, we aim to expand the dataset through automated extraction from larger chemical databases and through simulation-based augmentation.

## 5.2 Baseline Models

To establish a reliable benchmark for evaluating our proposed method, we implemented six baseline models spanning a range of machine learning paradigms. These models were trained to predict the Van der Waals constants $a$ and $b$ using either engineered molecular descriptors or molecular images. All models were trained using normalized input and target data, and evaluated using mean absolute error (MAE) on a held-out test set after inverse transformation.

### 5.2.1 Linear Regression

The linear regression model consists of a single fully connected layer with no activation function. It serves as a baseline for assessing the benefit of adding non-linearity and inductive biases.

**Architecture:**

- Input layer: 9 molecular features
- Output layer: 2 values ($a$, $b$)

**Training configuration:**

- Optimizer: Adam
- Learning rate: 0.01
- Loss function: Smooth L1 loss with output weights [1.0, 80.0]
- Early stopping: Patience of 10 epochs

### 5.2.2 Multi-Layer Perceptron (MLP)

The MLP adds non-linearity and regularization through multiple hidden layers and dropout. It is designed to capture more complex feature interactions than linear regression.

**Architecture:**

- Hidden layers: [64, 32, 16] with ReLU activations
- Dropout: [0.5, 0.3, 0.2]
- Output layer: 2 values ($a$, $b$)

**Training configuration:**

- Optimizer: Adam
- Learning rate: 0.01
- Scheduler: ReduceLROnPlateau (patience = 5, factor = 0.5)
- Loss function: Smooth L1 loss with output weights [1.0, 80.0]
- Early stopping: Patience of 10 epochs

### 5.2.3 Physics-Informed Neural Network (PINN)

The PINN uses the same core architecture as the MLP but introduces a physics-based loss function to incorporate domain knowledge into training. This constrains the model to make physically plausible predictions.

**Architecture:**

- Hidden layers: [64, 32, 16] with ReLU activations
- Dropout: [0.5, 0.3, 0.2]
- Output layer: 2 values ($a$, $b$)

**Training configuration:**

- Optimizer: Adam
- Learning rate: 0.01

- Physics loss weight: 0.1
- Loss function: Smooth L1 loss + physics constraints
- Early stopping: Patience of 10 epochs

**Physics constraints:**

- $a \geq 0$, $b \geq 0$
- $b \leq c \cdot a^{1/3}$, with $c = 0.1$
- Monotonic increase of $b$ with molecular weight

### 5.2.4 Physics-Informed Kolmogorov-Arnold Network (PKAN)

The PKAN model extends the PINN by replacing one hidden layer with a radial basis function (RBF)-based KAN layer, increasing its expressiveness while retaining the same physics constraints.

**Architecture:**

- Hidden layers: [64, 32 (KAN), 16] with ReLU activations
- KAN layer: 32 inputs, 32 outputs, 5 kernels
- Dropout: [0.5, 0.3, 0.2]
- Output layer: 2 values ($a$, $b$)

**Training configuration:**

- Optimizer: Adam
- Learning rate: 0.01
- Physics loss weight: 0.1
- Loss function: Smooth L1 loss + physics constraints
- Early stopping: Patience of 10 epochs

### 5.2.5 Random Forest Regressor

The random forest is a non-parametric ensemble model trained on the same numerical molecular descriptors. It offers interpretability and robustness, especially on small datasets.

**Architecture:**

- Ensemble of decision trees
- Optional feature selection using model importances

**Training Configuration:**

- Number of estimators: 100
- Maximum depth: None
- Feature selection: Median importance threshold
- Hyperparameter tuning: Grid search over depth, number of estimators, and split criteria
- Evaluation metrics: Mean Absolute Error (MAE) and $R^2$ on inverse-transformed predictions

### 5.2.6 Convolutional Neural Network (CNN)

The CNN operates on 256×256 RGB images of molecular structures generated using RDKit. It is designed to learn spatial features from visual representations of molecules.

**Architecture:**

- Convolutional layers: 32, 64, and 128 filters with kernel sizes 5 and 3
- Max pooling after each convolution
- Fully connected layers: [576, 144, 64] with ReLU
- Dropout: [0.5, 0.3]
- Output layer: 2 values ($a$, $b$)

**Training configuration:**

- Optimizer: Adam
- Learning rate: 0.01
- Loss function: Smooth L1 loss with output weights
- Data augmentation: Random rotation and affine transforms
- Early stopping: Patience of 10 epochs

### 5.2.7 Training and Evaluation Protocol

All models were trained on a dataset of 226 molecules using an 80/20 train-test split. Training was performed using mini-batches of size 32 for neural models. Each model was monitored using validation loss, and early stopping was applied to prevent overfitting. Final evaluation was based on mean absolute error computed after inverse-transforming the model outputs. Among all baselines, the PKAN architecture achieved the lowest test error, validating the benefit of combining domain-informed constraints with flexible function approximators.

## 5.3 Final Predictive Pipeline

This section outlines the final design, development, and evaluation of our core architecture, a modified U-Net designed for molecular image embedding. Our objective was to build an encoder capable of extracting meaningful, compact representations from 2D molecular visualizations generated from SMILES strings. These representations form the foundation for downstream predictions of real gas behavior. We detail both the internal component architecture and the rationale behind each design choice, the sequence of iterative modifications used to refine performance, and the specific roles played by every stage in our pipeline. The ultimate goal of this system is to enable physically consistent predictions of Van der Waals constants through a principled and testable machine learning workflow.

### 5.3.1 Internal Component Design

The final design consists of the following major components:

1. **U-Net Embedding Generator** Transforms molecular images into latent embeddings via a masked autoencoding process. The U-Net architecture was adapted to our chemical imaging task with several key components:

| Component | Purpose |
|---|---|
| Patch Masking | Randomly masks 75% of the input image patches to encourage the encoder to learn robust representations and not rely on spatial redundancy. |
| Channel Attention | Learns to weight chemical color channels differently, emphasizing parts of the molecule such as ring systems or heteroatoms. |
| No Skip Connections | Prevents information from bypassing the bottleneck, ensuring that all information used in reconstruction is encoded in the latent vector. |
| Dropout Layers | Improves generalization and enables Monte Carlo inference for uncertainty estimation. |
| Custom Loss Function | Combines MSE and foreground-weighted L1 loss to emphasize chemically relevant regions, such as bonded atoms and active sites. |

Table 5.1: Summary of U-Net architectural components and their purpose.

   The U-Net outputs a compressed latent vector $\mathbf{z} \in \mathbb{R}^d$ representing the molecule, which serves as input to the PKAN predictor.

2. **Physics-Informed KAN Predictor (PKAN)** Maps molecular embeddings to the Van der Waals constants $(a, b)$ using a physics-aware decoder network.

| Component | Purpose |
|---|---|
| Frozen Encoder | Freezes the trained U-Net encoder to prevent degradation of learned visual features. |
| Configurable Decoder | Layers defined in YAML to enable flexible experimentation. Includes fully connected (FC) and KAN layers. |
| KAN Layers | Performs localized kernel-based transformations per input feature to increase nonlinearity while preserving interpretability. |
| Gradient Clipping | Prevents exploding gradients, especially important during physics loss backpropagation. |
| Mixed Precision | Increases training efficiency and stability on CUDA-enabled devices. |
| Physics-Informed Loss | Enforces physical plausibility via three terms:<br>(a) $a \geq 0$, $b \geq 0$ (positivity)<br>(b) $b \leq c \cdot a^{1/3}$ (empirical relation)<br>(c) $b$ increases with molecular weight (monotonicity) |

Table 5.2: PKAN decoder components and their physical relevance.

The total loss $\mathcal{L}$ used to train PKAN is:

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{smooth}}(\hat{y}, y)}_{\text{data loss}} + \lambda \cdot \underbrace{\mathcal{L}_{\text{physics}}(\hat{y}, \text{MW})}_{\text{physics-informed penalty}} \tag{5.1}$$

where $\hat{y}$ are predictions, $y$ are ground truth constants, and MW denotes molecular weight. The factor $\lambda$ tunes the strength of the physical constraints.

### 5.3.2 Pretraining Dataset for U-Net Encoder

To enable effective molecular representation learning, our U-Net encoder was first pretrained on a large dataset of molecular structures sourced from the ZINC20 database [34]. ZINC20 is a free resource of commercially available compounds for virtual screening, which includes a diverse set of small organic molecules relevant to drug discovery and chemical modeling. From this resource, we extracted over 50,000 SMILES strings representing various gas-phase and small-molecule compounds.

Each SMILES string was converted into a 2D molecular image using RDKit. These images were then augmented with random rotations and affine transformations to improve model robustness to spatial variance.

The pretraining objective was a masked image reconstruction task. During training, random image patches were masked, and the U-Net was trained to reconstruct the original image. This self-supervised masked autoencoding approach allowed the network to develop chemically meaningful embeddings without requiring labeled physical property data.

After pretraining, the encoder's bottleneck representations were used as input features for downstream EOS coefficient prediction models. This decoupled approach of unsupervised

structure learning followed by supervised fine-tuning aimed to provide better generalization and stronger performance when paired with the PKAN model for predicting Van der Waals constants.

### 5.3.3 Iterative Development and Modifications

Multiple design iterations were necessary to achieve optimal performance:

- **Skip Connection Removal**: Early versions of the U-Net included skip connections, but these were removed after observing that the decoder was circumventing the embedding bottleneck, leading to poor downstream prediction performance.
- **Noise Augmentation**: Random affine transformations and slight noise were introduced during training to improve generalization and reduce overfitting.
- **Channel-Attention Tuning**: Attention layers were added and tuned to highlight chemically relevant regions and interactions in the image.
- **Physics Loss Refinement**: The physics-informed loss was revised to include empirical constraints between $a$ and $b$, as well as sorted monotonicity checks with molecular weight.
- **Decoder Reconfiguration**: The decoder architecture and hyperparameters were iteratively tuned through random search to optimize validation loss.

These modifications were guided by empirical results and aimed at improving both embedding quality and physical plausibility of predictions.

