Deep Learning for Drug Response Prediction

Project Overview

Objective

- Predict drug response (IC50 values) for cancer cell lines
- Enable personalized treatment selection
- Accelerate drug discovery process

Dataset: GDSC (Genomics of Drug Sensitivity in Cancer)

- Large-scale pharmacogenomic database
- Multiple cancer types and therapeutic compounds
- Comprehensive molecular profiling data

Prediction

IC50 = Drug concentration needed to kill 50% of cancer cells

Lower IC50 = More effective drug (less needed to work)

Higher IC50 = Less effective drug (more needed to work)

Dataset

Real experiments testing cancer drugs on cell lines

200,000+ drug-cell line combinations

Collected by Wellcome Sanger Institute

Model Architecture - Feedforward

Input (12 features)
↓
Dense Layer (256 neurons) → BatchNorm → Dropout (40%)
↓
Dense Layer (128 neurons) → BatchNorm → Dropout (30%)
↓
Dense Layer (64 neurons) → BatchNorm → Dropout (30%)
↓
Dense Layer (32 neurons) → Dropout (20%)
↓
Output Layer (1 neuron) → Predicted LN_IC50

Results and What This Means Clinically - Feedforword

Training Strategy

64% Training (learning patterns)

16% Validation (tuning during training)

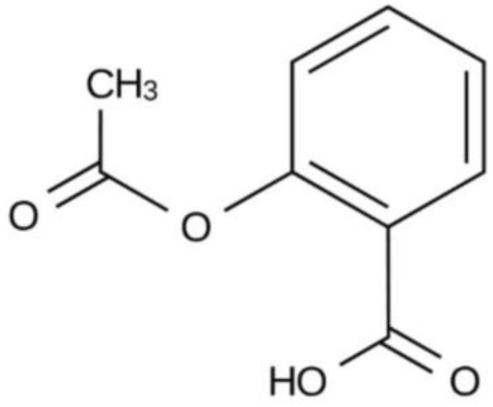
20% Test (final unseen evaluation)

Metric	Value	Interpretation
R^2 Score	0.7247	Explains 72% of drug response variation
MAE	1.1057	Average error of 1.1 on log scale
RMSE	1.15	Root mean squared error

Model Quality Proof - Feedforword

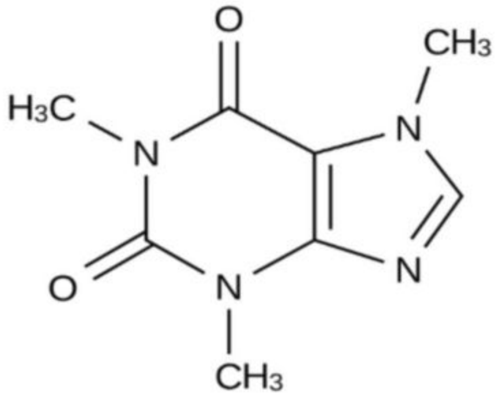
Dataset	R² Score
Training	0.73
Validation	0.72
Test	0.7247

Drug encoding improvement: SMILES



Acetylsalicylic Acid (Aspirin)

