A Real-time Research Project / Societal Related Project

Report on

Gen AI Med Diagnosis And Drug Design

Submitted in Partial fulfillment of requirements for B.Tech II Year II Semester course

By

A. Akhil Varma	**23BD1A050A**
D. Sai Sehitha	**23BD1A0523**
G. Tejaswi	**23BD1A052F**
Jayansh Adithya J	**23BD1A052W**
L. Shivani	**23BD1A053U**
S. Srija	**23BD1A053Z**

Under the guidance of

Dr. R. Devika Ruby
Associate Professor , R & D Department

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH.
Narayanaguda, Hyderabad, Telangana-29

2024-25

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH
Narayanaguda, Hyderabad, Telangana-29

CERTIFICATE

This is to certify that this is a bonafide record of the project report titled "**Gen AI Med Diagnosis And Drug Design "** which is being presented as the **Real-time Research Project / Societal Related Project** report by

1.	**A.Akhil Varma**	**23BD1A050A**
2.	**D. Sai Sehitha**	**23BD1A0523**
3.	**G. Tejaswi**	**23BD1A052F**
4.	**Jayansh Adithya J**	**23BD1A052W**
5.	**L. Shivani**	**23BD1A053U**
6.	**S. Srija**	**23BD1A053Z**

In partial fulfillment for the II Year II Semester Course RTRP in KMIT affiliated to the Jawaharlal Nehru Technological University Hyderabad, Hyderabad

Mentors
(Dr. Devika Ruby)

Program Coordinator

(Mr Shailesh Gangakhedkar)

Submitted for Final Project Review held on _____

Vision & Mission of KMIT

Vision of KMIT

- To be the fountainhead in producing highly skilled, globally competent engineers.
- Producing quality graduates trained in the latest software technologies and related tools and striving to make India a world leader in software products and services.

Mission of KMIT

- To provide a learning environment that inculcates problem solving skills, professional, ethical responsibilities, lifelong learning through multi modal platforms and prepares students to become successful professionals.
- To establish an industry institute Interaction to make students ready for the industry.
- To provide exposure to students on the latest hardware and software tools.
- To promote research-based projects/activities in the emerging areas of technology convergence.
- To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
- To induce a spirit of nationalism which will enable the student to develop, understand India's challenges and to encourage them to develop effective solutions.
- To support the faculty to accelerate their learning curve to deliver excellent service to students.

DECLARATION

We hereby declare that the results embodied in the dissertation entitled **"**Gen AI Med Diagnosis And Drug Design **"** has been carried out by us together during the academic year 2024-25 as a partial fulfillment of the B.Tech II Year II Semester Course "**Real-time Research Project / Societal Related Project**". We have not submitted this report to any other Course/College.

Student Name	Rollno.
1. A.Akhil Varma	23BD1A050A
2. D. Sai Sehitha	23BD1A0523
3. G. Tejaswi	23BD1A052F
4. Jayansh Adithya J	23BD1A052W
5. L. Shivani	23BD1A053U
6. S. Srija	23BD1A053Z

ACKNOWLEDGEMENT

We take this opportunity to thank all the people who have rendered their full support to our project work. We render our thanks to **Dr. B L Malleswari**, Principal who encouraged us to do the Project.

We are grateful to **Mr. Neil Gogte,** Founder & Director and **Mr. S. Nitin,** Director, for facilitating all the amenities required for carrying out this project.

We express our sincere gratitude to **Ms. Deepa Ganu**, Director Academic for providing an excellent environment in the college.

We are also thankful to **Mr.Shailesh Gangakhedkar**, **Real-Time Research Project** Program Coordinator for providing us with time to make this project a success within the given schedule.

We are also thankful to our Project Mentor **Dr. Devika Ruby**, for her/his valuable guidance and encouragement given to us throughout the project work.

We would like to thank the entire KMIT faculty, who helped us directly andindirectly in the completion of the project.

We sincerely thank our friends and family for their constant motivation during the project work.

Student Name	Roll no.
A. Akhil Varma	**23BD1A050A**
D. Sai Sehitha	**23BD1A0523**
G. Tejaswi	**23BD1A052F**
Jayansh Adithya J	**23BD1A052W**
L. Shivani	**23BD1A053U**
S. Srija	**23BD1A053Z**

ABSTRACT

This web application integrates Generative AI and Machine Learning to advance lung disease diagnosis and accelerate drug discovery. Built on the MERN stack with AWS integration, it unifies medical imaging and pharmaceutical tools into a cohesive platform. Key features include AI-powered lung tumor segmentation (UNETR), 3D CT scan visualization, de novo drug design using reinforcement learning, SMILES structure completion, ligand prediction for Alzheimer's using Vision Transformers, protein-to-SMILES conversion, 3D protein structure rendering, and automated docking simulations. The system enables faster, more accurate diagnostics and drug development through intelligent automation and visualization.

CONTENTS

S. No.	Title	Page No.
1	Introduction	8
2	Lung Tumor Segmentation	9–11
3	3D Medical Image Viewer	12–13
4	De Novo Drug Molecules by Reinforcement Learning	14–18
5	Federated Learning for De Novo Drug Design	19
6	SMILES Masking Identification	20–22
7	Active/Inactive Ligand Identification for Alzheimer's Disease using ViT	23–25
8	Nucleotides and Amino Acids	26–28
9	Codon Sequences in GenBank	29–30
10	GenAI: Protein to SMILES Converter	31–33
11	Protein 3D Structure Visualization	34–35
12	Auto Docking Visualization	36–38
13	Web Hooks	39
14	Integration Workflow	40
15	AWS Deployment	41–42

1. INTRODUCTION

This web application leverages **Generative AI and Machine Learning** to enhance lung disease diagnosis and accelerate drug discovery. The system combines medical imaging and pharmaceutical research tools into a unified platform to support faster, more accurate outcomes.

Technologies: MERN, GenAI, ML, webhook, AWS

Tools & Features:

- **Lung Tumor Segmentation - UNETR**
  AI-based segmentation of CT images for early and accurate lung disease diagnosis.
- **3D Medical Image Viewer**
  Interactive visualization of lung CT scans for detailed clinical analysis.
- **De Novo Drug Molecule Generation (RL)**
  Uses reinforcement learning to design novel drug-like molecules.
- **SMILES Masking Identification**
  Identifies and completes chemical structures using GenAI techniques.
- **Ligand Identification for Alzheimer (ViT)**
  Applies Vision Transformers to predict ligands effective against Alzheimer-related targets.
- **GenAI: Protein to SMILES Converter**
  Predicts drug-like SMILES strings directly from protein sequences.
- **Protein 3D Structure Visualization**
  Renders and explores protein structures for structural biology insights.
- **Auto Docking Visualization**
  Automates protein-ligand docking simulations to predict binding affinity.

2.Medical Image Analysis

Lung Tumor Segmentation – using UNETR

Aim:

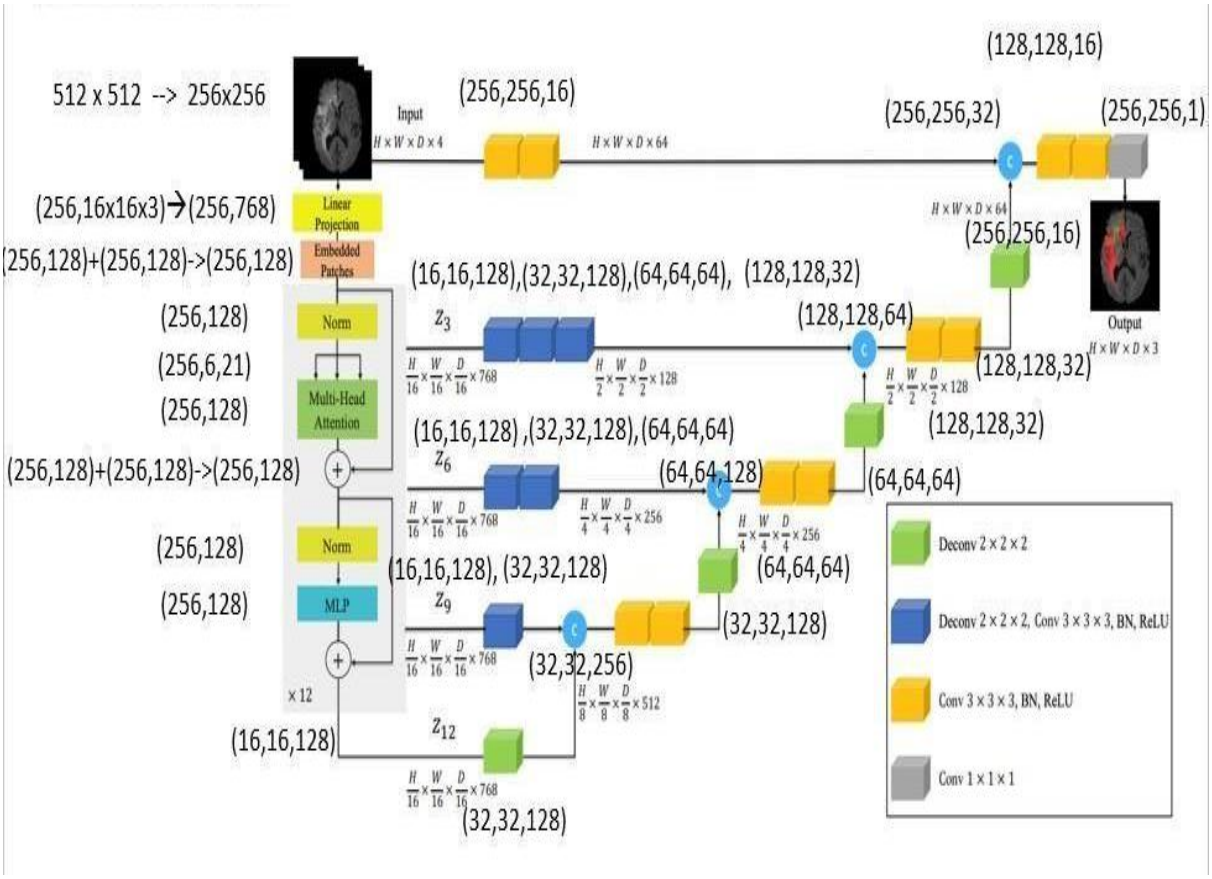
To achieve highly accurate segmentation and classification of lung tumors in CT images using **UNETR (UNEt Transformer)**, a transformer-based architecture optimized for 3D medical image analysis.

Dataset used:

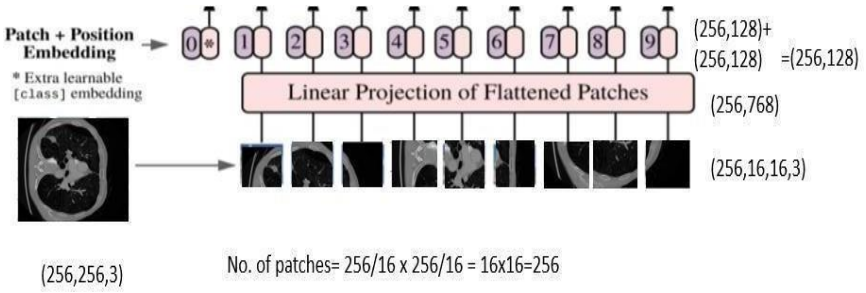
https://www.kaggle.com/datasets/rasoulisaeid/lung-cancer-segment

The original data is provided by the
Medical Segmentation Decathlon Challenge

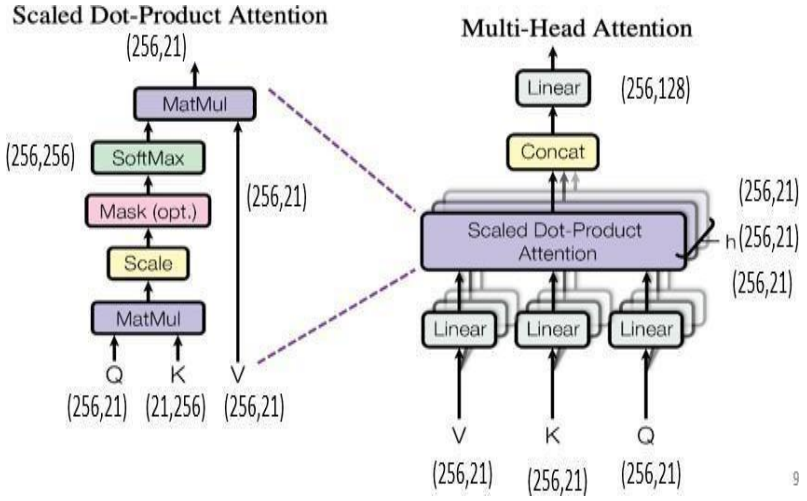
UNETR Architecture:



Patch Embedding and Positional encoding



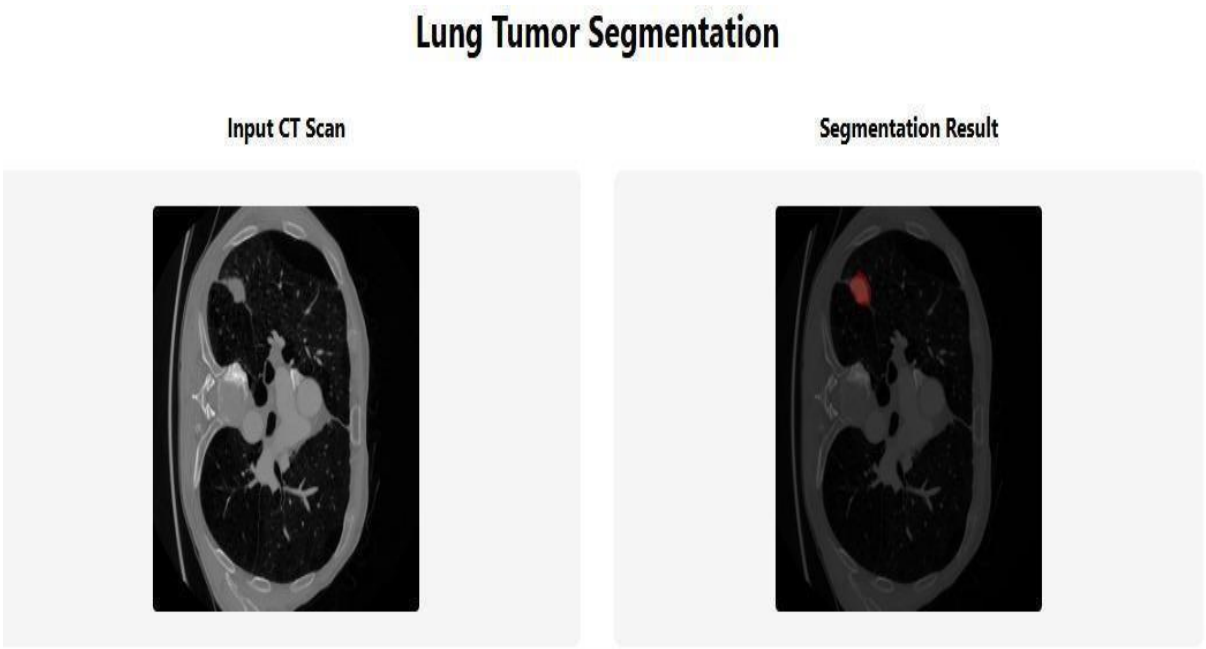
Multi head Attention



Model Accuracy

```
Epoch 66: val_loss did not improve from 0.31921
721/721 ━━━━━━━━━━━━━━━━━━━━ 125s 173ms/step - acc: 0.9981 - dice_coef: 0.6859 - loss: 0.3141 - val_acc: 0.9981 - val_dice_coef: 0.6583
Epoch 67/100
721/721 ━━━━━━━━━━━━━━━━━━━━ 0s 148ms/step - acc: 0.9981 - dice_coef: 0.6960 - loss: 0.3040
Epoch 67: val_loss improved from 0.31921 to 0.31909, saving model to files/model.keras
721/721 ━━━━━━━━━━━━━━━━━━━━ 145s 178ms/step - acc: 0.9981 - dice_coef: 0.6960 - loss: 0.3040 - val_acc: 0.9982 - val_dice_coef: 0.6809
Epoch 68/100
721/721 ━━━━━━━━━━━━━━━━━━━━ 0s 151ms/step - acc: 0.9982 - dice_coef: 0.7085 - loss: 0.2915
Epoch 68: val_loss improved from 0.31909 to 0.30783, saving model to files/model.keras
721/721 ━━━━━━━━━━━━━━━━━━━━ 136s 170ms/step - acc: 0.9982 - dice_coef: 0.7085 - loss: 0.2915 - val_acc: 0.9980 - val_dice_coef: 0.6922
Epoch 69/100
721/721 ━━━━━━━━━━━━━━━━━━━━ 0s 151ms/step - acc: 0.9983 - dice_coef: 0.7271 - loss: 0.2729
Epoch 69: val_loss improved from 0.30783 to 0.27067, saving model to files/model.keras
721/721 ━━━━━━━━━━━━━━━━━━━━ 150s 181ms/step - acc: 0.9983 - dice_coef: 0.7271 - loss: 0.2729 - val_acc: 0.9984 - val_dice_coef: 0.7293
Epoch 70/100
721/721 ━━━━━━━━━━━━━━━━━━━━ 0s 149ms/step - acc: 0.9984 - dice_coef: 0.7361 - loss: 0.2639
Epoch 70: val_loss did not improve from 0.27067
721/721 ━━━━━━━━━━━━━━━━━━━━ 131s 166ms/step - acc: 0.9984 - dice_coef: 0.7361 - loss: 0.2639 - val_acc: 0.9983 - val_dice_coef: 0.6943
Epoch 71/100
```

Sample Input and Output from Website:



3. 3D Medical Image Viewer

NiBabel is a Python library for reading/writing neuroimaging formats (e.g., NIfTI, Analyze, MINC, MGH), handling voxel data and metadata, and integrating with NumPy/SciPy for MRI/fMRI analysis and ML preprocessing.

To import Nibabel library we use:

import nibabel as nib

Papaya is a pure JavaScript medical research image viewer, supporting NIFTI formats, compatible across a range of web browsers. This orthogonal viewer supports overlays. The Papaya UI is configurable with many display, menu and control options and can be run on a web server or as a local, shareable file.

```
# Load the NIfTI file from the given path
nii_img = nib.load(file_location)

# Extract voxel data as a NumPy array (float64 by default)
nii_data = nii_img.get_fdata()
```

```
# Create a NIfTI image from the predicted volume using the original image's affine matrix
nii_pred = nib.Nifti1Image(predicted_volume, affine=original_img.affine)
# Save the NIfTI image to the specified path
nib.save(nii_pred, predicted_nii_path)
```

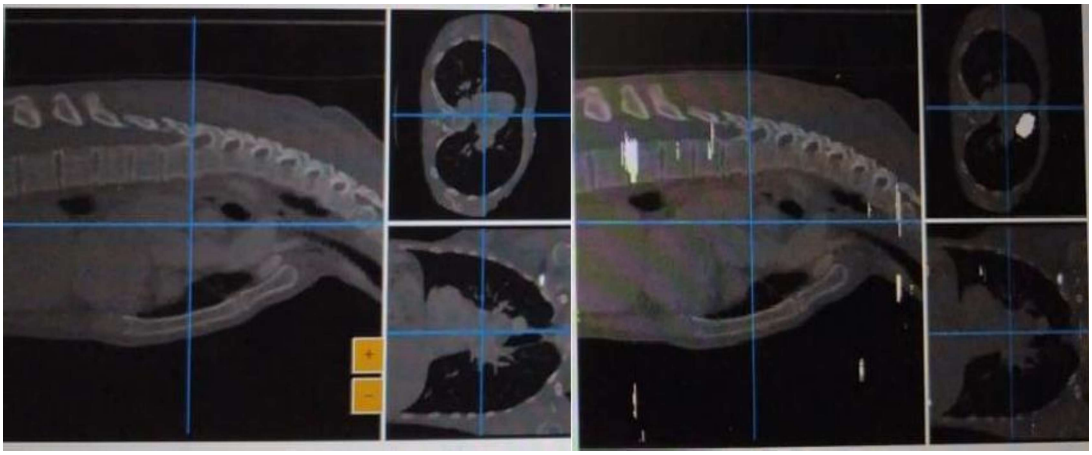
You can use it to visualize segmentation results in a React or Flask app by:

1. Saving your .nii.gz image and predicted mask.

2.Passing those as URLs to the data-params attribute.

3. Embedding Papaya in your results page to render both.

Sample input and output from website:



3 . DE-NOVO DRUG DESIGN

USING REINFORCEMET LEARNING

AIM :

Use a smiles-generating rnn with reinforcement learning to optimize molecules for target properties (like logp or jak2 activity) .

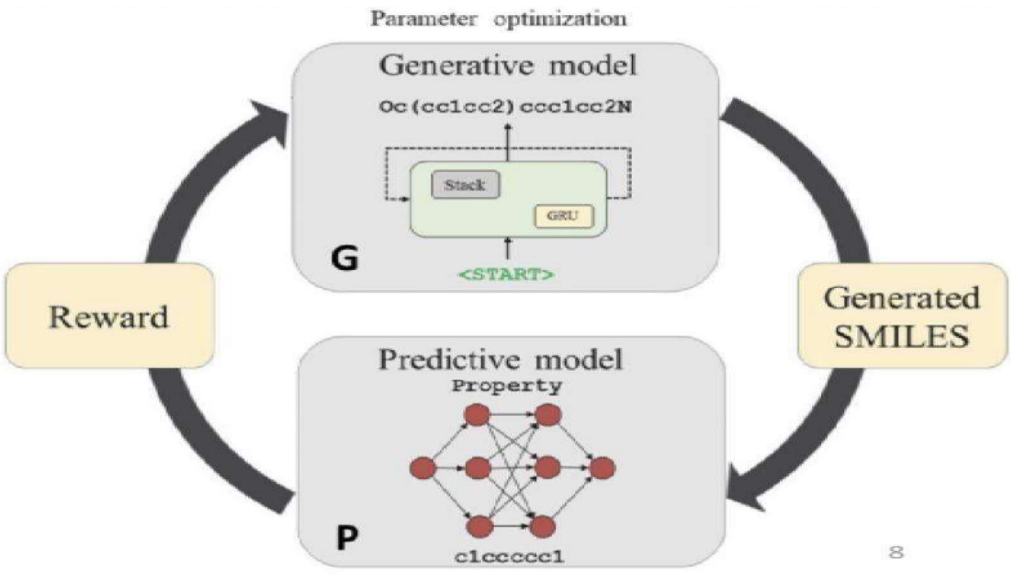
DATASET DESCRIPTION :

Chembl — A subset drug-like molecules from ChEMBL for training the generator.
https://github.com/isayev/ReLeaSE/blob/master/data/chembl_22_clean_1576904_sorted_std_final.smi

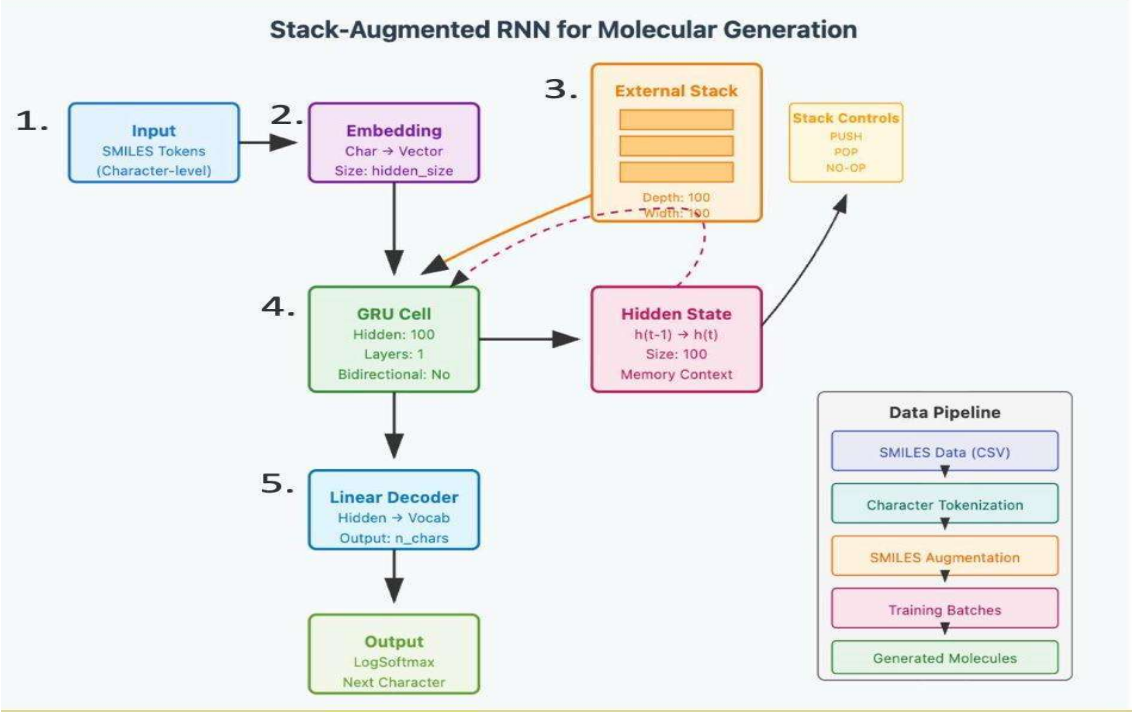
JAK2 — Experimental JAK2 kinase inhibition data for training predictive model.
https://github.com/isayev/ReLeaSE/blob/master/data/jak2_data.csv

logP_labels — A set of 14000 commercially available molecules for training the predictive model.
https://github.com/isayev/ReLeaSE/blob/master/data/logP_labels.csv

MODEL WORKFLOW :



GENERATIVE MODEL :



1.

```
def tokenize(smiles, tokens=None):

    if tokens is None:
        tokens = list(set(''.join(smiles)))
        tokens = list(np.sort(tokens))
        tokens = ''.join(tokens)
    token2idx = dict((token, i) for i, token in enumerate(tokens))
    num_tokens = len(tokens)
    print("Tokens \n",tokens)
    print("Tokens to index \n",token2idx)
    print("No of tokens:",num_tokens)
    return tokens, token2idx, num_tokens
```

2.

```
self.encoder = nn.Embedding(input_size, hidden_size)
```

3.

```
def stack_augmentation(self, input_val, prev_stack, controls):

    batch_size = prev_stack.size(0)

    controls = controls.view(-1, 3, 1, 1)
    zeros_at_the_bottom = torch.zeros(batch_size, 1, self.stack_width)
    if self.use_cuda:
        zeros_at_the_bottom = Variable(zeros_at_the_bottom.cuda())
    else:
        zeros_at_the_bottom = Variable(zeros_at_the_bottom)
    a_push, a_pop, a_no_op = controls[:, 0], controls[:, 1], controls[:, 2]
    stack_down = torch.cat((prev_stack[:, 1:], zeros_at_the_bottom), dim=1)
    stack_up = torch.cat((input_val, prev_stack[:, :-1]), dim=1)
    new_stack = a_no_op * prev_stack + a_push * stack_up + a_pop * stack_down
    return new_stack
```

```
    hidden_2_stack = hidden_.squeeze(0)
stack_controls = self.stack_controls_layer(hidden_2_stack)
print("stack controls:",stack_controls)
stack_controls = F.softmax(stack_controls, dim=1)
print("stack controls after applying softmax:",stack_controls)
stack_input = self.stack_input_layer(hidden_2_stack.unsqueeze(0))
print("stack input:",stack_input)
stack_input = torch.tanh(stack_input)
print("stack input after tanh:",stack_input)
stack = self.stack_augmentation(stack_input.permute(1, 0, 2),
                                stack, stack_controls)
stack_top = stack[:, 0, :].unsqueeze(0)
print("top of the stack:",stack_top)
inp = torch.cat((inp, stack_top), dim=2)
print("Input after adding the top",inp)
```