### 5.3.4 Step-by-Step Pipeline and Purpose

1. **Data Preprocessing**: SMILES strings are converted into 256×256 RGB molecular images using RDKit.
2. **U-Net Training**: Trained to reconstruct masked images, forcing the model to understand key molecular patterns. The encoder learns latent embeddings.
3. **Embedding Extraction**: Once trained, the decoder is discarded and only the encoder is retained.
4. **Decoder Construction**: Configurable via YAML, this component processes embeddings using KAN and FC layers.
5. **Predictor Training**: The full predictor model is trained with early stopping, scheduler, and physics-informed loss.
6. **Evaluation**: Model outputs are denormalized and compared to ground truth $a$ and $b$ values using MAE and distributional analysis.

Each step was included to build a robust, modular pipeline that could enforce physical laws while learning from high-dimensional molecular representations.

### 5.3.5 Baseline Comparison and Evaluation Strategy

To assess the effectiveness of our U-Net predictive pipeline, we compare its performance against a suite of baseline models, including traditional machine learning models (Random Forest, Linear Regression), standard neural architectures (MLP, CNN), and physics-informed methods (PINN). Each model is trained and evaluated on the same dataset split and compared using the standardized metrics defined in Table 2.4.

Our evaluation framework focuses on both statistical performance and project-level requirements. Metrics such as $R^2$ and MSE are used to compare predictive accuracy.

The U-Net encoder, pretrained via masked image reconstruction, is expected to produce superior representations that improve generalization and predictive power in downstream regression. This representation is fed into a physics-informed PKAN decoder, forming our final predictive pipeline.

## 5.4 GUI Design

### 5.4.1 General Layout

The GUI, shown in Figures 5.2 and 5.3, 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. The molecule image and SMILES string are saved to files on the user's hard drive. These data files are then fed into the models for prediction.
- **Output** - The Output section is split into four main components: the two graph display widgets, the numerical output section, and the debug window. This section also contains the run button used to initiate the prediction pipeline.
  - **Graph Displays** - The graph display populates when the pipeline completes its predictions. The selection bars at the top of the graph area allow the user to choose which model (PKAN, CNN, etc.) is shown as well as which prediction value (a or b) is displayed.
  - **Numerical Display** - This section displays all of the statistical information about the a and b prediction given by the selected model. It populates these values after the run button has been clicked and the pipeline has finished. The bar above this section allows the user to choose which model's information is being displayed.
  - **Debug Window** - Once the run button is pressed, the debug window will display the steps currently being done in the pipeline. If any errors were to occur in the process they will be displayed here. It is essentially a direct connection to the standard output of the model pipeline.

### 5.4.2 Monte Carlo-Based Statistical Output

To provide more informative predictions, the GUI leverages Monte Carlo (MC) sampling to estimate the uncertainty in the predicted Van der Waals constants. When the user initiates a prediction, the selected model is explicitly set to train() mode—even during inference. This forces dropout layers to remain active, introducing stochasticity in each forward pass.

The GUI executes the prediction 100 times with identical input data, each time sampling slightly different internal neuron activations due to dropout. This produces a distribution of 100 predicted values for both constants $a$ and $b$. From this distribution, the following statistics are computed and displayed in the Numerical Output section:

- Mean prediction for $a$ and $b$
- Median prediction for $a$ and $b$
- Mode prediction for $a$ and $b$
- Standard deviation (to express model uncertainty)
- 95% confidence interval with lower and upper bounds

This statistical approach helps the user assess not just a single deterministic prediction, but a range of possible values—giving insight into the model's confidence. This method is especially useful when using architectures like CNNs or PKANs where dropout is a standard regularization technique.

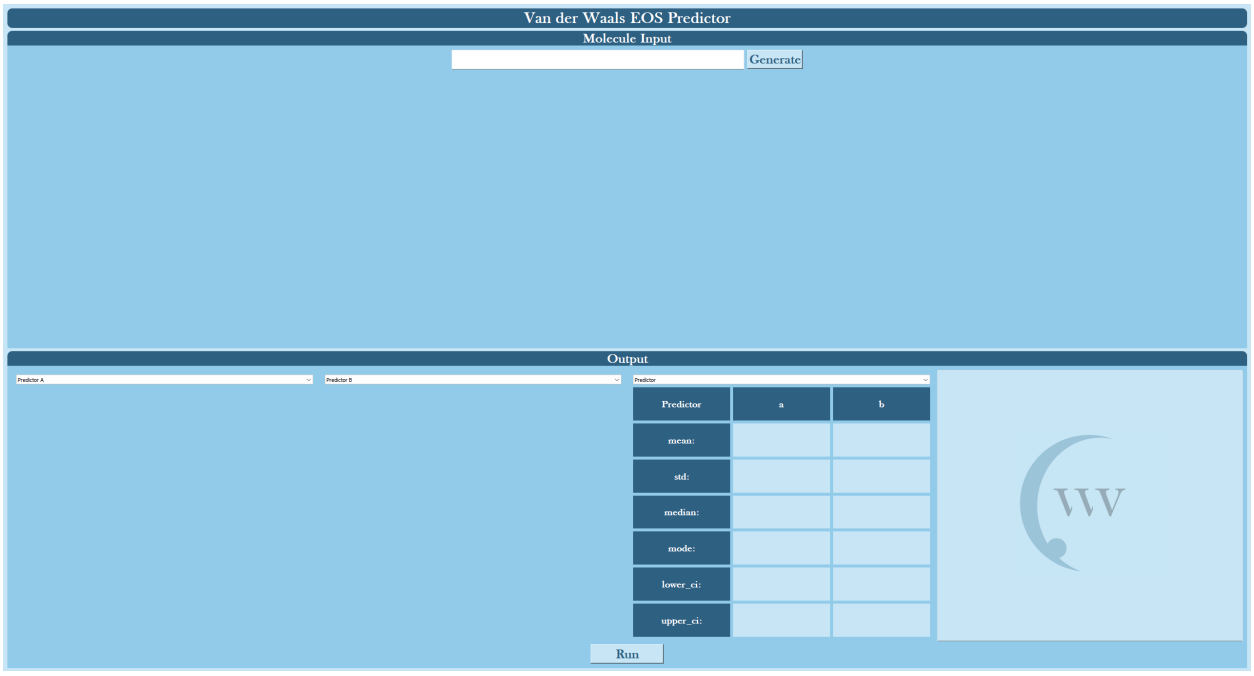


Figure 5.2: GUI upon application startup

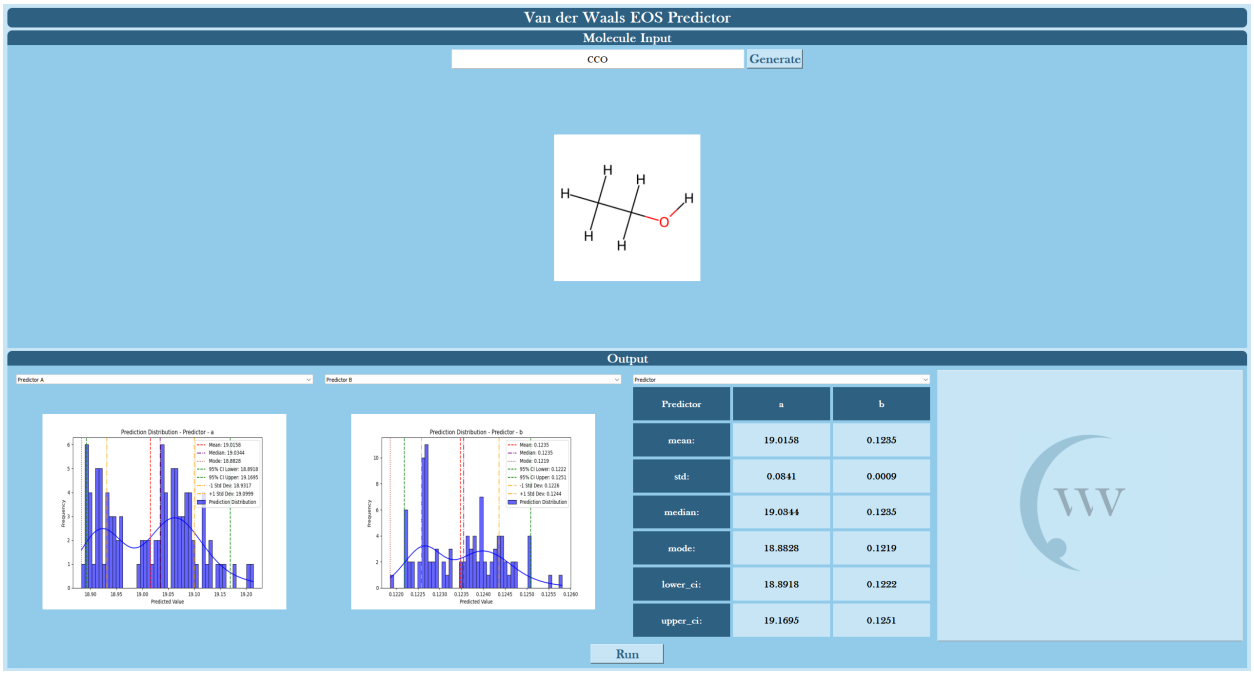


Figure 5.3: GUI after molecule has been generated and model is run

## 5.5 GUI Evaluation

### 5.5.1 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.

### 5.5.2 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.

## 5.6 Results

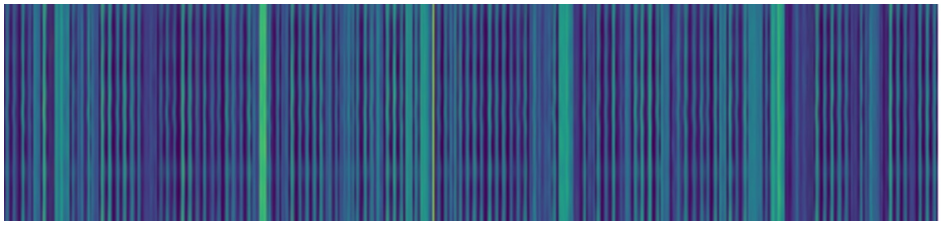

Figure 5.4: Visual representation of U-Net molecular embeddings

Figure 5.4 shows the embeddings of a batch of vertically stacked samples. The encoded vectors form distinct bands across the batch, showing a lack of diversity within the embedding space. This heavily implies that our model fails to learn to embed molecular properties, but rather to compress the images.