CC(=O)Oc1ccccc1C(=O)O



Epinephrine

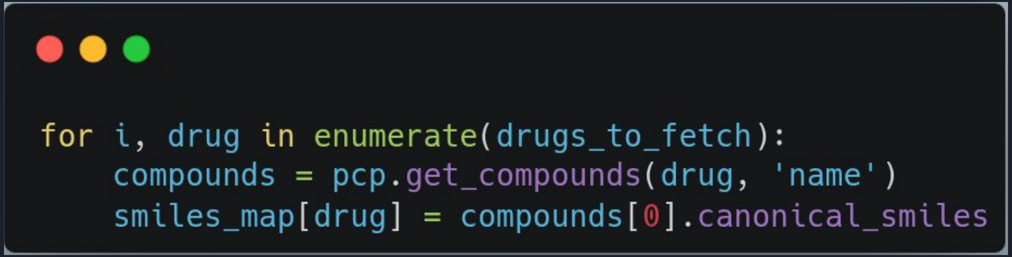
CNC[C@H](O)c1ccc(O)c(O)c1

Simplified Molecular Input Line Entry System

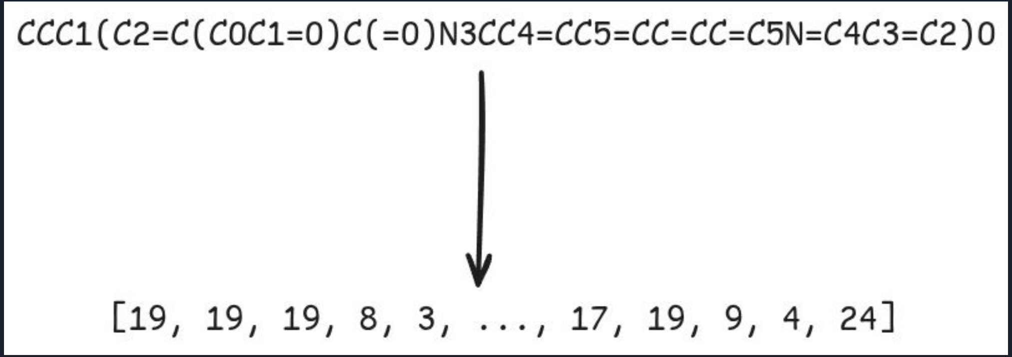
Text-based (ASCII) format to represent 2D or 3D chemical structures for drugs.

Can be passed to the model for it to understand the molecular structure of the drug

Drug encoding improvement: SMILES



```
for i, drug in enumerate(drugs_to_fetch):
    compounds = pcp.get_compounds(drug, 'name')
    smiles_map[drug] = compounds[0].canonical_smiles
```



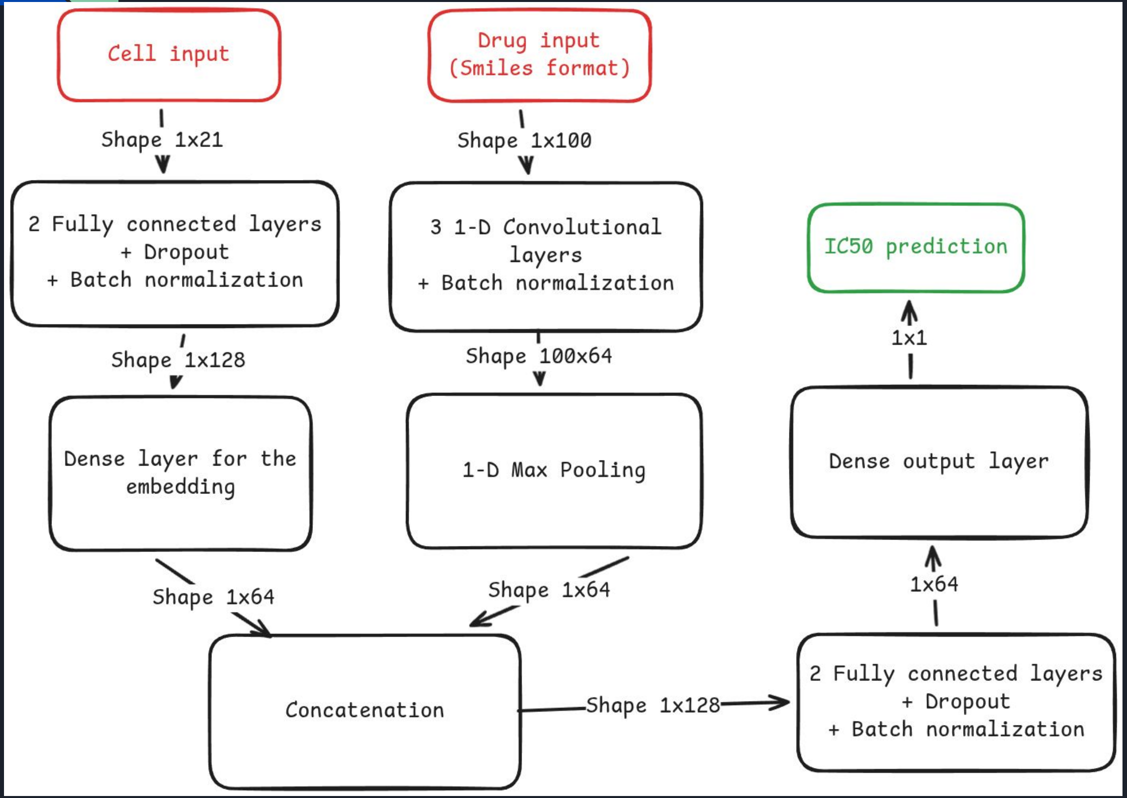
```
CCC1(C2=C(COC1=O)C(=O)N3CC4=CC5=CC=CC=C5N=C4C3=C2)O

[19, 19, 19, 8, 3, ..., 17, 19, 9, 4, 24]
```

