Cairo University
Faculty of Engineering
Department of Computer Engineering

ViroAlert

What is the next pandemic?

A Graduation Project Report Submitted

to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

Presented by

Nesma AbdEl-Kader Shafie
Sara Gamal Gerges
Eman Ibrahim Elbedwihy
Yousef Osama Mohamed

Supervised by

Dr. Abd El Moniem Bayoumi
Virology Expert: Dr. Ebrahim Hassan

2025

Abstract

Viruses are among the most critical threats to global health due to their rapid evolution and ability to infect new hosts, as witnessed in pandemics like COVID-19. Traditional methods for viral risk assessment and drug discovery are time-consuming and resource intensive. In response, we introduce **ViroAlert**, an AI-driven system that integrates deep learning, machine learning, graph neural networks, and generative modeling to accelerate the detection of human-adaptable viruses and the discovery of effective antiviral drugs. ViroAlert consists of four key modules: a deep learning model for predicting viral human adaptability, a machine learning model for biological interpretability, a drug-target interaction predictor based on GNNs, and a generative model for designing novel drug candidates. This unified pipeline enables faster, more interpretable, and scalable analysis of viral threats, supporting early intervention and drug development efforts.

الملخص

تُعد الفيروسات من أخطر التهديدات التي تواجه الصحة العالمية بسبب قدرتها السريعة على التطور والتكيف مع مضيفين جدد، كما ظهر في جائحة كوفيد-19. الطرق التقليدية لتقييم مخاطر الفيروسات واكتشاف الأدوية المضادة تستغرق وقتًا طويلاً وتتطلب موارد كبيرة. استجابةً لذلك، نقدم ViroAlert، وهو نظام ذكي يعتمد على الذكاء الاصطناعي ويجمع بين تقنيات التعلم العميق، والتعلم الآلي، والشبكات العصبية البيانية، والنماذج التوليدية لتسريع عملية اكتشاف الفيروسات من أربعة مكونات رئيسية: نموذج تعلم عميق ViroAlert القابلة للتكيف مع الإنسان واقتراح علاجات فعالة لها. يتكوّن للتنبؤ بقابلية الفيروس للتكيف مع البشر، ونموذج تعلم آلي لتفسير بيولوجي أوضح، ومُتنبئ لتفاعل الدواء مع الفيروس باستخدام الشبكات العصبية البيانية، ونموذج توليدي لتصميم أدوية جديدة. يوفّر هذا النظام المتكامل تحليلاً سريعًا وقابلاً للتفسير وقابلًا للتوسع لدعم التدخل المبكر وتطوير الأدوية.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to our academic supervisor, Dr. Abd El Moniem Bayoum, for his continuous guidance, support, and valuable feedback throughout the project.

We would also like to thank Dr. Ibrahim Essam, Virology Expert at Nile University, for his valuable support in providing virological insights and datasets.

This project is dedicated to everyone who has suffered from the impact of pandemics and to the healthcare professionals and researchers who continue to fight on the frontlines.

Table of Contents

List of Figures

List of Tables

List of Abbreviation

Abbreviation	Full Form
AI	Artificial Intelligence
APAAC	Amphiphilic Pseudo Amino Acid Composition
BI-GRU	Bi-Directional Gated Recurrent Unit
CNN	Convolutional Neural Network
CTDD	Composition Transition and Distribution Descriptors
CTriad	Conjoint Triad
DDE	Dipeptide Deviation from Expected Mean
DL	Deep Learning
DTI	Drug-Target Interaction
EA	Evolutionary Algorithms
ESM	Evolutionary Scale Modeling
GA	Genetic Algorithms
GAN	Generative Adversarial Network
GC	Graph Convolution
GCN	Graph Convolutional Network
GNN	Graph Neural Network
k-mer	A substring of length *k* derived from biological sequences
LSTM	Long Short-Term Memory
ML	Machine Learning
MSE	Mean Squared Error
PCA	Principal Component Analysis
P-HIPSTer	Pathogen-Host Interactome Prediction using Structure Similarity
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
SA	Simulated Annealing
SMILES	Simplified Molecular Input Line Entry System

List of Symbols

Abbreviation	Full Form
α	Learning rate
D	Discriminator
G	Generator
IC50	Half Maximal Inhibitory Concentration
Kd	Dissociation Constant
L	Loss function
P	Probability
R	Reward

Contacts

Team Members

Name	Email	Phone Number
Nesma AbdEl-Kader Shafie	nasma.mahmoud02@eng-st.cu.edu.eg	+2 0111 224 29362
Sara Gamal Gerges	sarah.gerges01@eng-st.cu.edu.eg	+20 128 504 3196
Iman Ibrahim El-Bedwihy	iman.al-badwaihi02@eng-st.cu.edu.eg	+20 101 858 9574
Yousef Osama Mohamed	Youssef.Tawfiq02@eng-st.cu.edu.eg	+20 155 285 1677

Table 1.1 Team contacts

Supervisor

Name	Email	Number
Dr. Abd El Moniem Bayoumi	abayoumi@cu.edu.eg	+2 0111 121 5582

Table 1.2 Supervisor Contacts

Chapter 1: Introduction

Viruses are among the most significant threats to global public health, as seen in recent pandemics like COVID-19. The rapid emergence and evolution of viruses make it critical to develop intelligent computational tools that can predict viral behavior and support early drug discovery efforts. In this project, we present **ViroAlert**, an AI system that leverages deep learning, machine learning, graph neural networks, and generative modeling to help researchers and healthcare professionals detect potential viral threats and suggest effective antiviral treatments.

ViroAlert is composed of four integrated modules:

1. A **Deep Learning-based Virus Host Adaptability Predictor** that determines whether a virus can infect humans.

2. A **Machine Learning-based Virus Host Adaptability Predictor** that enhances interpretability through biological features.

3. A **Drug-Target Interaction (DTI) Predictor** that estimates how well existing drugs can bind to viral proteins.

4. A **Novel Drug Generator** that creates new drug-like molecules using generative adversarial networks (GANs) and reinforcement learning.

Together, these modules aim to provide a fast, scalable, and interpretable AI-driven platform for understanding and responding to viral threats.

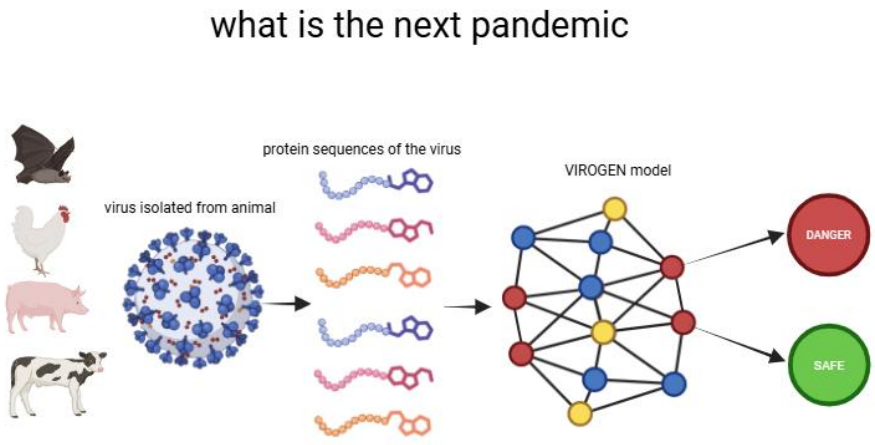

Figure 1.1: ViroAlert Idea

1.1. Motivation and Justification

Viruses can cause serious risks due to their ability to rapidly mutate and adapt to new hosts. Traditional laboratory-based methods for understanding host adaptability and developing antiviral drugs are time-consuming and expensive. Also, the lack of early warning tools and the slowness of drug development process contribute to delayed responses during outbreaks.

Our project addresses this gap by building a fully automated AI pipeline capable of:

- Predicting whether a virus can infect humans based on its protein sequences.
- Estimating how strongly a drug is likely to bind to a viral protein (i.e., its effectiveness).
- Generating new potential antiviral compounds tailored to a given virus.

The motivation behind ViroAlert is to provide a tool that can help in pandemic preparedness and rapid antiviral discovery. ViroAlert enables scalable, fast, and accurate analysis that would otherwise require months of manual experimentation.

1.2. Project Objectives and Problem Definition

The main objectives of the ViroAlert project are:

- **Objective 1**: Build an AI model that predicts human-infecting potential of viruses using their protein sequence data.
- **Objective 2**: Predict drug-target interaction using a graph-based deep learning model.
- **Objective 3**: Generate new, chemically valid drug molecules tailored to specific viruses using GANs and reinforcement learning.

Each of these objectives targets a specific problem in the viral analysis and drug development pipeline. Combined, they define our main problem:

> Design and develop an AI-based system that predicts human adaptability of viruses, evaluates drug-target interactions, and generates new antiviral drug candidates to accelerate viral threat assessment and treatment development.

1.3. Project Outcomes

The project has produced the following outcomes:

- A web-based application powered by four AI modules for real-time virus and drug analysis.
- A trained deep learning model for virus-host adaptability classification with visual interpretability.
- A machine learning model for virus-host adaptability classification.
- A Drug-Target Interaction Predictor using Graph Neural Networks that outputs predicted binding strength (pIC_{50} values).
- A generative model capable of producing new SMILES strings representing valid and novel drug candidates for a specific virus.

1.4. Document Organization

This report is organized into the following chapters:

- **Introduction**: This chapter introduces the ViroAlert project, describing the motivation behind it, the biological and computational problems it addresses. It clearly defines the project's goals, outcomes and outlines the structure of the document.
- **Market Visibility Study**: This chapter reviews existing bioinformatics tools and platforms related to virus-host prediction. It evaluates the advantages and disadvantages of each, highlighting the unique value ViroAlert provides in comparison.
- **Literature Survey**: This chapter provides the scientific and technical foundations required to understand the ViroAlert system. It includes background knowledge on the key technologies and methods used in our modules. It also reviews relevant prior research and studies that inspired or informed the development of each component of the project.
- **System Design and Architecture**: This chapter provides a detailed overview of ViroAlert's modular architecture. Each module—host prediction (DL & ML), DTI predictor, and drug generator—is described in terms of its function, data flow, model structure, and interconnectivity within the system.

- **System Testing and Verification**: This chapter summarizes how ViroAlert was evaluated and tested. It includes performance metrics for each module
- **Conclusions and Future Work**: In this chapter, we conclude with the key outcomes of ViroAlert and its potential in accelerating antiviral research. We also highlight areas for future improvement.
- **References**: This chapter lists all scientific articles that were used or cited throughout the report.

Chapter 2: Market Visibility Study

The ViroAlert project aims to improve how viral infections are diagnosed and treated by using AI to detect viruses from genetic sequences, identify vulnerability markers, and predict how well antiviral drugs will work. It uses advanced machine learning, including Generative Adversarial Networks (GANs) to help design new antivirals.

There is a growing demand for such tools, especially after global outbreaks like the **Spanish Flu** and **COVID-19**, which showed the need for faster and more accurate virus detection.

This chapter looks at the market for ViroAlert, including potential customers, competing products, and why it makes sense to bring ViroAlert to market.

2.1. Targeted Customers

ViroAlert's main customers are research labs and pharmaceutical companies working in virology and infectious diseases.

- **Research labs** use it to study viral genes and support vaccine development.
- **Pharmaceutical companies** benefit from its predictions of antiviral effectiveness and AI-generated drug candidates, which help speed up and lower the cost of drug discovery.

2.2. Market Survey

2.2.1. Competitive Project 1: DeepViral

2.2.1.1. Overview

DeepViral is a computational tool that uses deep learning models to predict virus-host interactions. It leverages CNNs to analyze viral and host genome sequences, aiming to determine whether a virus can infect a particular species.

2.2.1.2. Pros

- **Accurate Prediction:** DeepViral achieves high accuracy in predicting virus-host interactions due to the use of advanced neural architectures.

- **Sequence-Based:** It only requires genome sequences, making it relatively accessible.

- **End-to-End Pipeline:** Offers a complete computational framework from input to prediction.

2.2.1.3. Cons

- **Limited Interpretability:** The model functions as a black box, offering no insights into *why* a virus may infect a host or not.

- **No Drug Recommendation:** DeepViral does not predict or recommend antiviral drugs, which limits its practical use in treatment development.

- **Lack of Generalizability:** May not perform well on novel or mutated strains without retraining.

2.2.2. Competitive Project 2: P-HIPSTer

2.2.2.1. Overview

P-HIPSTER is a structure-based platform that predicts interactions between viral proteins and human host proteins using 3D structural similarity and known interactomes.

2.2.2.2. Pros:

- **Structural Insight:** Offers understanding by focusing on protein structure similarity.
- **Biologically Informed:** Takes into account real protein-protein interactions, improving biological validity.
- **Web-Based Interface**: Freely accessible and easy to use.

2.2.2.3. Cons:

- **Structure Dependency:** Relies heavily on the availability of accurate 3D protein structures, which are not always available.
- **No Drug Recommendation**: P-HIPSTer does not predict or recommend antiviral drugs, which limits its practical use in treatment development.

2.3. Business Case and Financial Analysis

2.3.1. Business Case:

Given the ongoing emergence of new viral strains and the increasing demand for personalized antiviral treatments, we estimate a compound annual growth rate (CAGR) of ~15% in demand for computational virology tools.
The price is $300/year.

Year	Expected Clients	Annual Revenue
1	100	$30,000
2	300	$95,000
3	600	$182,000
4	1000	$300,000
5	1500	$45,000

Table 2.1 Business Case

2.3.2. Financial Analysis

CAPEX (One-Time Costs)

Item	Cost (USD)
Cloud GPU servers (first year)	$5,000
Software Development	$4,000
Legal, licensing, and IP	$2,000
Marketing Launch Materials	$2,000

Office Equipment + Setup	$2,000
Total CAPEX	**$15,000**

Table 2.2 CAPEX

OPEX (Monthly Recurring Costs)

Item	Monthly Cost (USD)
Salaries people	$1,000
Cloud hosting	$500
Marketing & sales	$250
Admin & support	$250
Total OPEX	**$2,000**

Table 2.3 OPEX (Monthly Recurring Costs)

Cash flow table:

Month	Capex	Opex	Revenue	Net Cash Flow	Cumulative Cash Flow
1	15,000	2,000	2,500	-14,500	-14,500
2	0	2,000	2,500	500	-14,000
12	0	2,000	2,500	500	-8,500
13	0	2,000	7,917	5,917	-2,583

24	0	2,000	7,917	5,917	60,167
25	0	2,000	15,167	13,167	73,333
36	0	2,000	15,167	13,167	218,833
37	0	2,000	25,000	23,000	241,833
48	0	2,000	25,000	23,000	471,833
49	0	2,000	37,500	35,500	507,333
60	0	2,000	37,500	35,500	824,833

Table 2.4 Cash flow table

Break-even occurs approximately at:

- Month 14.
- So, break-even is reached in September 2026.

Chapter 3: Literature Survey

3.1. Background on 1D Convolution

A **1D convolutional layer** is a neural network layer designed to perform convolution operations on one-dimensional data. Unlike 2D convolutions, which slide a kernel across two spatial dimensions (as in image processing), 1D convolutions move the kernel along a single axis, making them ideal for sequential data**.** These layers are particularly effective at capturing local dependencies and patterns within sequences, such as relationships between neighboring words or characters. As a result, 1D convolutions are widely used in Natural Language Processing (NLP) tasks, including text classification.

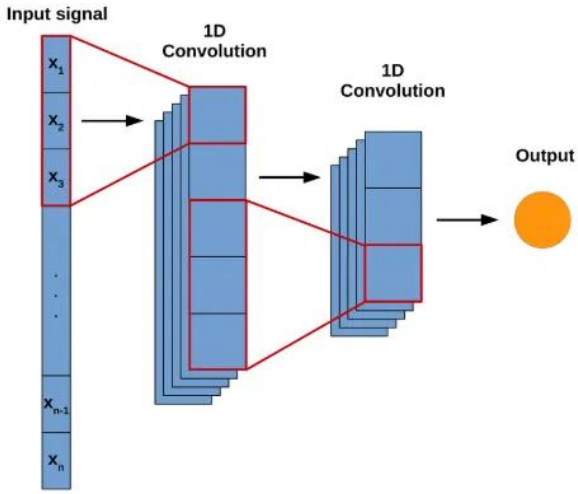

Figure 3.1 :1D CONV

3.2. Background on AVG Pooling

Average pooling is a pooling operation in convolutional neural networks (CNNs) that reduces the spatial dimensions of feature maps by calculating the average value of each region, typically a filter window, within the input. This operation helps to reduce computation and memory requirements.

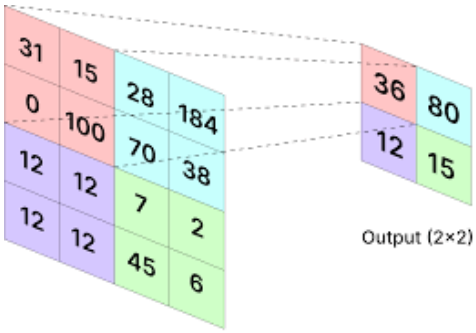

Figure 3.2.2: Average Pooling

3.3. Background on CNN

CNNs consist of multiple layers:

1. **Convolutional Layers**: This is the layer that is used to extract the features from the input dataset. It applies to a set of learnable filters

2. Activation Layer: By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network

3. Pooling layer: This layer is periodically inserted in the convnets, and its main function is to reduce the size of the volume, which makes the computation fast, reduces memory, and also prevents overfitting.
   We used MIL to aggregate features from multiple segments of the viral protein sequence, improving accuracy by allowing the model to consider all parts of the sequence. We used MIL to aggregate features from multiple segments of the viral protein sequence, improving accuracy by allowing the model to consider all parts of the sequence.
4. Flattening: The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers

5. Fully Connected Layers: It takes the input from the previous layer and computes the final classification task.

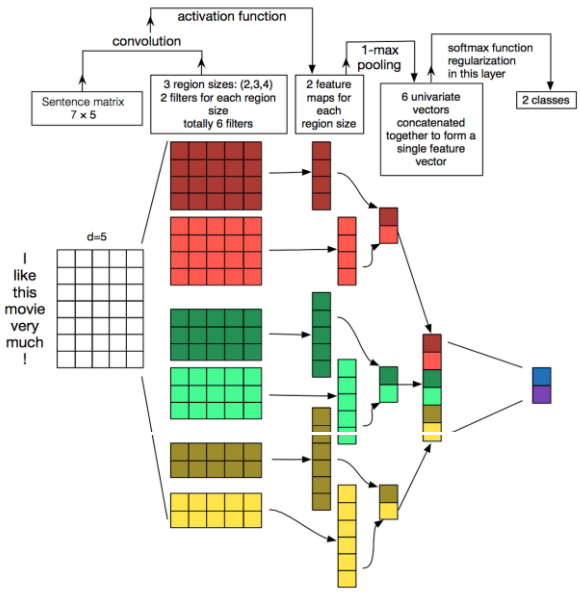

Figure 3.3 :CNN on text

3.4. Background on Transformer Encoder

The encoder is composed of multiple layers that repeatedly transform the input through a combination of attention mechanisms and feed-forward networks.

The encoder processes the input data **step by step**:

1. **Input Embedding**:
    - Convert the input tokens into word embeddings.
    - Embeddings allow the model to work with numerical representations of words, capturing information about word relationships in the input sequence

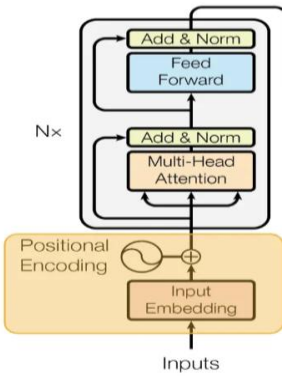

Figure 3.4:Transformer Encoder Layer

2. **Positional Encoding**:
    - Transformers have no inherent sense of word order because they process inputs in parallel.
    - To address this, the encoder adds positional encoding to the input embeddings
    - This additional information helps the model understand the order of tokens in a sequence.