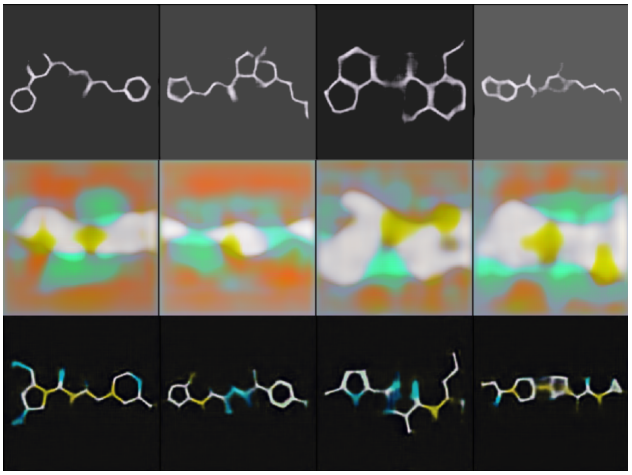

Figure 5.5: Iterative view of improving reconstruction results

While the embeddings showed a severe lack of diversity, Figure 5.5 shows that the model was able to progressively produce more accurate molecular images as we iterated upon the design. The first row shows a model which was able to capture structural information, while missing colors from the original image. These colors correlate to different elements, and are necessary to be included in the final reconstruction for reliable encoding. The second row shows a model which was able to capture color information, but lost the ability to create fine structures. The third row shows the results of our final model, able to capture and reconstruct structure and color information.

Table 5.3: Model's $R^2$ and MSE Comparison for Van der Waals Constants a and b. Full refers to all models evaluated on the full test dataset and Z-score 3 refers to all models evaluated on all values in the test dataset with a z-score of 3 or less.

| Model | R2_a | R2_b | MSE_a | MSE_b |
|---|---|---|---|---|
| LR (Full) | 0.451 | 0.137 | 91.257 | 0.008 |
| LR (Z-Score 3) | 0.699 | 0.642 | 44.251 | 0.001 |
| RF (Full) | 0.473 | 0.158 | 87.670 | 0.008 |
| RF (Z-Score 3) | 0.688 | 0.650 | 46.109 | 0.001 |
| MLP (Full) | 0.474 | 0.144 | 87.506 | 0.008 |
| MLP (Z-Score 3) | 0.783 | 0.695 | 30.110 | 0.001 |
| CNN (Full) | 0.445 | 0.067 | 92.273 | 0.008 |
| CNN (Z-Score 3) | 0.657 | 0.618 | 50.377 | 0.001 |
| PINN (Full) | 0.463 | 0.175 | 89.390 | 0.007 |
| PINN (Z-Score 3) | 0.767 | 0.706 | 32.244 | 0.001 |
| PKAN (Full) | 0.490 | 0.125 | 84.905 | 0.008 |
| PKAN (Z-Score 3) | 0.905 | 0.710 | 12.914 | 0.001 |
| Predictor (Full) | 0.000 | 0.002 | 166.346 | 0.009 |
| Predictor (Z-Score 3) | 0.000 | 0.011 | 166.346 | 0.003 |

As shown in Table 5.3, traditional machine learning models such as Random Forests (RF), Linear Regression (LR), and Multi-Layer Perceptrons (MLP) demonstrated reasonable performance when provided with hand-crafted molecular features. Among these, the PKAN model trained on descriptor features performed the best, achieving an $R_a^2$ of 0.905 and an $R_b^2$ of 0.710 on inlier test data (z-score $\leq 3$), along with the lowest corresponding mean squared errors.

Table 5.4: Model Compliance with $R^2$ and MSE Requirements for $a$ and $b$ (Z-score 3 subset)

| **Model** | $R_a^2 \geq 0.75$ | $\text{MSE}_a \leq 30$ | $R_b^2 \geq 0.75$ | $\text{MSE}_b \leq 0.05$ |
|---|---|---|---|---|
| Linear Regressor (LR) | ✗ | ✗ | ✗ | ✓ |
| Random Forest (RF) | ✗ | ✗ | ✗ | ✓ |
| MLP | ✓ | ✓ | ✗ | ✓ |
| CNN | ✗ | ✗ | ✗ | ✓ |
| PINN | ✓ | ✗ | ✗ | ✓ |
| PKAN | ✓ | ✓ | ✓ | ✓ |
| U-Net + PKAN Predictor | ✗ | ✗ | ✗ | ✗ |

The results show a clear advantage to incorporating domain-specific knowledge into the architecture. PKAN, which combines a physics-informed loss function with Kernel Activation Networks, excelled at modeling the nonlinear relationships inherent in the Van der Waals equation of state.

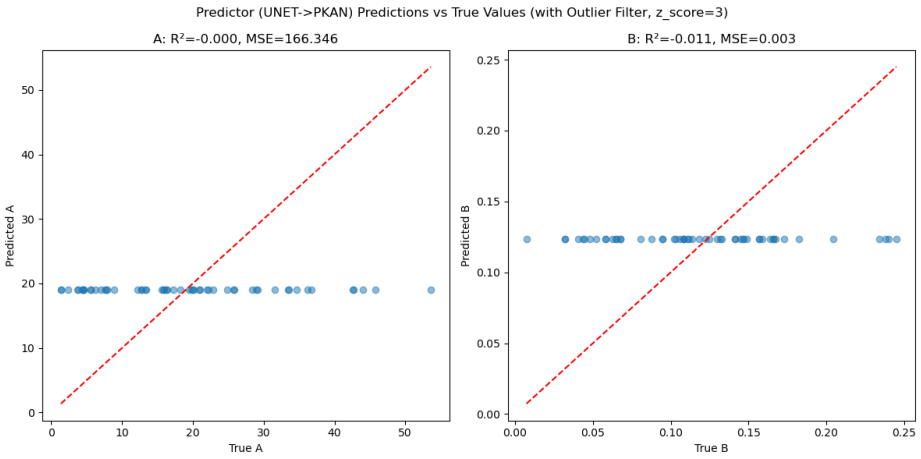


Figure 5.6: Predictor Test Results for a and b without outliers

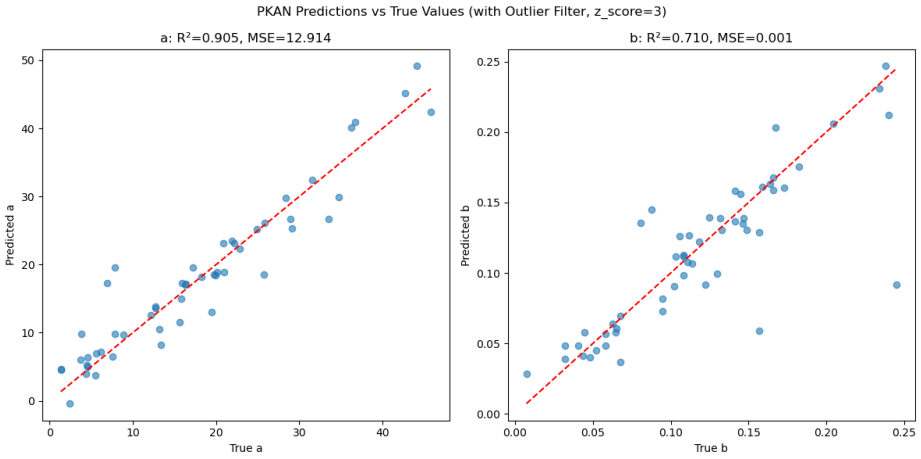


Figure 5.7: PKAN Test Results for a and b without outliers

When we transitioned to our end-to-end model, which uses a U-Net encoder and PKAN-based decoder, the performance significantly degraded. Both $R^2$ values dropped to nearly zero and MSE increased substantially, even on filtered data, shown in Table 5.3. Figures 5.6 and 5.7 plot the true and predicted values for $a$ and $b$ for our predictor and the baseline PKAN. A higher predictive accuracy is reflected by how well the predicted distribution fits the red line. Because the slope of our prediction line is constant in Figure 5.6 we see that the model only predicts the mean of the dataset. In comparison, the PKAN's predictions fit the line much better. This outcome highlights a core limitation in our approach: while the U-Net was successful in reconstructing molecular images visually, the learned embeddings lacked diversity and failed to encode chemically meaningful features, as discussed in Figures 5.4 and 5.5.

The issue stems from a mismatch between the reconstruction task and the downstream prediction task. The U-Net, trained solely to fill in masked pixels, optimized for image similarity rather than encoding quantitative chemical information. As a result, the embeddings

extracted by the encoder were not suitable as inputs to the PKAN predictor, leading to poor generalization and underperformance.

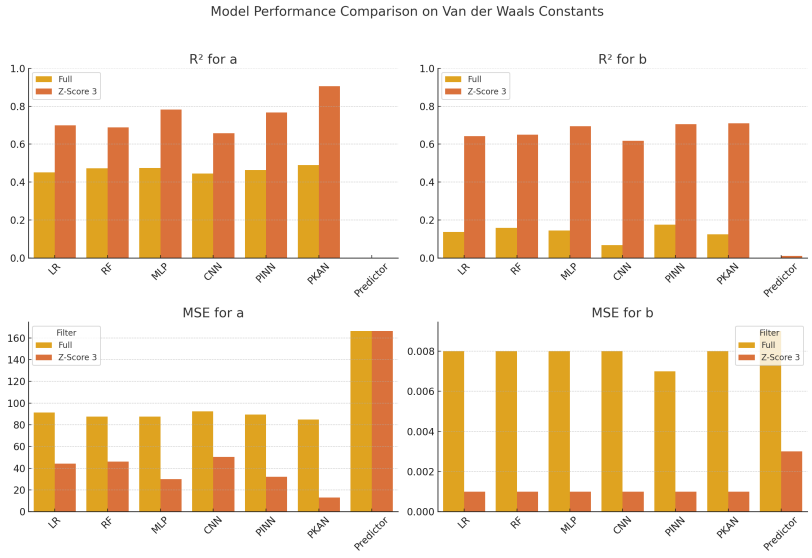


Figure 5.8: Bar comparison of all baseline model's vs. Predictor Model's (UNET-PKAN) $R^2$ and MSE values on the full test data set (Full) and the test data removing outliers with a z-score of greater than 3 (Z-Score 3).

