IMPLEMENTATION OF GNN FOR PROPERTY PREDICTION USING MOLECULAR STRUCTURE

BY

KANAD SEN (22M1674)
ABHISHEK RAJ (22M1677)

GUIDED BY: PROF. ALANKAR ALANKAR

DEPARTMENT OF MECHANICAL ENGINEERING , IIT BOMBAY

STAGE 2

ME-793

TEAM 10

INTRODUCTION TO THE PROBLEM STATEMENT

- In this project we are trying to understand what a Graph Convolutional Neural Network is and understand it's uses. In real world scenario it is very difficult to get data in csv format. Each and every data is correlated to something or the other. This is what we call as Relational Data. Some examples of Relational Data is the data set of social media platforms like Twitter, Facebook etc where we combine the data of users and their followers and following. This is a sort of directed graph. Thus, if we want to build a recommendation system for the users we need some model which can work on graph Data sets. This is where GNNs come into play.

- In this project we are trying to recreate and improve the results found in the paper "**Graph neural networks for the prediction of molecular structure-property relationships**" **[2]** especially the given regression model. Since no code has been given for the said paper we have tried to build a model from scratch and understand the mathematics behind it. By understanding the mathematics we have tried to come up with a new flow logic which helped us improve upon the said results.

- In the given paper, we have the dataset in SMILES format on which we cannot apply any model. Thus, we have to convert into molecular graphs to be worked upon. Because we don't want to use any experimental information we want to build our model on basic molecular properties for this reason we have to build a molecular graph from the basic information.

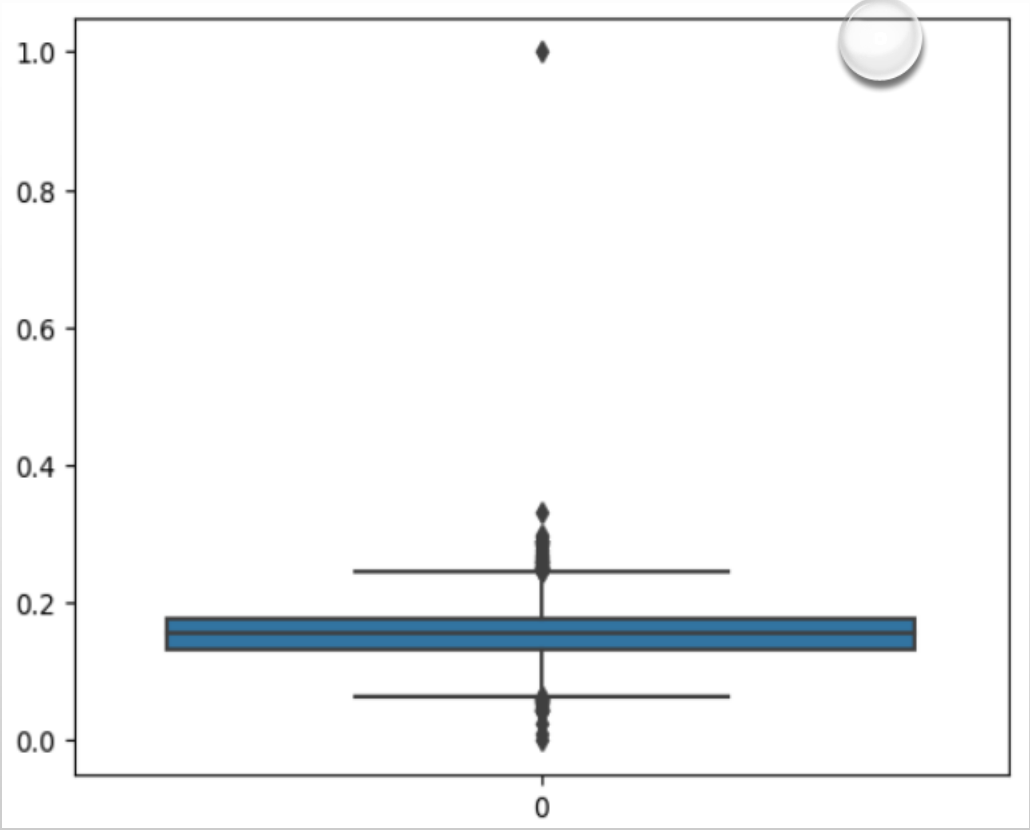
- Since the problem to be solved is a regression problem we will be using R2 score, residual plots, parity plots and RMSE values for comparison of our models. We also will use transfer learning in later stage to check the compatibility of our model in later stages.