4.

```
elif self.layer_type == 'GRU':
    self.rnn = nn.GRU(rnn_input_size, hidden_size, n_layers,
                      bidirectional=self.is_bidirectional)
    self.decoder = nn.Linear(hidden_size * self.num_dir, output_size)
self.log_softmax = torch.nn.LogSoftmax(dim=1)
```

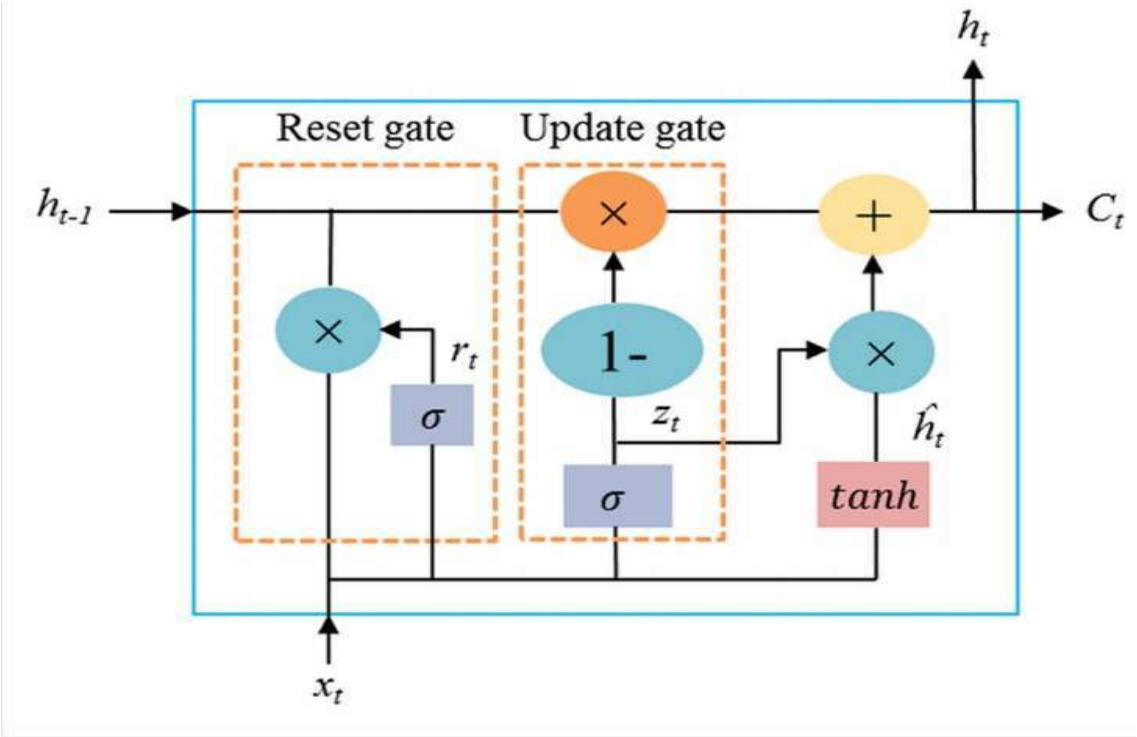
```
output, next_hidden = self.rnn(inp.view(1, 1, -1), hidden)
print("output",output)
print("Next hidden",next_hidden)
```

```
self.decoder = nn.Linear(hidden_size * self.num_dir, output_size)
```

5.

```
output = self.decoder(output.view(1, -1))
```

GRU ARCHITECTURE :



FORMULAS :

1. Update Gate:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

2. Reset Gate:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

3. Candidate Hidden State:

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$$

4. Final Hidden State:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

1. Sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

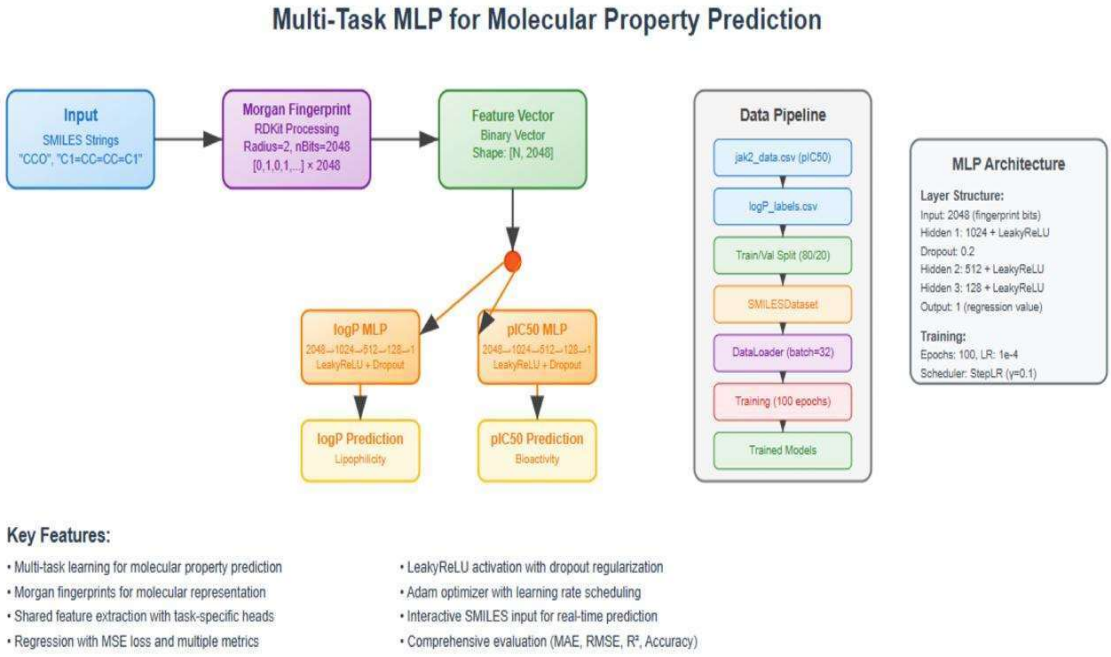
- Range: $[0, 1]$

2. Hyperbolic Tangent:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- Range: $[-1, 1]$

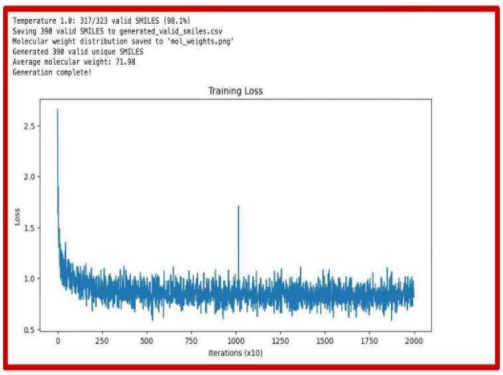
PREDICTIVE MODEL :



Key Features:

- Multi-task learning for molecular property prediction
- Morgan fingerprints for molecular representation
- Shared feature extraction with task-specific heads
- Regression with MSE loss and multiple metrics
- LeakyReLU activation with dropout regularization
- Adam optimizer with learning rate scheduling
- Interactive SMILES input for real-time prediction
- Comprehensive evaluation (MAE, RMSE, R², Accuracy)

GENERATIVE MODEL TRAINING :




```
Training pIC50 model...
Epoch 1: Val R2: 0.3380, MAE: 0.7500, RMSE: 0.9375, Accuracy: 89.43%
Epoch 2: Val R2: 0.4817, MAE: 0.6550, RMSE: 0.8296, Accuracy: 90.63%
Epoch 3: Val R2: 0.5295, MAE: 0.6074, RMSE: 0.7904, Accuracy: 91.28%
Epoch 4: Val R2: 0.5433, MAE: 0.5951, RMSE: 0.7788, Accuracy: 91.47%
Epoch 5: Val R2: 0.5552, MAE: 0.5850, RMSE: 0.7685, Accuracy: 91.69%
Epoch 6: Val R2: 0.5458, MAE: 0.5882, RMSE: 0.7766, Accuracy: 91.65%
Epoch 7: Val R2: 0.5609, MAE: 0.5737, RMSE: 0.7635, Accuracy: 91.92%
Epoch 8: Val R2: 0.5704, MAE: 0.5636, RMSE: 0.7553, Accuracy: 91.99%
Epoch 9: Val R2: 0.5571, MAE: 0.5781, RMSE: 0.7669, Accuracy: 91.83%
Epoch 10: Val R2: 0.5416, MAE: 0.5785, RMSE: 0.7802, Accuracy: 91.85%
Final pIC50 - R2: 0.7011, MAE: 0.5096, RMSE: 0.7223, Accuracy: 92.12%
Final LogP - R2: 0.8477 , MAE: 0.5033, RMSE: 0.7141, Accuracy: 96.55%
```

PREDICTIVE MODEL TRAINING

MODEL SUMMARY

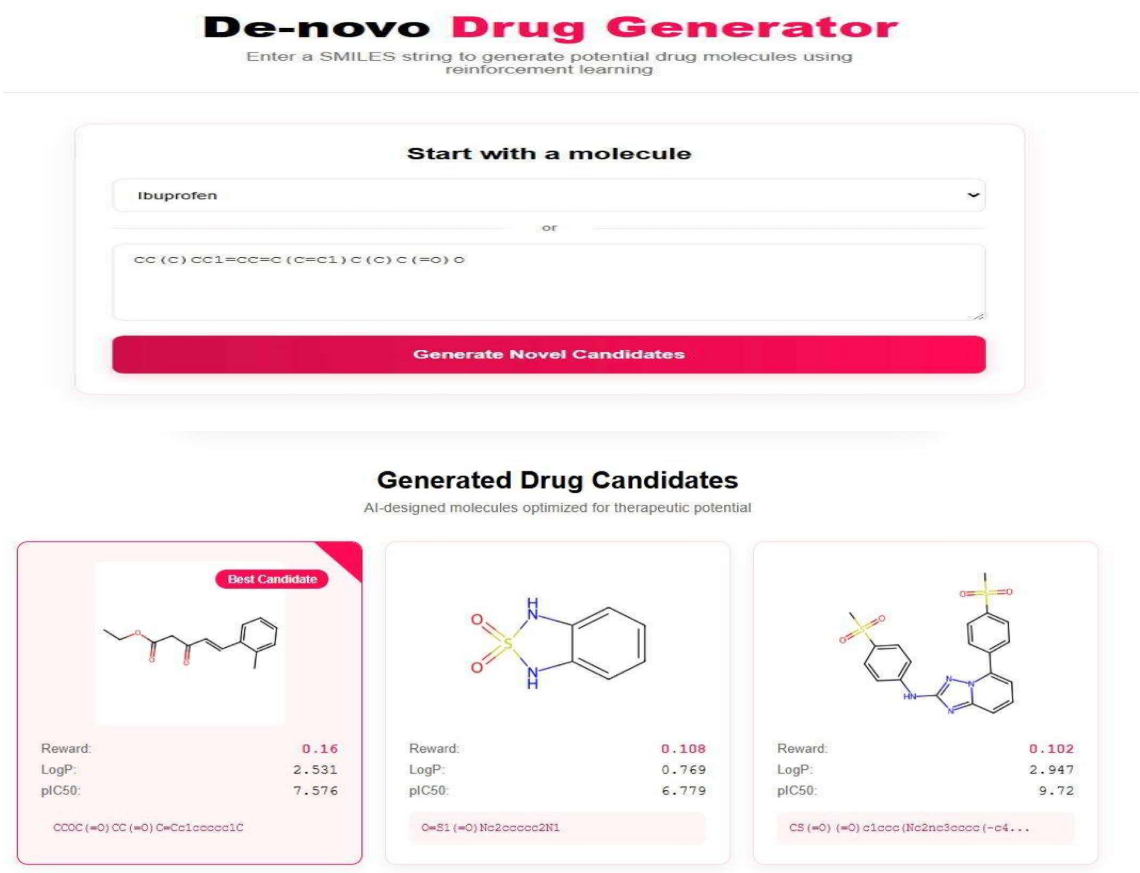
GENERATIVE MODEL

Layer	Param #
stack_controls_layer.weight	3000
stack_controls_layer.bias	3
stack_input_layer.weight	250000
stack_input_layer.bias	250
encoder.weight	45000
rnn.weight_ih_l0	3750000
rnn.weight_hh_l0	3000000
rnn.bias_ih_l0	3000
rnn.bias_hh_l0	3000
decoder.weight	45000
decoder.bias	45
Total parameters	7099298
Trainable parameters	7099298
Non-trainable parameters	0

PREDICTIVE MODEL

Layer	Param #
stack_controls_layer.weight	3000
stack_controls_layer.bias	3
stack_input_layer.weight	250000
stack_input_layer.bias	250
encoder.weight	45000
rnn.weight_ih_l0	3750000
rnn.weight_hh_l0	3000000
rnn.bias_ih_l0	3000
rnn.bias_hh_l0	3000
decoder.weight	45000
decoder.bias	45
Total parameters	7099298
Trainable parameters	7099298
Non-trainable parameters	0