### 5.6.1 Future Work

To improve the end-to-end prediction pipeline, future efforts should focus on refining the embedding process. One possible direction is to incorporate contrastive or supervised objectives into the U-Net training, forcing the latent space to reflect property-relevant variations. Exploring joint training of the encoder and predictor, rather than sequential training, could help align the learned features more closely with the prediction task. Other methods of molecular representation, such as SMILES embeddings, could be used in tandem with other encoding architectures, such as Transformers.

Despite the shortcomings of the final model, the performance of descriptor-based PKAN models suggests that our physics-informed architecture is a promising approach when combined with appropriate features.

# Chapter 6

# Social Impact Evaluation

## 6.1 Recognizing and Defining Targeted Needs in a Social Context

This project addresses a growing demand for efficient, scalable molecular property prediction tools. In fields like chemical engineering, pharmaceuticals, and environmental science, the ability to model real-gas behavior without costly laboratory experiments can save time, reduce material waste, and expand research access. For instance, early-stage drug design and reaction engineering often require real gas data—especially under non-ideal conditions—yet many research labs and academic institutions lack the resources to gather this data manually. Our system seeks to democratize access to these thermodynamic insights, empowering users to conduct exploratory molecular modeling with minimal infrastructure. In doing so, it directly supports under-served educational and research environments.

## 6.2 Broader Impacts of Engineering Solutions

Machine learning-based scientific tools like ours reshape how discoveries are made. Our project supports sustainable research practices by minimizing physical waste and maximizing computational throughput. It promotes open scientific exchange by building on open-source platforms and using publicly available datasets. More broadly, this design highlights a shift toward automation and reproducibility in the sciences—trends that benefit industrial R&D and academic communities alike. By replacing repetitive, high-cost processes with fast, automated prediction, we help accelerate innovation across domains including synthetic chemistry, green energy storage materials, and personalized medicine.

## 6.3 Ethical and Professional Responsibilities

As engineers, we have a responsibility to ensure that our tool behaves ethically and remains reliable under all use cases. We have adopted the following strategies:

- **Transparency:** All datasets, models, and training routines are documented and will be released as open source, facilitating peer verification and reproducibility.

- **Fairness:** We assess performance across diverse molecular structures to mitigate dataset-induced bias.
- **Integrity:** We communicate limitations of our model clearly, especially regarding generalization outside the training distribution.

To ensure reproducibility, all model training scripts were executed using a fixed random seed (torch.manual_seed(42)), and experiments were run on a consistent hardware configuration. The dataset, derived from the Zinc20 standard, is available at [34]. Our codebase, including data preprocessing, model training, and evaluation, is documented and available at [35].

## 6.4 Communication on Societal Issues

Our project team is committed to communicating the scientific, ethical, and societal implications of AI-driven molecular modeling. Through written documentation, poster presentations, and software demonstrations, we will share how predictive systems like ours can improve chemical design while reducing ecological and financial costs. These discussions will include not only results, but design decisions, tradeoffs, and ethical constraints. We aim to present our findings in both academic and public forums, including undergraduate research expos and open-source repositories.

## 6.5 Understanding Professional and Ethical Responsibilities

Throughout this project, our team has developed a deeper understanding of the responsibilities involved in building engineering tools for scientific use:

- We follow best practices for software development, version control, and documentation to ensure the tool can be trusted and reused.
- We emphasize interpretability in our models, selecting architectures and outputs that can be understood by domain experts.
- We approach collaboration with humility and openness, understanding that our tool fits within a larger ecosystem of researchers, educators, and engineers working toward shared scientific goals.

# 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 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 ($R^2$ and mean squared 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.

## 7.4 Milestones

| **Milestone** | **Completion Date** | **Contributors** |
|---|---|---|
| Initial Planning and Research | 9/27/24 | Chmayssani, Heath, and Milstead |
| Data Collection and testing molecular encodings | 11/15/24 | Chmayssani and Milstead |
| Preliminary GUI prototype | 11/18/24 | Heath |
| CNN and PKAN development and testing | 12/1/24 | Chmayssani and Milstead |
| GUI SMILES input and image generation | 12/5/24 | Heath |
| Unconnected Run button and Output display | 12/7/24 | Heath |
| Implemented CNN and PKAN into pipeline | 12/8/24 | Chmayssani and Milstead |
| Random Forest implemented | 1/7/25 | Chmayssani and Milstead |
| Connected GUI to current pipeline | 1/12/25 | Chmayssani, Heath, and Milstead |
| Debug window implemented | 1/17/25 | Heath |
| Graph implementation and display | 1/31/25 | Chmayssani, Heath, and Milstead |
| Refined data output and created Numerical Output section of GUI | 2/18/25 | Heath |
| Completed and implemented UNET architechture for novel prediction pipeline | 2/26/25 | Chmayssani and Milstead |
| Connected MLP, Linear Regressor, and PINN to baseline model pipeline | 3/2/25 | Chmayssani and Milstead |
| Implemented Model switching logic for graph and numerical display | 3/10/25 | Heath |
| Testing and adjustment from user feedback | 3/20/25 | Chmayssani, Heath, and Milstead |
| Project Completion | 4/9/25 | Chmayssani, Heath, and Milstead |

Table 7.1: Project Timeline

# 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 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.

## 8.3 Project Timeline

Table 8.2 summarizes the key milestones and deliverables for the project:

| 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

This timeline and role allocation will be revisited at regular intervals to ensure alignment with project goals and to address unforeseen challenges. The final project timeline is given below.

| **Milestone** | **Deadline Week** | **Description** |
|---|---|---|
| Initial Planning and Research | Week 1 | Reviewed literature, defined project scope, and selected appropriate modeling strategies. |
| Data Collection and Testing Molecular Encodings | Week 8 | Gathered molecular data and evaluated encoding methods (SMILES, RDKit images, descriptors). |
| Preliminary GUI Prototype | Week 9 | Designed and implemented initial user interface framework. |
| CNN and PKAN Development and Testing | Week 11 | Implemented baseline CNN and advanced PKAN models; ran performance tests. |
| GUI SMILES Input and Image Generation | Week 12 | Integrated SMILES input parsing and RDKit image generation into the GUI. |
| Unconnected Run Button and Output Display | Week 12 | Created GUI elements for running predictions and displaying output (not yet functional). |
| Implemented CNN and PKAN into Pipeline | Week 12 | Integrated trained models into the inference pipeline for backend processing. |
| Random Forest Implemented | Week 16 | Added traditional ML baseline (Random Forest) for comparison. |
| Connected GUI to Current Pipeline | Week 17 | Linked GUI controls to model pipeline; enabled end-to-end predictions. |
| Graph Implementation and Display | Week 17 | Visualized prediction outputs using embedded graphs in GUI. |
| Refined Data Output and Created Numerical Output Section | Week 17 | Improved formatting and layout for numerical predictions within the GUI. |
| Project Completion and Testing | Week 30 | Final round of testing, code cleanup, and evaluation of full system functionality. |

Table 8.2: Finale Project Timeline

# 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 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

| **Category** | **Estimated Cost (USD)** | **Description** |
|---|---|---|
| Software Licenses | 0 | All used software is free |
| Cloud Services | 50 | Google Colab computing hours, GPU computing credits ($10/100hrs) |
| Miscellaneous | 0 | Expenses for documentation, presentations, and potential third-party APIs for molecular data. |
| **Total Estimated Cost** | 50 | |

Table 9.2: Final Budget

## 9.3 Engineering Time Allocation

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

## 9.4 Milestones

## 9.5 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.

| **Milestone** | **Estimated Hours** |
|---|---|
| Initial Planning and Research | 15 hours |
| Data Collection and Testing Molecular Encodings | 20 hours |
| Preliminary GUI Prototype | 10 hours |
| CNN and PKAN Development and Testing | 35 hours |
| GUI SMILES Input and Image Generation | 12 hours |
| Unconnected Run Button and Output Display | 10 hours |
| Implemented CNN and PKAN into Pipeline | 15 hours |
| Random Forest Implemented | 8 hours |
| Connected GUI to Current Pipeline | 15 hours |
| Graph Implementation and Display | 9 hours |
| Refined Data Output and Created Numerical Output Section | 11 hours |
| Completed and implemented UNET architechture for novel prediction pipeline | 20 hours |
| Connected MLP, Linear Regressor, and PINN to baseline model pipeline | 15 hours |
| Implemented Model switching logic for graph and numerical display | 9 hours |
| Testing and adjustment from user feedback | 20 hours |
| Final tweaks and project completion | 10 hours |
| **Total Estimated Hours** | **234 hours** |

Table 9.3: Estimated Hours for Project Milestones

# 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] "Ideal gas law and real gas behavior," https://www.discoverengineering.org/ideal-gas-law-and-real-gas-behavior/, accessed: 2025-05-02.

[3] Y. e. a. Zhang, "Accurate prediction of molecular properties and drug targets using a molecular image-based deep learning framework," *Nature Machine Intelligence*, vol. 4, pp. 279–287, 2022.

[4] Q. Li, N. Fu, S. S. Omee, and J. Hu, "Md-hit: Machine learning for materials property prediction with dataset redundancy control," 2023. [Online]. Available: https://arxiv.org/abs/2307.04351

[5] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988.

[6] "Van der waals interactions - chemistry libretexts," https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Physical_Properties_of_Matter/Atomic_and_Molecular_Properties/Intermolecular_Forces/Specific_Interactions/Van_Der_Waals_Interactions, accessed: 2025-05-02.

[7] J. Deng, Z. Yang, H. Wang, I. Ojima, D. Samaras, and F. Wang, "A systematic study of key elements underlying molecular property prediction," *Nature Communications*, vol. 14, no. 1, p. 6395, Oct 2023.