DATASET AND DATA PREPROCESSING

- Normal boiling point data set contains **5,276 boiling points for 5,089 molecules** of various classes, i.e., pure hydrocarbons, hydrocarbons with additional atoms such as oxygen, nitrogen, or fluorine, as well as multi-functional molecules.
- We randomly select 10% of the data set for testing, i.e., 509 molecules are set aside. The test set is kept unchanged in the following. The remaining 4,651 data points will be split randomly into **90% training set** and **10% validation.** The training set and the validation set may be changed as required.
- We will convert the molecules provided as SMILES strings to attributed molecular graphs with the atom and bond features provided in **Table 1** and **Table 2 (as depicted in next slide)** respectively, with the help of **RDKit**.
- We extend the atom features stated in table 1 to the atom types occurring in the data set **(C, O, N, F, S, Cl, P, I, Br, Si)** and exclude atom features that do not occur or do not vary, i.e., **sp3d,** resulting in **30 atom features and 6 bond features** in total.
- We **standardize the boiling** point values to a gaussian with a zero mean and standard deviation of one (z-score normalization) for training purposes. Lastly we scale the boiling points **between (0,1)** for an efficient neural network.

$$X_{norm} = (X - mean)/Std\ dev.$$

$$X_{scalling} = (X_{norm} - X_{min}/(X_{max} - X_{min})$$



Boxplot diagram Showing outliers

Out of 5089 molecules, **21 molecules** are found to be in outliers and are removed from further analysis.

Atom features for initial node feature vector** [2]

Feature	Description/Exemplary values
Atom type	Type of atom (C, N, O, P, S, F, Cl, Br, I, Si)
Atomic Number	Atomic Number of element **(Added to our model)**
Is in ring	Whether the atom is a part of ring
Is aromatic	Whether the atom is a part of an aromatic system
Hybridization	sp, sp2,sp3, sp3d2
Charge	Formal charge of the atom
Bonds	Number of bonds the atom is involved in
Hs	Number of bonded hydrogen atoms

Table 1

Bond features for initial edge feature vector** [2]

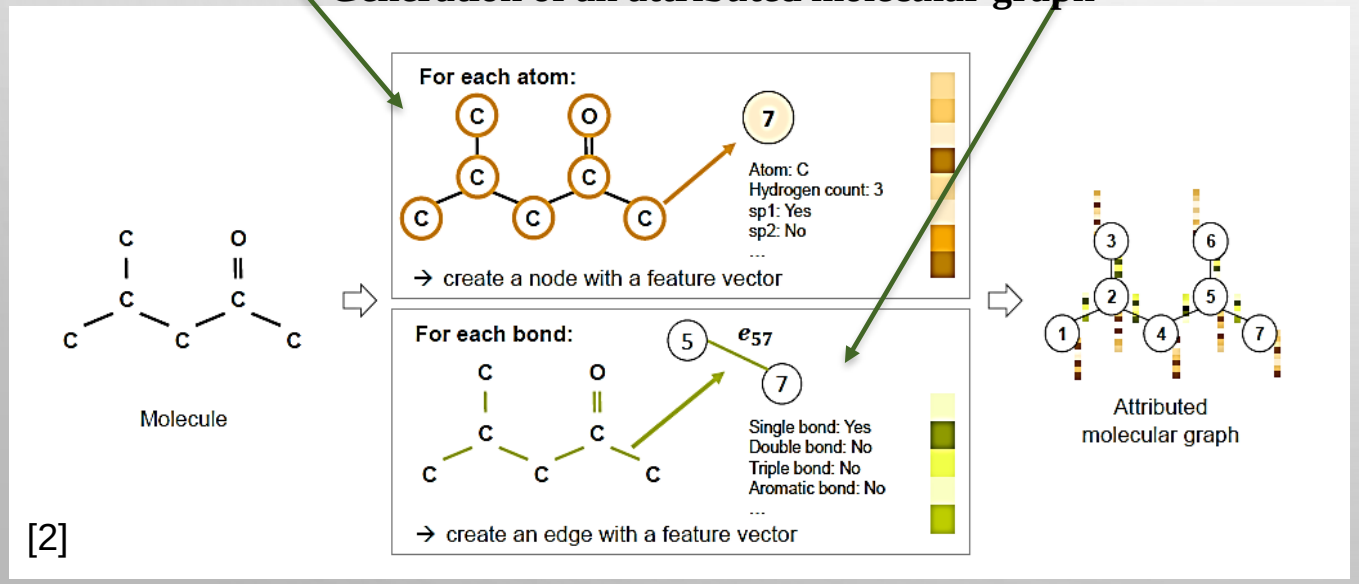
Feature	Description/Exemplary values
Bond type	Single, double, triple or aromatic
Conjugated	Whether the bond is conjugated
Is in ring	Whether the bond is a part of a ring

Table 2

Stereoisomer's (cis/trans) form of the molecules have not been included in the bond feature due to lack of information in our SMILES dataset.