Formulation for Positional Encoding

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

3. **Self-Attention**:
    - Each word attends to other words in the sequence, identifying relationships and context.

   - For each word (token) in the input sequence, the model computes three vectors: Query (Q), Key (K), and Value (V).
   - The attention score is calculated by taking the dot product of the Query vector of one word with the Key vectors of all other words. This score determines how much focus (or attention) the model should give to each word in the sequence.
   - The attention scores are then used to create a weighted sum of the Value vectors, which forms the output of the self-attention layer.
4. **Layer Normalization**:
   - Normalize the output to ensure stable training.
5. **Residual Connections**:
   - Residual connections allow the output of each layer to "skip" over the next layer and be added directly to the output of the following layer.
   - Residual connections help the model retain information from earlier layers, even as it goes through deeper and deeper layers.
6. **Feed-Forward Layer**:
   - Apply a fully connected network to refine the output.

3.5. Background on LSTM

LSTMs can capture long-term dependencies in sequential data.
It solves the two main problems in RNN:

1. **Vanishing Gradient**: When training a model over time, the gradients that help the model learn can shrink as they pass through many steps. This makes it hard for the model to learn long-term patterns since earlier information becomes almost irrelevant.
2. **Exploding Gradient**: Sometimes gradients can grow too large, causing instability. This makes it difficult for the model to learn properly, as the updates to the model become erratic and unpredictable.

LSTM architecture involves the memory cell, which is controlled by three gates:

1. Input gate: Controls what information is added to the memory cell.
2. Forget gate: Determines what information is removed from the memory cell.

3. Output gate: Controls what information is output from the memory cell.

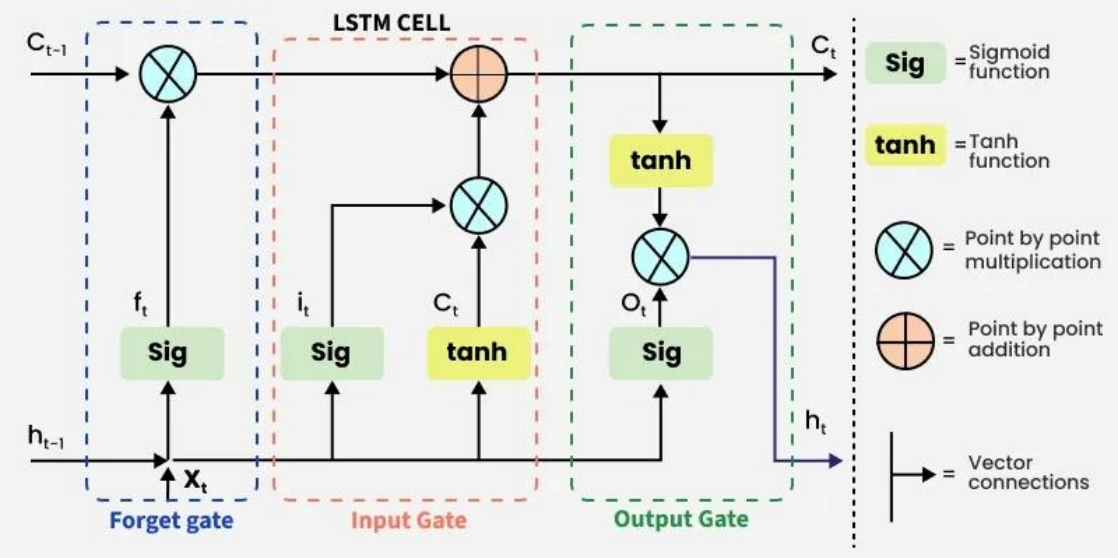

Figure 3.5: LSTM Architecture

3.6. Background on Multi-Instance Learning

- Multiple instance learning (MIL) is a form of weakly supervised learning where training instances are arranged in sets, called bags.
- Labels are provided for entire bags rather than for the individual instances contained in them.
- **Attention Mechanism:**
  - It is possible to modify the embedding-based approach to be interpretable using an attention-based MIL pooling method
  - Let H = {h_1,…,h_k} be a bag of K embeddings. The following attention-based MIL pooling can be used:

$$\mathbf{z} = \sum_{k=1}^{K} a_k \mathbf{h}_k$$

where:

$$a_k = \frac{\exp\{\mathbf{w}^\top \tanh(\mathbf{V}\mathbf{h}_k^\top)\}}{\sum_{j=1}^{K} \exp\{\mathbf{w}^\top \tanh(\mathbf{V}\mathbf{h}_j^\top)\}}$$

- V and w are trainable parameters that ultimately allow the model to learn the embedding features that are responsible for triggering a positive bag label

- **Gated Attention Mechanism:**
  - While the standard tanh non-linearity helps capture relationships, it may be limited in modeling complex interactions.
  - To overcome this, a gated attention mechanism is introduced, which combines tanh and sigmoid activations for better expressive power:

$$a_k = \frac{\exp\{\mathbf{w}^\top \left(\tanh\left(\mathbf{V}\mathbf{h}_k^\top\right) \odot \mathrm{sigm}\left(\mathbf{U}\mathbf{h}_k^\top\right)\right)\}}{\sum_{j=1}^{K} \exp\{\mathbf{w}^\top \left(\tanh\left(\mathbf{V}\mathbf{h}_j^\top\right) \odot \mathrm{sigm}\left(\mathbf{U}\mathbf{h}_j^\top\right)\right)\}}$$

3.7. Background on Fast-Text

FastText is an extension of traditional word embedding models like Word2Vec and GloVe, with a key innovation: instead of treating each word as a distinct atomic unit, FastText represents words as a collection of subword units (character n-grams). This enables it to generate meaningful word embeddings even for out-of-vocabulary (OOV) words—those not seen during training—by leveraging shared subword structures.

FastText employs neural network-based architecture, primarily CBOW (Continuous Bag of Words) and Skip-gram, to learn word representations:

- **CBOW** predicts a target word based on its surrounding context words.
- **Skip-gram** does the reverse: it predicts surrounding context words given a target word.

These models produce static word embeddings, meaning each word (or subword) is associated with a single, fixed vector regardless of context. Despite this limitation,

FastText remains highly effective, especially in morphologically rich languages and tasks where OOV handling is important.

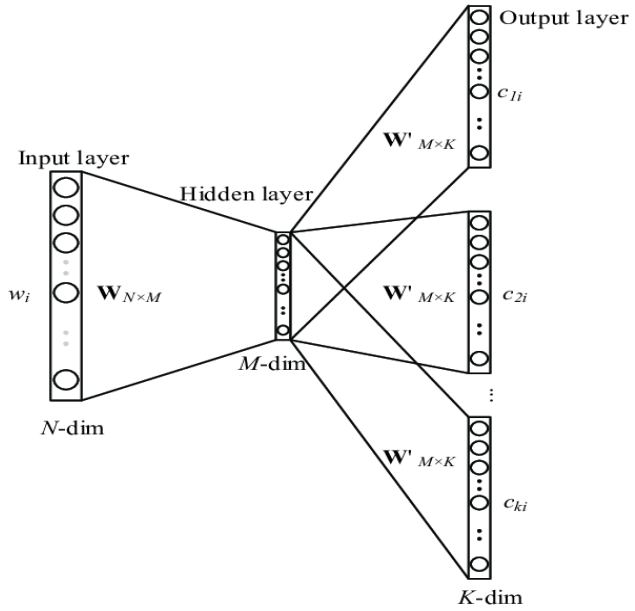

Figure 3.7 :Skip Gram Network Architecture

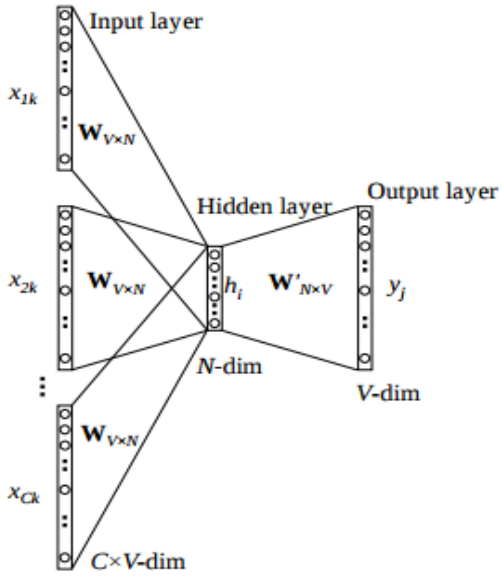

Figure 3.6 :CBOW Network Architecture

3.8. Background on Graph Convolution Network

- Graph Convolutional Networks (GCNs) are a class of neural networks designed to operate directly on graph-structured data.
- Unlike traditional convolutional neural networks (CNNs), which work on regular grid-like data such as images, GCNs can model complex relationships and interactions between entities represented as nodes in a graph.
- In a GCN, each node represents an entity, and the edges represent the relationships between these entities.
- The primary goal of GCNs is to learn node embeddings, which are vector representations of nodes that capture the graph's structural and feature information.
- **How GCNs Work:** GCNs aggregate information from a node's local neighborhood to update its feature representation. At each layer of the GCN, a node's new representation is computed as a function of:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

where:

- $H^{(l)}$: Node features at layer l
- $\tilde{A}=A+I$: Adjacency matrix with added self-loops
- $\tilde{D}$: Degree matrix of $\tilde{A}$ (counts the number of edges for the corresponding node i)
- $W^{(l)}$: Trainable weight matrix at layer l
- σ: Activation function (e.g., ReLU)

- This operation performs neighborhood aggregation, allowing the network to capture both node attributes and graph topology.

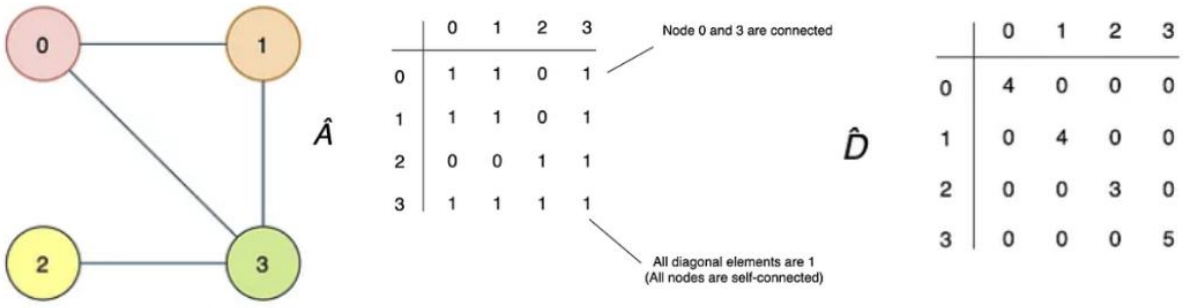

Figure 3.8 :GCN Computations

3.9. Background on Needleman-Wunsch

The Needleman-Wunsch algorithm is a dynamic programming approach used for global sequence alignment. It aligns two sequences (e.g., DNA, RNA, or protein sequences) over their entire length to identify similarities, accounting for insertions, deletions, and substitutions. The algorithm constructs a scoring matrix where each cell represents the optimal alignment score up to that position, using a scoring scheme that includes:

- A match score (typically +1) for identical residues,
- A mismatch penalty (e.g., -1) for differing residues,
- A gap penalty (e.g., -1) for insertions or deletions.

The process entails:

- Pre-filling the matrix with zeros or gaps penalties on the edges.
- Fill up the matrix by calculating the scores as the maximum of the three possible movements: diagonal (match/mismatch), left (insertion), or top (deletion).
- Back trace from bottom-right cell to reconstruct the aligned sequences, indicating matches, mismatches, or gaps.

This method assures an optimal global alignment with a time complexity of O(mn) and space complexity of O(mn), where m and n are the sizes of the two sequences. This method is used generally in bioinformatics for purposes like phylogenetic analysis and sequence comparison and provides a solid foundation for the sequence alignment component of ViroGen.

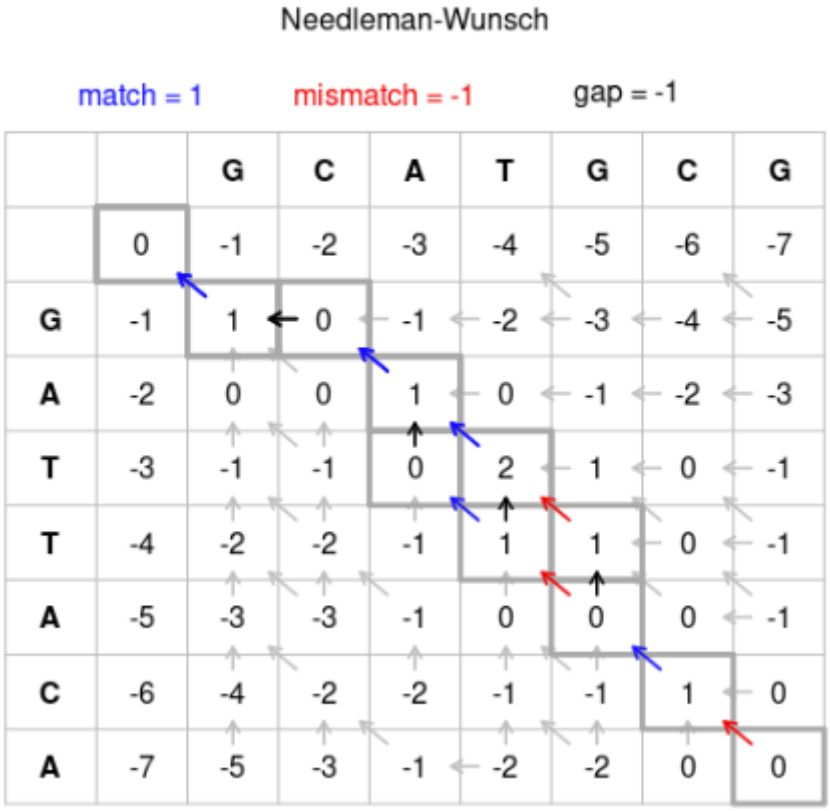

Figure 3.9: Needleman-Wunsch algorithm

3.10. Background on GANs Architecture

Generative Adversarial Networks (GANs) are a class of deep learning models designed to generate realistic synthetic data by training two neural networks in opposition to one another. These two networks, the Generator and the Discriminator, form a minimax game where each improves by challenging the other.

1. **Generator (G)**: The generator is a deep neural network that starts with a random noise vector as input and attempts to produce data that mimics the real dataset, such as images, text, or audio. Its goal is to generate samples so realistic that the discriminator cannot distinguish them from real data. During training, it updates its internal parameters to better fool the discriminator.

2. **Discriminator (D)**: The discriminator is a binary classifier trained to differentiate between real samples (from the actual dataset) and fake samples (produced by the generator). It outputs a probability indicating whether a given input is real or generated. For image data, the discriminator typically uses convolutional layers to extract features and improve classification accuracy.

How GANs Work:

1. Generator's First Move
    - The generator receives a random noise vector (e.g., Gaussian or uniform distribution) and transforms it into a synthetic data sample. This could be a generated image that imitates real-world data.
2. Discriminator's Evaluation
   The discriminator receives two types of input:
    - Real samples from the training dataset.
    - Fake samples were generated by the generator.
      It then outputs a probability score between 0 (fake) and 1 (real), indicating its confidence about the authenticity of the input.
3. Adversarial Training
    - The generator tries to minimize the probability that the discriminator detects its samples as fake.

    - The discriminator tries to maximize its accuracy in distinguishing real from fake.
      This forms a Minimax Loss function defined as:

    - Training continues in an alternating fashion, where each model learns from the other's improvements, leading to progressively better generated data. The generator tries to minimize this loss (to fool the discriminator), and the discriminator tries to maximize it (to detect fakes accurately).

$$min_G\ max_D(G, D) = [\mathbb{E}_{x \sim p_{data}}[log\ D(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(g(z)))]$$

3.11. Comparative Study of Previous Work

3.11.1 DL Viral Host Prediction:

Among the most notable works is **ViRNN** (2024), a deep learning model specifically designed to predict the host species of Coronaviruses using genomic sequence data. ViRNN employs a hybrid CNN+RNN architecture (with GRU or LSTM layers) and achieves 90.55% accuracy on subsequences and 95.3% on full-length sequences across 11 host classes. It significantly outperforms traditional models such as K-Nearest Neighbors (KNN) and SVM.
Another highly relevant work is the study titled **"Prediction of virus-host associations using protein language models and multiple instance learning"**, which presents a scalable and interpretable framework that leverages pre-trained protein language models to generate embeddings of viral protein sequences. These embeddings are then processed using a Multi-Instance Learning (MIL) strategy, where each virus is modeled as a “bag” of protein representations, enabling host prediction across different taxonomic levels.

Another significant advancement is **HostNet** (2023), a deep learning framework aimed at improving virus-host prediction from genomic sequences, particularly under conditions of class imbalance and limited data. HostNet integrates a Transformer-CNN-BiGRU architecture with two innovative sequence representation techniques like K2v and ASW.

3.11.2. ML Viral Host Prediction

Virus2Vec is a biologically inspired embedding technique that transforms protein sequences into fixed-length vectors using minimizers. It builds Position Frequency Matrices (PFM), normalizes them into Position Probability Matrices (PPM), and finally computes a Position Weight Matrix (PWM) through log-likelihood scoring. Each m-Mer is scored against the PWM to produce an embedding that captures both compositional and positional information.

MFV (Minimizer-based Feature Vector) focuses on creating alignment-free fixed-length vectors by counting the frequency of minimizers (lexicographically smallest *m-Mers*) across a sequence. This method is simple yet powerful, as it preserves key patterns in local sequence context and scales well with large datasets. MFV has proven effective in traditional machine learning pipelines like SVM and Random Forest, particularly when computational efficiency and interpretability are priorities

PWM2Vec is an alignment-based technique that assigns position-specific weights to each *k-Mer* based on values from a Position Weight Matrix constructed from

multiple sequence alignments. Unlike MFV, PWM2Vec considers evolutionary conservation and structural significance at specific sequence positions, which allows for capturing nuanced biological signals. However, its dependency on aligned sequences limits its applicability to well-curated datasets

PSWM2Vec enhances PWM2Vec by using minimizers instead of *k-Mers* and generating a Position-Specific Weight Matrix (PSWM), which is then flattened to produce the final feature vector. This hybrid approach reduces redundancy while retaining alignment-based precision, making it particularly suitable for datasets with rich sequence alignments but a need for compact representation

3.11.3. Drug-Target Interaction

Recent work on **Drug-Target Interaction prediction** has explored using neural networks and graph-based methods to model the binding affinity between small molecules and target proteins. Models such as **DeepDTA** and **GraphDTA** have shown that protein sequence embeddings, combined with drug structure encodings (e.g., SMILES or molecular graphs), can achieve high accuracy in predicting interaction scores.

These models typically use a regression-based framework to predict binding affinity values and have been used to screen existing drugs for repurposing against emerging pathogens like SARS-CoV-2.

3.11.4. Drug Generation

Several studies have proposed **generative models** for molecule synthesis using SMILES-based representations. Models such as:

- **MolGAN**: A GAN-based model that generates molecular graphs with a reinforcement signal for chemical validity.

- **Reinforced GANs**: Which apply policy gradient techniques to optimize molecules using custom reward functions (e.g., validity, binding affinity, novelty).

3.12. Implemented Approach

Module	Used Approach	Justification
Viral Host Prediction	ML based on Virus2vec and m-Mer	Virus2vec embedding technique that captures contextual and positional relationships between amino acid fragments.
	DL based on multi-instance learning, CNN and encoder layers	We used MIL to aggregate features from multiple segments of the viral protein sequence, improving accuracy by allowing the model to consider all parts of the sequence.
Drug-Target Interaction	Graph smile and drugs model	Graph Neural Networks (GNNs) were chosen because molecular structures and proteins are naturally represented as graphs, which capture complex relationships better than linear models.
Drug Generation	Reinforced GANs	Reinforced GANs combine the generative power of GANs with reinforcement learning to generate chemically valid and effective molecules while being effective against specific virus.