[8] Y. Li, B. Liu, J. Deng, Y. Guo, and H. Du, "Image-based molecular representation learning for drug development: a survey," *Briefings in Bioinformatics*, vol. 25, no. 4, 2024.

[9] 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.

[10] 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

[11] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, "Fast and accurate modeling of molecular atomization energies with machine learning," *Physical Review Letters*, vol. 108, no. 5, p. 058301, 2012.

[12] D. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

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

[14] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox, "Gaussian˜16 Revision C.01," 2016, gaussian Inc. Wallingford CT.

[15] F. Neese, F. Wennmohs, U. Becker, and C. Riplinger, "The orca quantum chemistry program package," *The Journal of Chemical Physics*, vol. 152, no. 22, p. 224108, 2020.

[16] B. P. Brown, O. Vu, A. R. Geanes, S. Kothiwale, M. Butkiewicz, E. W. Lowe, R. Mueller, R. Pape, J. Mendenhall, and J. Meiler, "Introduction to the biochemical library (bcl): An application-based open-source toolkit for integrated cheminformatics and machine learning in computer-aided drug discovery," *Frontiers in Pharmacology*, vol. 13, 2022, accessed: 2025-05-02. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphar.2022.833099

[17] M. E. Khatib and W. A. de Jong, "Ml4chem: A machine learning package for chemistry and materials science," 2020. [Online]. Available: https://arxiv.org/abs/2003.13388

[18] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[19] 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.

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

[21] P. S. Foundation, *Python Language Reference, version 3.10*, 2025, https://www.python.org/.

[22] R. C. Limited, *PyQt5 Reference Guide*, 2025, https://www.riverbankcomputing.com/static/Docs/PyQt5/.

[23] A. Paszke *et al.*, *PyTorch: An Open Source Machine Learning Framework*, 2024, https://pytorch.org/.

[24] F. Pedregosa *et al.*, *Scikit-learn: Machine Learning in Python*, 2011, https://scikit-learn.org/.

[25] S. Marcel and Y. Rodriguez, *Torchvision: Datasets, Transforms and Models Specific to Computer Vision*, 2025, https://pytorch.org/vision/.

[26] C. R. Harris *et al.*, *NumPy*, 2020, https://numpy.org/.

[27] W. McKinney, *pandas: a foundational Python library for data analysis and statistics*, 2010, https://pandas.pydata.org/.

[28] J. D. Hunter, *Matplotlib: A 2D Graphics Environment*, 2007, https://matplotlib.org/.

[29] M. Waskom and the Seaborn Development Team, *Seaborn: Statistical Data Visualization*, 2023, https://seaborn.pydata.org/.

[30] T. Authors, *TensorBoard: Visualizing Learning*, 2025, https://www.tensorflow.org/tensorboard.

[31] O. Ben-Kiki, C. Evans, and I. d. Net, *YAML Ain't Markup Language (YAML)*, 2009, https://yaml.org/.

[32] J. Developers, *Joblib: running Python functions as pipeline jobs*, 2025, https://joblib.readthedocs.io/.

[33] K. C. Blake Milstead, "Van der waals constants dataset," https://huggingface.co/datasets/BlakeMilstea/Van_Der_Waals_Constants, 2024, accessed: 2025-05-02.

[34] J. J. Irwin, K. G. Tang, J. Young, C. Dandarchuluun, B. R. Wong, M. Khurelbaatar, Y. S. Moroz, J. Mayfield, and R. A. Sayle, "Zinc20—a free ultralarge-scale chemical database for ligand discovery," *Journal of Chemical Information and Modeling*, vol. 60, no. 12, pp. 6065–6073, 2020.

[35] B. Milstead, K. Chmayssani, and T. Heath, "Eos predictor," https://github.com/sussykeem/eos_predictor, 2025, accessed: 2025-05-02.