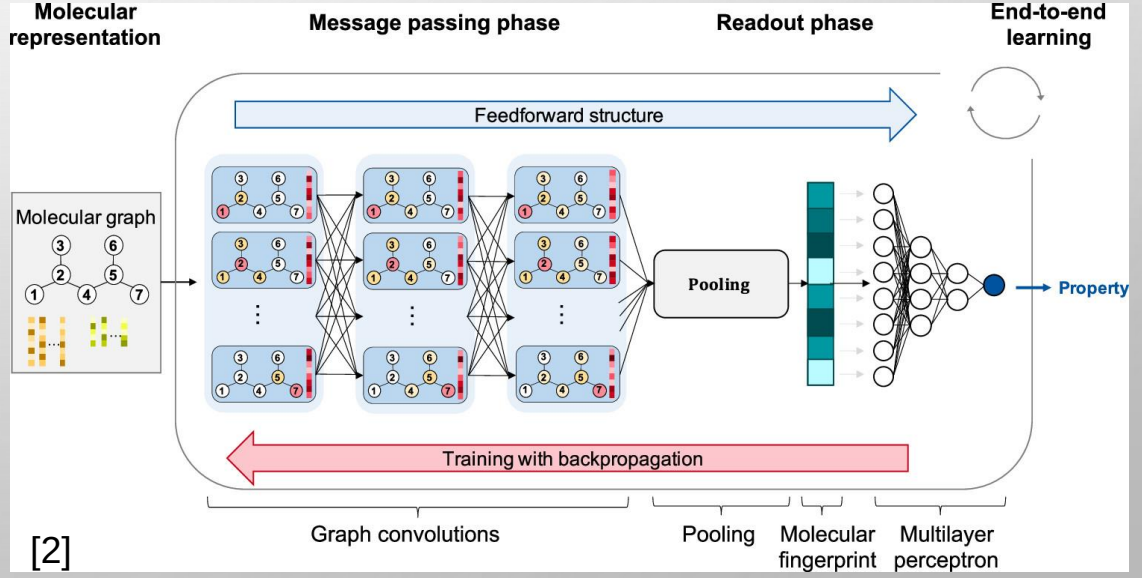
**** All features are implemented as one-hot encoders.**

Generation of an attributed molecular graph



[2]

SAMPLE GNN STRUCTURE FOR PREDICTION



[2]

MESSAGE PASSING NEURAL NETWORK

MPNNs which operate on undirected graphs G with node features $\mathbf{x}_v$ and edge features $\mathbf{e}_{vw}$. It is trivial to extend the formalism to directed multigraphs. The forward pass has two phases, a message passing phase and a readout phase. The message passing phase runs for **T** time steps and is defined in terms of message functions $\mathbf{M}_t$ and vertex update functions $\mathbf{U}_t$. During the message passing phase, hidden states $\mathbf{h}^t_v$ at each node in the graph are updated based on messages $\mathbf{m}_v^{t+1}$ according to where in the sum, **N(v)** denotes the neighbors of **v** in graph **G**. [11,12]

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

The readout phase computes a feature vector for the whole graph using some readout function **R** according to

$$\hat{y} = R(\{h_v^T | v \in G\})$$

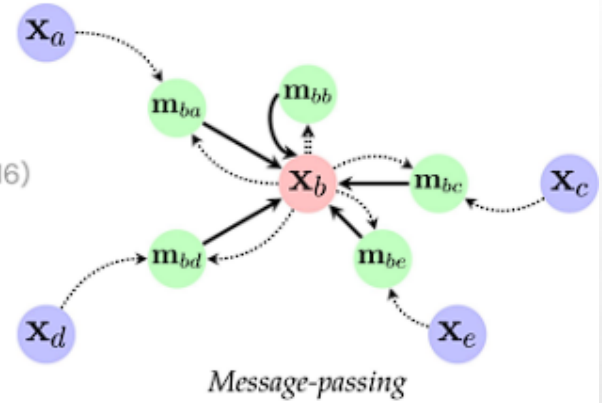
we extract all newly update hidden states and create a final feature vector describing the whole graph. This feature vector can be then used as input to a standard machine learning model.