Table 3.1 Implemented Approaches for Each Module

Chapter 4: System Design and Architecture

This chapter presents the complete design and implementation details of our proposed system. It serves as the core body of our work, describing what has been achieved and how each part of the system was developed. The chapter walks through the full project lifecycle, starting from data collection and preprocessing, through model development and integration, and concluding with the generation of final analytical results and reports. Each stage is carefully explained to provide clear insight into the system's workflow and to support reproducibility by future researchers.

4.1. Overview and Assumptions

In designing ViroAlert system, we aimed to build a modular pipeline that streamlines the process of analyzing viral sequences and predicting potential therapeutic strategies. The architecture is composed of four core modules: the Virus Host Adaptability Predictor using deep learning, a complementary machine learning-based predictor, a Drug-Target Interaction (DTI) module, and a Novel Drug Generator that combines GANs with reinforcement learning. These modules function collaboratively, each receiving specific inputs (such as protein sequences or drug SMILES) and producing outputs that feed into the overall analysis workflow. Our design assumes access to reliable viral genomic or protein sequences, accurate representation of drugs in SMILES format, and sufficient computational resources to handle deep learning and graph-based processing. These assumptions informed our architectural decisions and helped ensure the system remains scalable, interpretable, and applicable to real-world virology and drug discovery challenges.

4.2. System Architecture

The proposed system consists of four distinct modules, each with a well-defined functionality. These modules are connected in a pipeline, where the output of one module serves as the input for the next, enabling an end-to-end workflow from viral classification to drug generation. Below is a description of each module:

1. **Virus Host Adaptability Predictor (Deep Learning-based)**
   This module classifies whether a virus exhibit features indicative of human

adaptation, whether it has the potential to cause a pandemic. It uses deep learning to analyze protein or genomic sequences and outputs both the prediction and attention-based highlights of the most informative subsequences that influenced the classification decision.

2. **Virus Host Adaptability Predictor (Machine Learning-based)**
   Complementing the deep learning module, this version relies on biologically inspired, engineered features (such as nucleotide compositions). It also predicts human adaptability and identifies the most influential features contributing to the prediction.

3. **Drug-Target Interaction Predictor (DTI Model)**
   This module estimates the interaction between viral proteins and drug molecules. By analyzing the chemical properties of both, it predicts which commercially available drugs are most likely to bind effectively to the viral target, thereby helping prioritize candidates for drug repurposing.

4. **Novel Drug Generator (GAN + Reinforcement Learning)**
   This generative module uses a GAN architecture combined with reinforcement learning to generate new, valid drug-like molecules. Initially trained to produce chemically plausible drugs, the model is later fine-tuned using domain-specific rewards (e.g., similarity to known inhibitors) to generate compounds that are more likely to be effective against a specific virus.

4.2.1. System Block Diagram

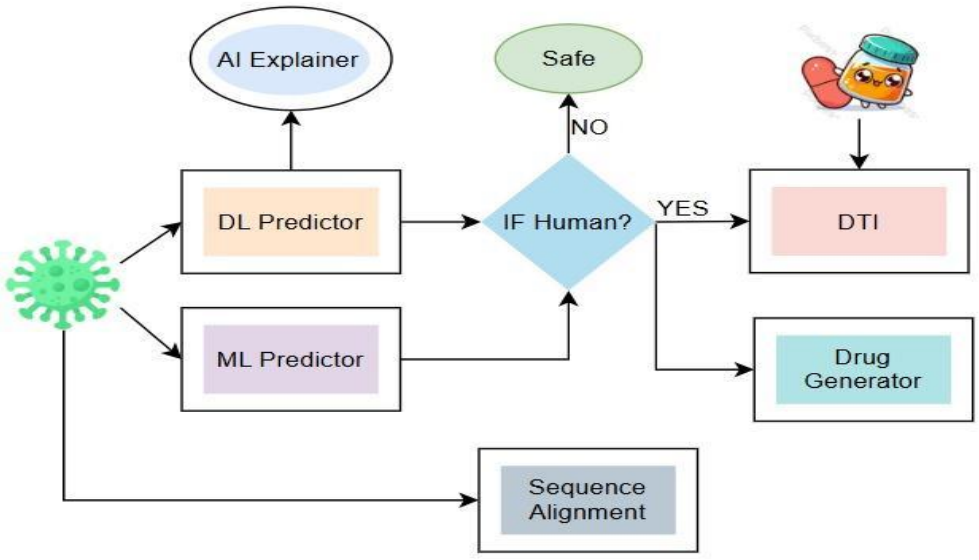

Figure 4.1 System Design

4.3. Viral Host Adaptability Predictor DL Approach

4.3.1. Functional Description

This module serves as the core component of the system, responsible for learning host adaptability signals from viral protein sequences. Its primary objective is to provide early warnings to scientists by identifying viruses with the potential to cause pandemics. The module aggregates information from segmented viral genomes and predicts whether a given virus is likely to infect humans. It is built upon a **Multi-Instance Learning (MIL)** framework, where each virus is modeled as a "bag" comprising multiple "instances" (i.e., subsequences or genome segments). This structure enables the model to analyze and integrate information across multiple parts of the viral genome, capturing both local and global patterns associated with host adaptation.

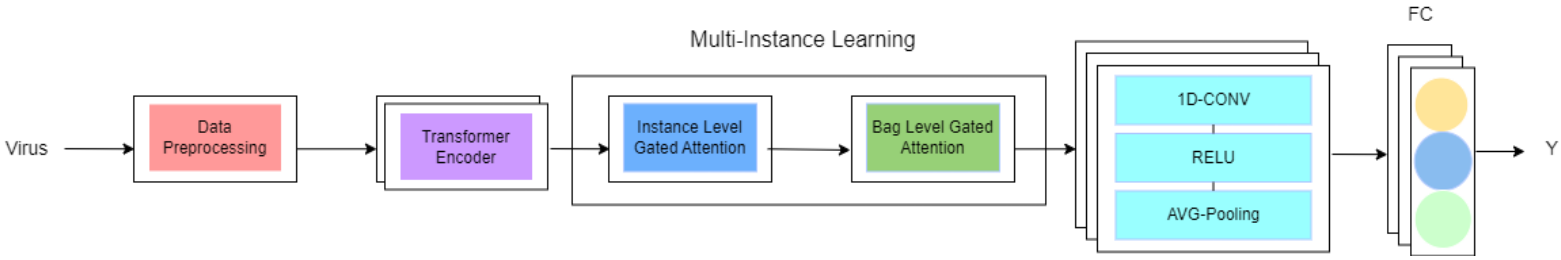

Figure 4.2: Viral Host Adaptability Predictor with DL Block Diagram

4.3.2. Modular Decomposition

4.3.2.1 Data Preprocessing:

1. **Duplicate Removal**: We eliminated duplicate sequences to prevent redundancy and reduce the risk of model overfitting. Duplicate entries can bias the model towards overrepresented sequences, impairing its ability to generalize to unseen data.

2. **Filtering Incomplete Sequences**: We excluded incomplete sequences, such as those that were truncated, partially annotated, or missing substantial regions. This step ensures that only full-length or nearly complete sequences are included in the training process, maintaining consistency in input length and information content.

3. **Ambiguous Character Filtering**: We discarded sequences containing a high proportion of ambiguous nucleotide symbols, particularly the character ‘X’. A threshold for allowable ambiguity was defined, and any sequence exceeding this threshold was removed to avoid feeding unreliable information into the model.

4. **High-Similarity Sequence Reduction**: Viral genome datasets often include multiple sequences that are highly similar or nearly identical, particularly those derived from the same outbreak or region. To reduce redundancy and overfitting, we used sequence similarity analysis to identify and remove overly similar sequences, ensuring a more diverse and representative dataset.

4.3.2.2 Embedding:

1. . We first utilize a pre-trained protein language model, which has been trained on a large corpus of viral protein sequences, to extract rich and informative representations that capture the underlying biological features of each protein.

2. In the next step, we customized the embedding process to better suit the specific requirements of our problem. We used the following techniques:

- **Tokenization with Adaptive Sliding Window**: To handle variable-length viral genome sequences, we apply an Adaptive Sliding Window (ASW) tokenization strategy. This approach segments each sequence into overlapping, fixed-length subsequences (tokens) while ensuring consistency across different sequence lengths. For an input sequence of length l, with a desired number of tokens n and token length l_sub, the stride s is computed as:

$$s = \left\lfloor \frac{l - l_{\text{sub}}}{n - 1} \right\rfloor$$

- **Initial Embedding via FastText (CBOW) Model:** We use the FastText model, based on the Continuous Bag-of-Words (CBOW) approach, to capture semantic relationships between tokens by predicting each token from its surrounding context, similar to its application in natural language processing. Each token is passed through a trained FastText model to obtain an initial dense vector representation, which is then fed into the neural network for further refinement.

- **Contextual Enhancement with Transformer Encoder:** To model contextual dependencies and relationships between tokens, we

employ a two-layer Transformer Encoder. This encoder uses self-attention mechanisms to enhance each token's embedding by capturing its interactions with neighboring tokens within the same sequence. This step ensures that the final embeddings reflect both local patterns and broader sequence-level structures.

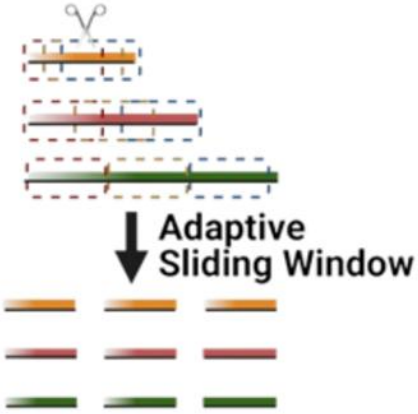

Figure 4.3: Adaptive Sliding Window Technique

4.3.2.3 Hierarchical Gated Attention Multi-Instance Learning:

4.3.2.3.1 Instance-Level Gated Attention

For each subsequence (instance), we compute a gated attention score as follows:

$$\alpha_k = \frac{\exp\left\{w^T\left(\tanh(Vx_k) \odot \sigma(Ux_k)\right)\right\}}{\sum_{j=1}^{K}\exp\left\{w^T\left(\tanh(Vx_j) \odot \sigma(Ux_j)\right)\right\}}$$

where $x_k \in R^M$ is the encoded instance vector, $V, U \in R^{LxM}$ are learnable projection matrices, $w \in R^L$ is the attention vector, σ denotes the sigmoid function, and $\odot$ represents element-wise multiplication. The attention scores α_K weight each token's contribution, and the sequence representation is computed as:

This mechanism prioritizes tokens encoding biologically significant regions, such as those associated with receptor-binding domains critical for host adaptability.

$$s = \sum_{k=1}^{K} \alpha_k x_k$$

4.3.2.3.2 Bag-Level Gated Attention

At the virus level, we treat each virus as a bag of sequence representations (bags of bags). A second gated attention layer aggregates these sequence representations into a single virus-level vector using the same formulation but applied to sequence embeddings s_i . This hierarchical approach captures both local token-level patterns (e.g., specific nucleotide motifs) and global sequence-level interactions (e.g., inter-segment dependencies), accommodating the multi-segment nature of many viruses**.**

4.3.2.4 Classification:

The final virus-level representation is fed into a deep convolutional neural network (CNN) for binary classification (human vs. non-human host). The CNN architecture is designed to capture both local and global patterns:

1. Convolutional Layers: Three 1D convolutional layers with kernel sizes of 4, 5, and 7, respectively, each followed by batch normalization and ReLU activation. These layers detect local sequence motifs.

2. Pooling Layers: Average pooling is applied after the convolutional layers to reduce dimensionality while preserving salient features, improving computational efficiency, and reducing overfitting.

3. Fully Connected Layers: The flattened feature map is passed through three dense layers (256 and 128 units) with batch normalization and ReLU activation, enabling the model to learn complex, non-linear relationships between features.

4. Output Layer: A single neuron with sigmoid activation outputs the probability of human host adaptability, providing a clear decision boundary for classification.

4.3.3. Visualization of Virus-Level Embeddings

To evaluate the ViroAlert model's capacity to encode host-specific features, we applied t-SNE to reduce the 30-dimensional virus-level embeddings, generated post-bag-level attention, to a 3D representation, as illustrated in the following Figure.

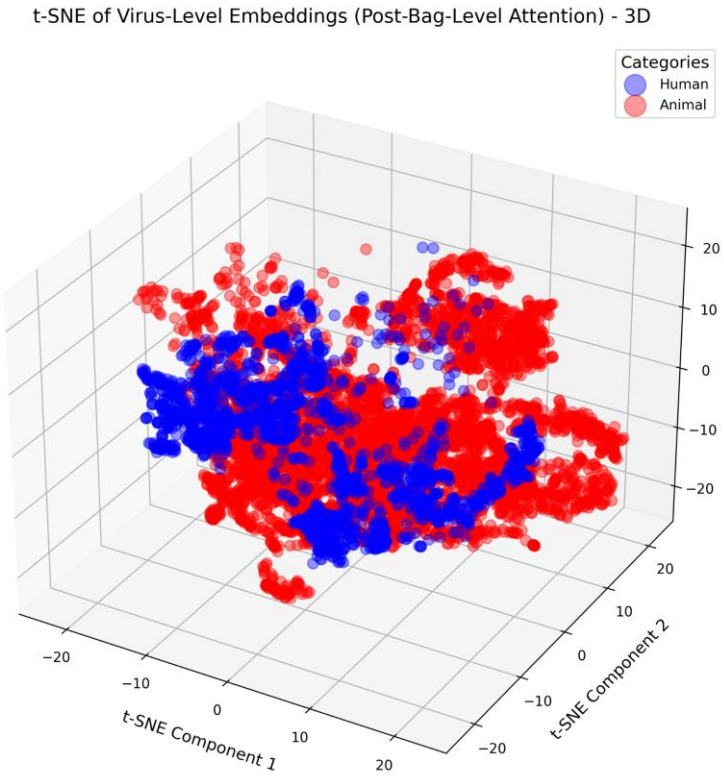

Figure 4.4 : Clustered Data

This visualization is based on a random subset of 10,000 sequences, encompassing approximately 5,000 unique viruses, with each virus associated with 1 to 3 sequences as identified during processing. The embeddings are color coded, with blue representing human viruses (label 0) and red indicating animal viruses (label 1). The 3D t-SNE plot reveals distinct clustering patterns between human and animal viruses across the three components, underscoring the model's ability to capture host-related characteristics.

4.3.4. Attention Weight Analysis

To explain the ViroAlert model's attention mechanism, we visualized the instance-level (A) and bag-level (A_2) attention weights using heatmaps, as shown in the following Figures:

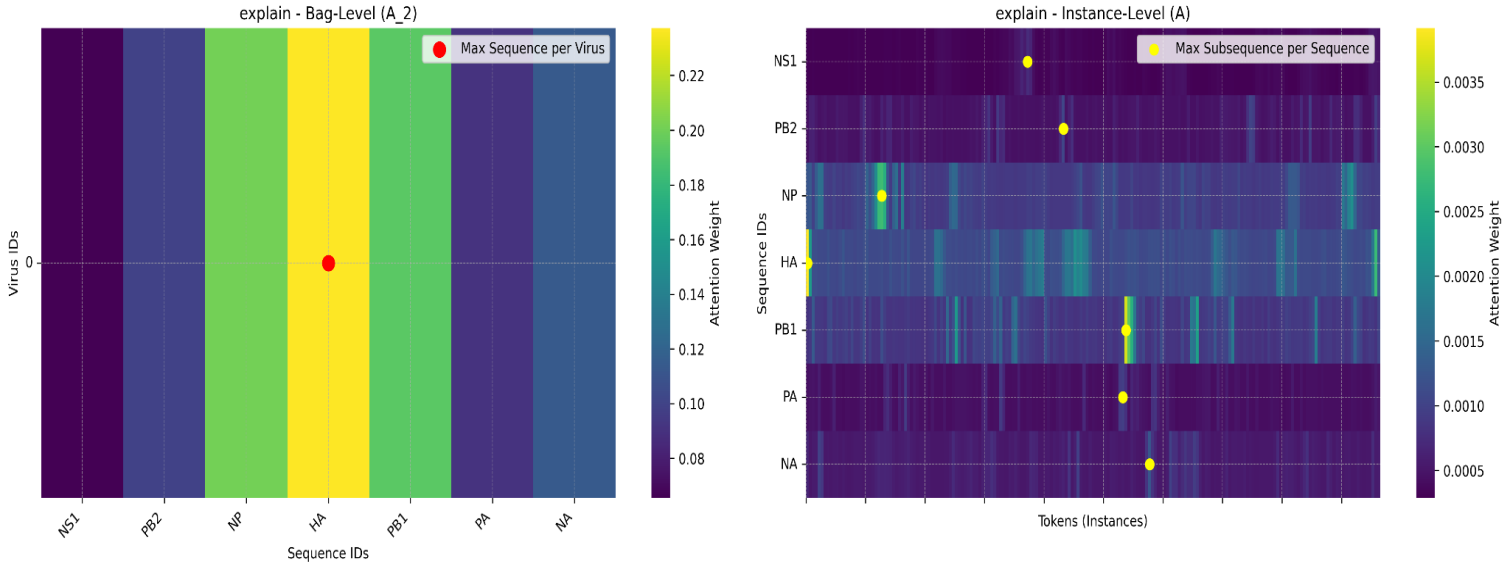

Figure 4.5: Attention weights

This analysis was performed on a single virus from the dataset, represented by 7 sequences, with each sequence containing 193 tokens. The instance-level heatmap highlights token importance within sequences, marked with yellow dots for the maximum attention per sequence. The bag-level heatmap illustrates the relative importance of sequences within the virus, with a red dot indicating the global maximum. These patterns suggest that the model effectively weighs tokens and sequences. Further analysis showed that the attention weights often focused on key viral proteins like hemagglutinin (HA), which is widely recognized in biology as the most critical sequence for infectivity due to its essential role in mediating viral attachment to host cell receptors, thereby confirming the model's biological accuracy.

4.3.5. Design Constraints

1. We experimented with using a pretrained model for sequence embedding. However, fine-tuning the model to better suit our specific problem proved infeasible due to the high computational resource requirements, which exceeded our available capacity. So we could not use it.
2. **2. Memory Constraints During Attention Mechanism Computation** The use of hierarchical gated attention required storing and computing large intermediate matrices for instance- and bag-level attention, especially when working with viruses composed of many long sequences. This limited the batch size we could use during training and constrained our ability to scale up the model.
3. **t-SNE Scalability for Visualization** Although t-SNE provided valuable insight into virus-level embeddings, it has poor scalability for large datasets and high dimensions. Even with dimensionality reduction to 30D before t-SNE, visualization was only practical on a limited subset of the full dataset (10,000 sequences), potentially underrepresenting subtle embedding patterns.
4. **Trade-off Between Window Size and Sequence Context** The Adaptive Sliding Window (ASW) approach required selecting optimal token length and stride. Larger windows preserved more context but led to fewer tokens per sequence, reducing the effectiveness of attention. Smaller windows, on the other hand, improved local resolution but increased noise and computational overhead. Finding a universally optimal configuration proved difficult.

4.4. Viral Host Adaptability Predictor ML Approach

This is a machine learning module designed to classify whether a viral genetic sequence is likely to infect human hosts. The prediction is based on features extracted from the sequences using various encoding and representation techniques that capture biologically relevant patterns. Several approaches were explored to find the most effective feature extraction strategies.

4.4.1. Functional Description

The main function of this module is to take an input protein sequence (in FASTA format), extract biologically meaningful features, and predict its adaptability to human hosts.