# Appendix A

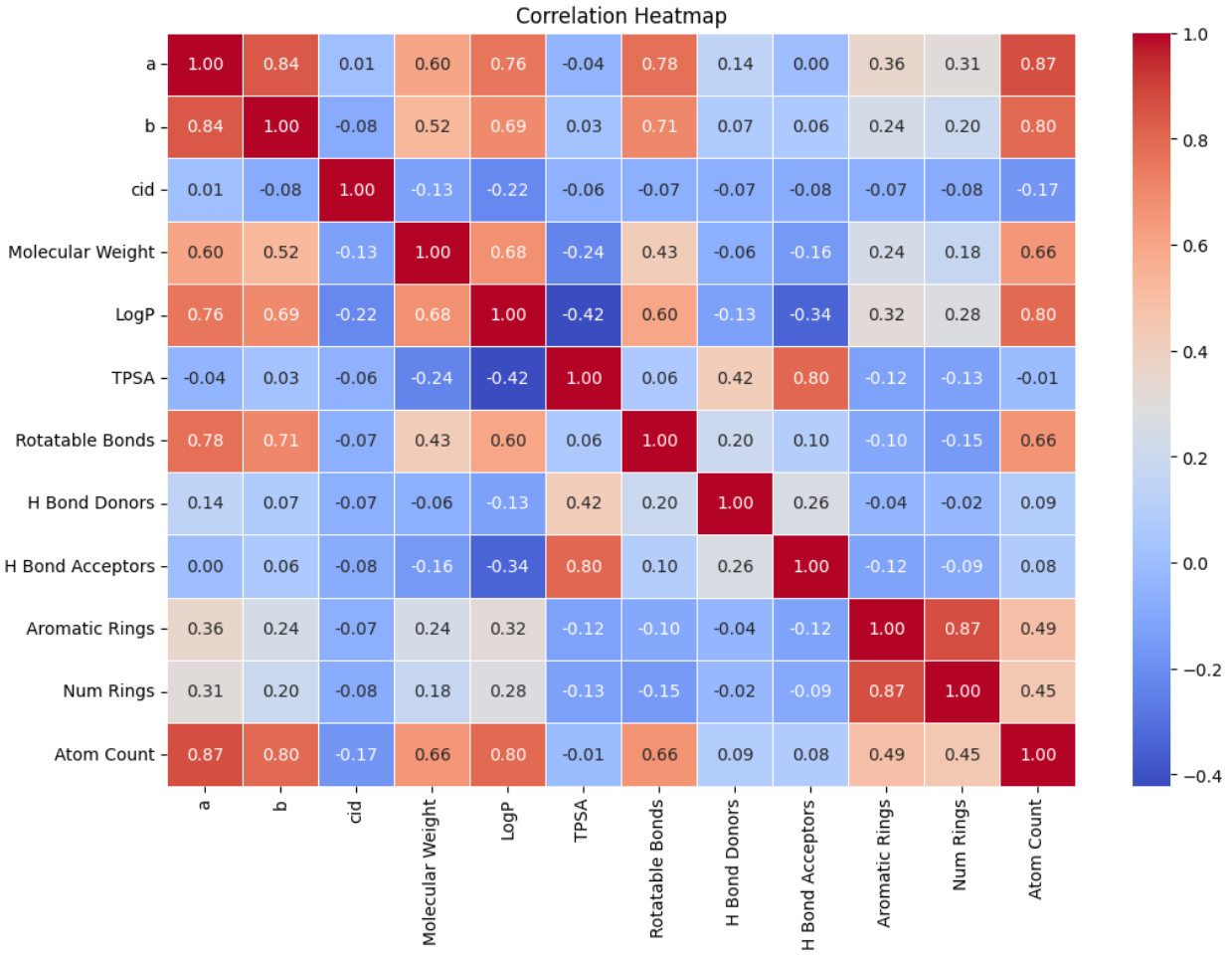


Figure A.1: Molecular Feature Correlation Heatmap


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

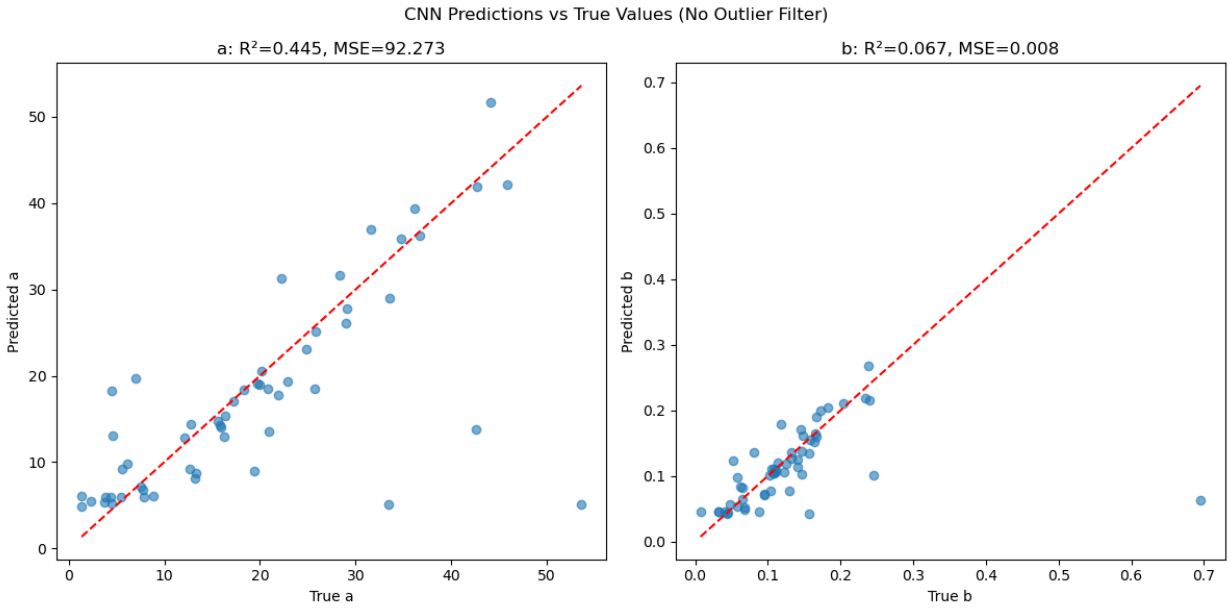


Figure A.3: CNN true vs predicted $a$ and $b$ with no outliers

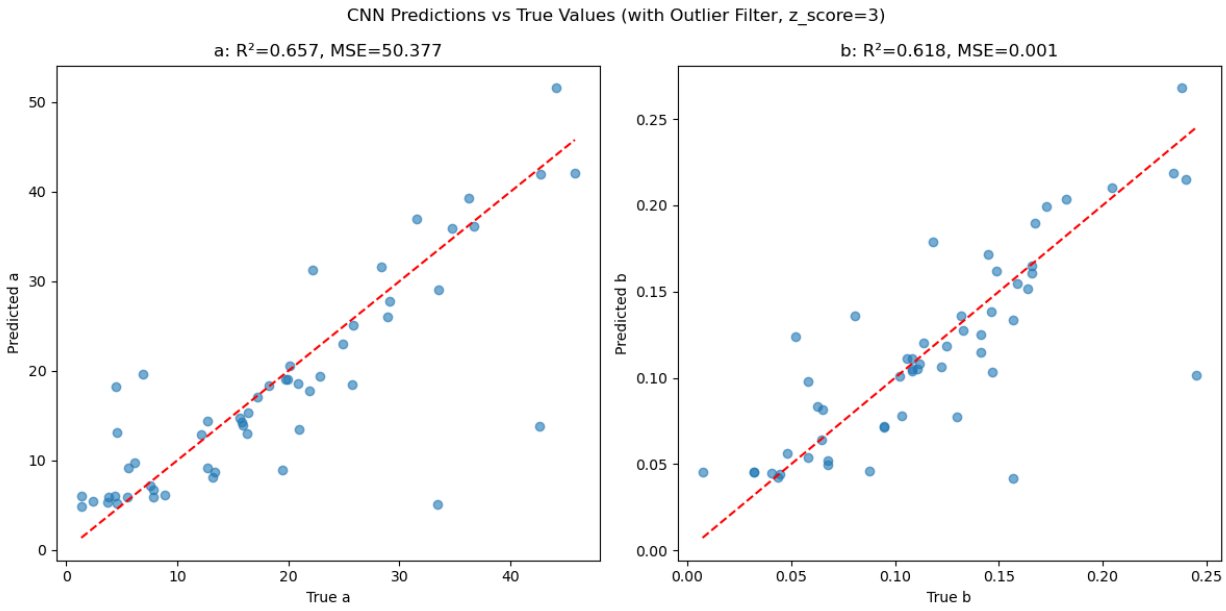


Figure A.4: CNN true vs predicted $a$ and $b$ with outliers

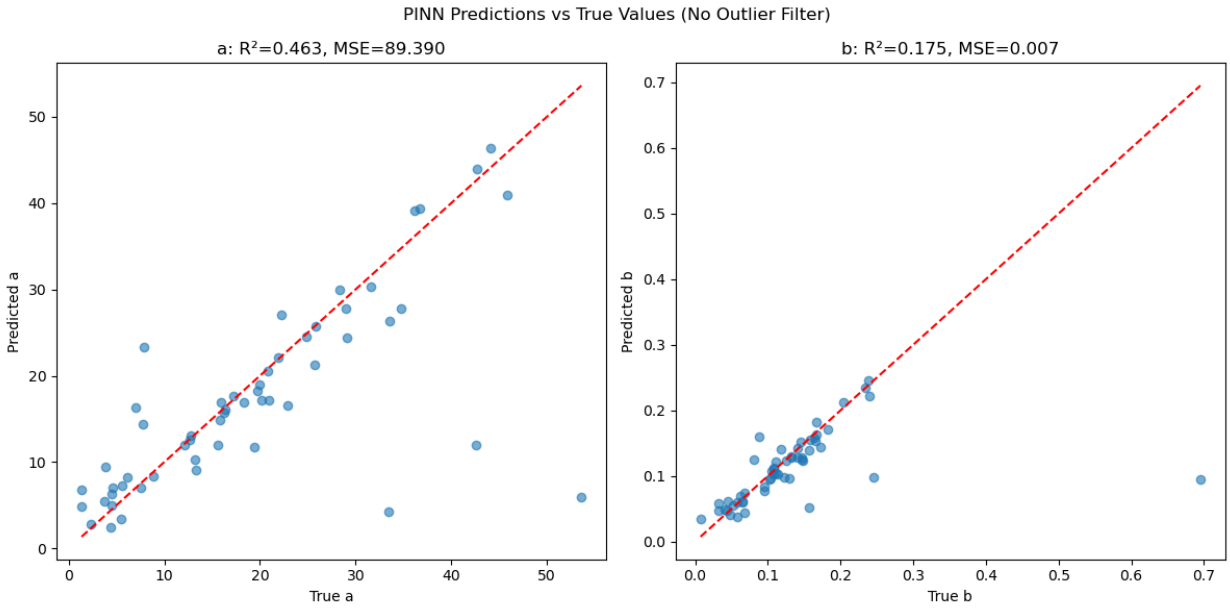


Figure A.5: PINN true vs predicted $a$ and $b$ with no outliers

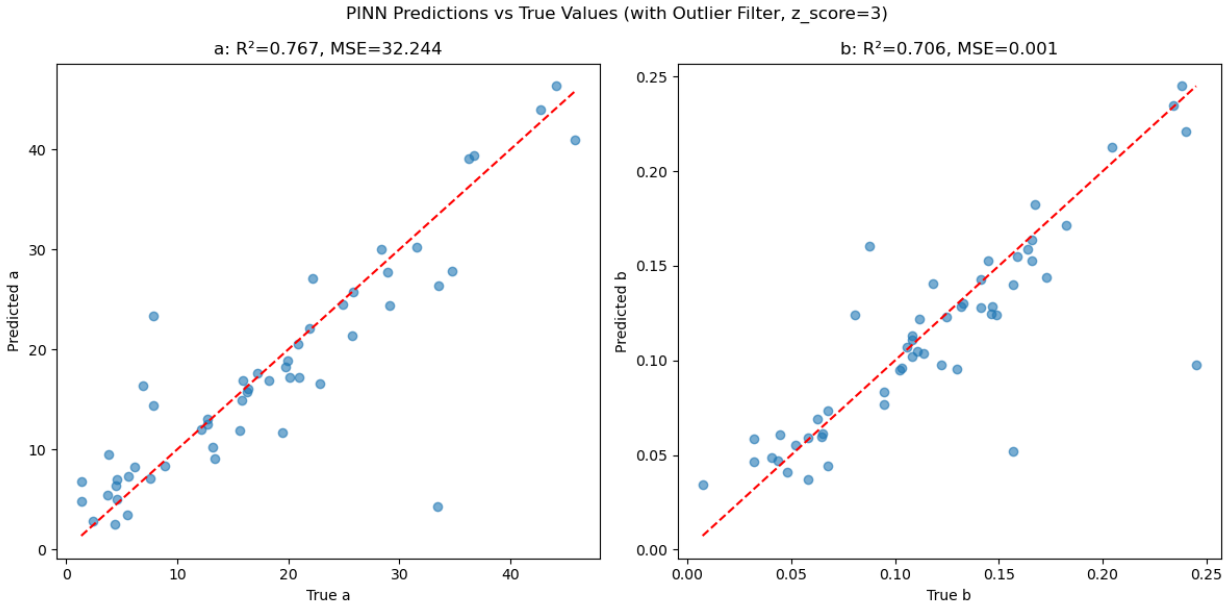