General Structure of Message Passing Network

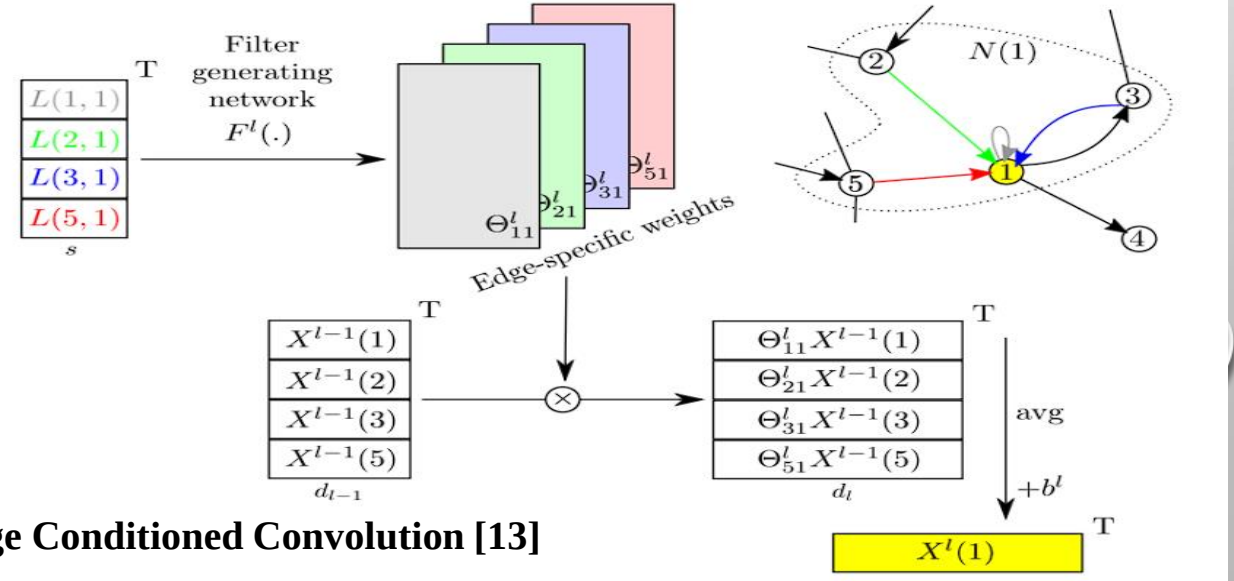
- Compute **arbitrary vectors** (*"messages"*) to be sent across edges

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

- Messages computed as $\mathbf{m}_{ij} = \psi(\mathbf{x}_i, \mathbf{x}_j)$
  - Interaction Networks (Battaglia *et al.*, NeurIPS'16)
  - MPNN (Gilmer *et al.*, ICML'17)
  - GraphNets (Battaglia *et al.*, 2018)
- Most **generic** GNN layer
  - May have *scalability* or *learnability* issues
  - Ideal for *computational chemistry, reasoning* and *simulation*