We use the PubChemPy library to fetch the SMILES value of each drug of the dataset

Then we encode the string into an array of integers

Model Architecture - Dual Tower


Cell input
Drug input (Smiles format)
Shape 1x21
Shape 1x100
2 Fully connected layers + Dropout + Batch normalization
3 1-D Convolutional layers + Batch normalization
IC50 prediction
Shape 1x128
Shape 100x64
1x1
Dense layer for the embedding
1-D Max Pooling
Dense output layer
Shape 1x64
Shape 1x64
1x64
Concatenation
Shape 1x128
2 Fully connected layers + Dropout + Batch normalization

Each input has its own branch

Then we concatenate both output

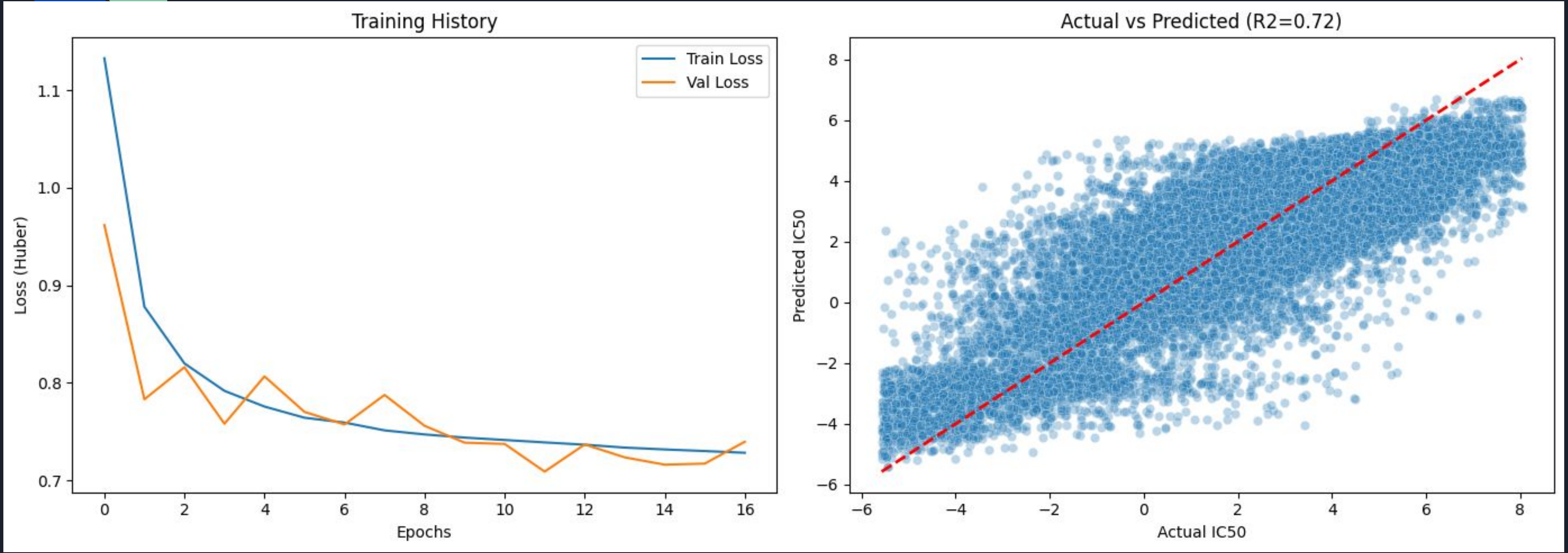
Allows for independent processing of the two types of input

Training Strategy - Dual Tower

- Training set: 80%
- Testing set: 20%

- Learning rate: 0.001 (with Reduce LR on Plateau)
- Epochs: 50 (with Early Stopping)
- Batch size: 64
- Dropout rate: 0.3