Figure A.6: PINN true vs predicted $a$ and $b$ with outliers

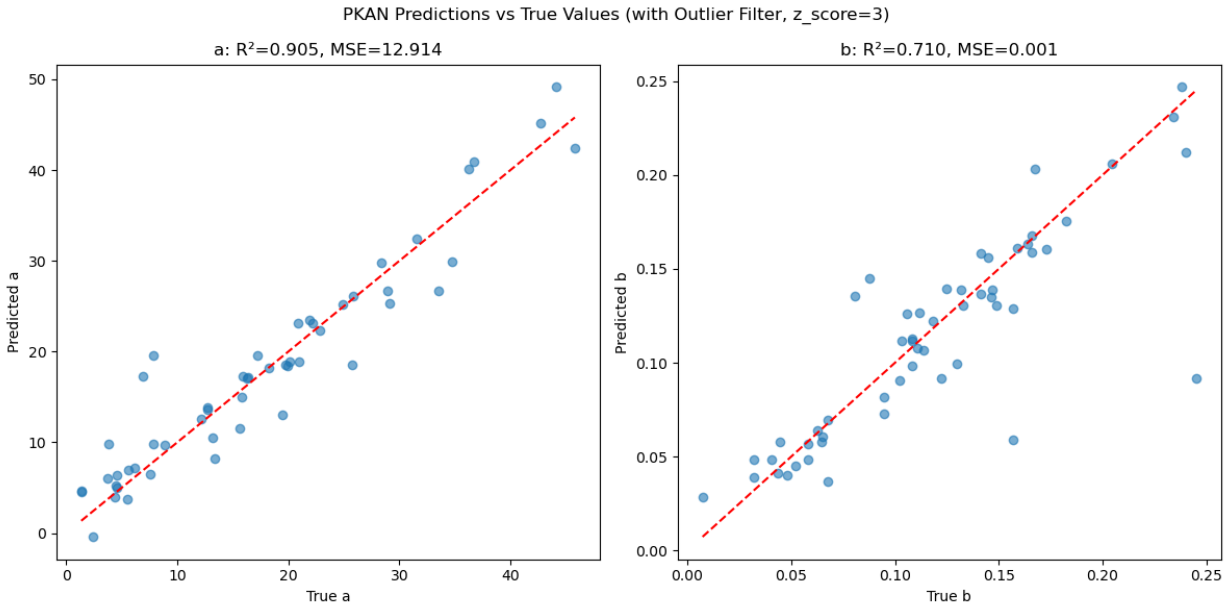


Figure A.7: PKAN true vs predicted $a$ and $b$ with outliers

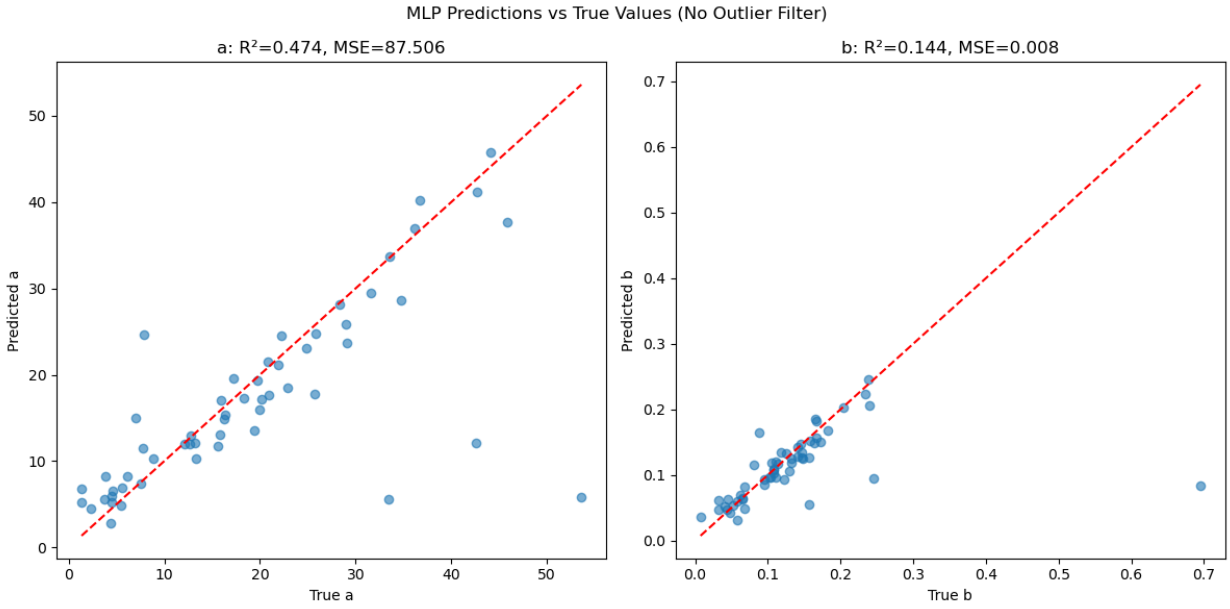


Figure A.8: MLP true vs predicted $a$ and $b$ with no outliers

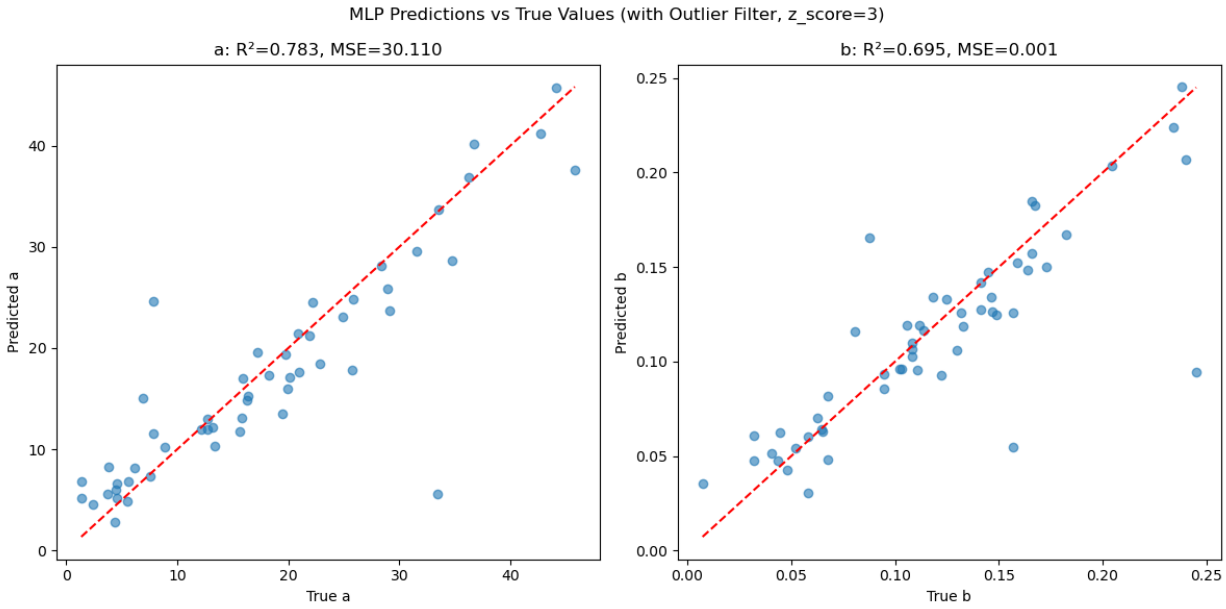


Figure A.9: MLP true vs predicted $a$ and $b$ with outliers

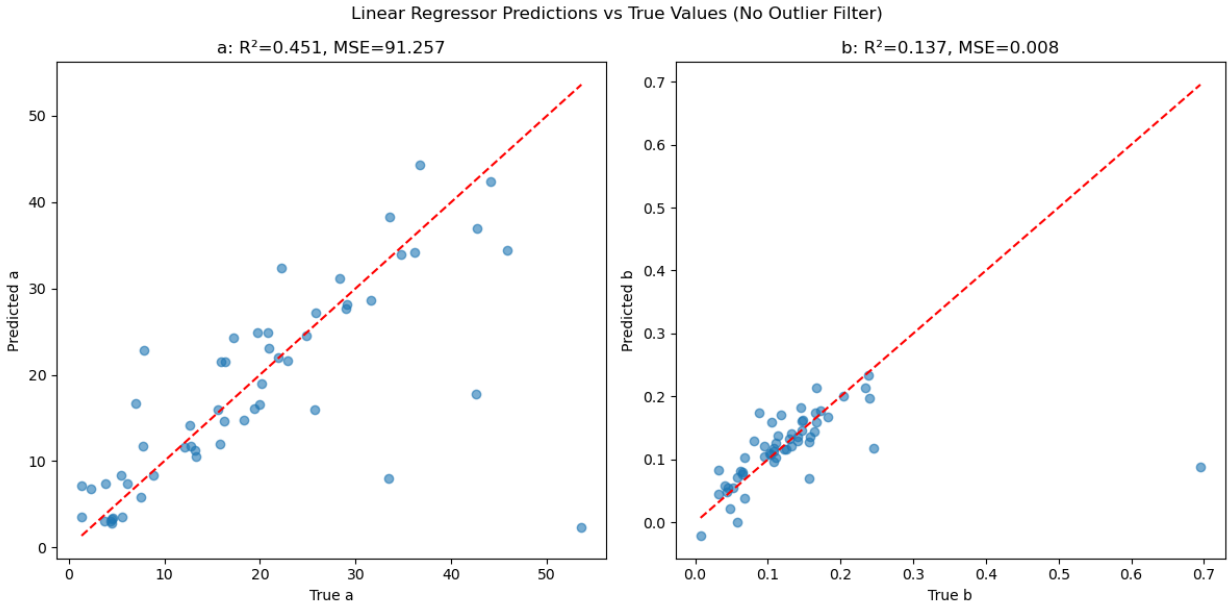


Figure A.10: Linear Regressor true vs predicted $a$ and $b$ with no outliers

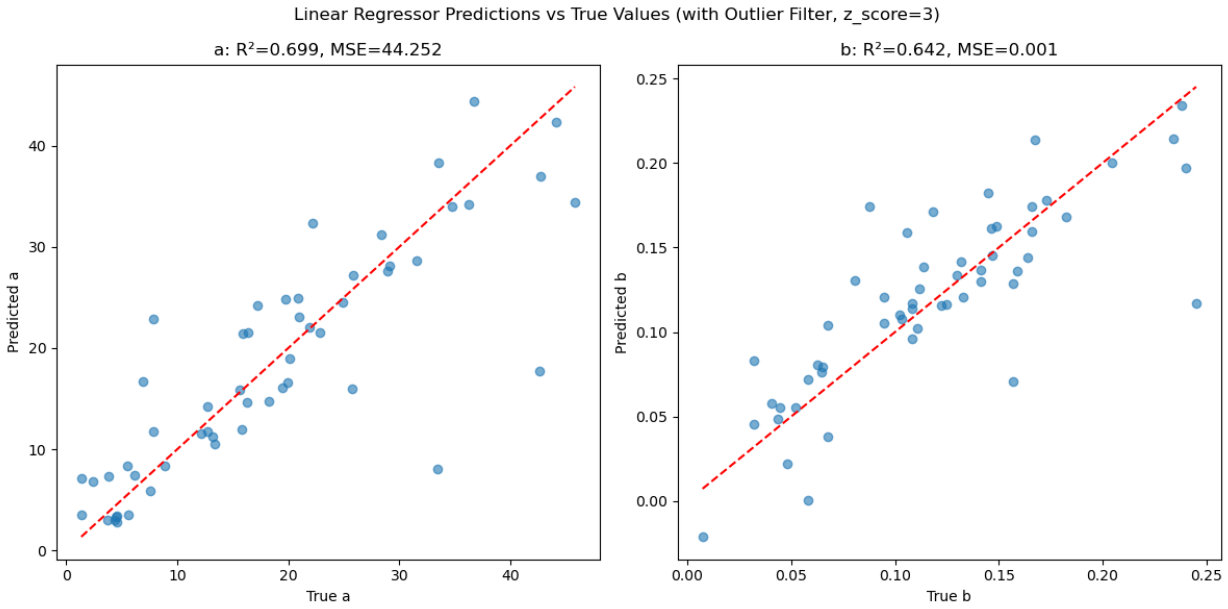