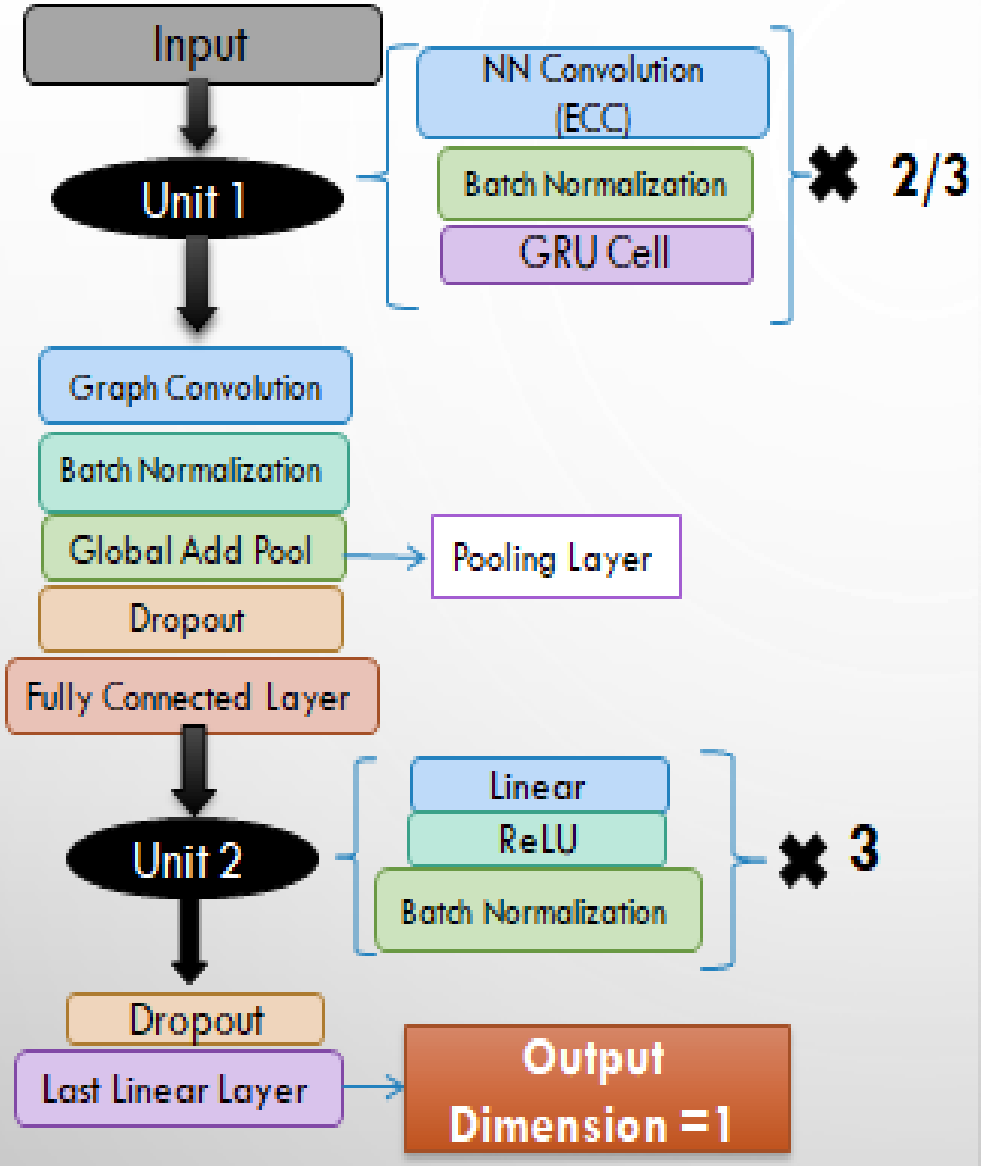


[10]



Edge Conditioned Convolution [13]

MODEL / ARCHITECTURE USED



Block Diagram of Model architecture used

NNConv Layer: (Built on the Base Message Passing Class)

NNConv Layer is a **Edge Conditioned Convolutional Layer (ECC)** that takes into account the structure of a graph when performing convolutions. The input is represented as a graph structure, where each node corresponds to a feature vector and each edge represents a relationship between nodes.

The output of the ECC operation on node i is a **new feature vector** that captures the information from its neighbors in the graph. **This operation can be applied to all nodes** in the graph to obtain a new set of feature vectors, which can be used for downstream tasks. **[13]**

$$\mathbf{x}'_i = \mathbf{\Theta}\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j \cdot h_{\mathbf{\Theta}}(\mathbf{e}_{i,j}),$$

where $h_{\mathbf{\Theta}}$ denotes a neural network, *.i.e.* a MLP.

$\mathbf{x_j}$ = node feature

$\mathbf{e_{ij}}$ = edge feature

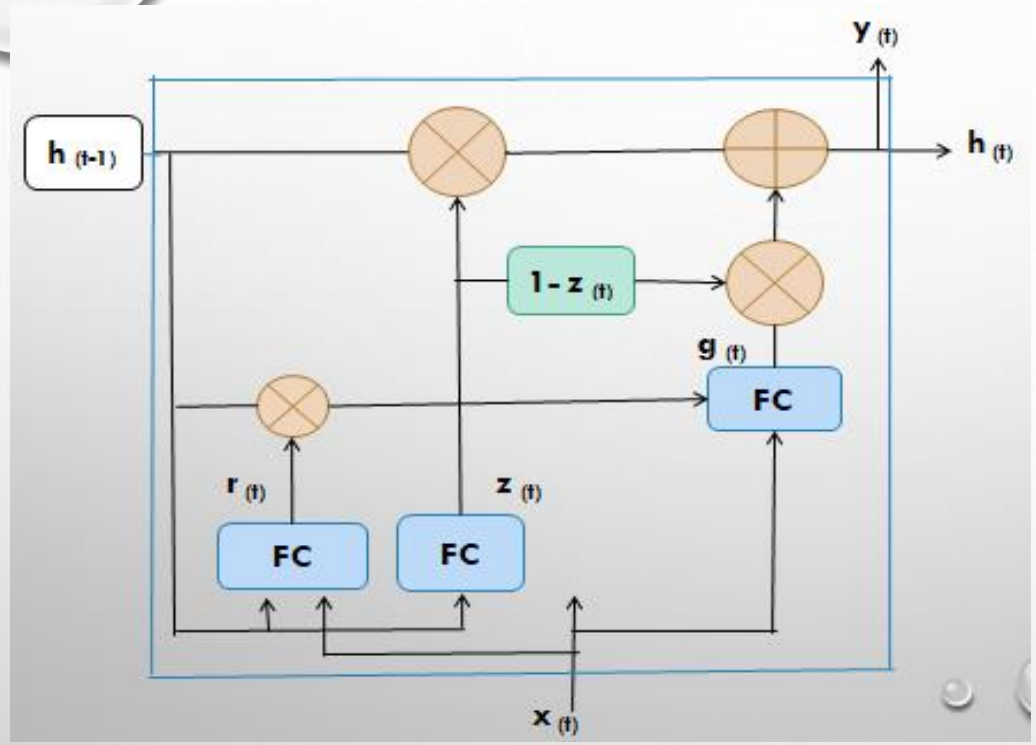
The NNConv layer uses **dynamic kernel** which is a learnable parameter that is used to compute the convolutional filter weights for each edge in the graph. Unlike traditional convolutional networks, where the kernel weights are fixed, the dynamic kernel in ECC Networks allows the network to adapt the convolutional filter to the local structure of the graph.