The steps include:

1. **Preprocessing** Raw viral protein sequences are preprocessed to ensure consistency and completeness. This includes removing duplicates, validating FASTA formatting, and handling ambiguous amino acids.

2. **Feature extraction** Biologically relevant features are extracted using a combination of:
   2.1. **k-Mer based embedding techniques** such as **Virus2Vec**, **PSWM2Vec**, and **MFV**.
   2.2. Other compositional and evolutionary descriptors, which help represent the underlying biological structure of the viral proteins

3. **Model Training** Multiple classical machine learning classifiers (e.g., **Support Vector Machine**, **Random Forest**, **Logistic Regression**) are trained on a labeled dataset indicating whether each sequence is human-adaptive or not.

4. **Hyperparameter Fine-Tuning** Each classifier undergoes **hyperparameter optimization** using cross-validation on the training/validation split to maximize model performance and avoid overfitting.

5. **Evaluation of Generalization** To assess the robustness and generalization of the trained models, we evaluate them on **independent pandemic test sets**, which are not seen during training. These include:
   5.1. Spanish Flu (1918–1920)
   5.2. Asian Flu (1957–1958)
   5.3. Hong Kong Flu (1968–1970)
   5.4. Swine Flu (2009–2010)
   5.5. COVID-19 Pandemic (2019–2022)

4.4.2. Modular Decomposition

The Machine Learning Viral Host Adaptability predictor module is decomposed into five fine-grained submodules, each responsible for a critical stage in the machine learning pipeline. Below is a detailed breakdown of each component:

4.4.2.1. Sequence Preprocessing

1. **Duplicate Removal**
   Eliminate duplicate sequences to prevent redundancy and reduce the risk of model overfitting. Duplicate entries can bias the model towards overrepresented sequences, impairing its ability to generalize to unseen data.

2. **Filtering Incomplete Sequences**
   Exclude incomplete sequences, such as those that were truncated, partially annotated, or missing substantial regions. This step ensures that only full-length or nearly complete sequences are included in the training process, maintaining consistency in input length and information content.

3. **Ambiguous Character Filtering**
   Discard sequences containing any character **not part of the standard 20 amino acids** (e.g., 'X', '*', or other undefined symbols) to avoid feeding unreliable information into the model.

4. **High-Similarity Sequence Reduction**
   Viral genome datasets often include multiple sequences that are highly similar or nearly identical, particularly those derived from the same outbreak or region. To reduce redundancy and overfitting, we used sequence similarity analysis to identify and remove overly similar sequences, ensuring a more diverse and representative dataset.

5. **Train-Test Overlap Removal**
   Ensure that **no sequence appears in both the training and testing datasets**. This step prevents data leakage and guarantees a fair evaluation of the model's generalization ability.

Purpose: These preprocessing steps improve data quality, reduce redundancy, and ensure that the dataset is biologically meaningful, diverse, and free from ambiguity or leakage. This maximizes the reliability and validity of the machine learning models trained on the data.

4.4.2.2. Feature Extraction Approaches

We evaluated Virus2Vec, PSWM2Vec and MFV as baseline approaches for our deep learning model, using these machines learning-based sequence embedding methods to convert biological sequences into fixed-length feature vectors suitable for viral host classification

4.4.2.2.1. Virus2Vec

Virus2Vec is a biologically inspired, minimizer-based embedding approach that efficiently encodes viral sequences into fixed-length vectors. The core steps of the Virus2Vec pipeline are as follows:

1. **Minimizer Generation:**
   Given a sequence, overlapping **k-Mers** are extracted. For each k-Mer, the **minimizer** is defined as the **lexicographically smallest m-Mer** (where m < k) found within the k-Mer, considering both forward and reverse directions. This step reduces redundancy and memory usage by focusing on representative substrings.

2. **Position Frequency Matrix (PFM):**
   Each minimizer is used to build a **PFM**, which records the frequency of each character at every position within the m-mer. The matrix has dimensions $|\Sigma| \times m$, where Σ is the amino acid alphabet.

3. **Position Probability Matrix (PPM):**
   The PFM is normalized into a **PPM** to reflect the probability distribution of characters at each position. To handle zero probabilities (which could distort downstream calculations), **Laplace smoothing** (pseudo count of 0.1) is applied.

4. **Position Weight Matrix (PWM):**
   Using the PPM, a **PWM** is computed by taking the **log-likelihood** of each character c at position i concerning its background frequency:

$$Wc,i = log2(\frac{P(c,i)}{p(c)}), \quad p(c) = (\frac{n(c)}{61})$$

   This captures how informative each position is for the sequence context.

5. **Feature Vector Construction:**
   Each m-Mer is scored using the PWM, and the scores are aggregated into a

fixed-length vector of dimension |Σ| × m, which serves as the final embedding of the sequence.

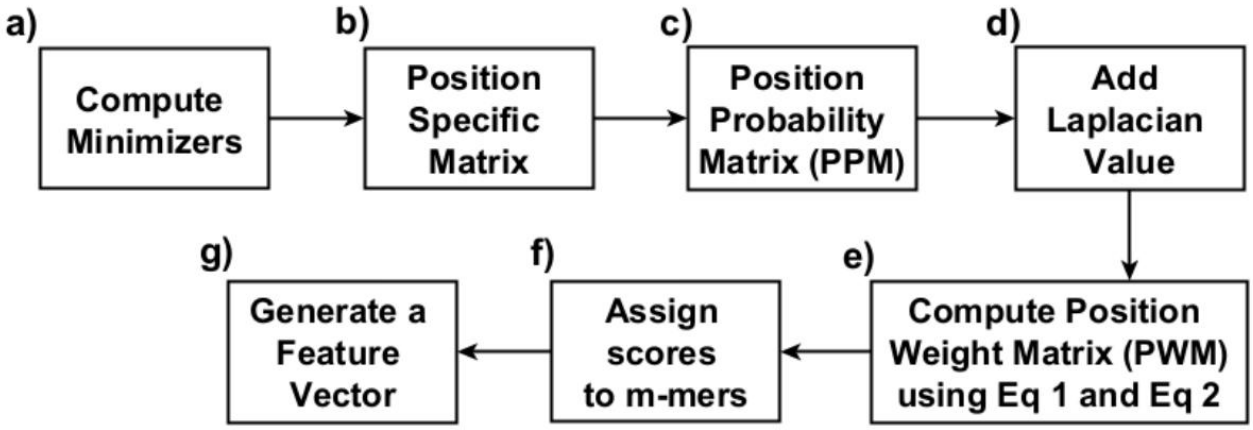

Figure 4.6: Virus2Vec Flow Diagram

4.4.2.2.2. PWM2Vec

PWM2Vec is a feature embedding technique that uses a **Position Weight Matrix (PWM)** to assign importance (weights) to each *k-Mer* in an aligned sequence.

The workflow includes:

- Generating a PWM from a set of aligned sequences.
- For each *k-Mer*, computing a score based on the PWM values at each position.
- Constructing a **feature vector** where the weight of each *k-Mer* is determined by these PWM-derived scores.

Key properties:

- The length of the feature vector is equal to the **number of *k-Mers*** in the aligned sequence.
- As unaligned sequences may have variable *k-Mer* counts, **PWM2Vec is applicable only to aligned sequences**.

4.4.2.2.3. PSWM2Vec

PSWM2Vec is a variant of PWM2Vec that introduces two key modifications:

1. Instead of using all *k-Mers*, it uses minimizers (m-Mers) to construct the initial matrix, which reduces dimensionality and noise.
2. Once the Position-Specific Weight Matrix (PSWM) is generated, it is flattened into a 1D feature vector.

Key properties:

- The resulting feature vector's length equals the total number of entries in the PSWM.
  This method inherits the alignment requirement from PWM2Vec and captures **position-specific statistical relevance** using a more compact representation.

4.4.2.2.4. MFV – Minimizer-Based Feature Vector

The Minimizer-Based Feature Vector (MFV) method encodes biological sequences by leveraging the concept of counting minimizers, which are lexicographically smallest substrings of a fixed length m (where $m < k$) found in each *k-mer*. Both the forward and reverse orientations of the *k-Mer* are considered when selecting the minimizer.

For each sequence:

- All *m-Mers* (minimizers) are extracted.
- A fixed-length frequency vector is constructed, where each bin corresponds to a unique minimizer from the alphabet Σ.
- The value in each bin represents the frequency of that minimizer in the sequence.

If $m = 3$ and Σ is the 20-character amino acid alphabet, the resulting feature vector has a length of $|\Sigma|^3 = 8000$.

This representation is efficient, alignment-free, and suitable for various machine learning algorithms.

4.4.2.2.5. Features

In addition to embedding-based representations, we explored a set of **handcrafted feature descriptors** that are widely used in protein sequence analysis. These features aim to extract **interpretable and biologically meaningful properties** from amino acid sequences and provide complementary inputs to classical machine learning models.

1. **APAAC (Amphiphilic Pseudo Amino Acid Composition):**
   Extends traditional amino acid composition by incorporating **sequence-order.**
   effects based on **hydrophobicity and hydrophilicity indices**. This allows the model to consider both the global composition and local distribution of amphiphilic properties.

2. **CTDD (Composition, Transition, Distribution Descriptors):**
   Captures the **positional distribution** of amino acids grouped by physicochemical properties (e.g., polarity, charge). CTDD provides insights into how amino acid classes are **spatially organized** along the sequence.

3. **CTriad (Conjoint Triad Descriptors):**
   Group amino acids into physiochemically similar categories and encode **sliding triads** of residues. This descriptor captures **local residue patterns**, which are important for understanding short-range biological motifs.

4. **DDE (Dipeptide Deviation from Expected Mean):**
   Quantifies the **deviation** between the **observed** and **expected frequencies** of all possible dipeptides. This helps in detecting **sequence irregularities** and biological signals related to codon bias or evolutionary pressures.

5. **Geary Autocorrelation:**
   Calculates autocorrelation values based on **physicochemical properties** (e.g., mass, polarity, flexibility) over different sequence **lags**. This captures how such properties fluctuate across the sequence, providing insights into **localized structural tendencies**.

4.4.2.3. Model Training

Objective: Train classical machine learning classifiers to distinguish between human-adaptive and non-human-adaptive viral sequences.

Steps:

1. Label each sequence in the dataset according to its known host adaptability (human or non-human).
2. Split the dataset into training and validation sets (e.g., 80/20).
3. Train several models (e.g., SVM, Logistic Regression, Random Forest) on the extracted features.
4. Optimize hyperparameters using grid search.
5. Select the best-performing model on the validation dataset.

4.4.2.3.1 Classifiers

We used different classical machine learning classifiers to detect whether a viral sequence is likely to adapt to infect humans. Below is a summary of the classifiers used and their configurations:

1. **SVM:** kernel is (rbf), C is (1)
2. **Random forest (the best):** n_istemators is (100)
3. **Logistic Regression:** max_iterations is (1000)

4.4.3. Design Constraints

The design of this module is affected by the following constraints:

- **Biological Variability:** Viral sequences can vary in length, mutation rate, and structure. Encoding methods must be robust and adaptable.
- **Data Imbalance:** Human-adaptive sequences are less common than non-human ones, requiring careful model selection and resampling techniques.
- **Computation Time:** Feature extraction (especially for long sequences or k-Mers) can be computationally expensive.
- **Explainability Requirements:** Since the system may assist virologists or public health officials, we must provide explainable outputs.
- **Model Generalizability:** The system must perform well across multiple viral families and pandemics, not just on a single training dataset.

4.5. Drug-Target Interaction Predictor

4.5.1. Functional Description

This module focuses on predicting the interaction strength between antiviral compounds (drugs) and viral proteins using a deep learning model built from graph neural networks (GNNs).

4.5.2. Modular Decomposition

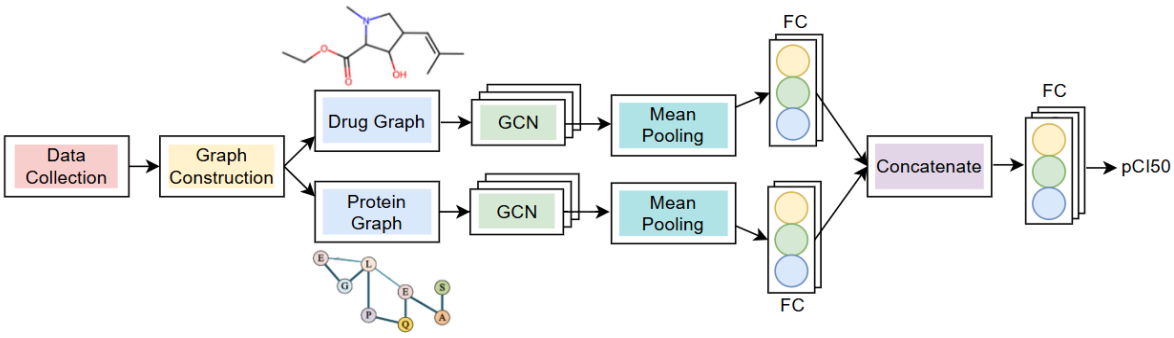

Figure 4.7: Block Diagram of Drug Target Interaction Predictor

4.5.2.1. Data Collection and Preprocessing

We started by collecting data from the ChEMBL database using its API, focusing on drug interactions with viral proteins that have IC50 values. For each entry, the corresponding viral protein sequence is retrieved from the UniProt database. The IC50 values are then converted to pIC50 values by applying negative log. This makes the values easier to work with for training the prediction model.

4.5.2.2. Graph Construction:

Both the drug molecules and viral proteins are converted into graph structures so that they can be used in a GNN.

4.5.2.2.1 Drug Graphs

Each drug is represented using its SMILES string, which describes the structure of the molecule. We used the RDKit library to convert this into a **molecular graph**:

- **Nodes**: Atoms in the molecule (e.g., carbon, oxygen).

- **Edges**: Chemical bonds between atoms.
- Each atom is described by a **feature vector** (e.g., atom type, degree, valence, hydrogen count, aromaticity).
- Bonds are treated as undirected edges and are encoded with features like bond type.

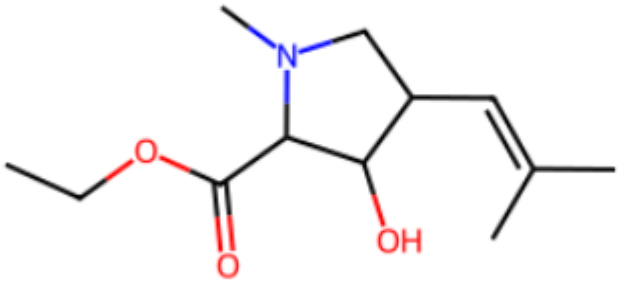

Figure 4.8: Chemical structure of antiviral drug for treating influenza A and B

4.5.2.2.2 Protein Graphs

Protein sequences are more complex and longer. To convert them into graphs:

- We use a pre-trained **ESM** model to predict **contact maps**, which show which amino acids are likely to interact with each other.
- **Nodes**: Amino acids, represented as one-hot vectors.
- **Edges**: Pairs of amino acids that are predicted to be in contact (based on a probability threshold).
- If the protein is too long, it is split into overlapping windows to make processing manageable, and all edges are combined at the end.

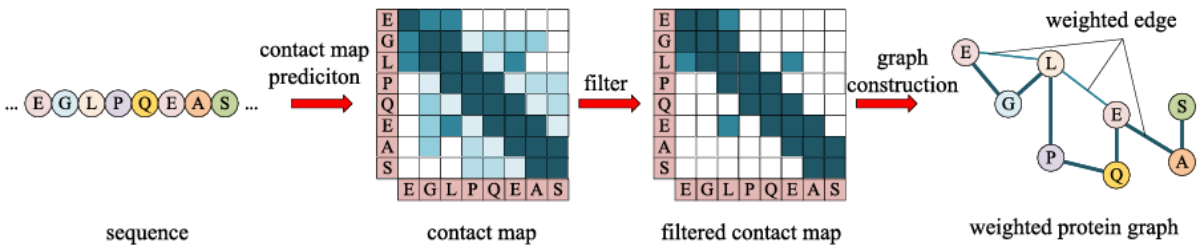

Figure 4.9: Construction of weighted protein graph.

By converting both drugs and proteins into graphs, the model can learn complex patterns that affect how they interact. These graphs are then passed to the GNN to learn meaningful features for predicting binding strength.

4.5.2.3. GNN Architecture:

The GNN architecture has two branches: one for drug graphs and one for protein graphs. Each branch uses GCN layers to learn features from the input graph. After the GCN layers, the features from all nodes are combined into a single vector using a global mean pooling layer. These two vectors—one from the drug and one from the protein—are then concatenated and passed to fully connected layers to predict the final pIC50 value, which represents the strength of the interaction between the drug and the viral protein.

4.5.2.4. Training and Evaluation:

The dataset is divided into training, validation, and test sets to ensure fair evaluation and prevent overfitting. The model is trained on the training set, while the validation set is used to tune key hyperparameters like batch size and hidden layer dimensions. After training, the model's performance is measured on the test set using evaluation metrics such as MSE, RMSE, and Pearson correlation, which help know how accurately the model predicts pIC50 values.

4.5.3. Design Constraints

1. **Protein Sequence Length Limitation (ESM Constraint)**
   The ESM (Evolutionary Scale Modeling) model used in this module only supports protein sequences up to 1024 amino acids in length so sequences longer than 1024 are split with overlaps and passed through the model in segments.
2. **Use of IC50 Instead of Kd Values**
   Initially, we intended to predict Kd (dissociation constant) values for drug-target affinity due to **insufficient and incomplete data** available for Kd; we used data with **IC50** values, which were more accessible.

4.5.4. Another approach

In addition to using graph-based representations for proteins, we tried another approach where the protein was encoded using manually engineered features instead of graphs. This included features such as APAAC, CTDD, CTriad, DDE, and clustered k-Mer frequency features based on BLOSUM62 similarity. These features

were combined into a high-dimensional vector of size 803, which was then reduced by using PCA. In this approach, the drug was still represented as a molecular graph and passed to GCN layers to extract drug features. The reduced PCA protein vector was passed to fully connected layers. Finally, the outputs of both branches—GCN for drugs and fully connected layers for protein features—were concatenated and passed to another set of fully connected layers to predict the pIC50 value. While this method performed reasonably well, the full graph-based approach gave better accuracy and was used in the final model.

4.6. Drug Generation using GAN

4.6.1. Functional Description

The goal of this module is to generate drug-like molecules represented as SMILES using GAN. The GAN is first trained on a large dataset of valid drug SMILES to learn how to generate chemically valid molecules. After this initial training, the model enters a fine-tuning phase to generate drugs that are more likely to inhibit a specific virus.

4.6.2. Modular Decomposition

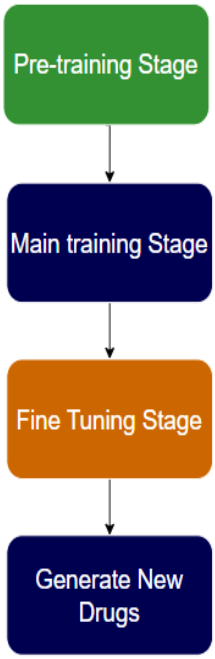

Figure 4.10: GAN Pipeline

4.6.2.1 Data Preparation

The dataset is preprocessed to ensure SMILES strings are valid and within the specified length range (32–128 characters), reflecting the typical size of drug-like molecules. The dataset's chemical diversity includes small molecules with functional groups relevant to antiviral activity. RDKit is used for validation, checking for correct valence and ring closure. The vocabulary includes 59 tokens, covering common atoms, bonds, and special characters, with a regular expression-based tokenizer ensuring accurate parsing. The dataset is split into training (80%), validation (10%),

and test (10%) sets, with statistics showing an average SMILES length of ~50 characters and a broad distribution of molecular weights.