SNIPPETS FROM WEBSITE



5. Federated Learning for De-novo drug design

Federated learning is used to train machine learning models collaboratively across decentralized devices or servers. While keeping data localized, ensuring privacy and reducing the need for centralized data storage.

```
Weight shapes for Client 1:
Layer 0 - Weights: shape = (2048, 32), size = 655
   First element: 0.0352933481335639995
Layer 0 - Biases: shape = (32,), size = 32
   First element: 0.0028766272589564323
Layer 1 - Weights: shape = (32, 32), size = 1024
   First element: -0.08521974086761475
Layer 1 - Biases: shape = (32,), size = 32
   First element: 0.00015512874233536422
Layer 2 - Weights: shape = (32, 16), size = 512
   First element: 0.026233581826090813
Layer 2 - Biases: shape = (16,), size = 16
   First element: 0.00022071160492487252
Layer 3 - Weights: shape = (16, 12869), size = 20
   First element: 0.007504452019929886
Layer 3 - Biases: shape = (12869,), size = 12869
   First element: -0.002588738687336445
```

```
Weight shapes for Client 2:
Layer 0 - Weights: shape = (2048, 32), size = 65536
   First element: 0.034680120646953358
Layer 0 - Biases: shape = (32,), size = 32
   First element: 0.0030924032907932997
Layer 1 - Weights: shape = (32, 32), size = 1024
   First element: -0.08521974086761475
Layer 1 - Biases: shape = (32,), size = 32
   First element: 0.00015512874233536422
Layer 2 - Weights: shape = (32, 16), size = 512
   First element: 0.026233581826090813
Layer 2 - Biases: shape = (16,), size = 16
   First element: 0.00022071160492487252
Layer 3 - Weights: shape = (16, 12869), size = 205904
   First element: 0.007504452019929886
Layer 3 - Biases: shape = (12869,), size = 12869
   First element: -0.002447094302624464
```

```
Weight shapes for Client 3:
Layer 0 - Weights: shape = (2048, 32), size = 65536
   First element: 0.033315811306238174
Layer 0 - Biases: shape = (32,), size = 32
   First element: -5.131165016791783e-05
Layer 1 - Weights: shape = (32, 32), size = 1024
   First element: -0.08679719269275665
Layer 1 - Biases: shape = (32,), size = 32
   First element: -0.0004575694038067013
Layer 2 - Weights: shape = (32, 16), size = 512
   First element: 0.026233581826090813
Layer 2 - Biases: shape = (16,), size = 16
   First element: 0.00022071160492487252
Layer 3 - Weights: shape = (16, 12869), size = 205904
   First element: 0.007504452019929886
Layer 3 - Biases: shape = (12869,), size = 12869
   First element: -0.002556709572672844
```

```
Weight shapes for Averaged Global Model:
Layer 0 - Weights: shape = (2048, 32), size = 65536
   First element: 0.03442975878715515
Layer 0 - Biases: shape = (32,), size = 32
   First element: 0.001972573110833764
Layer 1 - Weights: shape = (32, 32), size = 1024
   First element: -0.08574555069208145
Layer 1 - Biases: shape = (32,), size = 32
   First element: -4.9103971832664683e-05
Layer 2 - Weights: shape = (32, 16), size = 512
   First element: 0.026233581826090813
Layer 2 - Biases: shape = (16,), size = 16
   First element: 0.00022071161947678775
Layer 3 - Weights: shape = (16, 12869), size = 205904
   First element: 0.007504452019929886
Layer 3 - Biases: shape = (12869,), size = 12869
   First element: -0.002530847443267703
```

```
global_model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dense_8 (Dense)	(None, 32)	65,568
dense_9 (Dense)	(None, 32)	1,056
dense_10 (Dense)	(None, 16)	528
dense_11 (Dense)	(None, 12869)	218,773

Total params: 285,925 (1.09 MB)
Trainable params: 285,925 (1.09 MB)
Non-trainable params: 0 (0.00 B)

6. SMILES Masking Prediction

Aim:

Enable prediction of masked tokens using attention-based contextual understanding.

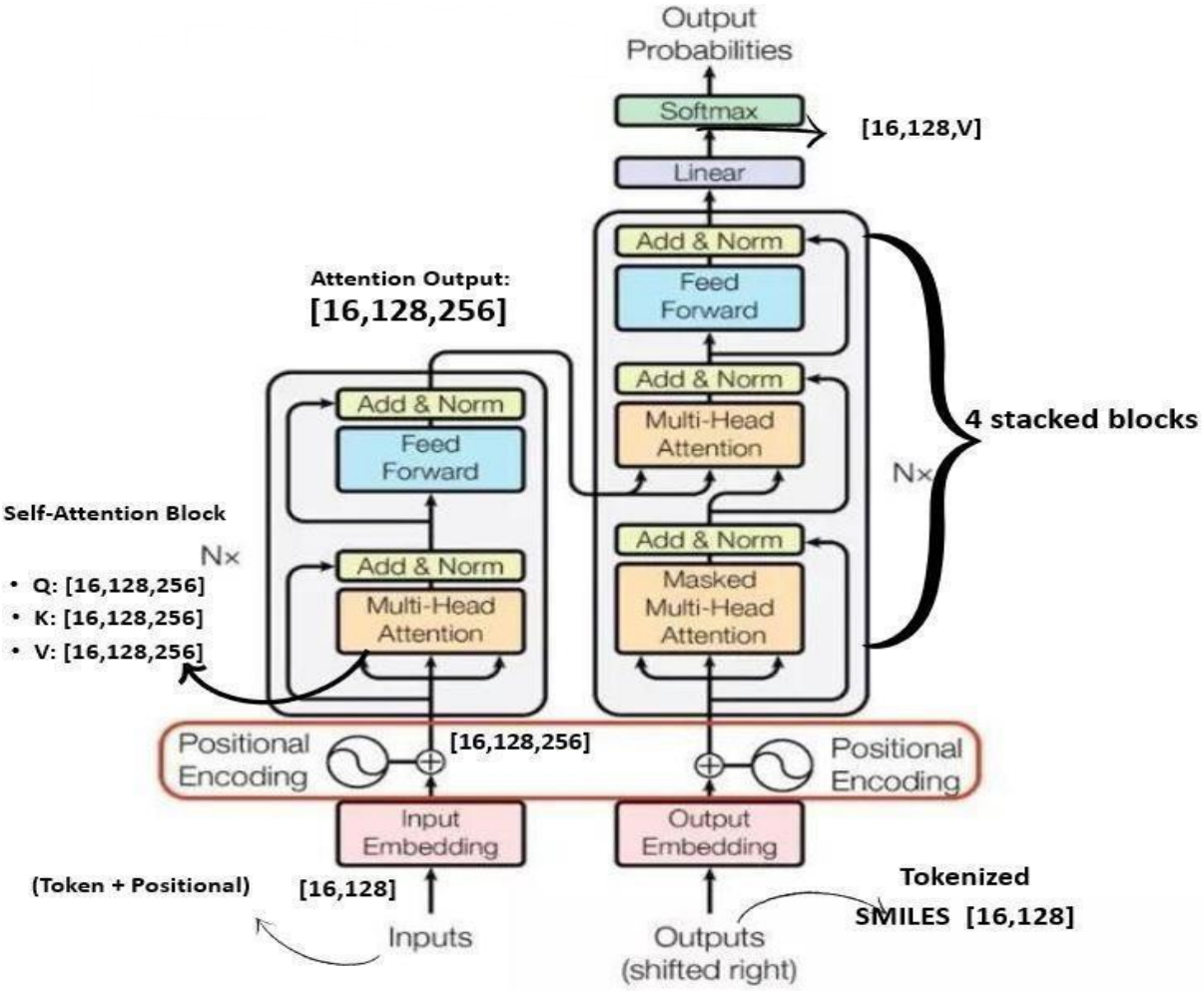
Data Description:

Source: 10,000 SMILES strings from jak2_10k_smiles.txt

Used: 500+ unique SMILES (subset for faster training)

Split: 80% training (~400), 20% testing (~100)

Architecture:



Model Summary:

```
RoBERTaForMaskedLM(
 (embedding): RoBERTaEmbedding(
  (token_embed): Embedding(num_embeddings=V, embedding_dim=256)
  (position_embed): Embedding(num_embeddings=128, embedding_dim=256)
  (norm): LayerNorm(normalized_shape=256)
  (dropout): Dropout(p=0.1)
 )
 (encoder): Sequential(
  (0): CustomTransformerBlock(
   (attn): MultiHeadSelfAttention(
    (q_proj): Linear(in_features=256, out_features=256)
    (k_proj): Linear(in_features=256, out_features=256)
    (v_proj): Linear(in_features=256, out_features=256)
    (out_proj): Linear(in_features=256, out_features=256)
    (dropout): Dropout(p=0.1)
   )
   (norm1): LayerNorm(normalized_shape=256)
   (ff): Sequential(
    (0): Linear(in_features=256, out_features=512)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=256)
   )
   (norm2): LayerNorm(normalized_shape=256)
   (dropout): Dropout(p=0.1)
  )
  ... (Repeated 3 more times for total of 4 transformer blocks)
 )
 (lm_head): Linear(in_features=256, out_features=V)
)
```

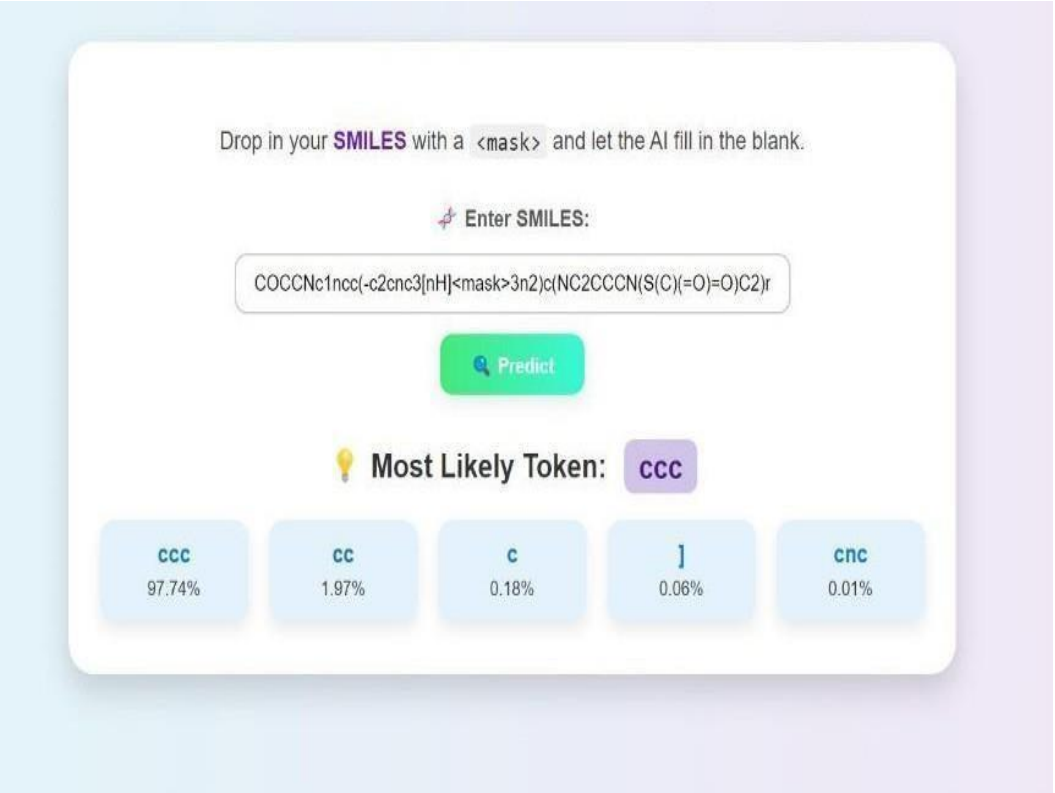
Model Accuracy:

Final Training Accuracy: 9.32%

Val loss: 0.0161

Val Accuracy:99.48%

Snippet from Website:



7. Active or Inactive Ligand Identification for Alzheimer disease – using VIT

Aim:

Identify Active or Inactive ligand for Alzheimer disease using Gen AI Vision Transformer technology.

Dataset description:

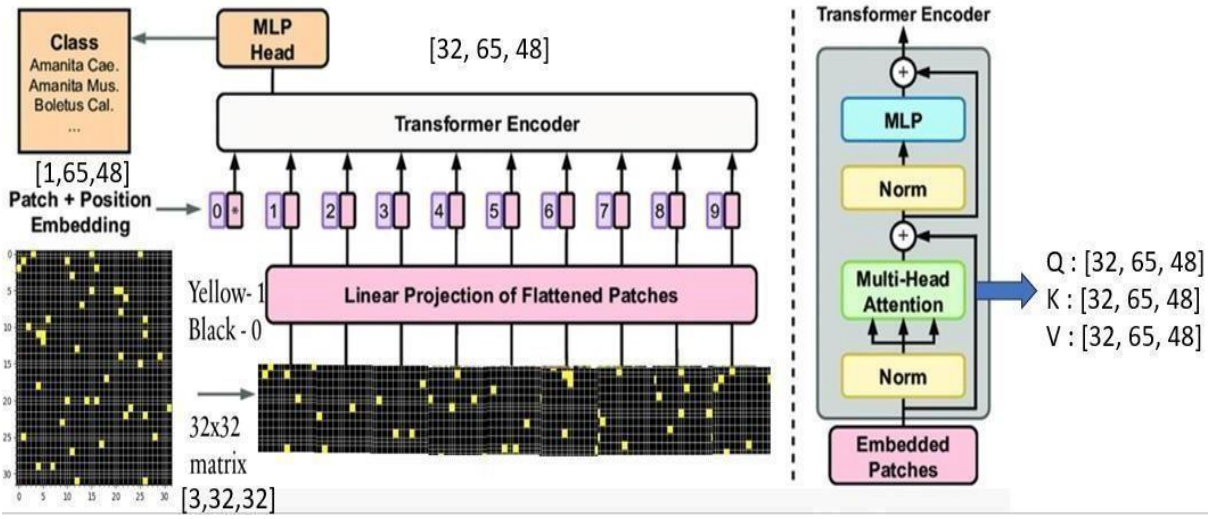
The Dataset being used to train the Vision Transformer model consists of The Canonical Smiles, Class and Morgan Fingerprints. The dataset consists of 4695 smiles out of which 3756 are for training and 939 testing.

Dataset Link:

https://www.ebi.ac.uk/chembl/explore/targets/eyJkYXRhc2V0Ijp7ImVudGl0eUlEIjoiVGFyZ2V0IiwiaW5pdGlhbFF1ZXJ5Ijp7fSwiZmFjZXRzU3RhdG UiOm51bGwsImN1c3RvbUZpbHRlcmluZyI6bnVsbCwicXVpY2tTZWFyY2hUZXJtIjoiYWNldHlsY2hvbGluZXN0ZXJhc2UiLCJleGFjdFRleHRGaWx 0ZXJzIjp7fX19

canonical_ class	MW	LogP	NumHDon	NumHAcc	0	1	2	3	4	5
CCOc1nn(-	active	312.325	2.8032	0	6	0	0	0	1	0	0
O=C(N1CC	active	376.913	4.5546	0	5	1	0	1	0	1	0
CN(C(=O)n	inactive	426.851	5.3574	0	5	0	0	0	0	0	0
O=C(N1CC	active	404.845	4.7069	0	5	0	0	1	0	1	0
CSc1nc(-c	active	346.334	3.0953	0	6	0	0	0	0	0	0
CSc1nc(-c	intermedia	338.436	4.07992	0	5	0	0	0	0	0	0
CSc1nc(-c	active	296.783	2.8501	0	5	0	0	0	0	0	0
CCCCCCSc	inactive	408.955	4.5712	0	6	0	0	0	0	0	0
COc1ccc(-	active	292.364	2.2053	0	6	0	0	0	0	0	0

Model Architecture:



Model Summary:

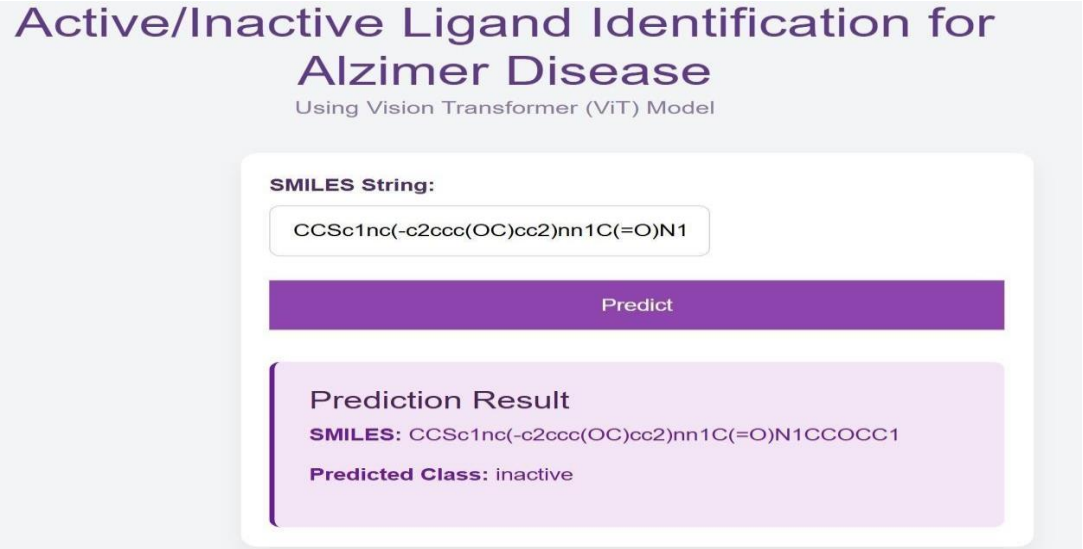
```
==========================================================================================
Layer (type:depth-idx)                                  Output Shape              Param #
==========================================================================================
ViT                                                     [1, 3]                    3,168
├─PatchEmbedding: 1-1                                   [1, 64, 48]               --
│    └─Conv2d: 2-1                                      [1, 48, 8, 8]             2,352
│    └─Flatten: 2-2                                     [1, 48, 64]               --
├─Dropout: 1-2                                          [1, 65, 48]               --
├─ModuleList: 1-3                                       --                        --
│    └─TransformerEncoderBlock: 2-3                     [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-1            [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-2                               [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-4                     [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-3            [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-4                               [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-5                     [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-5            [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-6                               [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-6                     [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-7            [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-8                               [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-7                     [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-9            [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-10                              [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-8                     [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-11           [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-12                              [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-9                     [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-13           [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-14                              [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-10                    [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-15           [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-16                              [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-11                    [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-17           [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-18                              [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-12                    [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-19           [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-20                              [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-13                    [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-21           [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-22                              [1, 65, 48]               298,128
│    └─TransformerEncoderBlock: 2-14                    [1, 65, 48]               --
│    │    └─MultiheadSelfAttentionBlock: 3-23           [1, 65, 48]               9,504
│    │    └─MLPBlock: 3-24                              [1, 65, 48]               298,128
├─Sequential: 1-4                                       [1, 3]                    --
│    └─LayerNorm: 2-15                                  [1, 48]                   96
│    └─Linear: 2-16                                     [1, 3]                    147
==========================================================================================
Total params: 3,697,347
Trainable params: 3,697,347
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 3.73
==========================================================================================
Input size (MB): 0.01
Forward/backward pass size (MB): 20.09
Params size (MB): 14.33
Estimated Total Size (MB): 34.43
```

Model Accuracy:

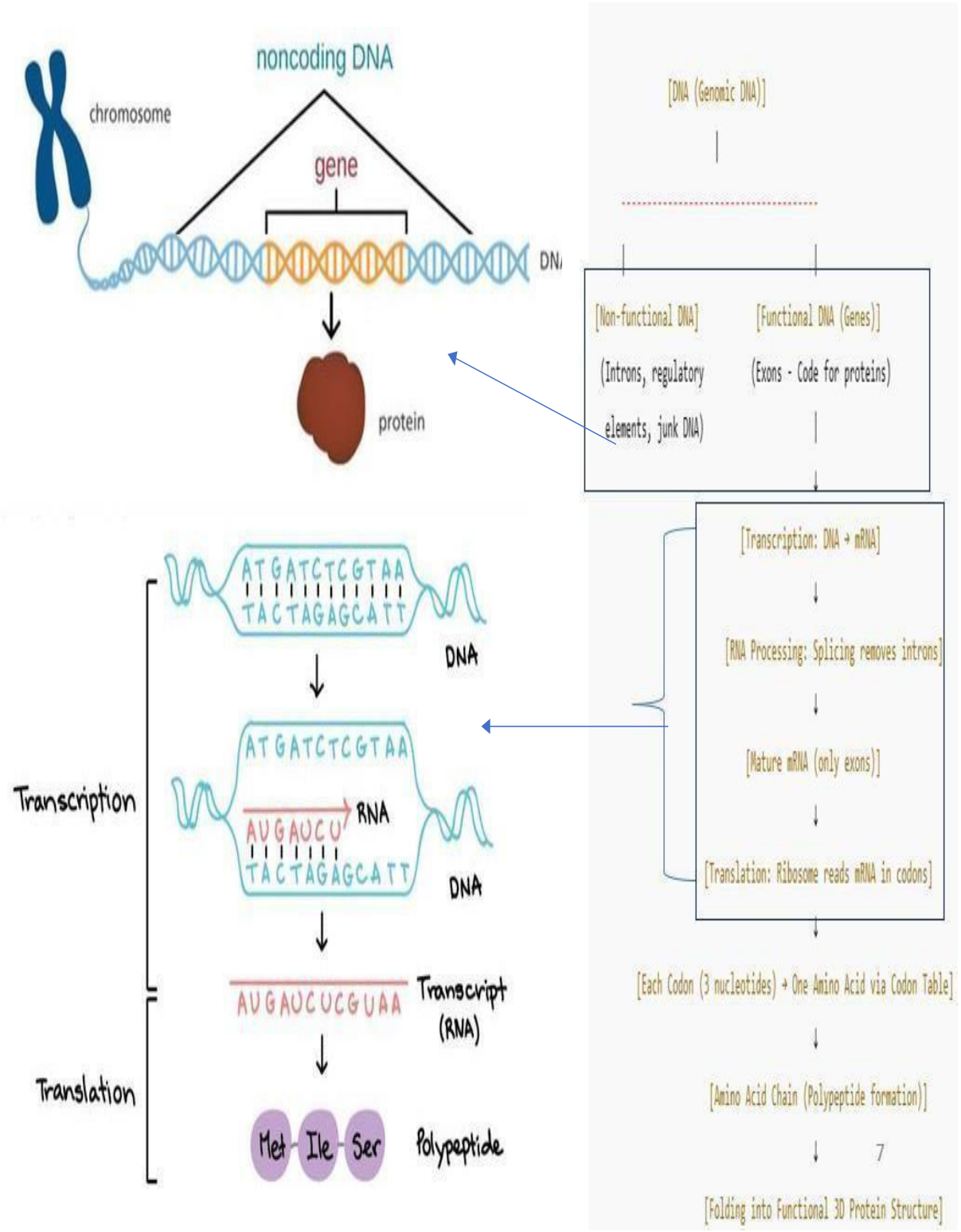
```
CUDA available: True
Device name: Tesla T4
[1/500] Loss: 1.0877 | Accuracy: 41.48%
[2/500] Loss: 1.0772 | Accuracy: 43.21%
[3/500] Loss: 1.0772 | Accuracy: 42.49%
[4/500] Loss: 1.0757 | Accuracy: 43.48%
[5/500] Loss: 1.0738 | Accuracy: 43.40%
[6/500] Loss: 1.0736 | Accuracy: 43.37%
[7/500] Loss: 1.0757 | Accuracy: 43.34%
[8/500] Loss: 1.0734 | Accuracy: 43.42%
[9/500] Loss: 1.0723 | Accuracy: 43.08%
[10/500] Loss: 1.0709 | Accuracy: 43.56%
```

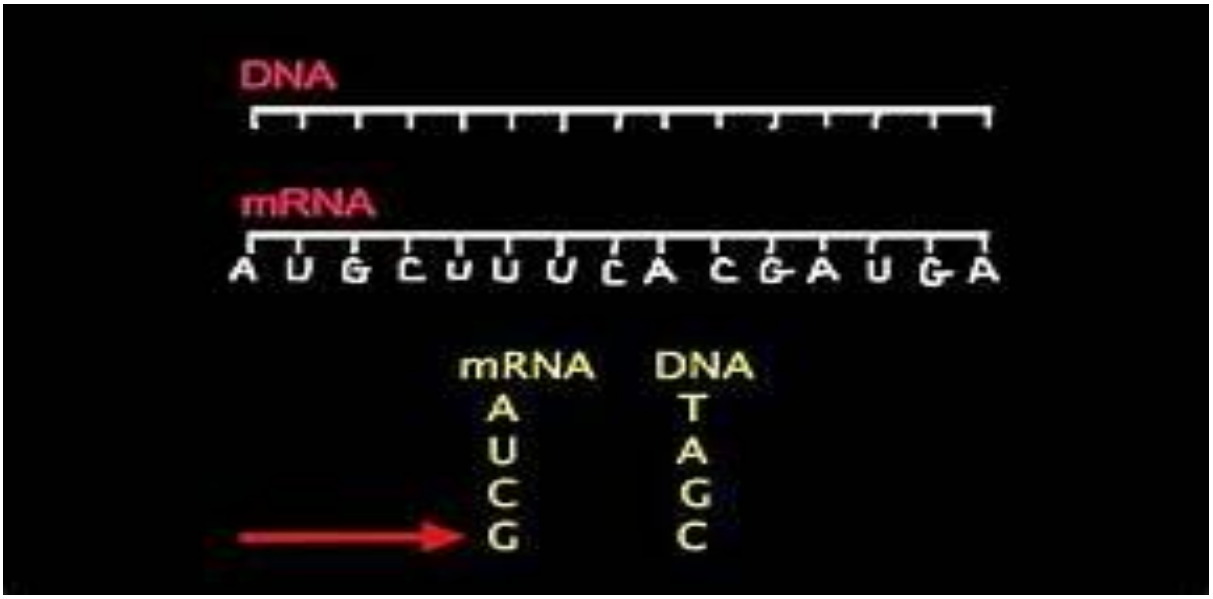
```
[433/500] Loss: 0.2449 | Accuracy: 90.63%
[434/500] Loss: 0.2416 | Accuracy: 90.55%
[435/500] Loss: 0.2454 | Accuracy: 90.39%
[436/500] Loss: 0.2342 | Accuracy: 90.73%
[437/500] Loss: 0.2413 | Accuracy: 90.42%
[438/500] Loss: 0.2351 | Accuracy: 90.89%
[439/500] Loss: 0.2393 | Accuracy: 90.47%
[440/500] Loss: 0.2454 | Accuracy: 90.79%
[441/500] Loss: 0.2458 | Accuracy: 89.75%
[442/500] Loss: 0.2327 | Accuracy: 91.24%
Early stopping.
```

Snippet from Website:



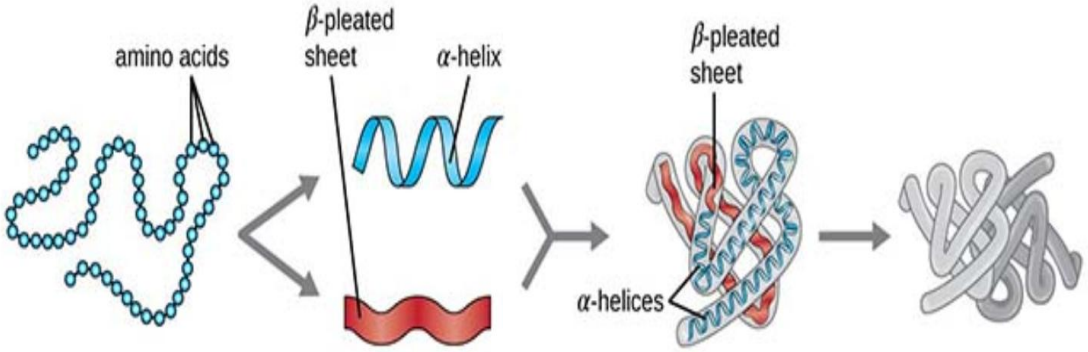
8. Nucleotides and Amino acids





Amino Acid	Codons
A (Ala)	GCT, GCC, GCA, GCG
C (Cys)	TGT, TGC
D (Asp)	GAT, GAC
E (Glu)	GAA, GAG
F (Phe)	TTT, TTC
G (Gly)	GGT, GGC, GGA, GGG
H (His)	CAT, CAC
I (Ile)	ATT, ATC, ATA
K (Lys)	AAA, AAG
L (Leu)	TTA, TTG, CTT, CTC, CTA, CTG
M (Met)	ATG
N (Asn)	AAT, AAC
P (Pro)	CCT, CCC, CCA, CCG
Q (Gln)	CAA, CAG
R (Arg)	CGT, CGC, CGA, CGG, AGA, AGG
S (Ser)	TCT, TCC, TCA, TCG, AGT, AGC
T (Thr)	ACT, ACC, ACA, ACG
V (Val)	GTT, GTC, GTA, GTG
W (Trp)	TGG
Y (Tyr)	TAT, TAC

Structure Of Proteins



Primary Protein Structure
Sequence of a chain of amino acids

Secondary Protein Structure
Local folding of the polypeptide chain into helices or sheets

Tertiary Protein Structure
three-dimensional folding pattern of a protein due to side chain interactions

Quaternary Protein Structure
protein consisting of more than one amino acid chain

9. Codon sequences in GenBank

GenBank

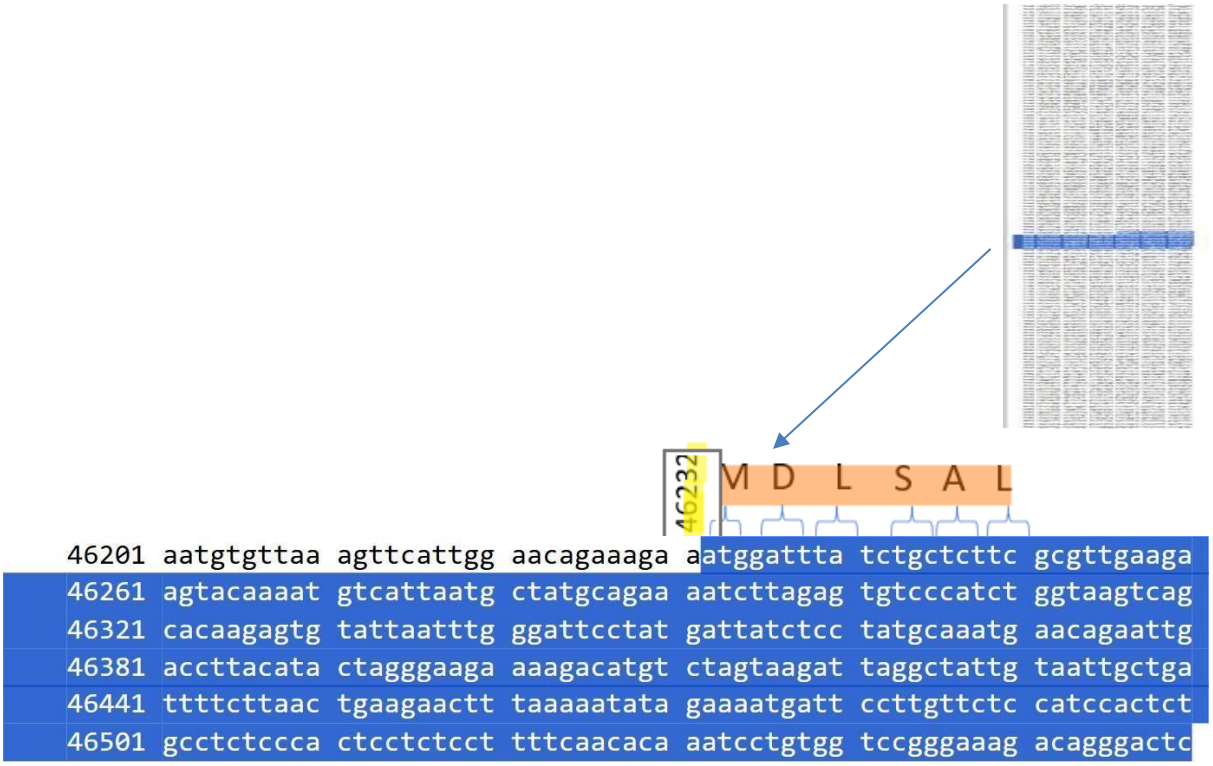
Homo sapiens chromosome 17, GRCh38.p14 Primary Assembly

NCBI Reference Sequence: NC_000017.11

