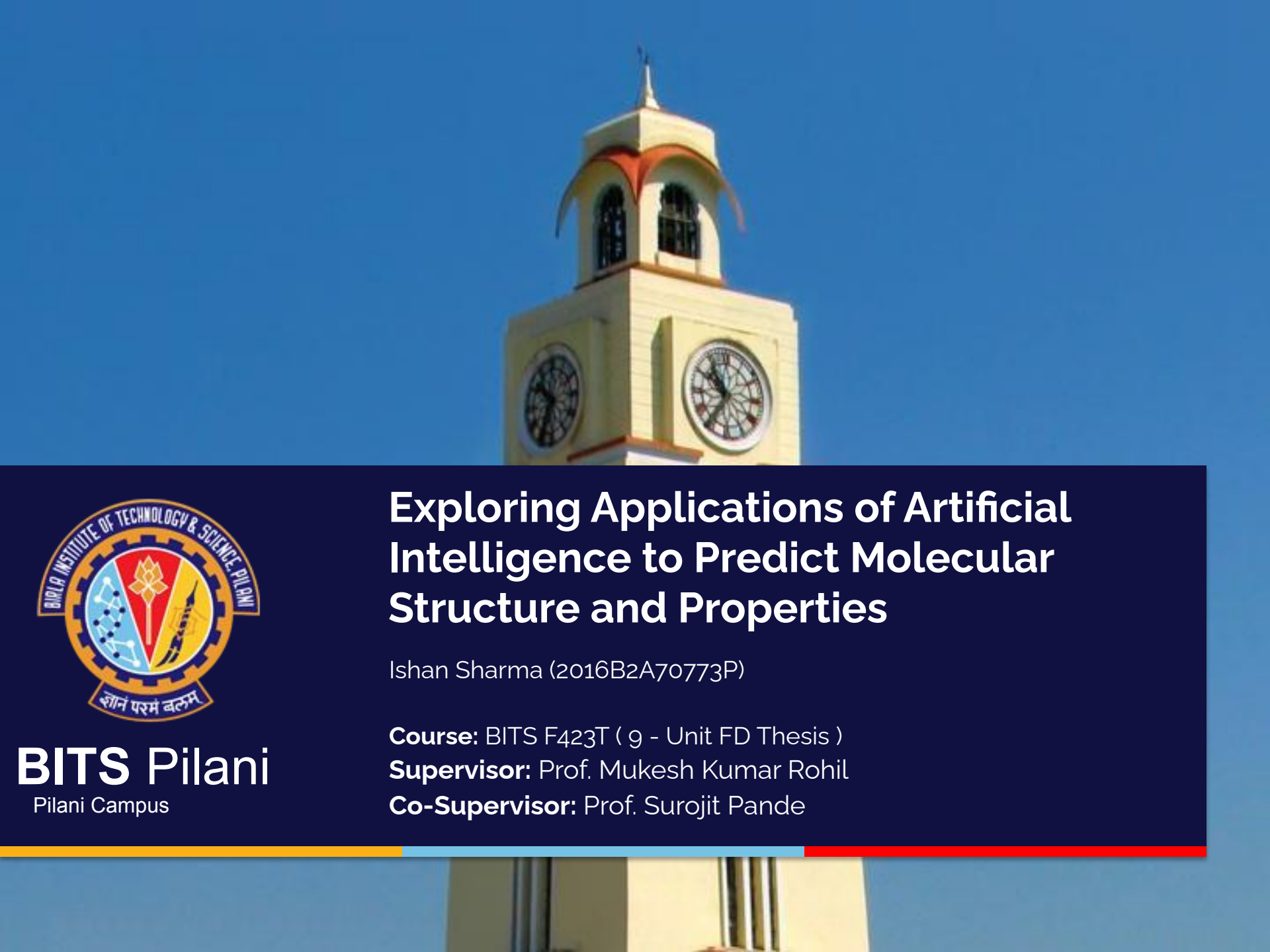


Exploring Applications of Artificial Intelligence to Predict Molecular Structure and Properties

Ishan Sharma (2016B2A70773P)

Course: BITS F423T (9 - Unit FD Thesis)
Supervisor: Prof. Mukesh Kumar Rohil
Co-Supervisor: Prof. Surojit Pande

BITS Pilani
Pilani Campus

Motivation and Research Goals

- Artificial Intelligence (AI) methods and algorithms are being frequently used by chemists to perform tasks that would be rather difficult to perform using traditional methods.
- AI in chemistry is fueled by the need to discover new molecules - drug discovery.
- New molecules need to be checked experimentally that they possess the right properties - iterative methods are too tedious.
- Hence, there is a need to design a system that can predict various properties based on structural and molecular features that these molecules possess.
- Idea1 - Design and implement an AI model which will help me achieve the ultimate goal of predicting certain properties for any given molecule.
- Idea2 - Design and implement an AI model which will help me achieve the ultimate goal of generating new molecules given a reference molecule.