4.6.2.2 Generator

The generator is an LSTM-based model with 4 layers, 512-dimensional embeddings, and 256 hidden units per layer, implemented in PyTorch. It takes a random vector and generates sequences up to 128 tokens, outputting log-probability over the vocabulary. During sampling, a temperature parameter (default 1.0) controls diversity. The LSTM architecture was chosen for its ability to model sequential dependencies in SMILES strings, outperforming simpler RNNs or GRUs in preliminary tests.

4.6.2.3 Discriminator

The discriminator is a CNN-based model with an embedding layer (256 dimensions), two convolutional layers (3 and 6 filters, kernel sizes 5 and 7, respectively), and a fully connected layer. It processes padded sequences of 128 tokens, outputting a probability of authenticity. Dropout (0.3) is applied to prevent overfitting. CNNs were selected for their ability to capture local patterns (e.g., functional groups) in SMILES strings, with padding ensuring consistent input sizes. The discriminator is trained to distinguish real SMILES from generated ones, using a sigmoid output for binary classification.

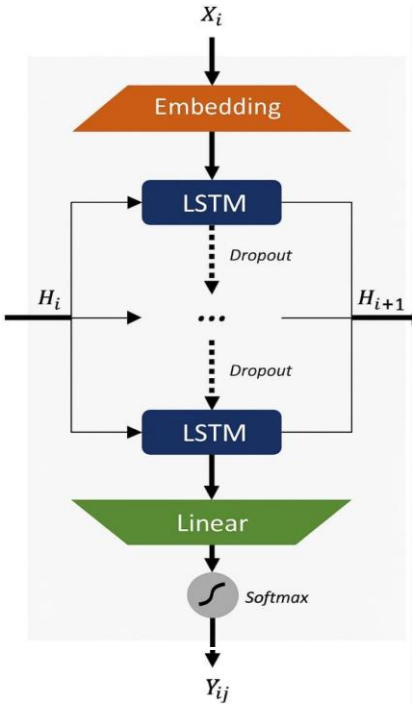

Figure 4.12: Generator Architecture

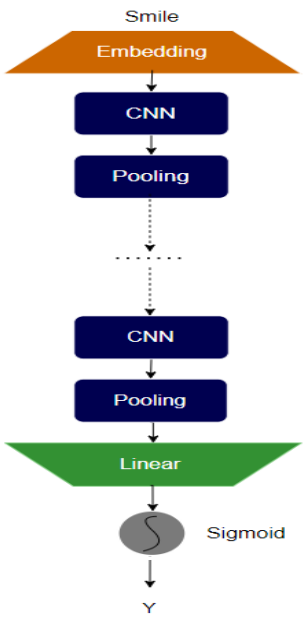

Figure 4.11 : Discriminator Architecture

4.6.3. GAN Training Pipeline

1. Before adversarial training, the model pre-trains to set the generator and discriminator with stable and interpretable parameters:
    - Maximum Likelihood Estimation (MLE) pre-trains the generator. During this stage, it learns to abstract the underlying distribution of training data through minimizing cross-entropy loss between generated token distributions and ground truth sequences. This supervised learning enables the generator to produce syntactically correct and coherent sequences as a good starting point before adversarial learning is introduced.

$$L\_G = -\sum_{t=1}^{T} log\, P(y_t \mid y_{1:t-1}, \theta_G)$$

        - y_t: Ground truth token at position t.
        - $P(y_t \mid y_{1:t-1}, \theta_G)$: Probability of token y_t given previous tokens and generator parameters θ_G.
        - T: Sequence length.

    - The discriminator is also pre-trained simultaneously with a binary classifier. It is trained to recognize real sequences drawn from the dataset and generated sequences produced by the generator. By minimizing binary classification loss, the discriminator becomes able to predict high probabilities for real sequences and low probabilities for fake sequences, effectively learning difference-inducing features between real and generated data.

$$L\_D = -[\, y\, log(D(x)) + (1-y)\, log(1 - D(G(z)))\,]$$

        - D(x): Discriminator's probability that input x is real.
        - G(z): Sequence generated by the generator from noise z.
        - y: Binary label (1 for real, 0 for fake).
2. Once the pre-training stage is completed, the model then moves to the adversarial training process, in which the generator and discriminator are trained together in a cooperative but competitive manner to better quality and relevance of output sequences.

- The generator begins by sampling new sequences of molecules (for example, SMILES strings) attempting to create realistic, chemically valid structures that might potentially be drug candidates.
- Each sequence is then evaluated using Monte Carlo sampling, which estimates the expected reward of the sequence by looking ahead to a few rollouts of the generation process. The reward signal is a mixture of two main parts:
    1. **Chemical validity**, ensuring that the sequence corresponds to a structurally plausible molecule.
    2. **Discriminator feedback**, reflecting how "real" the sequence appears based on the discriminator's learned representations

$$R = 0.5 * validity + 0.5 * D(G(z))$$

- The estimated reward is used to update the generator's parameters through **policy gradient methods** (specifically, the REINFORCE algorithm). so that the generator improves its policy for generating sequences by rewarding positive actions (i.e., token choice) that are rewarded more, thus guiding it to create more drug-like and real compounds.

$$\nabla_{\theta_G} J(\theta_G) = E\left[\sum_{t=1}^{T} R \cdot \nabla_{\theta_G} \log P(y_t \mid y_{1:t-1}, \theta_G)\right]$$

    - where $J(\theta_G)$ is the expected reward is, and the gradient is approximated via Monte Carlo rollouts.
- Concurrently, the **discriminator** is further trained to enhance its ability to differentiate between authentic sequences from the training dataset and synthetic sequences produced by the generator.

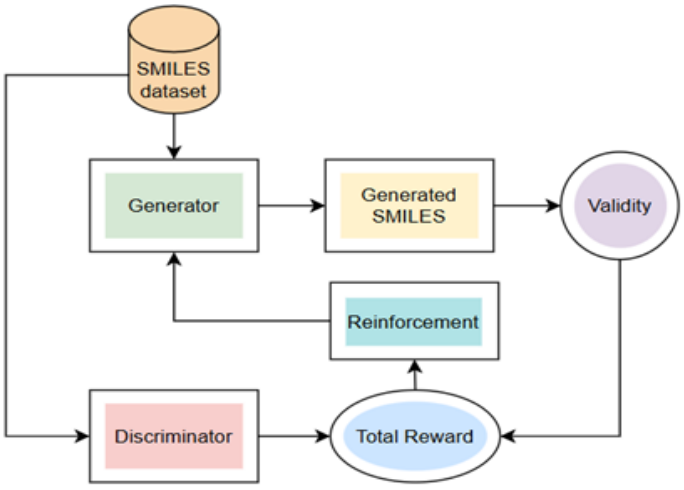

Figure 4.13: GAN Training Loop

3. After adversarial training, there is a fine-tuning phase where the generator is supervised to produce drug candidates that are specifically tailored to counteract a target viral protein (e.g., from an emerging virus)
    - A Drug-Target Interaction (DTI) model is used to identify a subset of known drugs with high predicted binding affinity against the target viral protein. The best-ranked drugs are selected based on their interaction scores and are termed reference compounds for the fine-tuning.
    - The generator is then fine-tuned with Monte Carlo sampling and reward function that encourages it to generate sequences with the properties of such high-scoring reference drugs. The reward is based on:

        1. **Chemical validity**, to maintain realistic molecular structure.
        2. **Tanimoto similarity**, a widely used metric for structural similarity between molecular fingerprints, to ensure the generated drugs share chemical features with known effective compounds.

        $$R_{fine\_tune} = 0.5 \cdot validity + 0.5 \cdot Tanimoto\_similarity$$

    - This targeted fine-tuning enables the model to generate novel drug-like molecules that not only adhere to chemical constraints but also align with therapeutic relevance to the specific virus of interest, making it a powerful approach for rapid antiviral drug design in response to emerging viral threats.

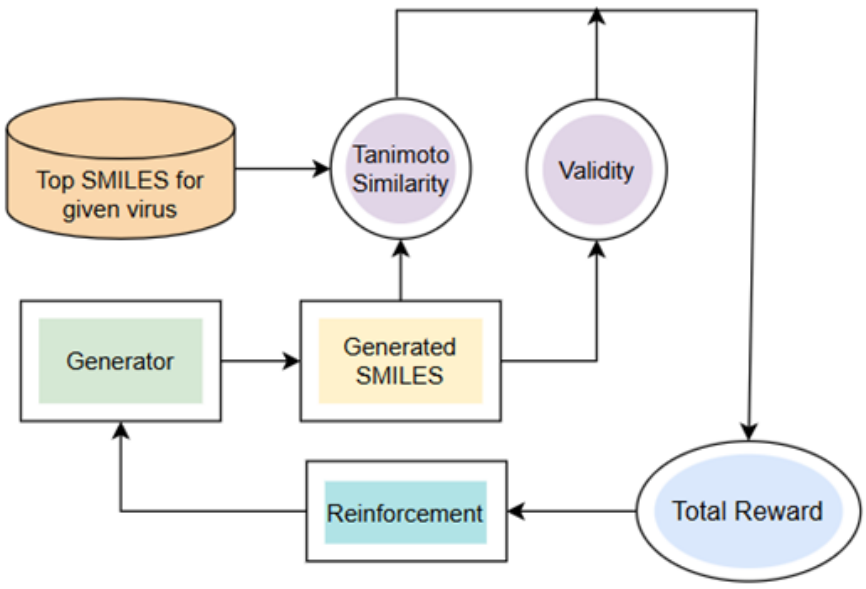

Figure 4.14: Fine Tuning Loop

4. Generating New Drugs
    - After the model has been fine-tuned toward a specific viral target, it is ready to enter the final stage: **drug generation**. In this stage, the generator can produce novel drug-like molecules that are not only valid but also highly relevant to the target virus.

4.6.4. Design Constraints

1. **SMILES Length Limit**: All generated sequences must be padded or truncated to a fixed maximum length.
2. **Discrete Output Problem**: GANs typically struggle with discrete outputs (like characters), so reinforcement learning with policy gradient is used.
3. **Resource Constraints**: Training GANs, especially for sequence generation, is known to be computationally intensive. Due to limited resources on the Kaggle platform, including restricted GPU time and session duration, we implemented checkpointing to save the model state periodically to be able to continue training after a session ends.
4. **Slowness of sequence generation**: Generating sequences using the generator was time-consuming when done character by character. To solve this, we used batching during sequence generation, which reduced computation time and improved overall training efficiency.
5. **Biological Relevance**: Not all valid molecules are biologically active. The integration with the DTI model helps guide the generator toward meaningful drug candidates.

4.7. Sequence Alignment Module

4.7.1 Functional Description

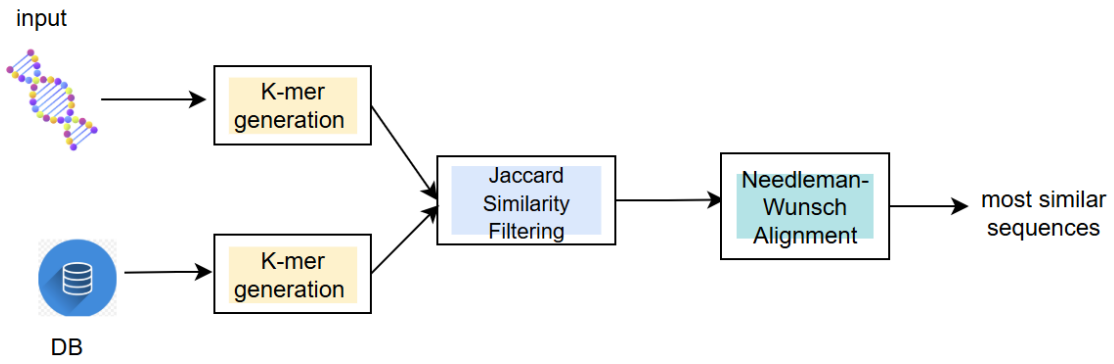

Figure 4.15: Alignment pipeline

The sequence alignment module is to compare an input virus sequence against a set of known sequences for closest matches to analyze viral similarity and potential evolutionary relationships. The module accepts a virus sequence as either a raw string or FASTA file and converts it to compute similarity measures and alignments. It uses k-mer similarity via Jaccard similarity for initial filtering and a dynamic programming Needleman-Wunsch algorithm for exact sequence matching. The module offers the top five nearest neighbors, such as aligned sequences, similarity scores, and visual match pointers, to enable virological investigations and antiviral drug discovery.

4.7.2 Modular Decomposition

The sequence alignment module is composed of several key components, each handling a specific aspect of the alignment process:

4.7.2.1 Input Processing

The module accepts virus sequences in two forms: FASTA files or raw sequence strings provided through form data or query parameters. For FASTA files, the module ensures a single sequence is provided by looking for over one header (lines that start with >). If there are multiple sequences in a FASTA file, an error is returned (HTTP 400). The input sequence is received by removing the FASTA header (if present) or merely by taking the provided raw sequence. The module uses the fs module in Node.js to read file content as well as to verify input format for valid input so that strong user-input data handling can be performed.

4.7.2.2 K-Mer Generation

The getKmers function generates k-Mers (size-k substrings with default size 5) from the query sequence and database sequences. The k-Mers are stored in a Set to render them distinct, reducing the computational cost for subsequent similarity calculation. Choosing k=5 strikes the balance between sensitivity and specificity and detects local sequence motifs without producing too much noise.

4.7.2.3 Jaccard Similarity Filtering

The Jaccard Similarity function computes the Jaccard similarity of two sets of k-Mers as the intersection size over the union size. It is a rapid approximate measure of sequence similarity. The module gets known sequences from a database (using Prisma ORM) and computes Jaccard similarity with each of these against the k-Mers of the input sequence. Sequences with more than a dynamically chosen threshold similarity value (initially set to 0.2 and lowered if less than five matches were found) are retained for further processing. The prefiltered sequences are ranked by Jaccard similarity in descending order, and the top 30 are aligned with the goal of achieving a balance between accuracy and efficiency.

4.7.2.4 Needleman-Wunsch Alignment

The align2seq process runs the Needleman-Wunsch algorithm for global sequence alignment using dynamic programming to compute the optimal alignment of the input sequence to each filtered known sequence. The algorithm uses:

- **Scoring Parameters**: Match score (+1), mismatch score (-1), and gap penalty (-1).
- **Dynamic Programming Matrix**: A scoring matrix tracks the optimal alignment score, and a traceback matrix records the path (diagonal for match/mismatch, up for deletion, left for insertion).
- **Traceback**: The algorithm reconstructs the aligned sequences (alignedSeq1, alignedSeq2) and a matchLine indicating matches (|), mismatches (:), or gaps (.).
- **Output**: For each alignment, the function returns the aligned sequences, a match line, and the alignment score.

The top five alignments, sorted by alignment score in descending order, are returned to the client, providing a clear visualization of sequence similarity and differences.

4.7.3 Design Constraints

- **Single Sequence Input**: The module enforces that FASTA files contain only one sequence to avoid ambiguity in alignment. Multiple sequences trigger an error, requiring users to preprocess their data.
- **K-Mer Length**: The default k-Mer length (k=5) is a tunable parameter. Smaller values increase sensitivity but may introduce noise, while larger values improve specificity but may miss subtle similarities.
- **Dynamic Thresholding**: The Jaccard similarity threshold (initially 0.2) is dynamically adjusted to ensure at least five matches, which may reduce specificity for highly divergent sequences.
- **Computational Efficiency**: The module limits alignment to the top 30 sequences (based on Jaccard similarity) to reduce the computational cost of the Needleman-Wunsch algorithm, which has a time complexity of O(mn) for sequences of lengths m and n.
- **Database Dependency**: The module relies on a Prisma-managed database to store known sequences. The database must be populated with relevant sequences for meaningful alignments.
- **Scoring Parameters**: The match (+1), mismatch (-1), and gap penalty (-1) scores are fixed. Adjusting these parameters could improve alignment accuracy for specific use cases but requires manual tuning.
- **Sequence Length**: Long sequences increase the computational cost of alignment. The module does not impose a strict length limit, but performance may degrade for very long sequences.

4.8. Backend

4.8.1. Functional Description

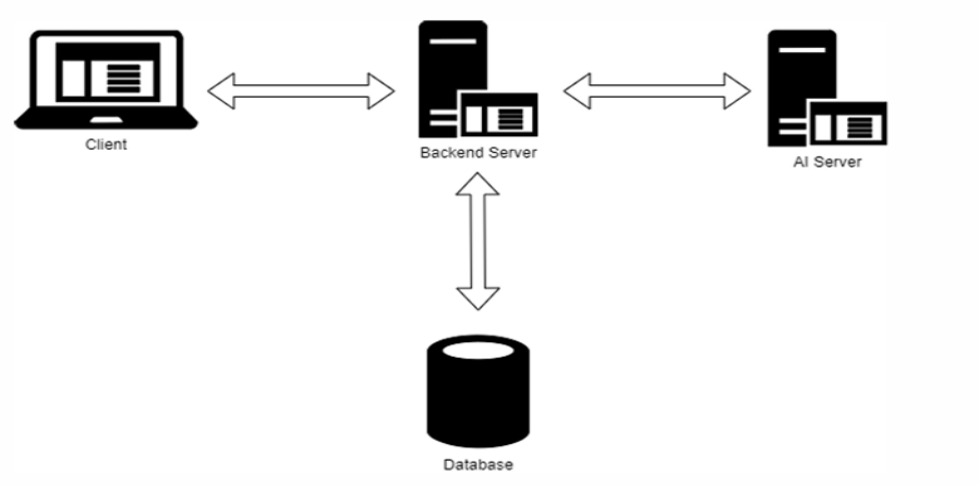

Figure 4.16 : System Flow

4.8.2. Modular Decomposition

The backend system is designed to manage the workflow of virus sequence analysis, host prediction, and antiviral drug candidate generation and evaluation. It integrates a REST API (built with Express.js in Node.js), an AI server (built with FastAPI in Python), and supporting middleware for authentication and file handling. This system enables seamless processing of virus sequences, prediction of host specificity, evaluation of antiviral drug effectiveness, and generation of potential antiviral compounds.

4.8.2.1. REST API

- Endpoint Management: The REST API, implemented in Express.js, provides endpoints for user authentication (signup, login, logout) and virus-related operations (host prediction, antiviral drug prediction, top antiviral candidates, and sequence alignment). These endpoints handle file uploads (FASTA files) and form data inputs for virus sequences and drug SMILES strings.
- Request Handling: The API validates incoming requests using middleware. It directs requests to the appropriate controllers, which communicate with the AI server for processing.
- Response Generation: After processing by the AI server or controllers, the REST API returns JSON responses containing results such as host predictions, drug effectiveness scores (pIC50), generated antiviral compounds, or sequence alignment data, along with appropriate status codes (e.g., 200 for success, 401 for unauthorized, 404 for not found).

4.8.2.2. AI Server

- Host Prediction: The AI server processes virus sequences (from FASTA files or form inputs) to predict whether a virus is likely to infect humans or non-humans. It uses two approaches: a machine learning model (predict-host with ML) that leverages Virus2Vec feature vectors and a more complex model (predict-host with DL) that includes attention mechanisms and generates 2D heatmap visualizations of sequence features.
- Antiviral Drug Prediction: The AI server evaluates the effectiveness of a given antiviral drug (specified by its SMILES string) against a virus sequence, returning a pIC50 score to indicate interaction (predict-antivirus).
- Top Antiviral Candidates: The server identifies top antiviral drug candidates for a given virus sequence from stored commercial antivirus, returning a list of SMILES strings with their predicted pIC50 scores (top-antivirus).
- Antiviral Drug Generation: The server generates novel antiviral drug candidates tailored to a specific virus sequence, using a generative model (generate-antivirus).