Global Add Pool

The feature map is summed over all of its spatial dimensions to produce a single scalar value for each feature map.

It is useful for reducing the dimensionality of the feature maps before the final layer of a neural network., where the output needs to be a scalar value.

GATED RECURRENT UNIT (GRU)



GRU Cell

The Symbols used in the GRU are :

$\mathbf{x}_{(t)}$: input vector
$\mathbf{r}_{(t)}$: reset gate controller
$\mathbf{g}_{(t)}$: activation vector
$\mathbf{z}_{(t)}$: update gate controller
$\mathbf{h}_{(t)}$: hidden states
FC : fully connected layer
⊗ / ⊕ : element-wise multiplication / addition

Why are we using GRU instead of other RNNs :

- GRU's don't have vanishing gradient problem unlike LSTM's.
- They have a simple architecture and have a much faster computation time than others.
- Also it stores only the previous iteration as the input gate and the forget gate are merged together into one gate.

$$z(t) = \sigma(W_{xz}^T x_{(t)} + W_{hz}^T h_{(t-1)} + b_z) \quad \dots (1)$$

$$r(t) = \sigma(W_{xr}^T x_{(t)} + W_{hr}^T h_{(t-1)} + b_r) \quad \dots (2)$$

$$g(t) = tanh(W_{xg}^T x_{(t)} + W_{hg}^T (r(t) \otimes h_{(t-1)}) + b_g) \quad \dots (3)$$

$$h(t) = z_{(t)} \otimes h_{(t-1)} + (1 - z_{(t)}) \otimes g_{(t)} \quad \dots (4)$$

Working of a GRU:

- There is no output gate; the full state vector is output at every time step. However, there is a new gate controller $\mathbf{r}_{(t)}$ that controls which part of the previous state will be shown to the main layer $\mathbf{g}_{(t)}$.
- A single gate controller $\mathbf{z}_{(t)}$ controls both the forget gate and the input gate. If the gate controller outputs a 1, the forget gate is open (=1) and the input gate is closed (1-1=0). If it outputs a 0, the opposite happens.
- Both state vectors are merged into a single vector $\mathbf{h}_{(t)}$.

BATCH NORMALIZATION

Batch normalization is a technique used in deep learning to improve the training of artificial neural networks. . This helps to reduce the effects of the "internal covariate shift" and can speed up training.

Given a batch of inputs X = {x1, x2, ..., xn} of size n, we can compute the mean and standard deviation of the batch as follows:

$$\mu = \sum Xi / N \qquad \sigma = \sqrt{\sum (Xi - \mu)^2 / N} \qquad \hat{y}_i = \frac{\varepsilon * (Xi - \mu)}{\sigma} + \beta$$

We then normalize each element of the batch by subtracting the mean and dividing by the standard deviation, and scale and shift the normalized values by learned parameters ε and β, respectively where $\hat{y}_i$ is the normalized output for the i-th input in the batch, σ is a small constant added for numerical stability.

Finally, the normalized outputs are passed through an activation function such as ReLU or sigmoid to produce the final output of the layer. This process is repeated for each layer in the neural network during training.