```
gene            1..126033
                /gene="BRCA1"
                /gene_synonym="BRCAI; BRCC1; BROVCA1; FANCS; IRIS; PNCA4;
                PPP1R53; PSCP; RNF53"
                /note="BRCA1 DNA repair associated; Derived by automated
                computational analysis using gene prediction method:
                BestRefSeq."
                /db_xref="GeneID:672"
                /db_xref="HGNC:HGNC:1100"
                /db_xref="MIM:113705"

mRNA            join(44964..45051,75468..78893,79296..79384,87753..87924,
                93717..93840,95807..95997,99090..99400,102633..102720,
                106377..106454,106955..106995,113193..113276,
                119211..119265,121134..121207,122625..122685,
                124526..126033)
                /gene="BRCA1"
                /gene_synonym="BRCAI; BRCC1; BROVCA1; FANCS; IRIS; PNCA4;
                PPP1R53; PSCP; RNF53"
                /product="BRCA1 DNA repair associated, transcript variant
                233"
                /note="Derived by automated computational analysis using

CDS             join(46232..46311,54549..54602,65372..65460,66067..66206,
                70448..70553,73039..73084,74406..74482,75468..78893,
                79296..79384,87756..87924,93717..93840,95807..95997,
                99090..99400,102633..102720,106377..106454,106955..106995,
                113193..113276,119211..119265,121134..121207,
                122625..122685,124526..124650)
                /gene="BRCA1"
                /gene_synonym="BRCAI; BRCC1; BROVCA1; FANCS; IRIS; PNCA4;
                PPP1R53; PSCP; RNF53"
                /note="isoform 93 is encoded by transcript variant 72;
                Derived by automated computational analysis using gene
                prediction method: BestRefSeq."
                /codon_start=1
                /product="breast cancer type 1 susceptibility protein
                isoform 93"
                /protein_id="NP_001394592.1"
                /db_xref="GeneID:672"
                /db_xref="HGNC:HGNC:1100"
                /db_xref="MIM:113705"
                /translation="MDLSALRVEEVQNVINAMQKILECPICLELIKEPVSTKCDHIFC
                KSLQESTRFSQLVEELLKIICAFQLDTGLEYANSYNFAKKENNSPEHLKDEVSIIQSM
                GYRNRAKRLLQSEPENPSLQETSLSVQLSNLGTVRTLRTKQRIQPQKTSVYIELGSDS
```


46232
M D L S A L
46201 aatgtgttaa agttcattgg aacagaaaga aatggattta tctgctcttc gcgttgaaga
46261 agtacaaaat gtcattaatg ctatgcagaa aatcttagag tgtcccatct ggtaagtcag
46321 cacaagagtg tattaatttg ggattcctat gattatctcc tatgcaaatg aacagaattg
46381 accttacata ctagggaaga aaagacatgt ctagtaagat taggctattg taattgctga
46441 ttttcttaac tgaagaactt taaaaatata gaaaatgatt ccttgttctc catccactct
46501 gcctctccca ctcctctcct tttcaacaca aatcctgtgg tccgggaaag acagggactc

10. Gen – AI protein to smiles converter

Aim:

To convert protein sequences into SMILES (Simplified Molecular Input Line Entry System) strings for enhanced molecular representation and analysis.

Dataset Description:

The data is collected from Bindingdb, and can generate new molecules that are optimized for binding to a target protein.

Total Protein Sequences and its corresponding smiles strings :5017

Training data:4500 sequences

Testing data:517 sequences.

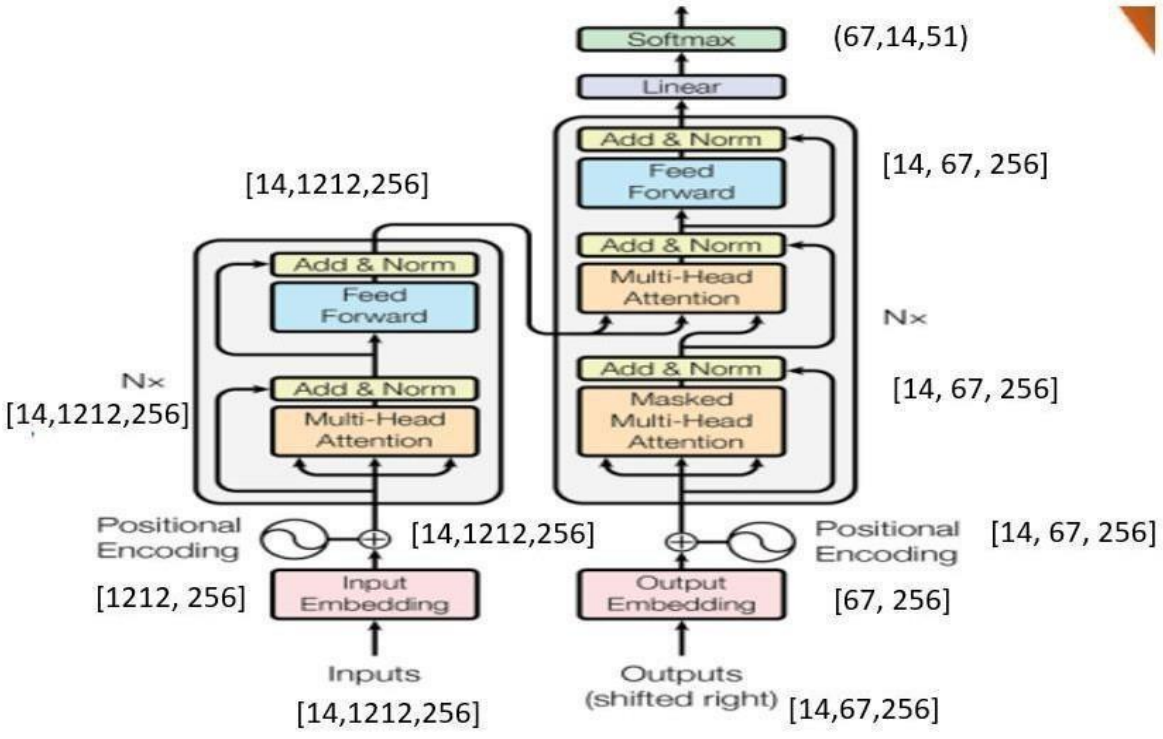
Protein Dataset

https://drive.google.com/file/d/1crds9AKCmp X3yW1Y8z2vvL3gE6Z_ecez/view

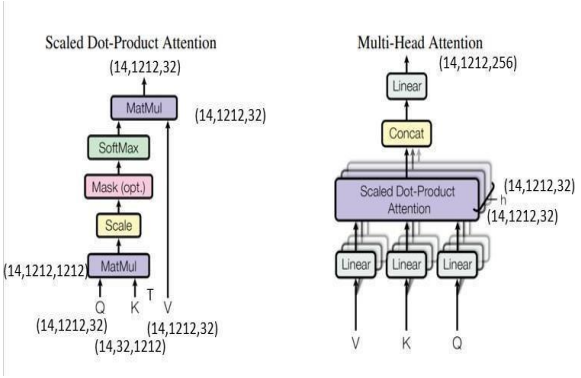
SMILES Dataset

https://drive.google.com/file/d/1urykQtrMGUYfF_ZTJ7j1iiIFr2a406Er/view

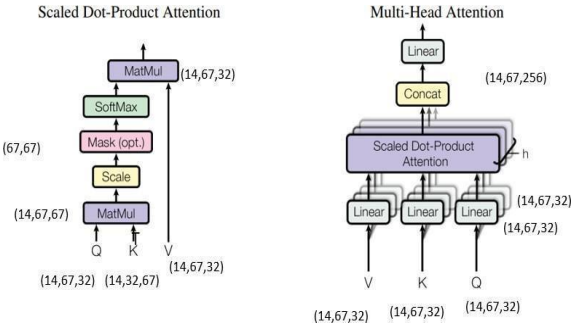
Transformer Architecture:



Multi-Head Attention in Encoder



Multi-Head Attention in Decoder



Model Summary:

```
print(model)

Transformer(
  (positional_encoder): PositionalEncoding(
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (input_embedding): Embedding(24, 128)
  (output_embedding): Embedding(51, 128)
  (transformer): Transformer(
    (encoder): TransformerEncoder(
      (layers): ModuleList(
        (0-2): 3 x TransformerEncoderLayer(
          (self_attn): MultiheadAttention(
            (out_proj): NonDynamicallyQuantizableLinear(in_features=128, out_features=128, bias=True)
          )
          (linear1): Linear(in_features=128, out_features=2048, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
          (linear2): Linear(in_features=2048, out_features=128, bias=True)
          (norm1): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
          (norm2): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
          (dropout1): Dropout(p=0.1, inplace=False)
          (dropout2): Dropout(p=0.1, inplace=False)
        )
      )
      (norm): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
    )
    (decoder): TransformerDecoder(
      (layers): ModuleList(
        (0-2): 3 x TransformerDecoderLayer(
          (self_attn): MultiheadAttention(
            (out_proj): NonDynamicallyQuantizableLinear(in_features=128, out_features=128, bias=True)
          )
          (multihead_attn): MultiheadAttention(
            (out_proj): NonDynamicallyQuantizableLinear(in_features=128, out_features=128, bias=True)
          )
          (linear1): Linear(in_features=128, out_features=2048, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
          (linear2): Linear(in_features=2048, out_features=128, bias=True)
          (norm1): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
          (norm2): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
          (norm3): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
          (dropout1): Dropout(p=0.1, inplace=False)
          (dropout2): Dropout(p=0.1, inplace=False)
          (dropout3): Dropout(p=0.1, inplace=False)
        )
      )
      (norm): LayerNorm((128,), eps=1e-05, elementwise_affine=True)
    )
  )
  (out): Linear(in_features=128, out_features=51, bias=True)
)
```

Model Accuracy:

```
Epoch 230/300
Train Loss: 0.3707 | Train acc: 0.8623 | Val Loss: 0.3207
Epoch 231/300
Train Loss: 0.3721 | Train acc: 0.8618 | Val Loss: 0.3502
Epoch 232/300
Train Loss: 0.3720 | Train acc: 0.8620 | Val Loss: 0.3395
Epoch 233/300
Train Loss: 0.3707 | Train acc: 0.8627 | Val Loss: 0.3298
```

Snippet from Website:

Conversion Results ← Convert Another Protein

Input Protein

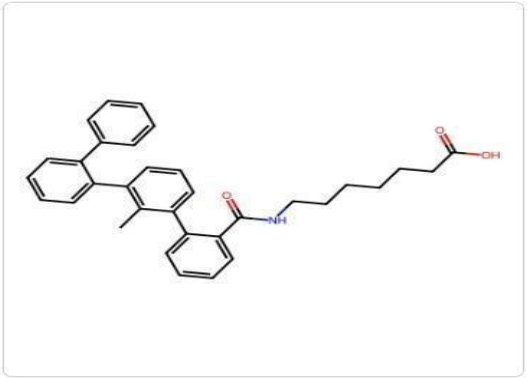
Human Insulin

GIVEQCCTSICSLYQLENYCNFVNQHLCGSHLVEALYLVCGERGFFYTPKT

SMILES Notation Copy

Cc1c(-c2ccccc2-c2ccccc2)cccc1-c1ccccc1C(=O)NCCCCCCC(=O)O

Molecular Structure Download Regenerate



11. Protein 3d structure visualization

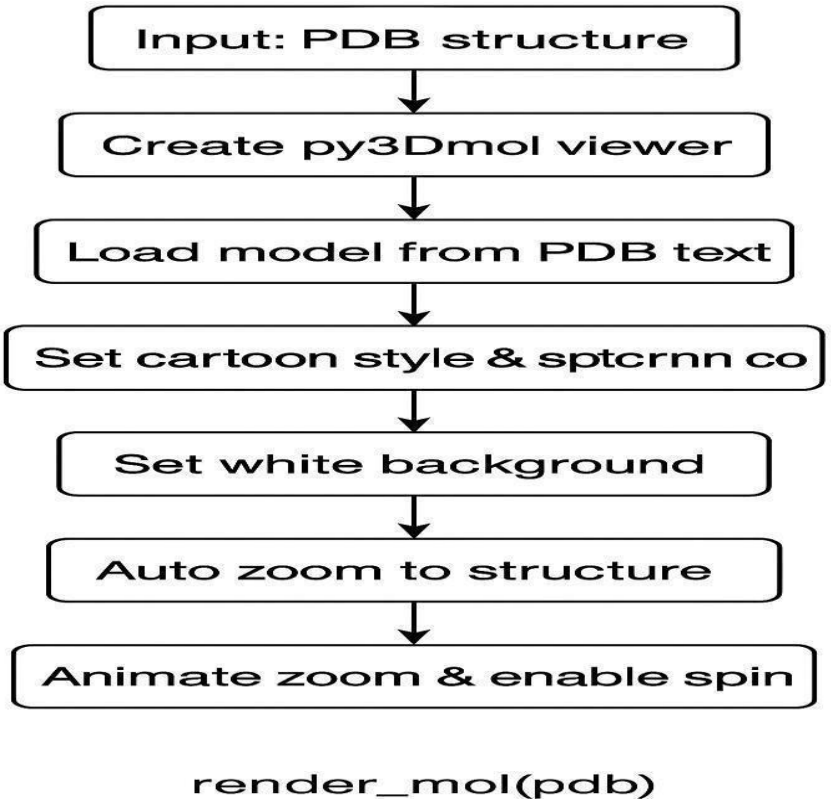
Aim:

To visualize protein 3d structure using esmfold.

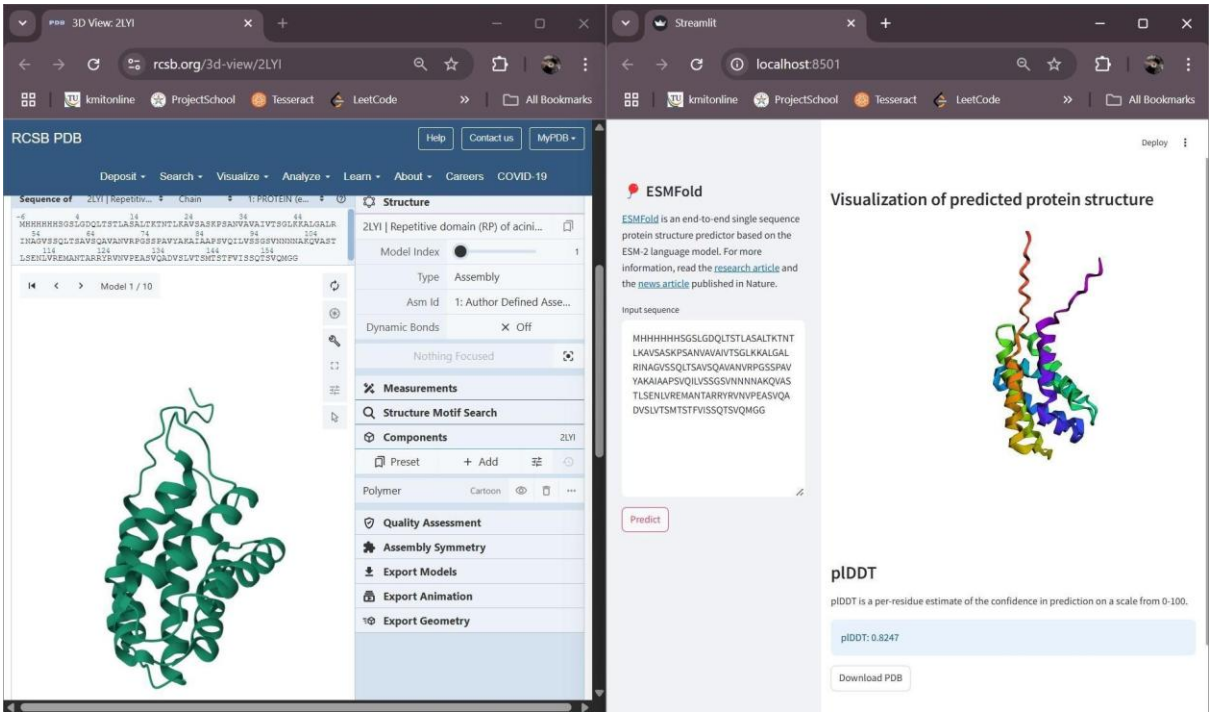
Dataset description:

PDB, NCBI, BindingDB, Genbank

Render mol function architecture for protein 3d structure visualization:



Side by side comparison of the RCSB PDB prediction with our model:

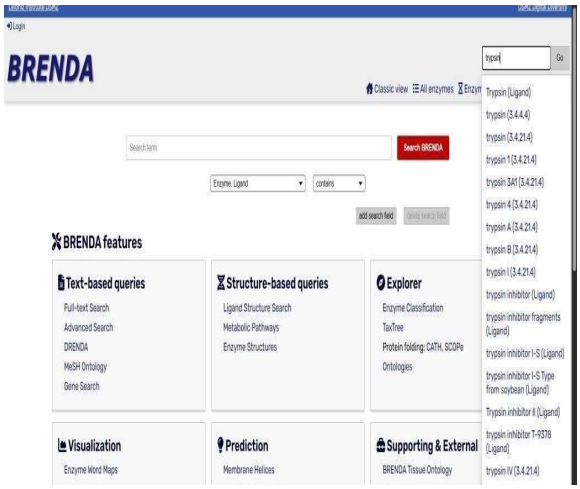


12 . AUTODOCKING VISUALIZATION

AutoDock Vina :

- AutoDocking is a computational technique to predict ligand binding to a target protein.
- AutoDock Vina is a fast, open-source docking software with improved accuracy
- The EC number stands for Enzyme Commission number, which is a numerical classification system for enzymes based on the chemical reactions they catalyze.
- Each number provides specific information about the enzyme's function(EC 1.1.1.1).
- A curated database containing functional and molecular data on enzymes classified by the EC number.
- Developed and maintained by the Braunschweig University of Technology, Germany.
- To get the ligand ID of a known SMILES notation of areceptor or ligand we can serach in

http://ligand-expo.rcsb.org/ld-search.html



Workflow of AutoDock Vina :

1. Prepare Receptor and Ligand (convert to PDBQT).
2. Define Grid Box (center & size).
3. Run Docking with Vina (CLI).
4. Analyze binding modes and affinity (output poses) and visualization.

Integration Workflow :

- Target Identification — Use EC numbers to select disease-relevant enzymes .
- Structure Retrieval — Fetch EC-tagged PDB structures
- Ligand Filtering — Query ChEMBL/PubChem for known EC ligands
- Docking Config — Customize docking (site residues, cofactors)

Tools/Resources :

- PDB, ChEMBL
- Docking tools: AutoDock Vina

Visual Workflow diagram with arrows :

EC Number → Protein Structure → Ligand Selection → Docking → Hit entification

Visualizing Docking with AutoDock Vina :

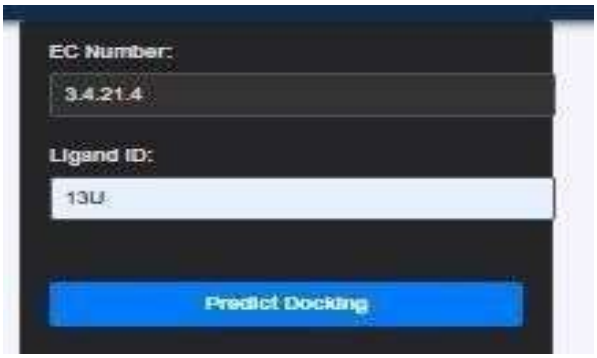
- Visualization Tools: PyMOL, Chimera, AutoDockTools.
- Ligand: Green/Magenta | Protein: Grey/Ribbon.
- Hydrogen bonds: Yellow dashed lines.

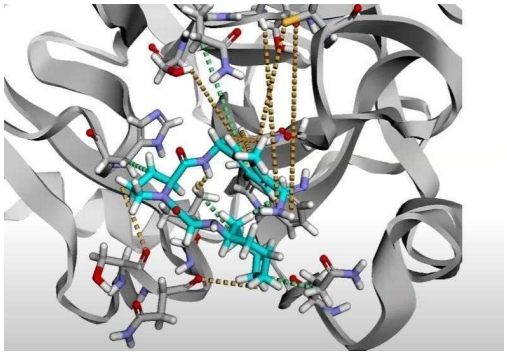
Color Map from AutoDock Vina

Carbon (C)	Grey / Black	Backbone of molecules
Hydrogen (H)	White	Often not shown explicitly
Oxygen (O)	Red	Electronegative atom
Nitrogen (N)	Blue	Common in amino groups
Sulfur (S)	Yellow	Found in cysteine, methionine
Phosphorus (P)	Orange	Common in DNA, ATP, etc.
Chlorine (Cl)	Green	Halogen atoms
Fluorine (F)	Light Green	Halogen
Bromine (Br)	Dark Red	Halogen
Iodine (I)	Dark Violet	Halogen
Metals (e.g., Fe, Zn)	Variable (e.g., rust or light purple)	Often customized

Color	Represents
Gr e/Magta	Ligand molecule (small molecule, drug candidate)
	Protein surface or backbone (non-highlighted)
	Protein secondary structure (α-helix, β-sheet)
	Hydrogen bonds (ligand ↔ protein interactions)
	Oxygen atoms (polar or reactive sites)
	Nitrogen atoms (common in H-bonds, active sites)
	Sulfur atoms (e.g., in cysteine)
	Sometimes used to distinguish another ligand po

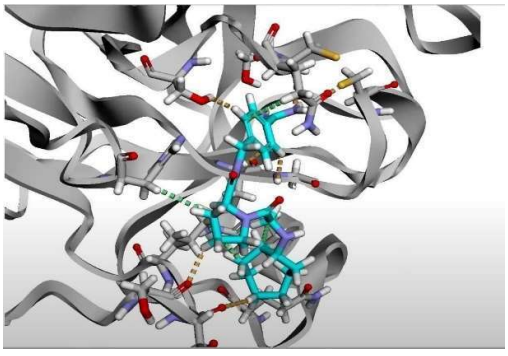
Website input and output





Energy Scores:

total	inter	intra	torsions	intra best pose
-7.809	-11.005	-1.103	4.299	-1.103
-7.049	-9.894	-1.142	3.968999999999999	-1.103
-6.929	-9.76	-1.106	3.939	-1.103
-6.259	-8.272	-1.651	3.8639999999999997	-1.103
-5.652	-6.926	-2.424	3.4979999999999993	-1.103



13 . WEBHOOKS

Using a webhook for caching means receiving data from an external service via a real-time HTTP POST request, and then storing that data locally (in a cache or database), so that future requests for the same data can be served quickly without recomputation.

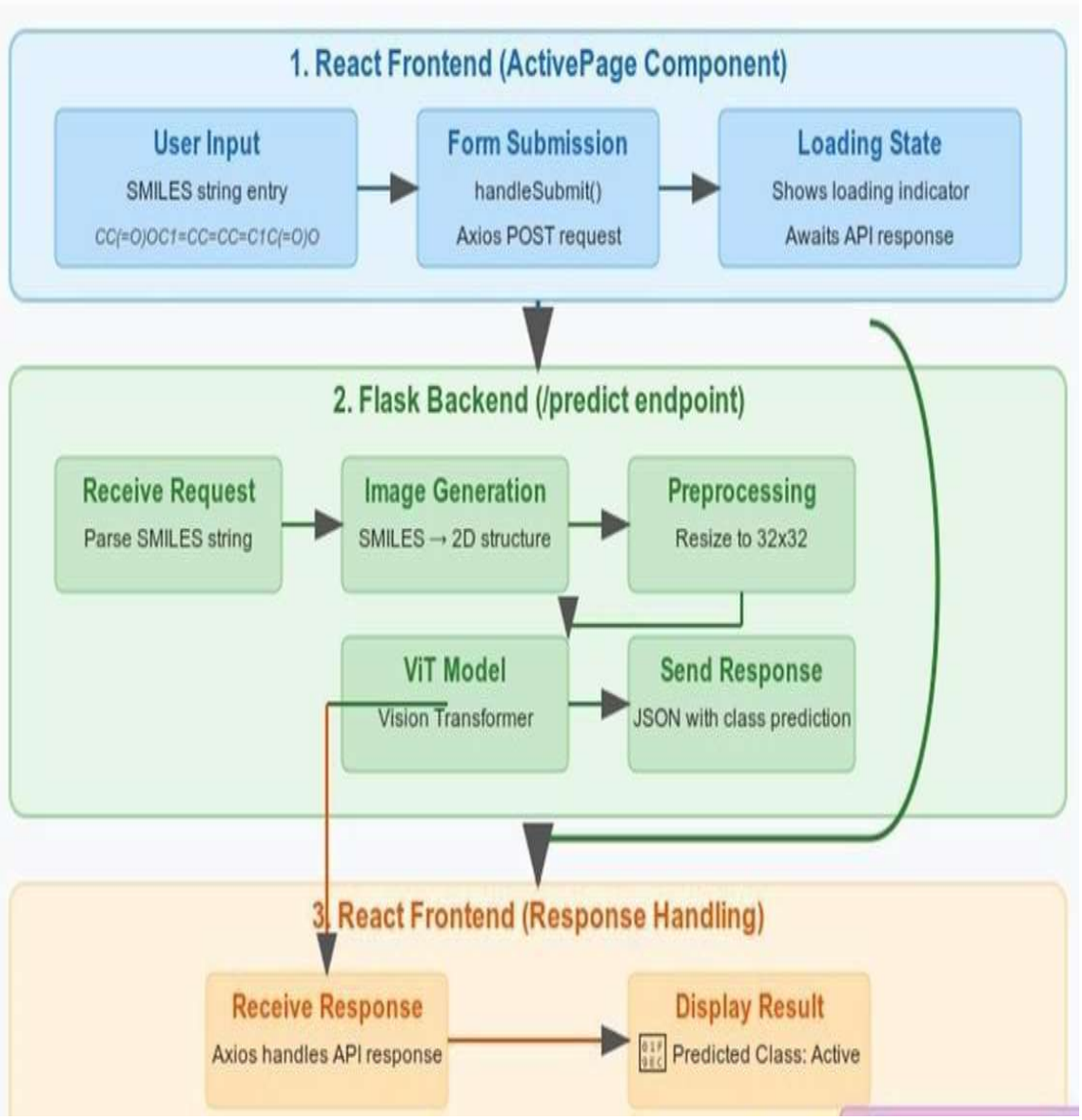
```
127.0.0.1 - - [05/May/2025 14:36:59] "OPTIONS /predict HTTP/1.1" 200 -
 Inserted into cache for SMILES: CC(C)CCCC(C)C(C)C1=CC(=CC(=C1)C(C)C)C(C)C(C)C
Current Cache: {'CN1C=NC2=C1C(=O)N(C(=O)N2C)C': 'O=C(CN1CCCCC1)Nc1ccc(Cl)c(Cl)c1', 'CC(=O)OC1=CC=CC=C1C(=O)O': 'CN(c1ccccc1CNc1cccn2nc
(Nc3ccccc3)nc12)S(C)(=O)=O', 'CCO': 'COc1ccc(COCc2cccc(CN3CCOCC3)c2)cc1', 'CC(C)CCCC(C)C(C)C1=CC(=CC(=C1)C(C)C)C(C)C(C)C': 'O=C1OCc2cc
ccc21'}
127.0.0.1 - - [05/May/2025 14:36:59] "POST /predict HTTP/1.1" 200 -
```

```
127.0.0.1 - - [05/May/2025 14:40:47] "OPTIONS /predict HTTP/1.1" 200 -
 Retrieved from cache for SMILES: CC(=O)OC1=CC=CC=C1C(=O)O
 Current Cache: {'CC(C)CCCC(C)C(C)C1=CC(=CC(=C1)C(C)C)C(C)C(C)C': 'O=C1OCc2ccccc21', 'CN1C=NC2=C1C(=O)N(C(=O)N2C)C': 'O=C(CN1CCCCC1)Nc
1ccc(Cl)c(Cl)c1', 'CC(=O)OC1=CC=CC=C1C(=O)O': 'CN(c1ccccc1CNc1cccn2nc(Nc3ccccc3)nc12)S(C)(=O)=O'}
127.0.0.1 - - [05/May/2025 14:40:47] "POST /predict HTTP/1.1" 200 -
```

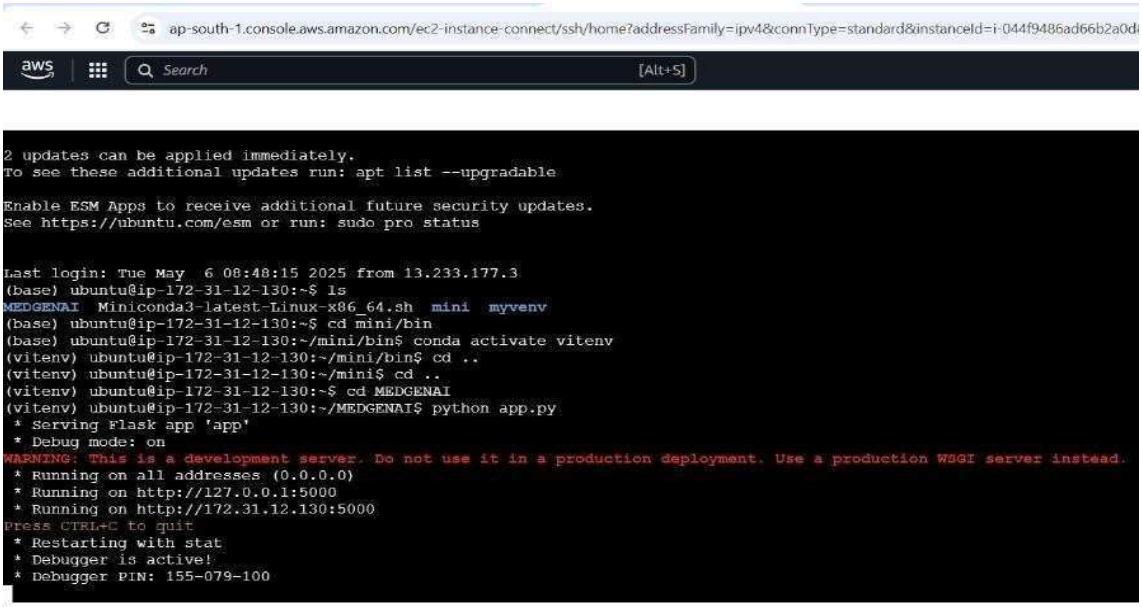
SNIPPETS FROM WEBSITE

14 . INTEGRATION WORKFLOW

ACTIVE/INACTIVE LIGAND IDENTIFICATION FOR ALZHEIMER DISEASE USING VIT MODEL

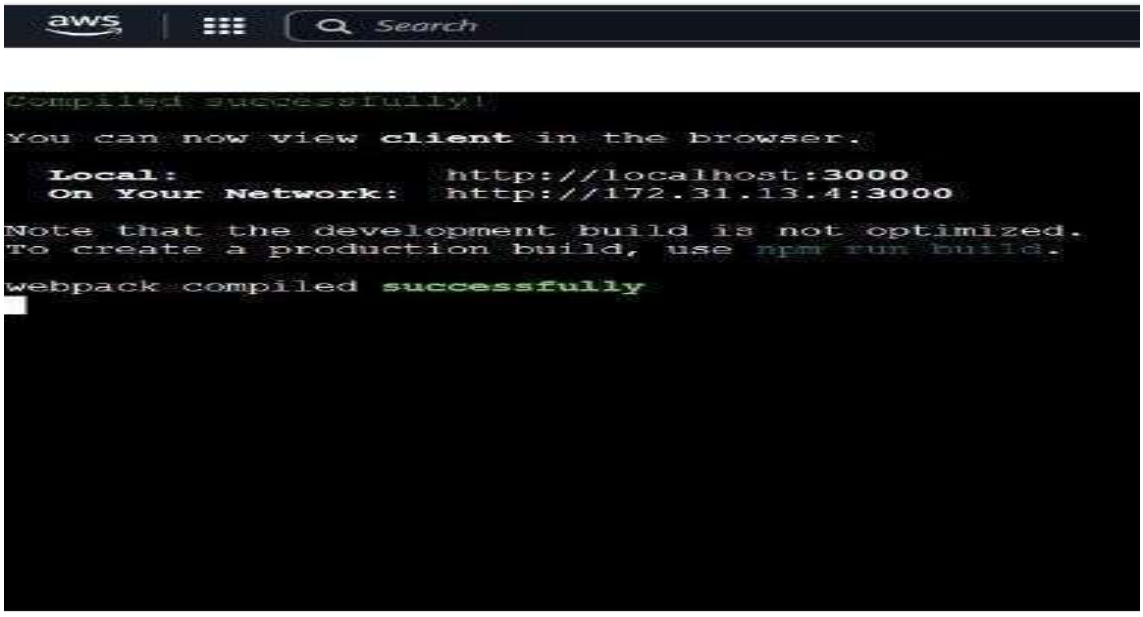


15 . AWS DEPLOYMENT



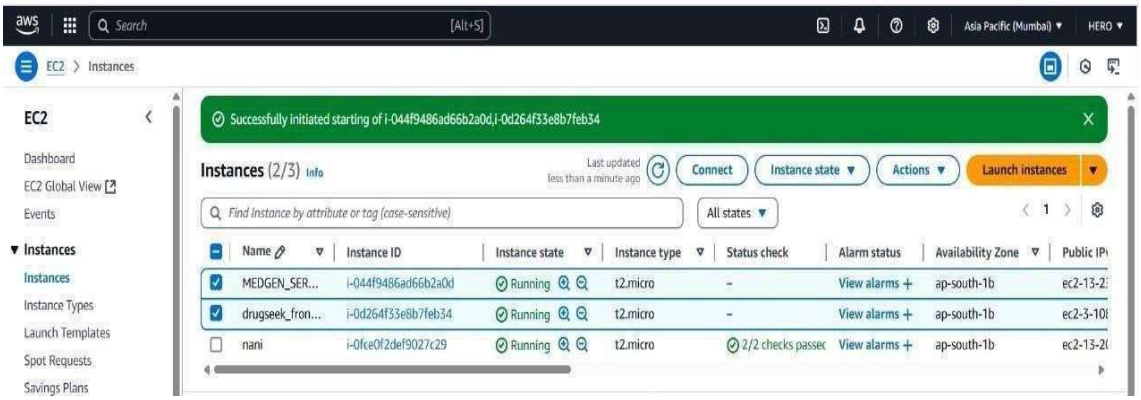
i-044f9486ad66b2a0d (MEDGEN_SERVER)

PublicIPs: 13.235.87.49 PrivateIPs: 172.31.12.130

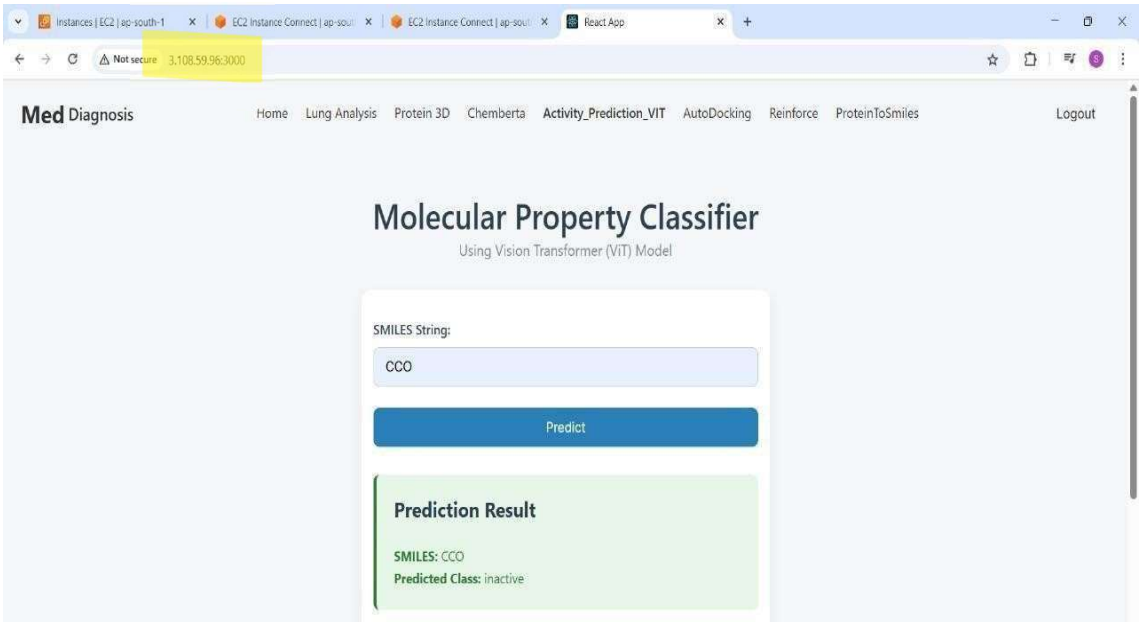


i-0d264f33e8b7feb34 (drugseek_frontend)

PublicIPs: 3.108.59.96 PrivateIPs: 172.31.13.4



VIT MODEL DEPLOYMENT



URL : 3.108.59.96:3000