Chapter 5: System Testing and Verification

To ensure the correctness, reliability, and robustness of our system, extensive testing and verification steps were implemented throughout the development lifecycle. The testing process included unit testing of individual components, integration testing to validate the interactions between modules, and performance evaluation under various scenarios using real biological datasets. Emphasis was placed on verifying both the accuracy of predictions and the stability of the software system, ensuring that each module performs its function as intended. Moreover, the evaluation included functional correctness checks, and performance benchmarking against baseline methods, ensuring the system delivers meaningful and reproducible results in the domain of computational virology.

5.1. Testing Setup

The ViroAlert system was tested in a controlled environment using modern hardware and key frameworks like PyTorch and RDKit. Viral and drug datasets were collected from trusted sources such as GISAID, NCBI and ChEMBL, then preprocessed and split into training, validation, and testing sets.

Each module was evaluated individually using appropriate metrics (e.g., accuracy, F1-score, ROC-AUC for classification; RMSE and Pearson correlation for regression; validity and uniqueness). Manual tests were also performed to ensure the pipeline worked smoothly from input sequence to output drug suggestion. Finally, the system's user interface was tested to ensure proper result visualization.

5.2. Testing Plan and Strategy

The testing strategy for this project was designed to systematically verify the correctness, robustness, and effectiveness of each stage in the virus-host interaction prediction pipeline. The methodology consisted of multi-level testing, including unit testing, integration testing, cross-validation. Each step was planned to ensure that the biological data was processed correctly and that the predictive models performed as intended.

5.2.1. Module Testing

5.2.1.1 Viral Host Adaptability Predictor DL Approach

We trained ViroAlert on a dataset of 80,000 viruses from NCBI GenBank and GISAID, collected prior to 2023, with an 80:20 split for training, and validation. The test datasets, spanning 2023–2025, included 3500 viruses from GISAID and NCBI covering diverse viral families. Performance metrics are presented in the following table.

NCBI				**GASAID**			
Class	**Precision**	**Recall**	**F1-Score**	**Class**	**Precision**	**Recall**	**F1-Score**
Human	0.98	1.00	0.99	**Human**	0.92	0.99	0.96
Animal	1.00	0.95	0.97	**Animal**	0.99	0.90	0.95
Accuracy	**0.99**			**Accuracy**	**0.96**		
Macro Avg.	0.99	0.97	0.98	**Macro Avg.**	0.96	0.95	0.95
Weighted Avg.	0.99	0.99	0.99	**Weighted Avg.**	0.96	0.95	0.95

Table 5.1 Viral Host Adaptability Predictor DL Approach Testing Accuracy

ViroAlert achieved exceptional performance, with F1-scores of 0.95 and 0.98 on the GISAID and NCBI datasets, respectively. The high recall for human-adapted viruses (0.99 on GISAID, 1.00 on NCBI) indicates ViroAlert's sensitivity to potential zoonotic threats, critical for early detection. The confusion matrices reveal minimal false positives, highlighting the model's precision in distinguishing human-adapted viruses.

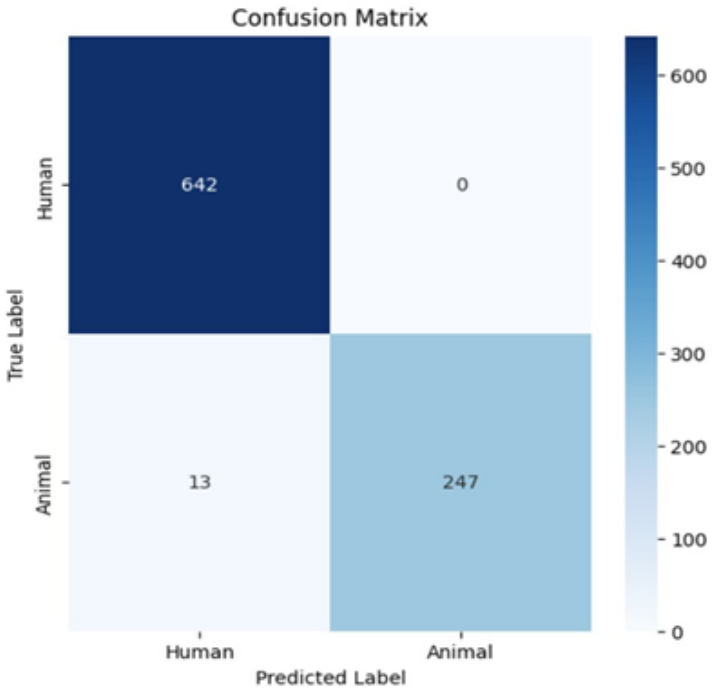

Figure 5.1 : Confusion Matrix on Test NCBI

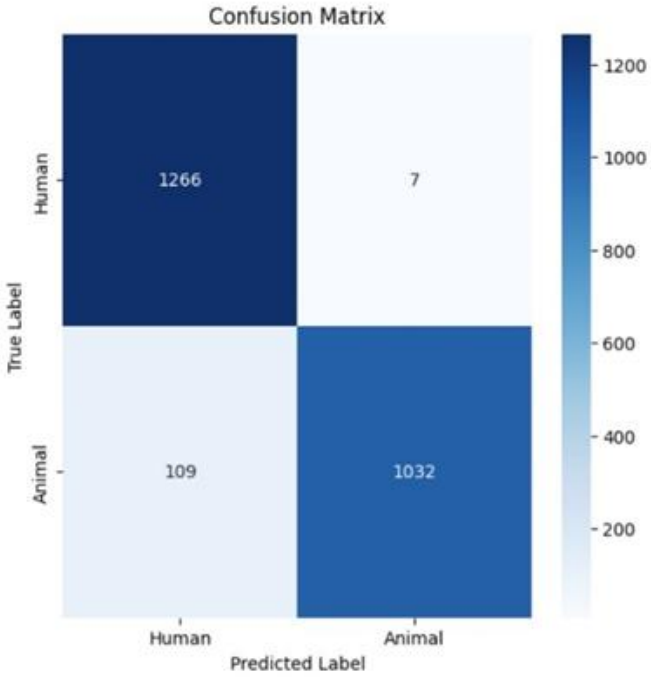

Figure 5.2 : Confusion Matrix on Test GASAID

Covid-19 Test

To assess early zoonotic detection, we evaluated ViroAlert **pre-2020 SARS-CoV-2** sequences from non-human hosts (e.g., bats). The model classified 85% of these sequences as human-adapted, despite their non-human origin, suggesting that ViroAlert can detect genomic signatures of human adaptability before a zoonotic spillover occurs.

This capability is pivotal for prioritizing high-risk strains for laboratory validation and enabling proactive public health interventions. The attention scores generated by the hierarchical Gated Attention mechanism further enhance interpretability, identifying key genomic regions driving classification decisions, which can guide targeted experimental studies.

5.2.1.2 Viral Host Adaptability Predictor ML Approach

We trained on a dataset of 80,000 viral protein sequences collected prior to 2023 from NCBI GenBank and GISAID. The data was split into 80% for training and 20% for validation.

To ensure a comprehensive evaluation, we implemented and compared multiple biologically meaningful feature extraction methods, including **Virus2Vec**, **PSWM2Vec**, **MFV** and other **Statistical ML Features.** Each method was used to generate feature vectors that were then fed into classical machine learning classifiers such as SVM, Random Forest, and Logistic Regression.

For generalization testing, we evaluated the models on independent datasets from 2023 to 2025, comprising 3,500 viral sequences from both GISAID and NCBI.

The results of these evaluations are summarized in the table below.

		Human-Adaptive Sequences				**Non-Human-Adaptive Sequences**			
Method	**Classifier**	Acc.	Prec.	Recall	F1	Acc.	Prec.	Recall	F1
Virus2Vec	SVM	0.87	0.90	0.87	0.89	0.92	**0.90**	0.92	0.90
	RF	0.87	**0.94**	**0.87**	**0.91**	**0.94**	0.87	**0.94**	**0.90**
	LR	**0.97**	0.94	0.79	0.86	0.94	0.84	0.94	0.87
PSWM2Vec	SVM	0.55	0.88	0.56	0.68	0.91	0.65	0.91	0.71
	RF	**0.84**	**0.94**	**0.84**	**0.89**	**0.95**	**0.84**	**0.95**	**0.89**
	LR	0.43	0.85	0.43	0.57	0.92	0.59	0.92	0.72
MFV	SVM	0.81	0.82	0.81	0.81	0.90	0.90	0.90	0.90
	RF	**0.85**	**0.85**	**0.86**	**0.85**	0.91	**0.92**	0.92	**0.92**
	LR	0.71	0.84	0.71	0.77	**0.92**	0.86	**0.93**	0.89
ML Features	SVM	**0.86**	0.93	**0.86**	0.89	0.92	**0.86**	0.92	0.89
	RF	0.85	**0.95**	0.85	**0.90**	**0.95**	0.85	**0.95**	**0.90**
	LR	0.74	0.92	0.74	0.82	0.93	0.77	0.93	0.84

Table 5.2 Performance comparison of different approaches on Unaligned Virus Sequences

5.2.1.3 Drug-Target Interaction Predictor

To evaluate the performance of our DTI prediction module, we conducted hyperparameter tuning and tested several configurations of hidden dimensions and batch sizes. The goal was to minimize the prediction error while maximizing correlation with the ground truth.

5.2.1.3.1. Hyperparameter Tuning Results

We experimented with different values for hidden_dim and batch_size: 64, 128, and 256. Evaluation metrics used included:

- MSE (Mean Squared Error)
- RMSE (Root Mean Squared Error)
- Pearson Correlation Coefficient

Then the optimal configuration was found to be:

- hidden_dim = 64
- batch_size = 64

With the following performance metrics

- MSE: 0.8800
- RMSE: 0.9381
- Pearson Correlation: 0.8375

5.2.1.3.2. Performance Overview

- Increasing the hidden dimension beyond 64 showed no significant improvement in MSE, RMSE, or Pearson metrics.
- Batch size had a more noticeable impact: larger batch sizes (128, 256) resulted in higher MSE and lower Pearson correlation, indicating reduced performance.
- Pearson correlation peaked at the lowest batch size (64), suggesting that smaller batches generalize better for this task.

5.2.1.3.3. Model Architecture Comparison

We initially trained the model using handcrafted feature vectors for viruses. After running the full hyperparameter search with this approach, we observed suboptimal results:

- **MSE**: 1.0421 -> Higher MSE
- **RMSE**: 1.0208 ->Higher RMSE
- **Pearson Correlation**: 0.7893 ->Lower Pearson

Following this, we switched to graph-based representations of viral proteins using GNNs and repeated the same training and tuning process. The performance significantly improved across all metrics, confirming that modeling viruses as graphs provide richer structural information and better supports the DTI prediction task.

As a result, our final model uses GNN-based viral representations, achieving the best generalization and accuracy.

5.2.1.3.4. Visualization Summary

- Bar plots show mean values across runs for each metric grouped by hyperparameter.

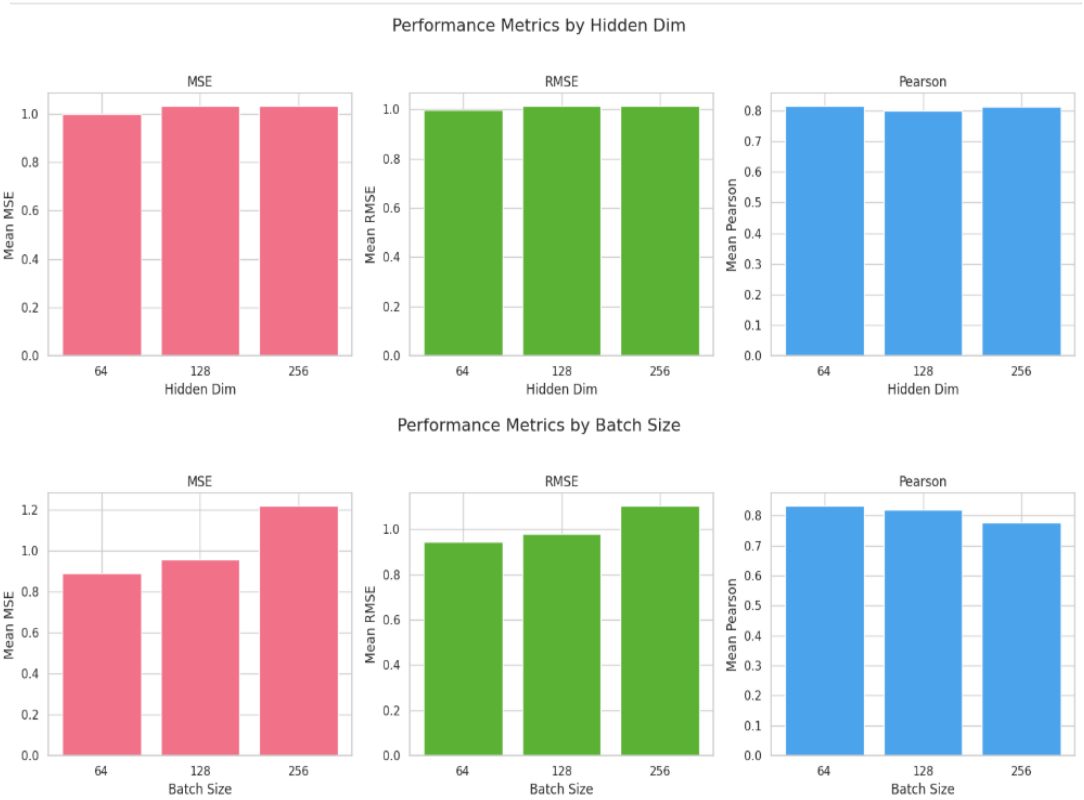

Figure 5.3: Bar Plot for Performance metrics

- Line plots with error bars show metric stability and variance, indicating that performance was more stable at lower hidden_dim and batch_size.

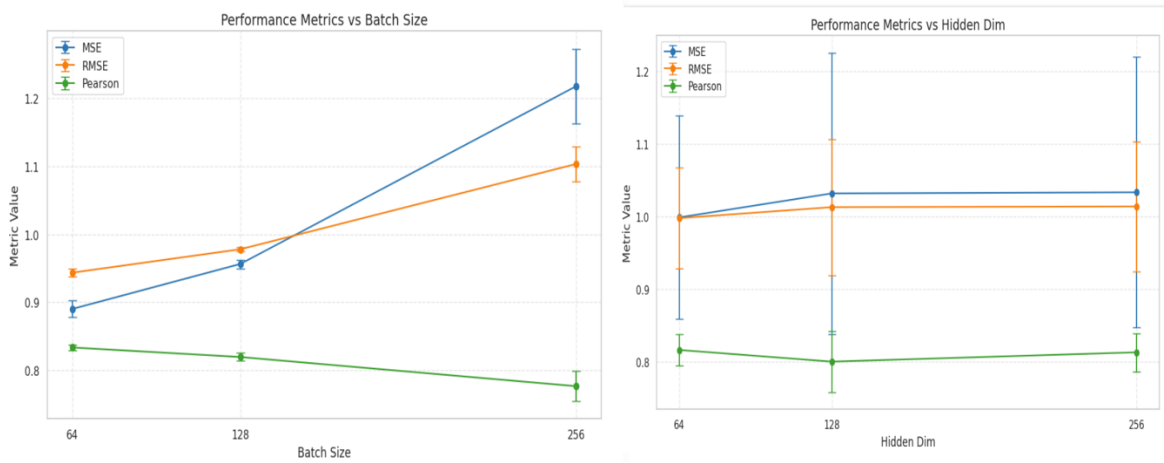

Figure 5.4: Line Plots of Performance Metrics

5.2.1.4 Drug Generation using GAN

During the training phase, we assessed the quality of the drug molecules generated based on three significant metrics to study the Generative Adversarial Network (GAN) model's performance. These metrics provide insight into the chemical validity, diversity, and novelty of the generated SMILES strings that play a critical role in ensuring the model produces viable drug candidates. We conceptualize the metrics as follows:

- **Validity:** Measures how many of the generated SMILES strings are chemically valid molecules.
- **Uniqueness:** Checks how many of the valid molecules are different from each other.
- **Novelty:** Tells us how many of the generated molecules are new (not seen in the training data).

We generated **10,000 drug** and got the following results:

- Valid SMILES: 9301 (93.01%)
- Unique SMILES: 9270 (99.66%)
- Novelty: 98.78%

In the fine-tuning phase, we focused specifically on SARS-CoV-2 and influenza A virus sequences. This involved integrating protein graph data and employing a reward-based mechanism to optimize for predicted activity, measured as pIC50 (a common metric for assessing the potency of a compound, where higher values indicate stronger binding affinity).

After training, the model generated drug candidates, and we found that from 100 generated molecules.

Results on SARS-CoV-2

- 22 molecules with a predicted pIC50 > 7 (strong activity).
- 53 molecules with pIC50 > 6 (moderate to strong activity).

Section	Metric/Compound	PIC50
Evaluation (SARS - Known Inhibitors)	CC1-CC=C(-C1)NC=O)C2CC=C(C2)C3-CC=CC=C3 (Nirmatrelvir (Paxlovid))	7.1266
	CCOC(=O)C1=NC=NC2=CC=CC=C2 (Favipiravir)	8.0472
Generation (SARS - New Candidates top 5)	Cc1ccc(-c2ccccc2)cc1CNC(=O)C(=O)NC(=O)C(C)C	8.5243
	COc1cc(CC(=O)NC(=O)NC(=O)C(=O)NC(=O)C(=O)NC(=O)C2C	7.9369
	CCOC(=O)C(=O)NC(=O)C(=O)C(=O)NC(=O)C(=O)	7.9039
	CCCN1CCCCC1CNC(=O)C(=O)NC(=O)C(=O)C(=O)C(=O)C(=O)N	7.8335

	CN(C)C(=O)C(=O)NC(=O)c1ccc(C)cc1	7.8200

Table 5.3 SARS-COVID-2 Results

Results on influenza A:

- 15 molecules with a predicted pIC50 > 7 (strong activity).
- 41 molecules with pIC50 > 6 (moderate to strong activity)

Section	Metric/Compound	PIC50
Evaluation (influenza A - Known Inhibitors)	CC1(C)C(=O)NC(=O)N(C1C)C2CC2	7.0331
	C1CN2CCC1C(C2)NC(=O)C(CO)NC(=O)C3=CC=CC=C3	6.5514
Generation (influenza A - New Candidates top 5)	Clc1cccc(-c2ccccc2C=C2CC(=O)NC2=O)c1	8.896
	CN(C)C(=O)c1cccc(C(=O)N2C(C(=O)N3CCCCC3)CC3CC32)c1	7.712
	CCc1cc(C(=O)c2ccccc2)c(=O)[nH]c1	7.67
	CN(C)C(=O)CCC(=O)N1CCc2cc(C(=O)Nc3ccccc3)ccc21	7.57
	Cc1c(C)c2ccccc2[n+]3c1C=CC(C)=C3	7.54

Table 5.4 Influenza A Results

5.2.2. Integration Testing

The ViroAlert system integration test was purposed to guarantee that there is no problem in the interaction between its four core modules: the Deep Learning (DL) Virus Host Adaptability Predictor, the Machine Learning (ML) Virus Host Adaptability Predictor, the Drug-Target Interaction (DTI) Predictor, and the Drug Generation module. Steps adopted in testing were:

1. **Module Integration Setup:**
    - The backend was built using FastAPI to serve the AI models, with Node.js handling server-side logic, Prisma as the ORM, and MySQL for data storage. The frontend was developed using Next.js to provide a user-friendly interface.
    - Input data pipelines were established to handle viral protein sequences (in .fasta) and drug SMILES strings, ensuring compatibility across modules.
2. **Drug-Target Interaction (DTI) Testing:**
    - Input Process: Users could upload a viral protein sequence and a drug SMILES string or paste them directly into the web application. The system processed these inputs through the GNN-based DTI Predictor to estimate interaction strength (pIC50).
    - Execution: The DTI module was tested by inputting a variety of viral sequences and known drug SMILES (e.g., Nirmatrelvir, Favipiravir) to predict pIC50 scores. The system displayed the predicted pIC50 value, where higher values indicated stronger inhibitory effects.
    - Validation: The predictions were compared against known pIC50 values for SARS inhibitors (e.g., Nirmatrelvir: 7.1266, Favipiravir: 8.0472) to verify accuracy.
3. **Sequence Match Analyzer Testing:**
    - Input Process: Users provided a viral sequence, which was analyzed for similarity against a database of known viral sequences using the Sequence Match Analyzer.
    - Execution: The analyzer computed alignment scores and match statistics, identifying top sequence matches (e.g., Influenza A virus strains). Visual outputs included alignment visualizations and similarity metrics.
    - Validation: Results were cross-checked with reference databases (e.g., GISAID, NCBI) to confirm the accuracy of sequence alignments and similarity scores.
4. **Human Adaptation Prediction Testing:**
    - Input Process: Viral protein sequences were input into both DL and ML Virus Host Adaptability Predictors to estimate the probability of human adaptation.