Quantitative Structure-Activity Relationship (QSAR)

QSAR models have been used in the past to predict molecular properties using physical laws or empirical relationships relating the structure of the molecule to their properties.

The physiological activity Φ was expressed as a function of chemical structure C

$$\Phi = f(C) \ldots\ldots\ldots\ldots [1]$$

Example: Predicting boiling points for different molecules.

A general trend was observed between number of carbons in alkanes and their boiling points. The boiling points for alkanes increases as the number of carbons increased in the molecules.

But generally, the patterns within structure and properties are not usually very straightforward but are very complex in nature.

Hence, we try to implement the same idea using Artificial Intelligence / Machine Learning so that complex structural-property relations could be extracted from molecules to predict accurately.

Graph Neural Networks: What? Why? How?

What are Graph Neural Networks ?

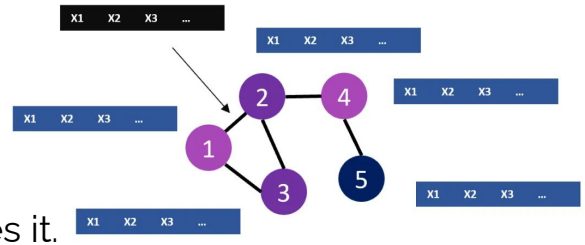
Graphs are a type of data structures having two components: Vertices (or Nodes) and edges which connect two nodes.

Graph neural networks (GNN) refer to the neural network architectures that operate on graph data.

Each node and edge have some set of features that defines it.

The aim of a GNN model is that for each node in the graph to gather information about its neighboring nodes.

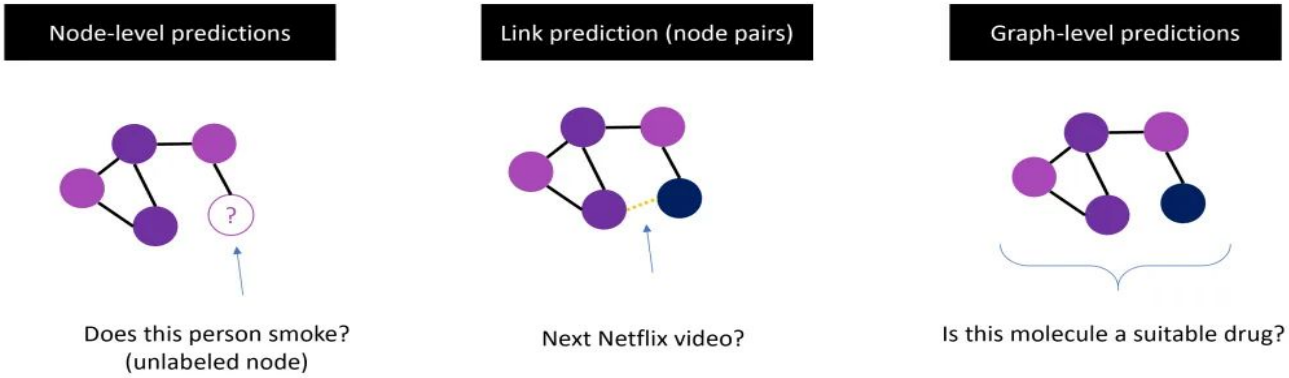
This step of gathering neighbour information is formally known as message passing.



G = (V,E)

	V1	V2	...
V1	0	1	...
V2	1	0	...
V3	1	1	...
...	...	...	...

Adjacency matrix



Why Graph Neural Networks ?