Results - Dual Tower



R^2 value: 0.72
Early Stopping at the 16th epoch

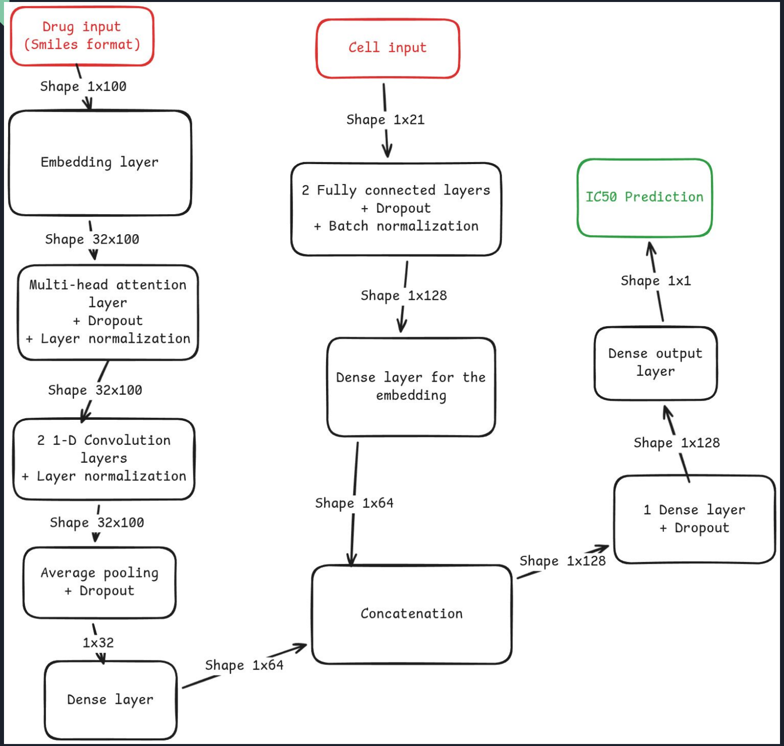
Results interpretation - Dual Tower

Same as the FNN

It seems that the Drug Tower did not improve the processing of the Drug Input

We need an architecture that can understand the SMILES input more efficiently

Model Architecture - Transformer


Drug input (Smiles format)
Shape 1x100
Embedding layer
Shape 32x100
Multi-head attention layer + Dropout + Layer normalization
Shape 32x100
2 1-D Convolution layers + Layer normalization
Shape 32x100
Average pooling + Dropout
1x32
Dense layer
Shape 1x64
Cell input
Shape 1x21
2 Fully connected layers + Dropout + Batch normalization
Shape 1x128
Dense layer for the embedding
Shape 1x64
Concatenation
Shape 1x128
1 Dense layer + Dropout
Shape 1x128
Dense output layer
Shape 1x1
IC50 Prediction

We modify the Drug Input branch to include a Multi-Head attention layer, which allows the model to understand the relations between molecules.

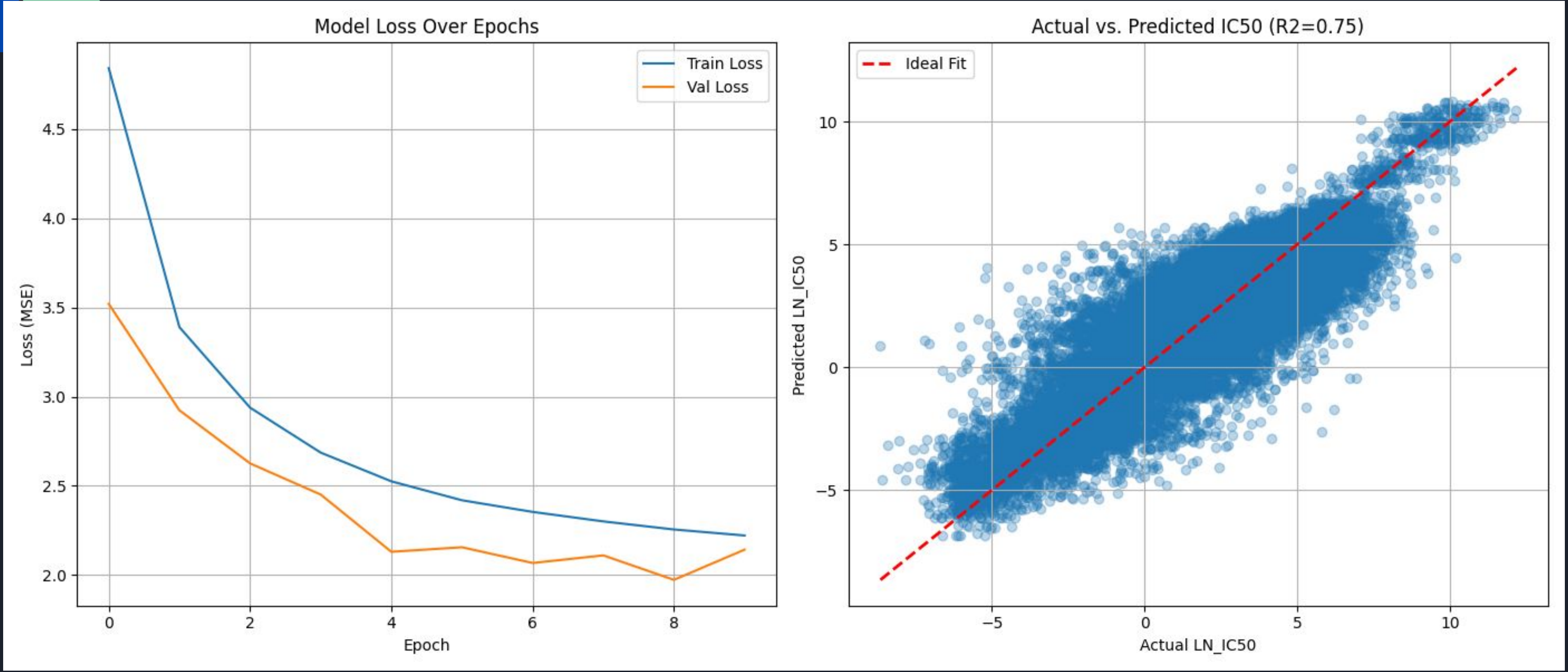
The SMILES input is processed like a LLM would process a sentence.