- Execution: The system displayed predicted probability and highlighted influential amino acids (tokens) using the hierarchical Gated Attention mechanism (for DL) and ML features and display the attention weights for each sequence and subsequence.
- Validation: Predictions were validated against pre-2020 SARS-CoV-2 sequences from non-human hosts, where the DL model correctly classified 85% as human-adapted, demonstrating its ability to detect zoonotic potential.

5. **Drug Generation Testing:**
   - Input Process: Users specified a target virus (e.g., SARS, Influenza A), and the GAN-based Drug Generation module generated novel SMILES strings.
   - Execution: The system produced 10,000 SMILES strings, evaluated for validity (93.01%), uniqueness (99.66%), and novelty (98.78%). For SARS-specific testing, 100 molecules were generated, with 22 achieving pIC50 > 7 and 53 with pIC50 > 6.
   - Validation: Generated SMILES were validated using RDKit to ensure chemical validity and compared against known inhibitors to assess novelty and predicted efficacy.
6. **End-to-End Pipeline Testing:**
   - A complete workflow was tested by inputting a viral sequence, predicting its human adaptability, analyzing sequence matches, predicting DTI for known drugs, and generating new drug candidates. The system ensured data flowed correctly between modules, with results stored in MySQL and visualized in the React frontend.
   - Postman was used to test API endpoints, ensuring robust communication between the frontend, backend, and AI models.

5.3. Testing Schedule

We started testing the modules in February and tried to enhance them until the end of June.

5.4. Comparative Results to Previous Work

5.4.1 Virus Host Adaptability Predictor (DL) module Results

model	Accuracy
our approach	96%
RNN using one sequence	85%
Neural Network using multiple sequences	86%
HostNet (research paper)	60%

Table 5.5 Comparative Results for Virus Host Adaptability Predictor (DL)

5.4.2 Virus Host Adaptability Predictor (ML) module Results

model	Accuracy
Virus2Vec	91%
Machine Learning Features	89%
MFV	85%
PSWM2Vec	72%

Table 5.6 Comparative Results for Virus Host Adaptability Predictor (ML)

5.4.3 Drug-Target Interaction Predictor module Results

model	MSE	Pearson
our approach	0.9	0.83
Sequence-based drug-target affinity prediction using weighted graph neural networks (research paper)	0.2	0.85
pre-trained model (deep purpose)	1.4	—

Table 5.7 Comparative Results for Drug Target Interaction Predictor

5.4.4 Drug Generation module Results

model	validity	uniqueness	novelty
our approach	93.01%	99.6%	98.7%
ReGen-DTI (research paper)	92.71%	93.55%	100%

Table 5.8Comparative Results for Drug Generator

Chapter 6: Conclusions and Future Work

This chapter presents a summary of the ViroAlert system, highlighting its main features, achievements, and limitations. It reflects on the development process, the outcomes of each module, and the system's overall contribution to the field of AI-driven antiviral discovery. In addition, this chapter outlines potential directions for future work to enhance the system's performance, scalability, and real-world applicability.

6.1. Faced Challenges

6.1.1 Virus Host Adaptability Predictor (DL) module Challenges

1. One of the challenges we faced while designing the deep learning module was our initial goal of using all available virus sequences for training. However, we discovered that datasets containing viruses with many sequences (e.g., 7 or more) caused the model to perform poorly on viruses that had only 1 or 2 sequences. To address this issue, we searched for new datasets that included viruses with a more balanced and variable number of sequences per class. This made the preprocessing pipeline more complex and significantly increased the size of the training data, leading to higher computational resource demands during training.

2. A particularly challenging aspect was our attempt to implement a **Multi-Instance Learning (MIL)** approach using a **"bag of bags"** concept, where each virus is represented by a bag of sequence bags. This hierarchical structure reflects the natural variation among sequences of the same virus but introduces significant complexity in model design and data representation. Due to the limited availability of resources, documentation, and examples, implementing and optimizing such a model proved to be extremely challenging.

6.1.2 Virus Host Adaptability Predictor (ML) module Challenges

1. **Data Imbalance**
   Viral sequence datasets often exhibit significant class imbalance, with far more sequences from non-human hosts than human-adaptive viruses. This imbalance can bias classifiers and lead to lower sensitivity for the minority class.

2. **Data Quality Issues**
   Many viral protein sequences in public databases suffer from **incompleteness** or contain **ambiguous amino acid characters** (e.g., 'X', 'B', 'Z'), which can degrade the quality of feature extraction and downstream model performance. These issues arise from low sequencing coverage, annotation errors, or uncertainty during experimental determination.

3. **Generalization to Unseen Viruses**
   While models perform well on validation data, predicting host adaptability for emerging or novel viruses remains a challenge due to domain shift—genetic differences between training and testing strains.

4. **High Dimensionality of k-Mer Features**
   K-Mer-based embeddings, especially for large k values, produce high-dimensional feature vectors, which can lead to increased computational costs and overfitting if not properly regularized or reduced.

5. **Sequence Length Variability**
   Viral protein sequences vary significantly in length, affecting the number and consistency of extracted k-mers, especially for alignment-based approaches. This variability complicates feature standardization across samples.

6.1.3 Drug-Target Interaction Predictor Module Challenges

1. **Limited Dataset Availability**: It was hard to find a large and useful dataset for drug-target interactions. Most of the data was spread across different platforms and had to be collected using APIs, which was slow and limited. It took time to gather enough records and prepare them for training.

2. **Initial Feature Representation Challenges**: At first, we used basic protein features without building protein graphs. The results weren't very satisfying, so we searched for research papers to find better ways. That led us to use graph-based models instead.

6.1.4 Drug Generation module Challenges

1. The GAN model was unstable — the discriminator became too strong, so the generator stopped learning as the discriminator loss dropped to zero.

2. Training GANs takes a lot of time and computing power, and we had very limited resources that made it even more difficult.

6.2. Gained Experience

1. We explored and implemented advanced concepts like **GANs**, **GNNs**, and **MIL**, which were new to us at the start.
2. We learned how to customize data loaders to handle different data formats that fit our specific problem.
3. We gained hands-on experience in reading, understanding, and applying research papers to solve real problems.
4. We also learned how to write a scientific paper and successfully wrote our first research paper as part of this project.

6.3. Conclusions

This project focused on the important task of predicting and responding to new viruses using advanced AI methods. We built four main modules that work together:

1. **Deep Learning for Viral Host Adaptability Predictor**: Trained on viral protein sequences to predict whether a virus has the potential to infect humans.

2. **Machine Learning for Viral Host Adaptability Predictor**: We also used classical machine learning methods to try a different approach and compare the results with the deep learning model to see which performs better.

3. **DTI Module:** Uses GNNs to represent molecular structures of drugs and viruses. This module predicts potential interactions, helping identify which existing drugs may be effective against a particular virus.

4. **Drug Generator Module:** Based on GANs combined with Reinforcement Learning, this component generates novel drug candidates that are structurally valid and tailored to a given viral target. This supports early-stage drug discovery by proposing new therapeutic options.

Together, these modules form a comprehensive AI pipeline for viral infectivity prediction and new drug generation. The project also deepened our understanding of multiple advanced AI techniques such as GANs, GNNs, and MIL, and provided us experience with model evaluation, customization, and optimization for biological data.

To make our system accessible, we built a web-based application where users can input viral sequences and receive predictions about whether the virus can infect humans, possible interactions between the virus and known drugs, and even suggested new drug candidates. The backend AI services are hosted via FastAPI, while the frontend and database layers use React, Node.js, Prisma, and MySQL, respectively.

In summary, this project shows the power of combining diverse machine learning and deep learning approaches to solve real-world problems in virology and drug discovery. It helps improve the use of AI in biology and creates a strong base for future work in fighting viruses and making new drugs.

6.4. Future Work

6.4.1 Virus Host Adaptability Predictor (DL)

- **Real-Time Adaptability and Continuous Learning**: The current model is trained on static datasets. A future direction is to enable continuous learning where the model updates as new viral sequences emerge (e.g., through GISAID). This would keep the system relevant during ongoing outbreaks or pandemics.

- **Multi-Host Prediction:** Currently, the model performs binary classification (e.g., human vs non-human). Future work could involve expanding to multi-host classification or predicting host probabilities across multiple species to reflect the complex nature of viral transmission.

6.4.2 Virus Host Adaptability Predictor (ML)

- **Hybrid Feature Architectures**
  Combining k-Mer-based embeddings with additional biologically informative features such as physicochemical properties, codon usage bias, and evolutionary conservation metric to construct more robust and expressive models.

- **Transfer Learning Across Viral Families**
  To enhance generalizability, we plan to investigate transfer learning strategies that enable models trained on one viral family to adapt to others, particularly in cases where limited labeled data is available for emerging viruses.

- **Explainable AI for Model Interpretability**
  Applying explainability techniques like SHAP or LIME will help reveal which parts of the viral sequence contribute most to predictions, offering valuable biological insights and model transparency.

- **Fine-Grained Host Specificity Prediction**
  Instead of a binary classification, future models could be extended to multi-class predictions that identify the specific host species a virus is likely to infect (e.g., human, avian, bat, swine), which is critical for zoonotic surveillance.

- **Ensemble Modeling with DNA and Protein Features**: promising direction is to train **classical ML models on DNA-level features** (e.g., nucleotide composition, codon usage) and **deep learning models on protein-level embeddings**, then integrate their outputs using an **ensemble or decision fusion strategy**. This hybrid approach could leverage complementary information from both genomic and proteomic levels, leading to more accurate and robust host adaptability predictions.

6.4.3 Drug-Target Interaction Predictor

- **Use 3D Molecular Structures:** Integrate 3D conformations of proteins and drugs for more accurate interaction predictions.
- **Explainable DTI:** Apply visualization to highlight why a model predicts a weak or strong interaction (useful for biomedical researchers).
- **Use More Advanced GNN Architectures:** Replace basic GCNs with stronger models like Graph Attention Networks (GAT) or Message Passing Neural Networks (MPNNs) to learn better complex relationships.
- **Using All Viral Sequences Instead of a Single Representative:** Currently, the module uses just one viral sequence to represent each virus. In the future, we can improve accuracy by using all available sequences of a virus. This way, the model can better understand variations in the virus and make more reliable predictions.

6.4.4 Drug Generation

- **Multi-objective Optimization**: Current training focuses primarily on generating chemically valid SMILES strings. A natural extension would be to optimize for multiple properties simultaneously, including Drug-likeness (QED) and Toxicity (via predictors).
- **Target-Specific Molecule Design**:In future versions, the model can be adapted to generate molecules tailored to specific viral proteins. This would require training separate conditional models for different viral targets
- **Explainability and Visualization**:Future work can also focus on improving the interpretability of the GAN's outputs, such as highlighting which parts of

the sequence influence certain generated molecular fragments. This can support better understanding of virus-drug interaction mechanics.

- **Using All Viral Sequences Instead of a Single Representative**: Currently, we use only one viral sequence to generate drugs, which may miss important features. Using all sequences can improve accuracy and results.

.

References

[1] P. Sphabmixay and B. Lash, “ViRNN: A Deep Learning Model for Viral Host Prediction,” Apr. 2024.

[2] D. Liu, F. Young, K. D. Lamb, D. L. Robertson, and K. Yuan, “Prediction of virus-host associations using protein language models and multiple instance learning,” PLOS Computational Biology, vol. 20, no. 11, p. e1012597, Nov. 2024.

[3] Z. Ming et al., “HostNet: improved sequence representation in deep neural networks for virus-host prediction,” BMC Bioinformatics, vol. 24, no. 1, Dec. 2023.

[4] G. E. Bastien, R. N. Cable, C. Batterbee, A. J. Wing, L. Zaman, and M. B. Duhaime, “Virus-host interactions predictor (VHIP): Machine learning approach to resolve microbial virus-host interaction networks,” PLOS Computational Biology, vol. 20, no. 9, p. e1011649, Sep. 2024.

[5] F. Mock, A. Viehweger, E. Barth, and M. Marz, “VIDHOP, viral host prediction with deep learning,” Bioinformatics, vol. 37, no. 3, pp. 318–325, Aug. 2020.

[6] J. Li et al., “Machine Learning Methods for Predicting Human-Adaptive Influenza A Viruses Based on Viral Nucleotide Compositions,” Molecular Biology and Evolution, vol. 37, no. 4, pp. 1224–1236, Nov. 2019.

[7] S. Ali, B. Bello, P. Chourasia, R. T. Punathil, Y. Zhou, and M. Patterson, “PWM2Vec: An Efficient Embedding Approach for Viral Host Specification from Coronavirus Spike Sequences,” Biology, vol. 11, no. 3, p. 418, Mar. 2022

[8] M. Jiang, S. Wang, S. Zhang, W. Zhou, Y. Zhang, and Z. Li, “Sequence-based drug-target affinity prediction using weighted graph neural networks,” BMC Genomics, vol. 23, no. 1, Jun. 2022

[9] H. Atas Guvenilir and T. Doğan, “How to approach machine learning-based prediction of drug/compound–target interactions,” Journal of Cheminformatics, vol. 15, no. 1, Feb. 2023

[10]J. Wang, J. Mao, M. Wang, X. Le, and Y. Wang, “Explore drug-like space with deep generative models,” Methods, vol. 210, pp. 52–59, Feb. 2023, doi: 10.1016/j.ymeth.2023.01.004.

[11] K. B. Sivangi, S. Amilpur, and C. M. Dasari, “ReGen-DTI: A novel generative drug target interaction model for predicting potential drug candidates against SARS-COV2,” Computational Biology and Chemistry, vol. 106, p. 107927, Oct. 2023.

Appendix A: Development Platforms and Tools

A.1. Hardware Platforms

We used Kaggle as our main platform for uploading datasets and training our models. It provided free access to powerful GPUs, which helped us train deep learning models efficiently.

A.2. Software Tools

We used a variety of tools and libraries throughout the project, including:

- PyTorch – for building and training deep learning and GAN models
- RDKit – for processing drug molecules and generating molecular fingerprints
- FastAPI – to serve the AI models through a fast and lightweight API
- Node.js – for handling backend logic and managing server-side operations
- Prisma – as an ORM (Object-Relational Mapping) tool to interact with the database
- MySQL – to store user inputs, results, and other data
- Next.js – for building the web application's user interface
- Postman – for testing and debugging API endpoints during development

Appendix B: Use Cases

B.1. Human Infection Prediction

virology researcher can input a viral protein sequence to predict whether the virus is likely to infect humans, using either deep learning or machine learning model.

B.2. Drug Interaction Prediction

Virology researchers can enter a viral sequence along with a drug (in SMILES format), and the system will predict whether the drug is likely to interact with the virus using our DTI model.

B.3. Drug Generation

The **virology researcher** can request the system to generate new drug candidates for a given virus. The GAN-based module will return valid SMILES strings for potential drug compounds.

Appendix C: User Guide

Welcome to the ViroAlert Platform, a revolutionary AI-powered tool designed to predict virus behavior, generate antiviral drug candidates, and evaluate their effectiveness. This guide will help you navigate and utilize the platform's features effectively.

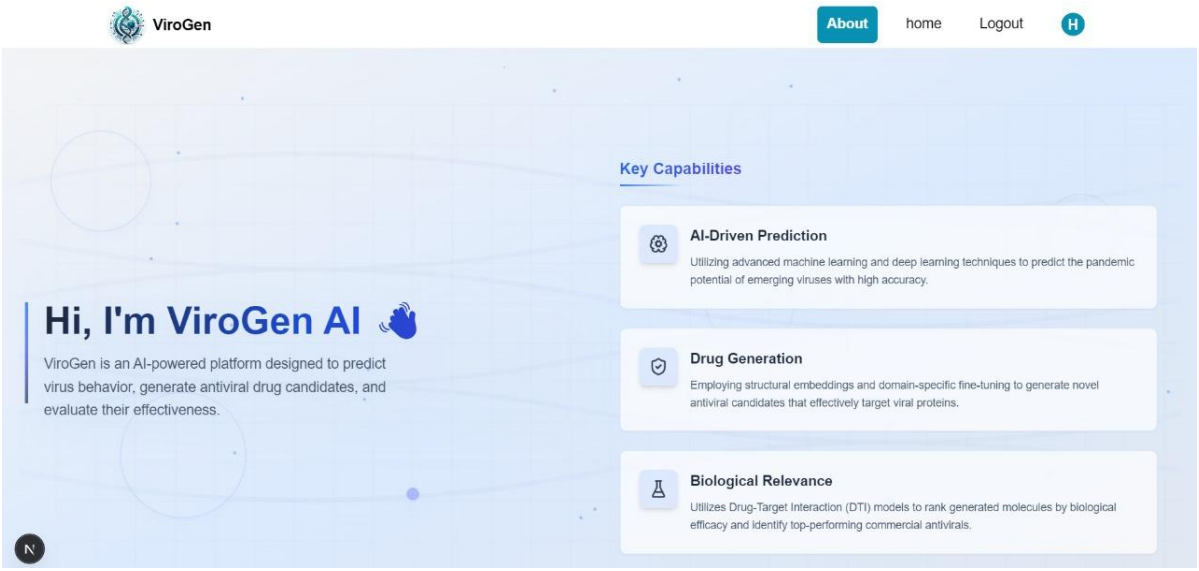

Figure C.1: About Page in ViroAlert App

Figure C.2: Home Page in ViroAlert App

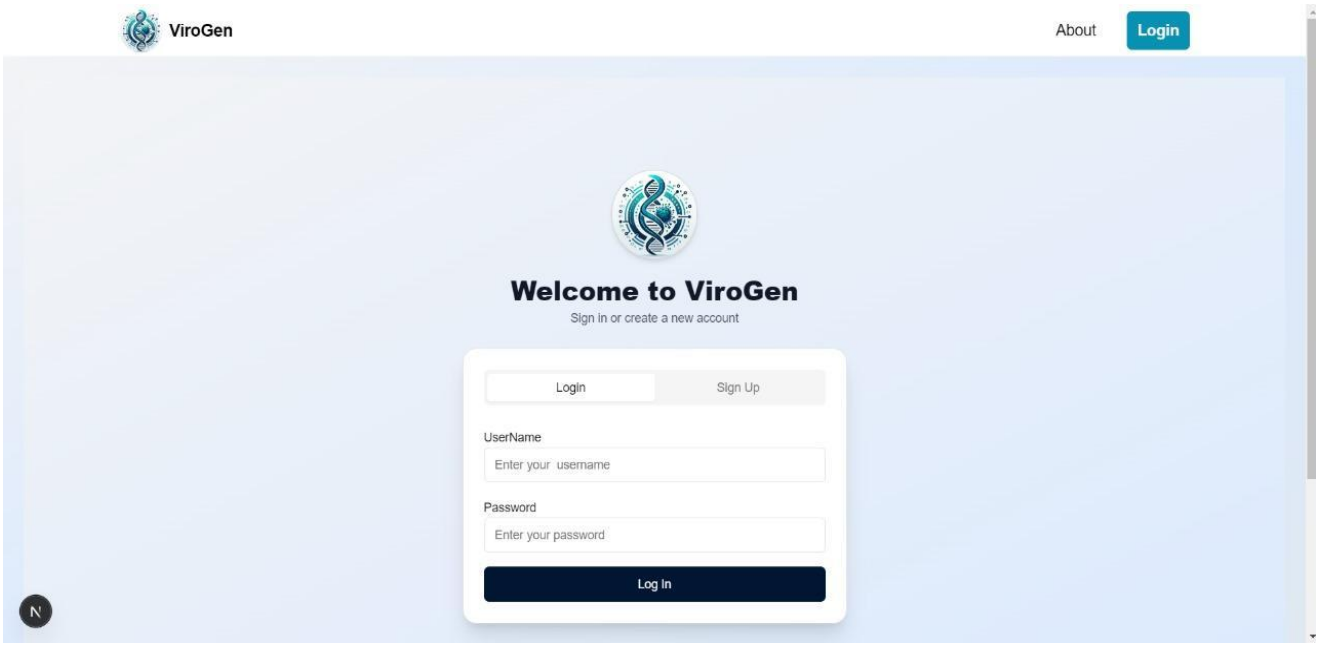

Figure C.4: Login Page in ViroAlert App

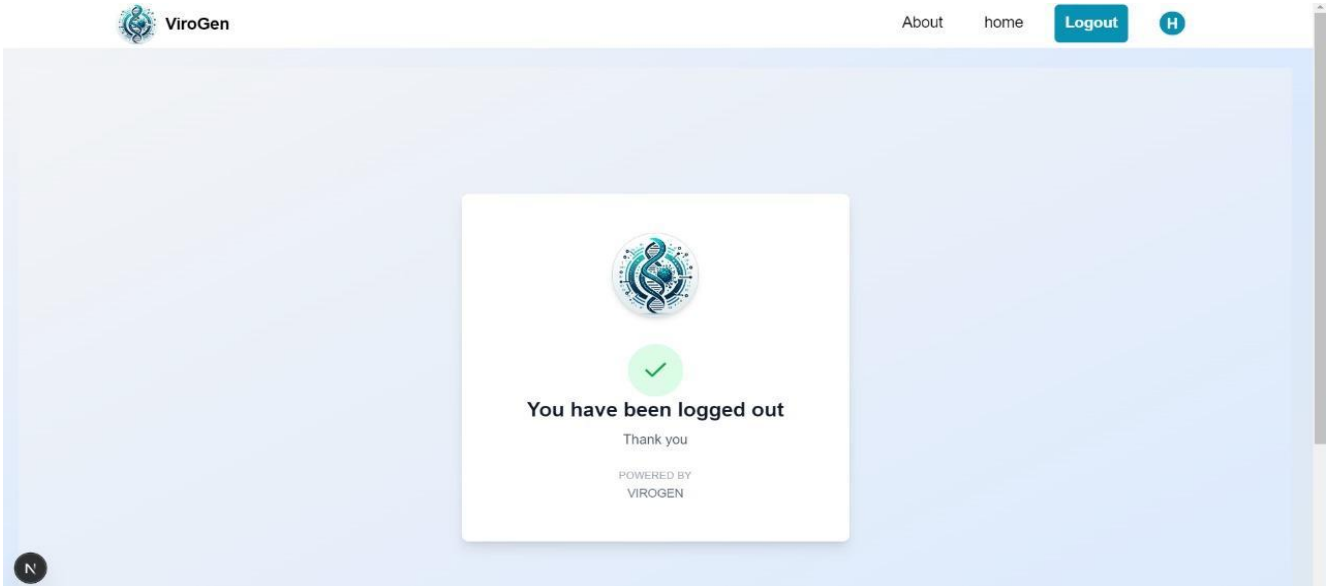

Figure C.3: Logout Page in ViroAlert App

C.1. Drug Target Interaction

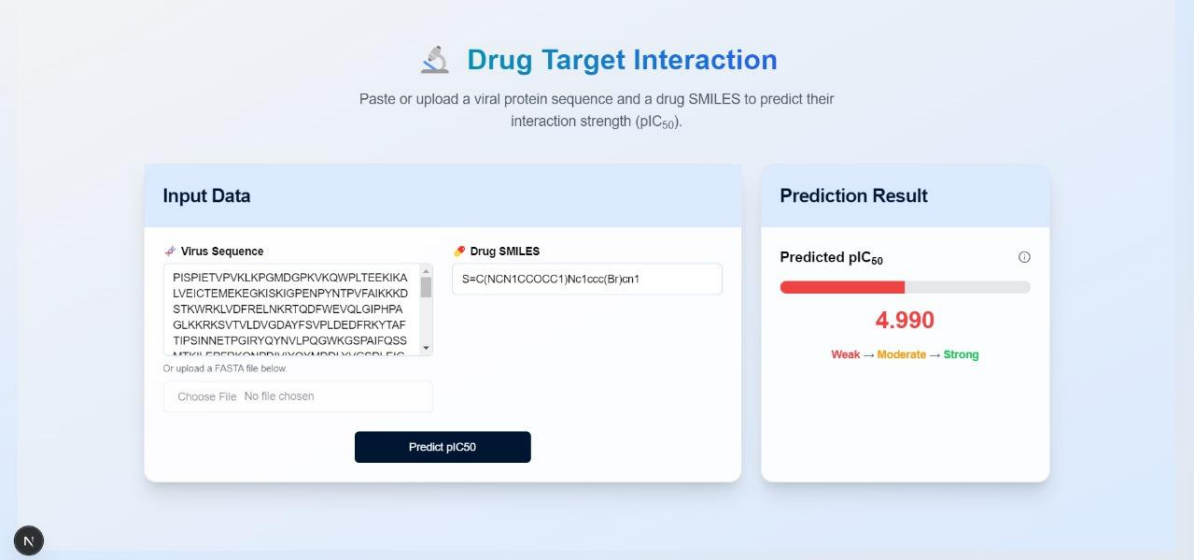

Figure C.5: Drug Target Interaction Page in ViroAlert App

The user can either upload a viral sequence file in **.fasta**, **.fa**, or **.txt** format, or directly paste the sequence into a provided text area. Additionally, the user must enter the **SMILES representation** of the drug to be tested. By clicking **"Predict pIC_{50}"**, the system will evaluate the potential interaction between the virus and the drug. The resulting **pIC_{50} score** will be display where a higher value indicates a stronger inhibitory effect of the drug on the virus.

C.2. Sequence Match Analyzer

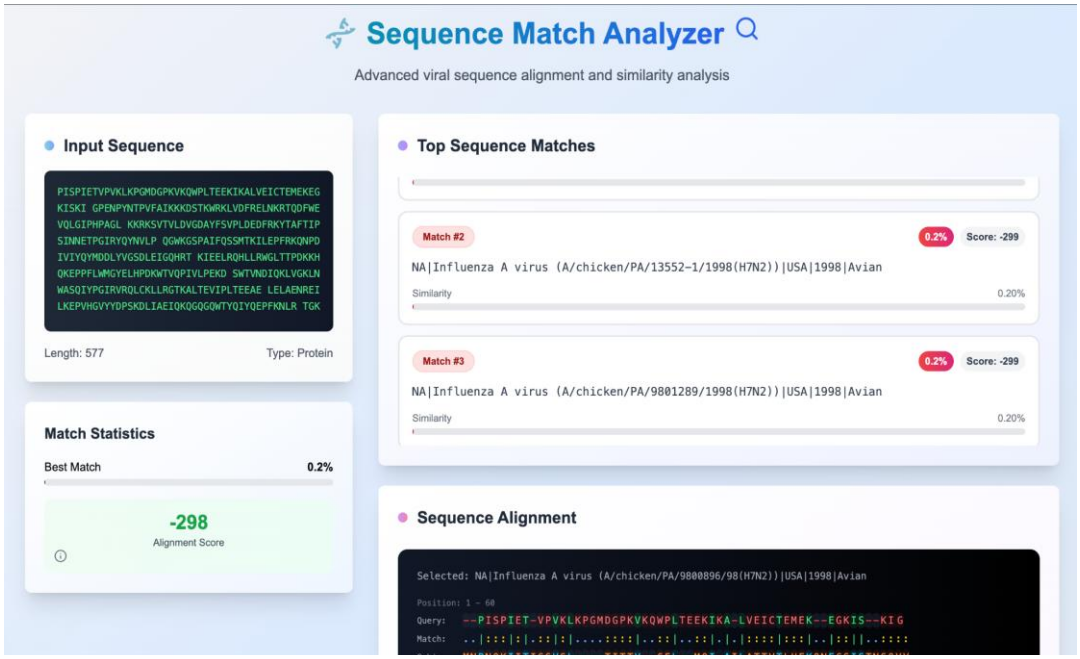

Figure C.6: Alignment Page in ViroAlert App

Figure C.7: Sequence Alignment In ViroAlert App

This tool enables users to analyze protein sequences from viral pathogens by identifying top sequence matches and displaying detailed alignment statistics against known viral strains.

1. Input the Protein Sequence

- Users can either:
  - Upload a sequence file in .fasta, .fa, or .txt format, or
  - Directly paste the protein sequence into the provided input text area.
- The system supports standard protein sequences (e.g., FASTA-formatted).

2. View Input Sequence Details

- The left-side panel shows:
  - The full input sequence.
  - The length of the sequence (e.g., 577 residues).
  - The type of the sequence (e.g., Protein).

3. Match Statistics

- The system calculates an Alignment Score for the top match.
- Best Match % reflects the degree of similarity (e.g., 84.3%).
- Alignment Score quantifies the match strength (e.g., 542 for high similarity, -298 for poor matches).

4. Review Top Sequence Matches

- Ranked matches are listed with:
  - Match ID (e.g., Match #2).
  - Matched virus name and metadata (e.g., strain, year, host).
  - Score (quantitative match strength).
  - Similarity % of aligned residues.

5. Analyze Sequence Alignment

- The bottom section shows a residue-level alignment of the input sequence with the selected viral protein:
  - Query: Your input sequence.
  - Match: Vertical bars (|) show matching residues.
  - MisMatch: (:) show different amino acids.
  - skipped: (.)skipped amino acids.
  - Subj: Matched reference sequence from the database.
  - Match percentage is given for each segment (e.g., 60/60 = 100%).

C.3. Human Adaptation Prediction

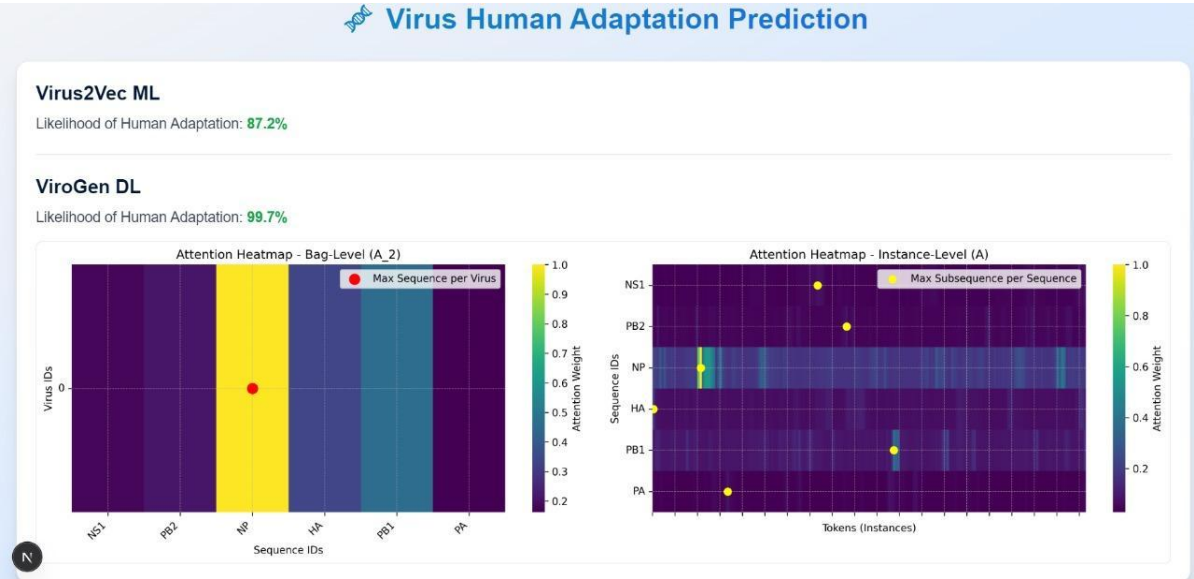

Figure C.8: Human Adaptation Prediction Page in ViroAlert App

After the user uploads a viral file containing multiple protein sequence in **.fasta**, **.fa**, or **.txt** format, or directly paste the sequence into a provided text area, the system will display:

- The predicted probability of human adaptation from both the machine learning model and the deep learning model.
- A heatmap image visualizing the importance (weights) of each sequence in contributing to the final prediction.
- A second image highlighting the most influential amino acids (tokens) within each sequence, helping researchers study mutation hotspots and understand which regions are most critical to the host adaptation decision.

C.4. Recommending Top Antiviruses && Generating New Antiviruses

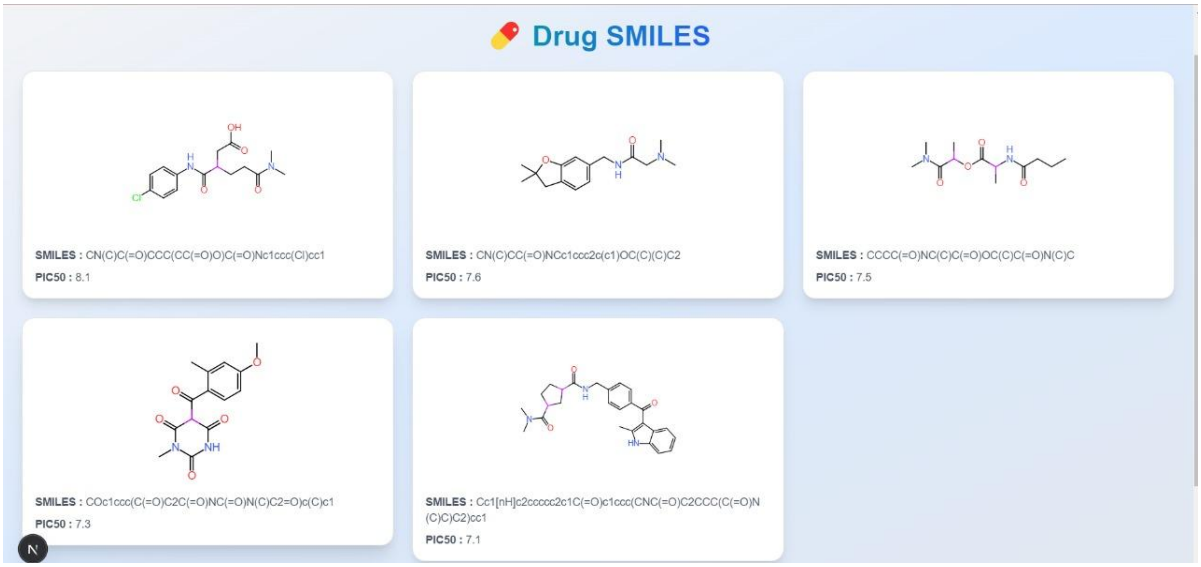

Figure C.9: Drugs Recommendation Page in ViroAlert App

Recommending Top Antiviruses:

The "Find Known Antivirals" module leverages the platform's Biological Relevance capability to search through a database of commercial antivirals. It identifies and recommends the top 5 drugs with the highest potential to bind effectively with a specified virus, based on Drug-Target Interaction (DTI) models**.**

Generating New Antiviruses:

This module allows users to fine-tune antiviral drug development by analyzing a given virus's genetic and structural data. It employs advanced machine learning techniques and domain-specific fine-tuning to generate novel antiviral drug

candidates tailored to effectively target the specified virus, offering a cutting-edge solution for rapid antiviral development.

Appendix D: Feasibility Study

1. **Technical Feasibility**
    - Tools such as PyTorch, RDKit, and Kaggle GPUs enabled efficient development and training.
    - **FastAPI, React, Node.js** for backend and frontend integration.

2. **Operational Feasibility**
    - The web-based system is easy for virology researchers to use: they just upload a virus sequence and get predictions.
    - The system produces clear outputs: host prediction, pIC50 drug scores, top antiviral candidates and generated drugs.
    - We used visualization tools for embeddings and attention heatmaps.

3. **Economic Feasibility**
    - We used free/open-source tools (e.g., PyTorch, RDKit, MySQL).
    - Training was done on Kaggle, which offers free GPU resources (reducing hardware cost)
    - No need for paid APIs or commercial datasets.

Appendix E: Project Progress Log

Meet (1): 10-9- 2024 (Online)

This was our first meeting. We discussed initial ideas for our graduation project and clarified what qualifies as a suitable topic. We considered several fields, including:

- Virology (e.g., virus analysis using AI/ML)
- Computer Vision (e.g., object detection, recognition)
- Self-Driving Cars (e.g., autonomous navigation systems)
- Other Ideas, such as:
    - Augmented Reality applications
    - Virtual Labs
    - gym and health application

Meet (2): 29 -9- 2024 (on Campus)

In this meeting, we continued discussing potential ideas for our graduation project. Our supervisor suggested that we focus our research in the field of virology, particularly exploring how artificial intelligence and machine learning could be applied to problems in this domain.

Project Status: Narrowed down the field of interest to virology using AI/ML.

Meet (3): 14-10-2024 (Online)

- In this meeting, we met with Dr. Ibrahim, a virology expert, who explained key virology concepts to support our understanding of the scientific aspects of the project.
- We discussed the modules for our project.

Project Status: Gaining domain knowledge and starting to define project modules.

Meet (4): 21-10-2024 (Online)

In this meeting, we continued our discussion with Dr. Ibrahim after conducting further research on the proposed idea. We shared our findings, asked follow-up questions, and discussed the feasibility and direction of the project based on the insights we gained from our readings.

Project Status: Refining the project scope and validating the idea's feasibility.

Meet (5): 2-12- 2024 (on Campus)

In this meeting, we met with our supervisor to present and review our seminar presentation. We received feedback and discussed necessary revisions to improve the content and clarity of our research direction.

Project Status: Seminar presentation prepared; revisions in progress.

Meet (6): 24 -12- 2024 (online)

In this meeting, we discussed the feedback we received from the doctors during the seminar. We focused on how to address the issues they highlighted, avoid the mistakes they warned us about, and effectively integrate the ideas and suggestions they provided to enhance our research.

Project Status: Refining project structure and addressing seminar feedback.

Meet (7): 09-03-2025 (Online)

In this meeting, we discussed the progress made in the antivirus and deep learning module. We reviewed the current results and addressed any challenges we faced in the implementation.

Project Status: Deep learning model under development; initial implementation completed.

Meet (8): 02-06-2025 (Online)

In this meeting, we explored the idea of integrating a Generative Adversarial Network (GAN) into our project. We discussed how GANs could enhance our system, particularly in generating new drug candidates for a specific virus.

Project Status: Deep learning, DTI, and machine learning module finished.. Research and planning stage for GAN integration.

Meet (9): 09-06-2025 (Online)

In this meeting, we presented our latest updates to the doctor and discussed the

overall progress of the project. Feedback was provided on our recent developments, helping us refine the remaining components.

Project Status: All finished except GAN(in progress) and the application.

Meet (10): 15-6- 2025 (on Campus)

In this meeting, Dr. Ibrahim was also present. We presented the final draft of our paper. The doctor reviewed it and suggested several improvements to enhance the quality and clarity of the work. We also discussed the structure and key points to highlight in our final presentation.

Project Status: Final paper draft completed; final presentation preparation in progress.