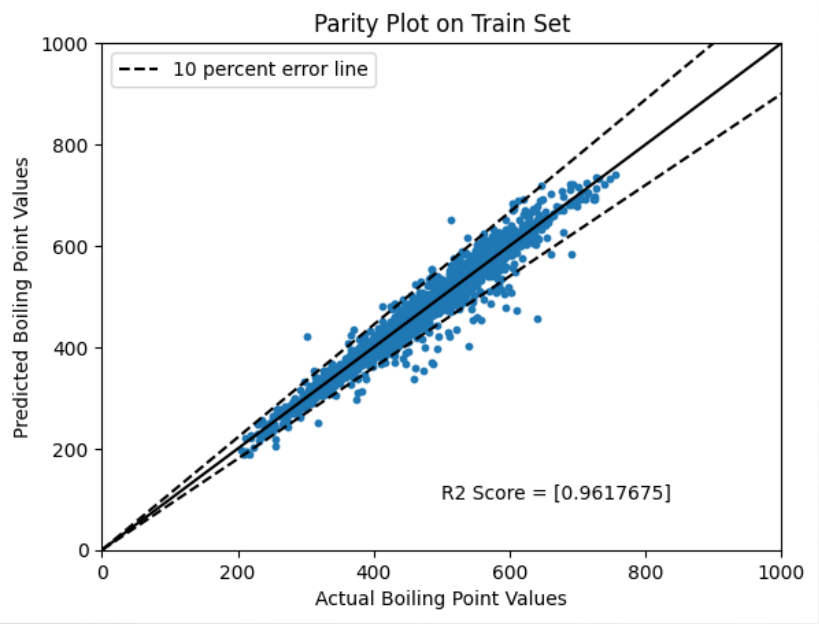
LOSS FUNCTION (HUBER LOSS)

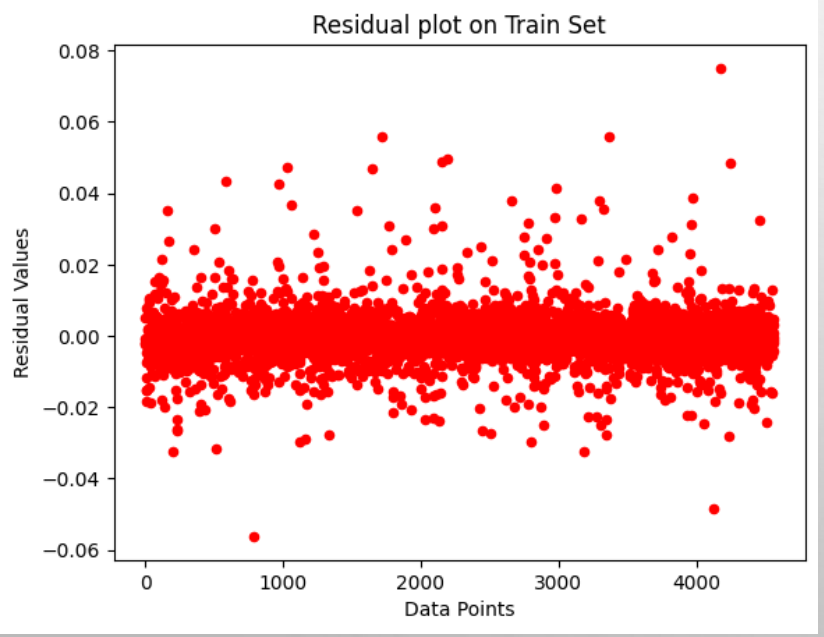
The Huber loss is a loss function that is often used in neural networks for regression tasks. It is a combination of the mean squared error (MSE) and the mean absolute error (MAE) and is less sensitive to outliers than the MSE. This has been used here to prevent our model from getting affected by outliers as graph networks are sensitive to that.

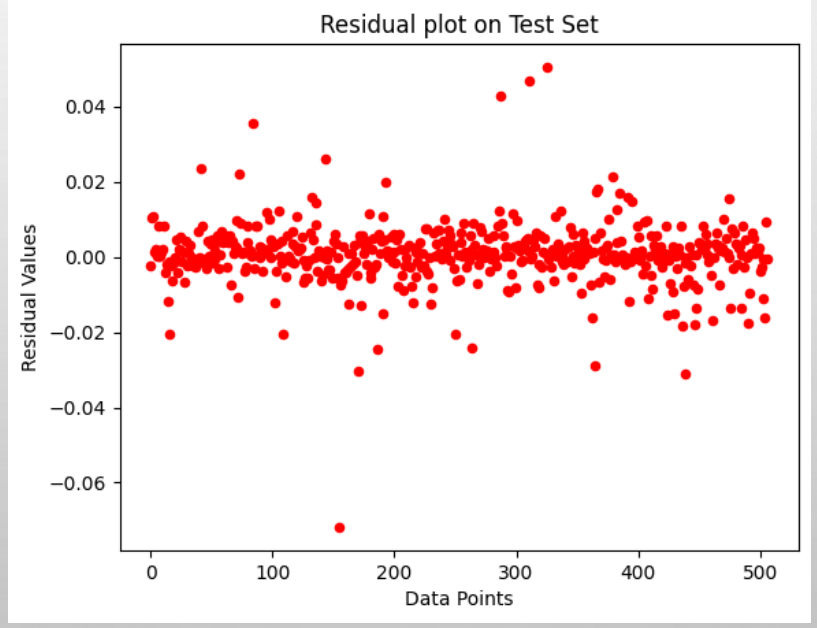
$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