Figure A.11: Linear Regressor true vs predicted $a$ and $b$ with outliers

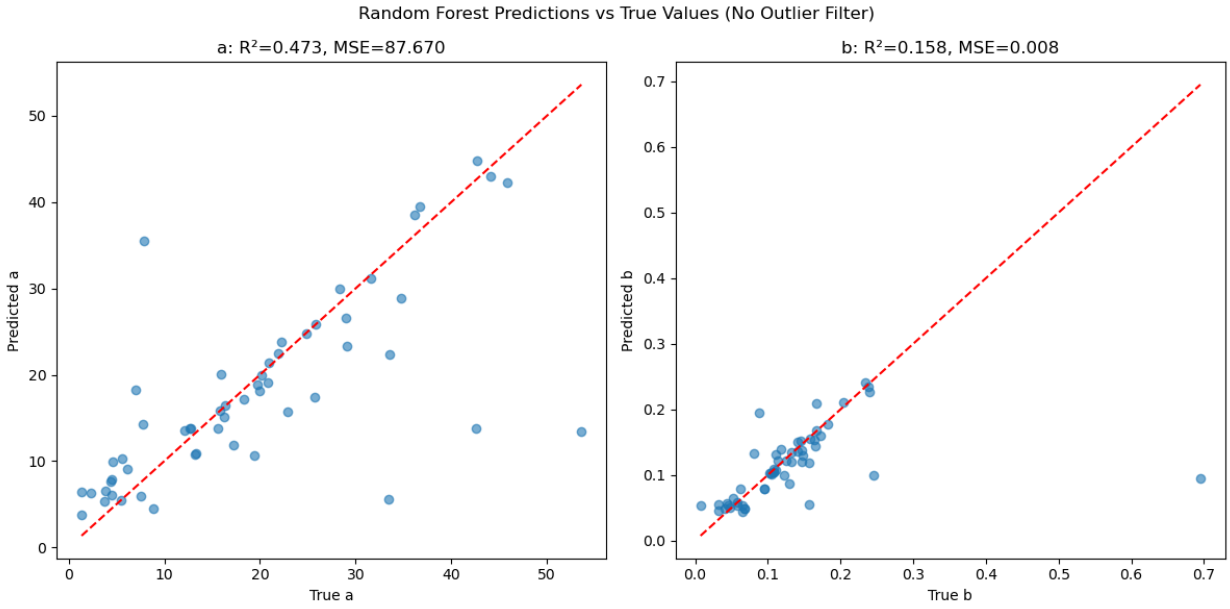


Figure A.12: Random Forest true vs predicted $a$ and $b$ with no outliers

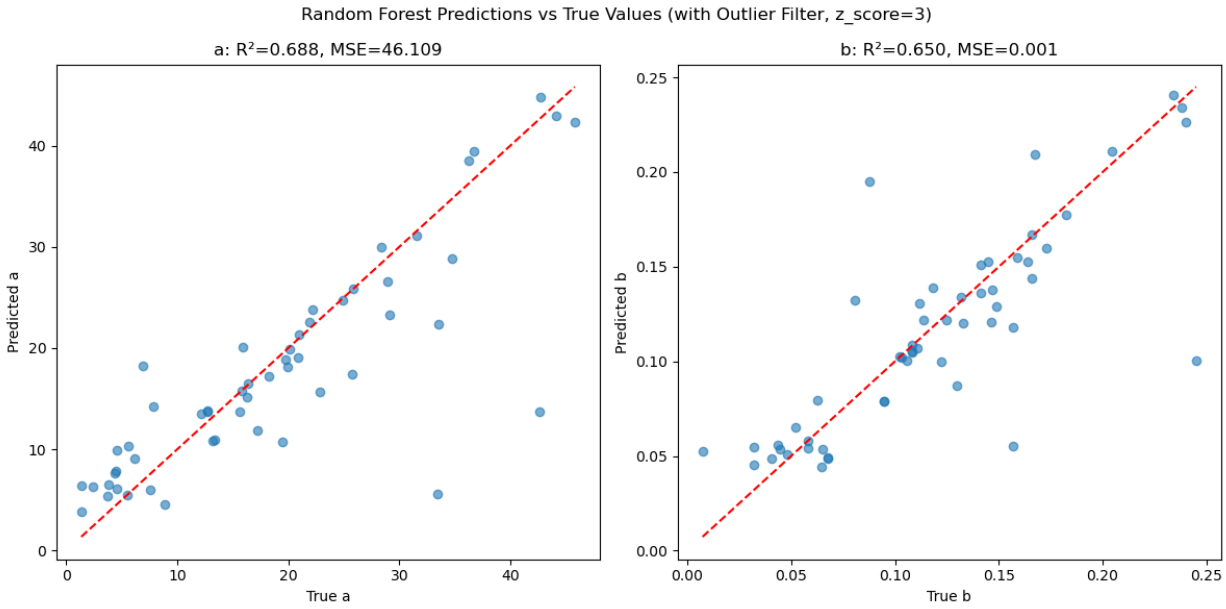


Figure A.13: Random Forest true vs predicted $a$ and $b$ with outliers

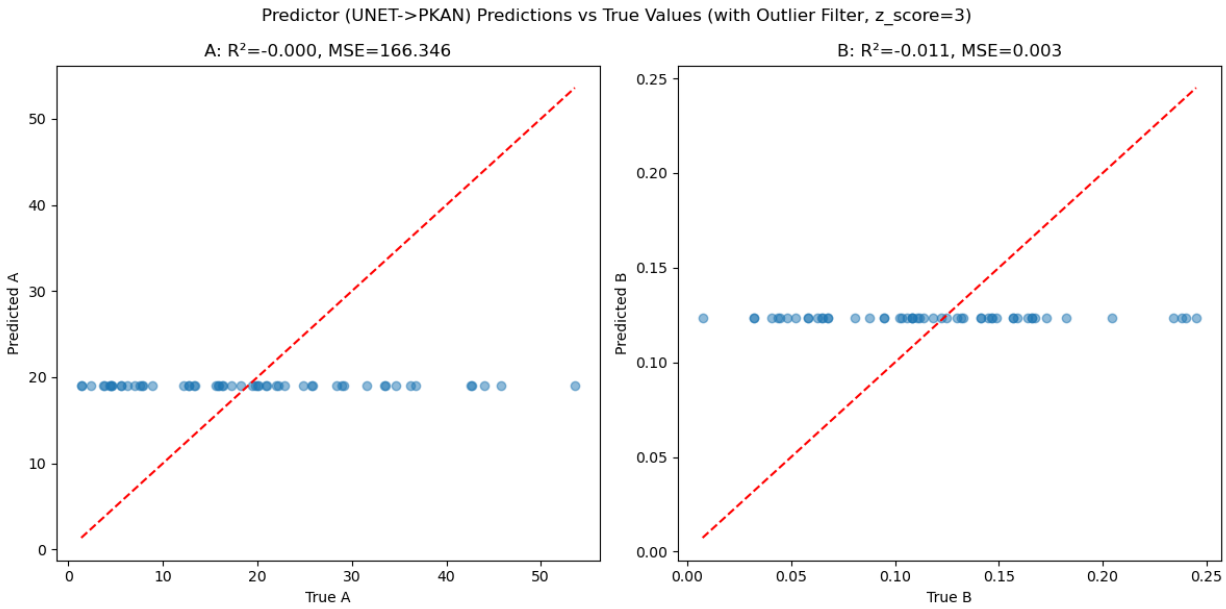


Figure A.14: Predictor true vs predicted $a$ and $b$ with outliers

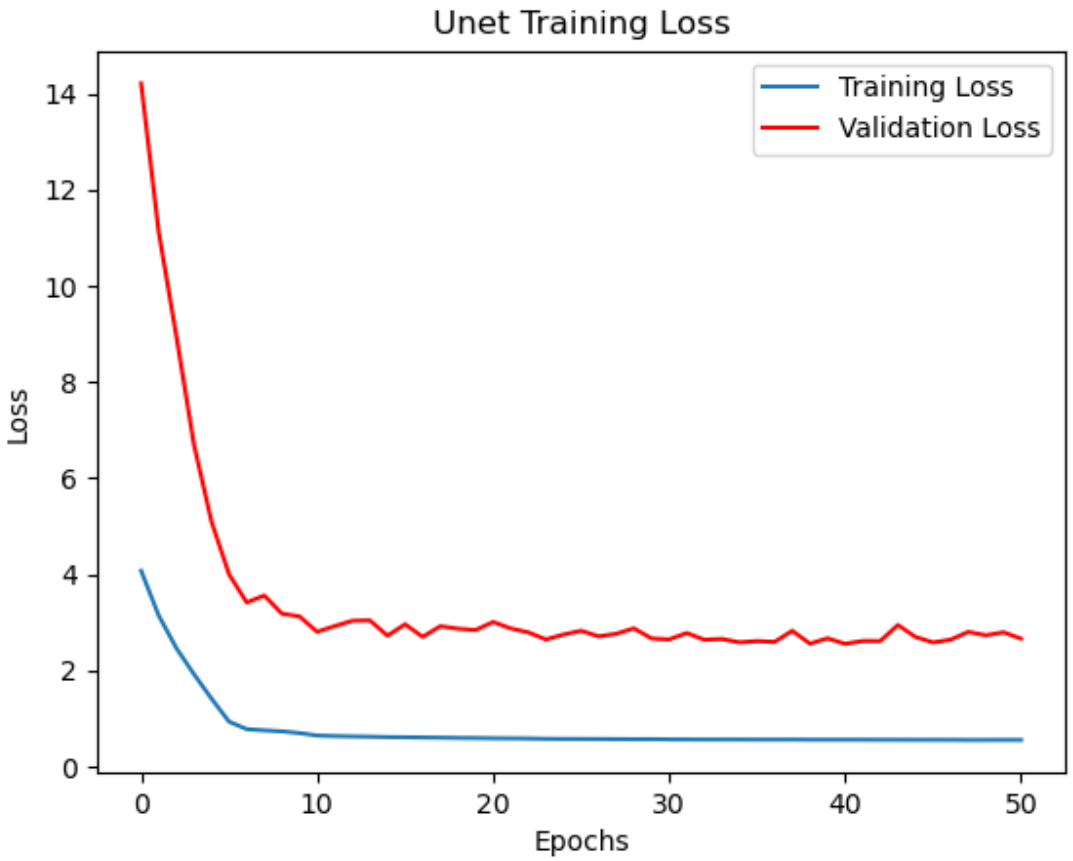


Figure A.15: U-Net Training and Validation Loss