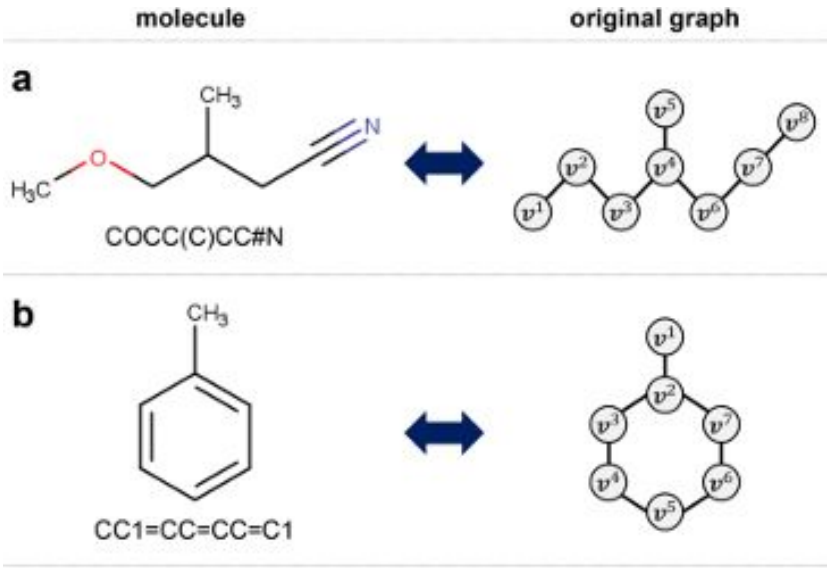
We want to build upon the QSAR relations and extract information from structural composition of a molecule.

We could use a simple feedforward recurrent neural network; - result would be the sequential grammar of molecular structure (SMILES strings) rather than extracting information from structural properties of it.

A molecule itself could be viewed as a graph.

The atoms correspond to the vertices and edges corresponds to the bonds between those vertices / atoms.

Vertices can contain features as electronic configuration, formal charge etc. Edges may contain features like bond type, bond distance etc.

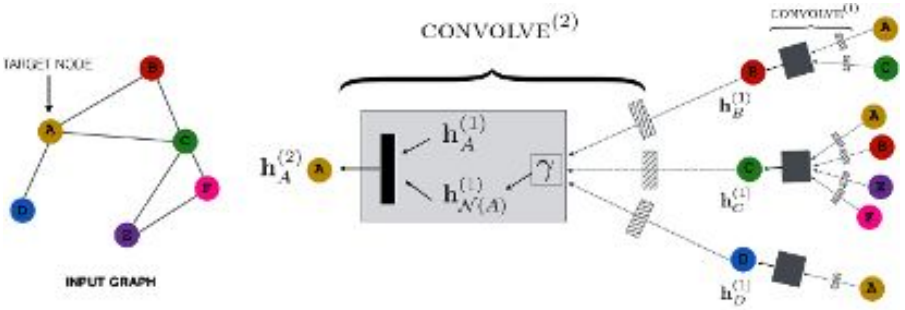


How does a Graph Neural Network work?

The GNNs work by gathering and passing information between nodes and its neighbours.

Nodes receive information from neighbours via edges.

Data aggregation (using many different techniques like max pooling, averaging etc.) - passed to an activation function of a new to get new set of embeddings for the node.



hv = f (Xv, Xco[v], hne[v], Xne[v]) …… [Eq 2]

Where,

f	is a local transition function
Xne[v]	denotes the features of the neighbours of v,
Xco[v]	denotes the edge features connected to v,
hne[v]	denotes the embeddings of the neighbours of v.

GNN Model: Molecular Property Prediction

Background To Datasets

1. **ESOL**: (1128 molecules) Water solubility for common organic molecules.
2. **Free Solv**: (642 molecules) Experimental and calculated hydration free energy of molecules in water.
3. **Lipophilicity**: (4200 molecules) Octanol/water distribution coefficient

Each dataset has to be preprocessed and convert into graph data.

Using rdkit library for python we can get atom level and bond level features for each atom and bond in all molecules.

Following are the 9 atom level features which I shortlisted:

1. Atomic Number
2. Chirality
3. Degree
4. Formal Charge
5. Number of hydrogens
6. Number of radical electrons.
7. Hybridization
8. Aromatic or not
9. Atom inside ring or not

Following are the 3 edge level features which I shortlisted:

1. Bond Type
2. Stereo configuration
3. Conjugation

Architecture

I have used PyTorch Geometric as the central framework as it has dedicated library functions to handle graph data and graph neural networks.

The message passing layers constitute of Graph Convolution Network Layer* with tanh activation function.

Layer	Components	Input Dimension	Output Dimension
First Message Passing	Graph Convolution with tanh activation	9	64
Second Message Passing	Graph Convolution with tanh activation	64	64
Third Message Passing	Graph Convolution with tanh activation	64	64
Fourth Message Passing	Graph Convolution with tanh activation	64	64
Fifth Message Passing	Graph Convolution with tanh activation	64	64
Global Pooling	Global Mean and Max Pooling	64	128
Linear Layer	Linear Regression Layer	128	1

Hyperparameters

Hyperparameters are parameters that cannot be changed during the learning process and are used to help the model learn other normal parameters.

1. **Activation Function:** tanh was chosen for every message passing layer due to higher performance of it over sigmoid or ReLu for convolutional systems.

2. **Loss Function:** The input data to the model is floating points numbers hence I used Mean Square Error as it performs well for Linear Regression models.

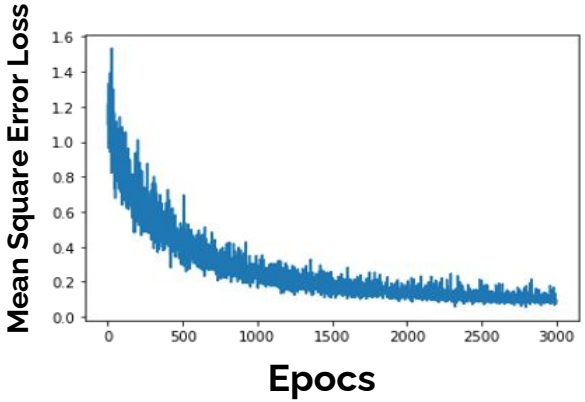
3. **Learning Rate:** After evaluating different ranges of learning rates, I finally decided to use 0.0005 as the learning rate.

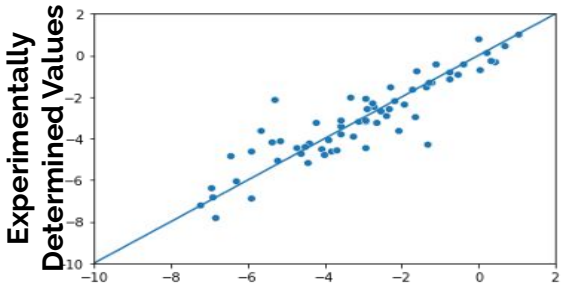
Performance Evaluation

innovate achieve lead

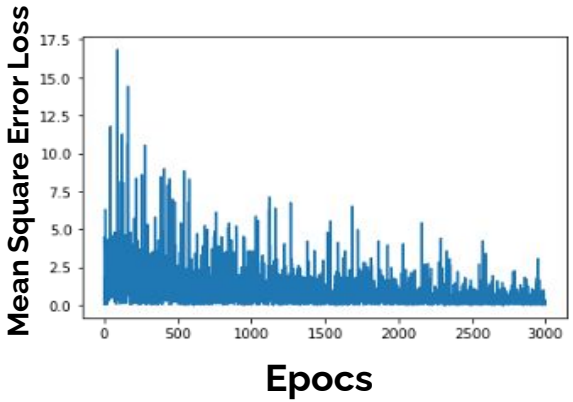
Datasets

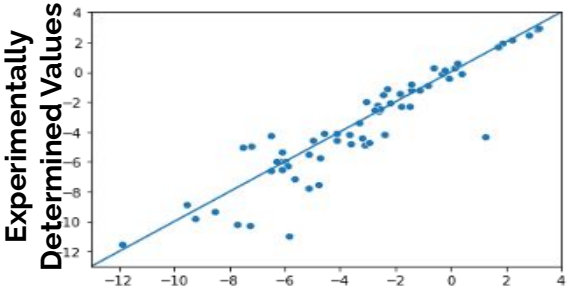
ESOL



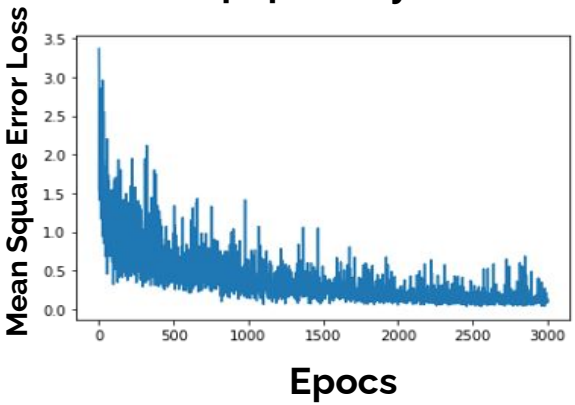


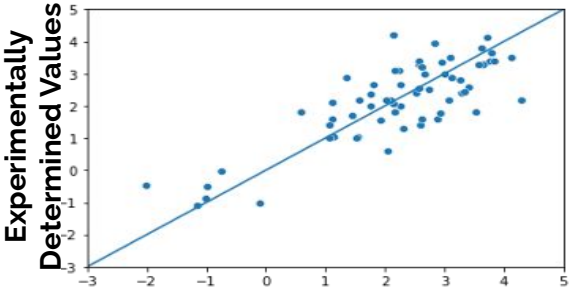
Free Solv





Lipophilicity





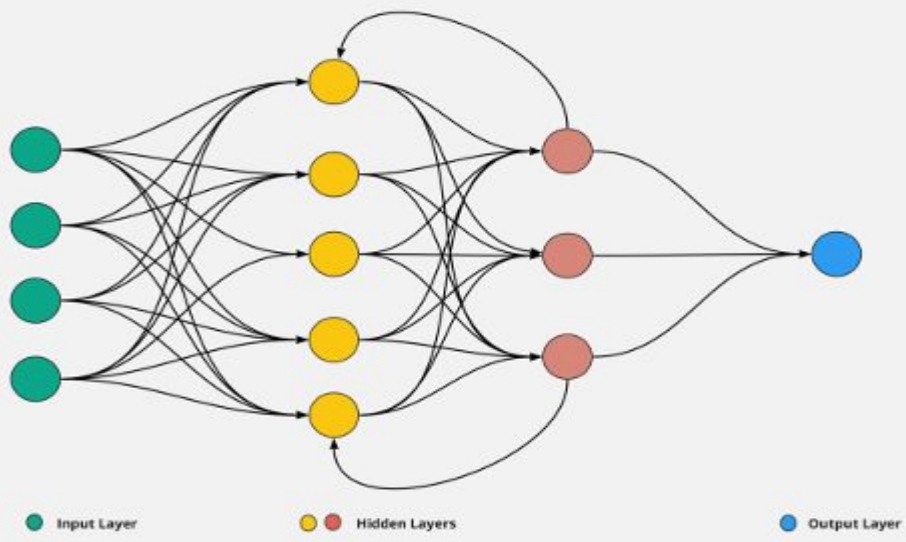
Recurrent Neural Networks: Similar Molecular Generator

What are Recurrent Neural Networks?

Recurrent Neural Networks are a class of Neural Network which operate on the output from the previous hidden step by feeding it as input to the current hidden step.

This feature of feeding the output of previous step can be very much necessary when we want to generate next word in some sentence or generate some regular pattern.

The memory unit in Recurrent Neural Networks helps in remembering information in various steps. This reduces the number of parameters in them as compared to traditional neural networks.



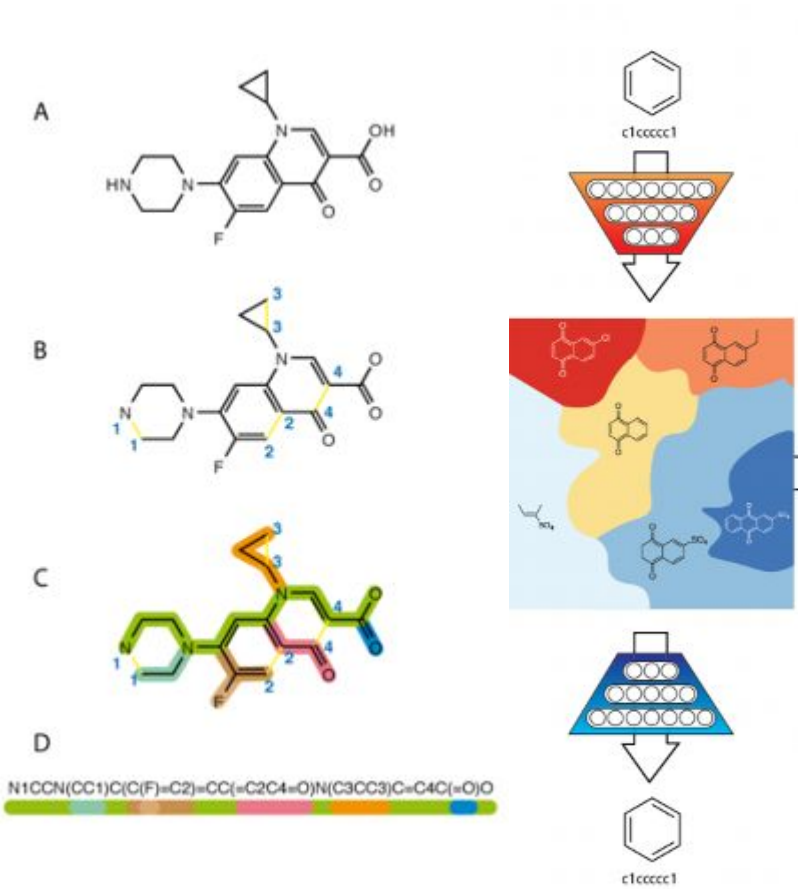
Why Recurrent Neural Networks?

In recent times many different techniques have been used to generate new molecules. One of these techniques used by Rafael Gómez-Bombarelli et al was to design an auto encoder with an encoder, decoder units to generate new molecules and also to predict properties for the same.

Generally the of molecules generated by these methods are often synthetically unreasonable. The yield for valid molecules are very low [8].

Recurrent neural networks come in handy for generating new molecules as these molecules can be presented as a set of simplified molecular-input line-entry system (SMILES).

By the natural language processing capabilities of recurrent neural networks. We can generate new molecules by giving the input as SMILES string and letting the neural network generate output character by character after being trained on various other molecules.



How does a Recurrent Neural Network Work?

Let the x1, x2, x3… represent the input sequence, y1, y2, y3… represent the predicted next words and h0, h1, h2, h3 … represent the information for the previous input sequence.

A general relation can be derived for Recurrent Neural Networks as follows. For some timestep t:

ht = f (W(h)ht-1 + W(x)xt) …….. [Eq 3]

This current information ht is then used to calculate the predicted sequence yt using the softmax function as –

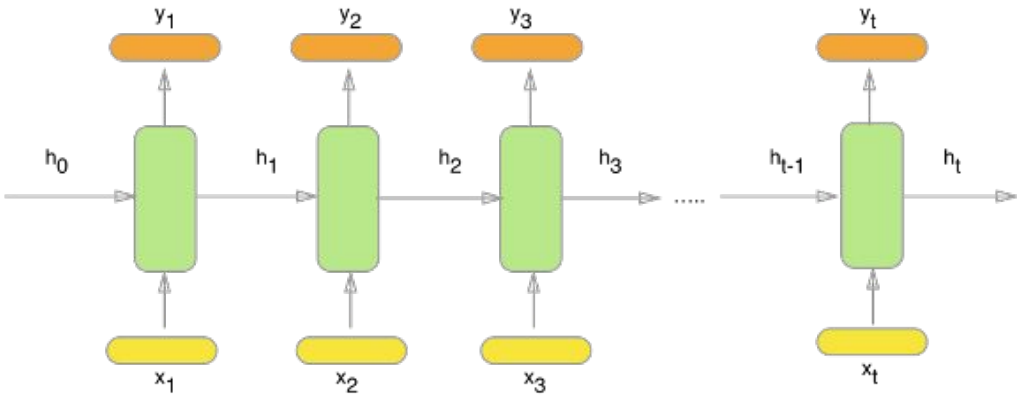
yt = softmax(W(S)ht) …….. [Eq 4]

Generally a cross entropy loss function is used to calculate the error between the predicted and actual sequence.

loss f = Σ (yt log yt) …….. [Eq 5]

This error is then used to adjust the weights (W(h), W(x),W(S)) in the back propagation step.

RNNSs generally have the problem of vanishing gradient, which can be resolved by having LSTM units in the architecture.



Background To Datasets

For this model, I have used the commercially free ZINC15 dataset to train the model.

The dataset contains 10,000 smiles strings in txt format. These molecules contain all type of organic molecules randomized. The SMILES strings in the data had to be encoded in some format to give the RNN model as input.

Every unique character in the SMILES string is mapped to an integer. These integers are then normalised by dividing them by the total unique characters in the dataset.

Architecture

The sequential architecture consists of 3 LSTM layers with Dropout regularization. The number of layer itself becomes a hyperparameter which can be altered while testing out the model. The dropout regularizations helps in model to not overfit the data.

A final dense neural network layer is added with softmax activation function to give output as vector of probability values that sum up to one.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 137, 256)          264192
_________________________________________________________________
dropout (Dropout)            (None, 137, 256)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 137, 256)          525312
_________________________________________________________________
dropout_1 (Dropout)          (None, 137, 256)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 256)               525312
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 34)                8738
=================================================================
Total params: 1,323,554
Trainable params: 1,323,554
Non-trainable params: 0
```

Hyperparameters

Hyperparameters are parameters that cannot be changed during the learning process and are used to help the model learn other normal parameters.
Examples of hyperparameters are the number of layers in the neural network, loss function and activation functions.

1. **Activation Function:** softmax was used as activation function for the dense layer to give a probabilistic prediction for the one hot encoded vector to generate new character.

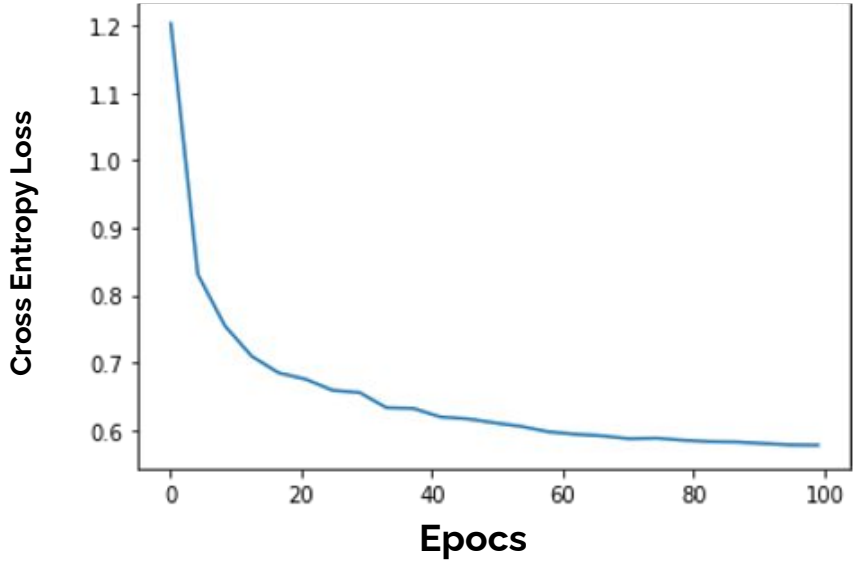
2. **Loss Function:** Categorical Cross Entropy function was used as the data is a set of one hot encoded binary vector with standard Adam optimizer.

Performance Evaluation

The model was trained with 10,000 molecules, a batch size of 128 and 100 epochs (overall time ~6hrs).

The model showed good descent in the cross entropy loss function suggesting good learning rate.

But even after the 6hrs of training, model was generating strings that did resemble SMILES strings but cannot be classified as valid SMILES for some molecule. Many factors could have contributed for the same, as these SMILES strings are very complex in nature, the model might require more LSTM layers, a smaller batch size or increase epochs. The hyperparameters could be tweaked to achieve a desired result.



Conclusion

It is known that physiological properties can be related to structural properties by some functions as we call these relations as QSAR relations.

As molecules can be viewed as graphs present in nature, the bonds correspond to the edges and atoms correspond to the atoms in the graph. It makes sense to use Graph Neural Network to extract structural information from the molecular graphs and use this information to predict various properties from it.

The graph neural network model was designed by having 5 linear layers of Graph Convolutional Network Layers with a catenated Global Mean and Max pooling layer with a final linear layer to predict the property.

The GNN model designed gave nearly accurate results. The model can be improved by adding more message passing layers, experimenting with different learning rates, changing atomic properties chosen for atoms etc.

This model can be very useful in prediction of new molecules that need to be designed in the field of molecule discover, drug design and various other fields.

Generating new molecules can be viewed as generating sequential patterns, as molecules can be represented as ASCII strings – SMILES. For this task Recurrent Neural Networks could be used to general sequential patterns like NLP.

The recurrent neural network model didn't quite achieve the results I was hoping for. The RNN might need more training data or training time or different hyperparameter values to learn such complex strings.

Thank You