Training Strategy - Transformer

- Training set: 80%
- Testing set: 20%

- Learning rate: 0.001 (with Reduce LR on Plateau)
- Epochs: 10 (with Early Stopping)
- Batch size: 64
- Dropout rate: 0.3

Results - Transformer



R² value: 0.75

Early Stopping at the 9th epoch

Results interpretation - Transformer

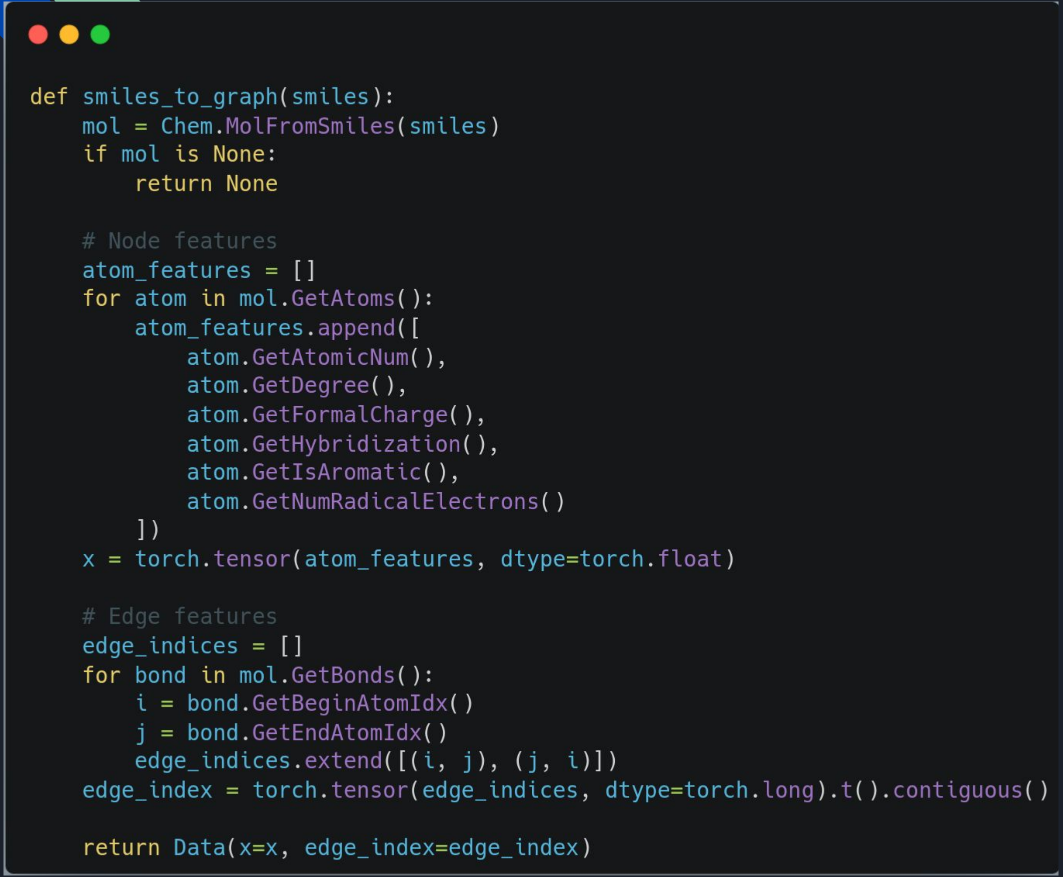
We can better results and a much faster training time

(stops at 9th epoch)

It seems that the Multi-head attention layer understands better the relations between molecules and their impact on the cells.

However it is probable that a GNN would be more adapted to modelize a molecular structure.

Data processing - GNN



```
def smiles_to_graph(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None

    # Node features
    atom_features = []
    for atom in mol.GetAtoms():
        atom_features.append([
            atom.GetAtomicNum(),
            atom.GetDegree(),
            atom.GetFormalCharge(),
            atom.GetHybridization(),
            atom.GetIsAromatic(),
            atom.GetNumRadicalElectrons()
        ])
    x = torch.tensor(atom_features, dtype=torch.float)

    # Edge features
    edge_indices = []
    for bond in mol.GetBonds():
        i = bond.GetBeginAtomIdx()
        j = bond.GetEndAtomIdx()
        edge_indices.extend([(i, j), (j, i)])
    edge_index = torch.tensor(edge_indices, dtype=torch.long).t().contiguous()

    return Data(x=x, edge_index=edge_index)
```

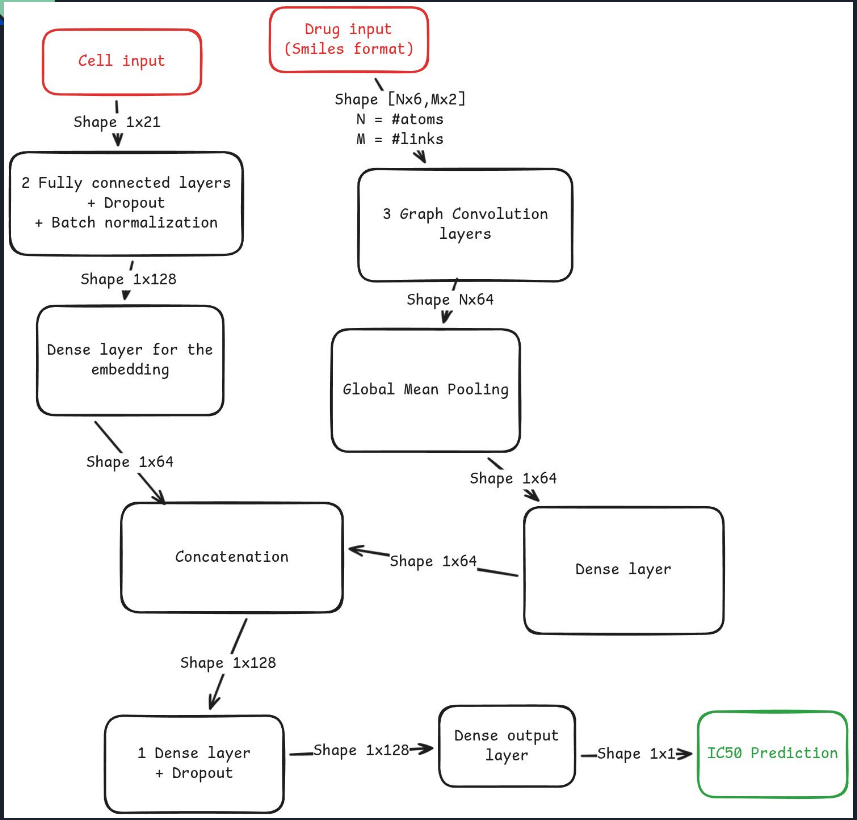
We use the rdkit library to load the molecular structure from the SMILES input.

Then we create a graph of the atoms for each drug.

Each atom has all their characteristics attached to it.

This graph will be passed as the input for the GNN.

Model Architecture - GNN



We introduce 3 Graph Convolution layers.

Instead of passing the SMILES input, we pass a graph represented by:

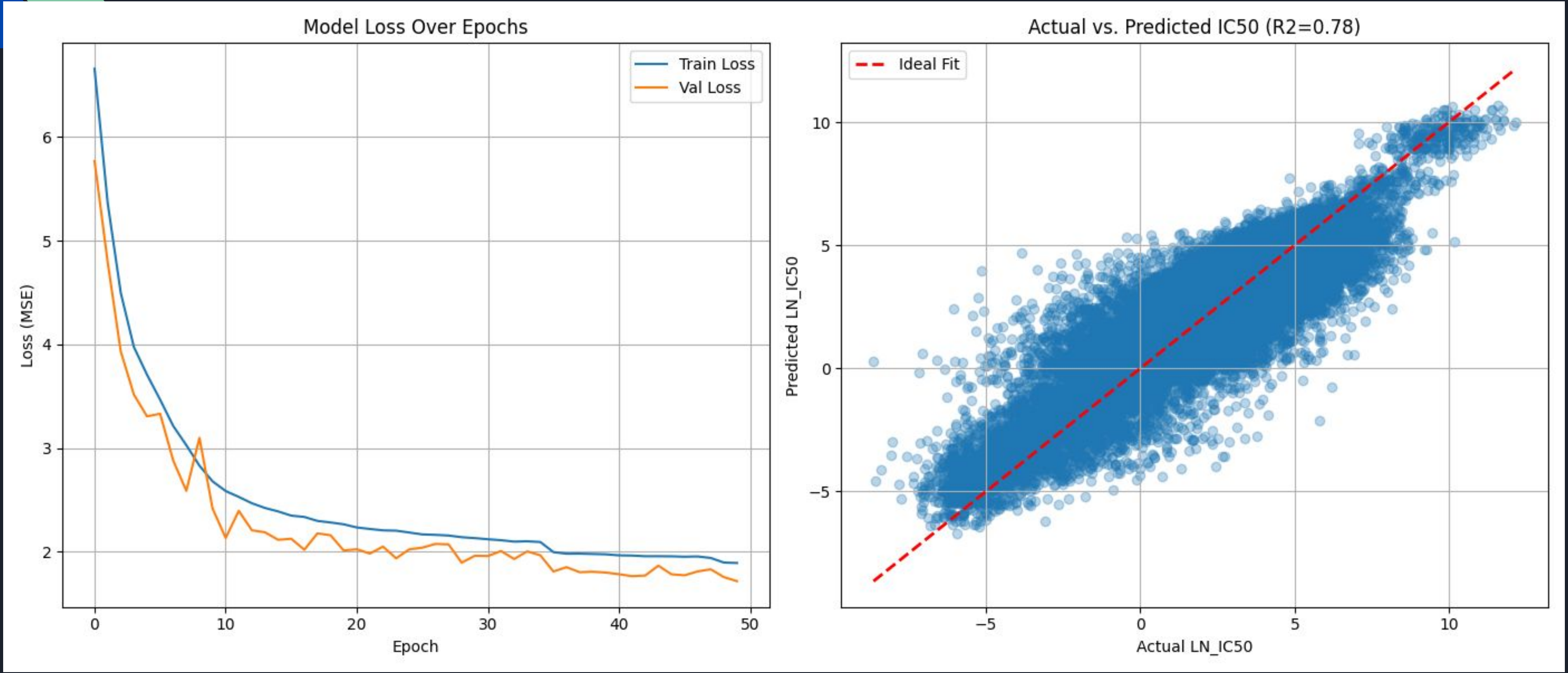
- A Node Feature Matrix Nx6 (N is the number of atoms) which contain the 6 characteristics of each atoms:
  - Atom number
  - Degree (number of atoms attached to it)
  - Formal charge (kind of electric state)
  - Hybridation (complicated)
  - Aromaticity (complicated)
  - Number of radical electrons (complicated)
- An Adjacency Matrix Mx2 (M is the number of links) which contains the connections between atoms

Training Strategy - GNN

- Training set: 80%
- Testing set: 20%

- Learning rate: 0.001 (with Reduce LR on Plateau)
- Epochs: 50 (with Early Stopping)
- Batch size: 64
- Dropout rate: 0.3
- Adam optimizer

Results - GNN



R^2 value: 0.78

Comparisons between models

Model	FNN	Simple Dual Tower	Transformer	GNN
Training time	8.5 min	3 min	3 min	8 min
Model depth	4-5	8	9	9
R^2	0.72	0.72	0.75	0.78