Where, **δ is a hyperparameter** that controls the threshold between using the MSE or the MAE. When the difference between **y_true** and **y_pred** is less than delta, the loss is calculated as MSE, and when it is greater than delta, the loss is calculated as MAE.

RESULTS







The plots given here show the R2 score on the training set and the test set. As seen we are getting an R2 score of **0.9356 on test set** and **0.9618 on the train set.** The closeness of R2 values on the train and test set represents that the model is not overfitting the given dataset. A highlight of how our model behaves when compared to actual paper is presented in the conclusion.

From the plots we can see that most of the data points lie in the error range of **10% error line** which is within the permissible range.

The residual plots give a uniform distribution which shows that the model made is a proper fit for the graph data.

From both the parity plots we can say that there is a huge presence of outliers which should be removed. However, we have kept them to train the model to train with respect to noise.

CONCLUSION AND INFERENCES

- From the previous iteration we have added a new Node feature called **Atomic Number**. Due to the addition of this feature we have actually seen an increase in R2 score on the **test set** by **1.68 %** from the previous set of features.
- The model which has been proposed by us works very well on the test set as seen from the results. We have managed to increase the R2 score on the test set by **10.05 %** over **single GNN model [2].** Not only that we are also finding an improvement of **3.944 %** on the test set as compared to the ensembled model of **40 GNNs.** Thus we can say that our model is able to capture most of the variances as seen from the R2 scores.
- The R2 score found on the test set is **0.9355** and on that of train set is **0.961** . We see the values of R2 scores of test and train set close enough due to the fact that we have applied a **weight_decay** of **5e-4** which acts as a L2 regularization parameter.
- We have kept the **dropout at 0**. This is because as we have tuned the **dropout percentage** we have found out that increase in this percentage reduces the R2 score due to the complexity of information in Graph Neural Networks.
- The model provides **best performances** if the dimension of all the layers is kept **at 128** as we have found after tuning the hyperparameters.

FUTURE PLAN

- Apply **Transfer Learning** on the current model and use it solve some another related regression problem like **Melting Point** using SMILES data to check the validity of the model and if this model is effective on all types of datasets without further or very few training.
- Create a webapp using **streamlit** such that if we give any smiles string as input it will give output of the required property. If time permits we will also allow to give an option for the user to give correct values and allow the model to train on the correct parameter.
- We will to try change the batch size, the size of the training set and the test set to check if we can get better results.

VIDEO PRESENTATION LINK

https://drive.google.com/drive/folders/1QG6YiYeRjwq5gI5sVYfnBZ_OzUp-ov2L?usp=sharing

REFERENCES:

1. https://en.Wikipedia.Org/wiki/simplified_molecular-input_line-entry_system
2. https://arxiv.org/abs/2208.04852 Graph neural networks for the prediction of molecular structure-property relationships.
3. Next generation pure component property estimation models: with and without machine learning techniques
4. Yann LeCun, Yoshua Bengio, and Georey Hinton. Deep learning. Nature, 521(7553):436{444, 2015.
5. Shuo Zhang, Yang Liu, and Lei Xie. Molecular mechanics-driven graph neural network with multiplex graph for molecular structures. arXiv preprint arXiv:2011.07457, 2020.
6. Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1):4{24, 2021.
7. Greg Landrum. RDKit: Open-source cheminformatics software. accessed on 01.04.2022.
8. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? ArXiv preprint arXiv:1810.00826v3, 2018.
9. Kristof T. Sch• utt, Huziel E. Sauceda, Pieter-Jan Kindermans, Alexandre Tkatchenko, and Klaus-Robert M• uller. SchNet - A deep learning architecture for molecules and materials. Journal of Chemical Physics, 48(24):1{11, 2018.
10. Theoretical Foundations of Graph Neural Networks
11. https://towardsdatascience.com/introduction-to-message-passing-neural-networks-e670dc103a87
12. https://paperswithcode.com/method/mpnn
13. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs