Using Graph Convolutional Neural Networks to predict physical properties

Otaviano da Cruz Neto[1]

INPE - Instituto Nacional de Pesquisas Espaciais, São José Dos Campos, Brazil
otavianocruz@id.uff.br

Abstract. In this work, we use the *Neural Message Passing* strategy present in *Graph Neural Networks* to be able to predict 19 physical properties of 134 thousand organic molecules present in the QM9 dataset. For this, it was necessary to evaluate several hyper parameter configurations (number of fully connected layers, number of neurons in each layer, number of convolutional layers, ...) in order to find the best topology for predicting properties. About 1800 configurations were evaluated, but with little effectiveness in the results of the models and the high recurrence of gradient disappearance as we increased the network, it was necessary to define two main points for the next evaluations. The first point is to add the stopping criterion for the gradient behavior and the second is to make use of optimized libraries for processing deep networks in graphs.

Keywords: Deep Learning · Neural Message Passing · Graph Neural Networks.

1 Introduction

Learning from data has revolutionized several areas of knowledge. Particularly, in computer vision, convolutional networks showed their great potential to identify objects, people and animals [citation]. The challenge now is, instead of using structured reality data, to make machine learning models capable of predicting behaviors present in unstructured data ie. graphs. The data structure organized in graphs has a great flexibility that until recently machine learning models could not handle due to representation problems.

With that understanding, we structured the work as follows. Section 2 will present the fundamentals of *Graph Neural Networks* and how to solve the graph representation problem. Section 3 will present some reputable works that were used to make this one. The methodology will be described in Section 4 showing which packages and libraries were used for the evaluation and training of models. The results and descriptions about the performed processes were presented in Section 5. Finally, the conclusions and next steps of the work were exposed in Section 6.

2 Graph Neural Networks

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ between we. We denote an edge going from the node $u \in \mathcal{V}$ to the node $v \in \mathcal{V}$ as $(u, v) \in \mathcal{E}$. The graph $\mathcal{G}$ can be conveniently represented by an adjacency matrix $\mathcal{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. The adjacency matrix can be constructed element by element following the rule of $A[u, v] = 1$, if $(u, v) \in \mathcal{E}$ or $A[u, v] = 0$, case $(u, v) \notin \mathcal{E}$.

This type of representation can be easily used by a *Multi Layer Perceptron* (MLP), however, for example, in a water molecule if we exchange the two hydrogen atoms they will result in two different adjacency matrices, even representing the same molecule. That is, we must look for a representation that is invariant and equivalent to permutations. The way we will use here to solve this representation problem is to introduce the *Neural Message Passing* (NMP) strategy [4].

The *Neural Message Passing* technique takes into account that the one node u under its neighborhood influence $\mathcal{N}(u)$, thus making the NMP h_u^{k+1} of iteration $k + 1$ of node u can be given by:

$$h_u^{k+1} = UPDATE^{(k)}(h_u^k, AGGREGATE^{(k)}(h_v^k, \forall v \in \mathcal{N}(u))) \tag{1}$$

$$h_u^{k+1} = UPDATE^{(k)}(h_u^k, m_{\mathcal{N}(u)}^{(k)}) \tag{2}$$

Here the functions $UPDATE$ and $AGGREGATE$ are differentiable and $m_{\mathcal{N}(u)}^{(k)}$ is the message that is aggregated from the neighborhood of u, as in Figure reffig:NMP. At the beginning $h_u^0 = x_u$, $\forall u \in \mathcal{V}$, so that x_u is the characteristic vector of the node u.

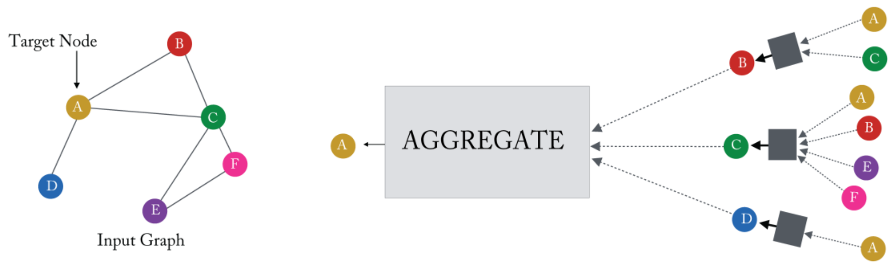


Fig. 1. Using the NMP for the node A, so that the node is influenced by the neighborhood $\mathcal{N}(A) = B, C, D$ and which in turn are also influenced by their own [7] neighborhood.

In general, this type of approach can be applied to nodes and edges that make up a $\mathcal{G}$ graph as below:

$$h_{(u,v)}^{k+1} = UPDATE_{edge}(h_{(u,v)}^k, h_u^k, h_v^k, h_{\mathcal{G}}^k) \tag{3}$$

$$m_{\mathcal{N}(u)} = AGGREGATE(h^k_{(u,v)}, \forall v \in \mathcal{N}(u)) \tag{4}$$

$$h^{k+1}_u = UPDATE_{node}(h^k_u, m_{\mathcal{N}(u)}, h^k_{\mathcal{G}}) \tag{5}$$

$$h^{k+1}_{\mathcal{G}} = UPDATE_{graph}(h^k_{\mathcal{G}}, \{h^{k+1}_u, \forall u \in \mathcal{V}\}, \{h^{k+1}_{(u,v)}, \forall (u,v) \in \mathcal{E}\}) \tag{6}$$

For this work, the NMP present in the Convolutional Graph Network (GCN) will be important, below:

$$h^{(k)}_u = \sigma\left(W^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{h_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right) \tag{7}$$

Where σ is the nonlinear activation function, W^k is the matrix of trainable parameters present in the GCN for the iteration k and $|\mathcal{N}(u)|$ is the quantity of neighbors in the vicinity of the node u.

3 Related Works

Due to the great potential of deep learning techniques aimed at graphs, there is great interest in Quantum Chemistry in seeking applications that can facilitate the calculation of physical properties of organic molecules, instead of performing computationally high cost calculations via Density Functional Theory (DFT).

With this in mind and aiming to compile and organize a database chair, Zhenqin Wu et al. in the article entitled 'MoleculeNet: A Benchmark for Molecular Machine Learning' [9] organizes 17 datasets that start from Quantum Mechanics calculations (QM7, QM7b, QM8, QM9), go to Chemistry Physics (ESOL, FreeSolv, Lipophilicity), contribute to biophysical calculations (PCBA, MUV, HIV, PDBbind, BACE), and fall into physiology (BBBP, Tox21, ToxCast, SIDER, ClinTox). In this work, we will focus on the QM9 set, which contains 134,000 molecules with information on 19 physical properties, listed in Table 3, in addition to their spatial information organized in graphs, as it is with them that Justin Gilmer et al. evaluate the performance of GNNs in predicting properties often defined by the structure of the molecule.

In their article Justin Gilmer et al. [6] uses neural networks for graphs (GNNs) and different NMP strategies as a way to learn the behavior of the 19 physical quantities present in QM9, and thus manage to avoid DFT calculations.

4 Methodology

The GCN model will be built from the Torch Geometric [5] library. All properties discussed in Section 2 about NMPs, architectures and approaches are already

Property	**Description**
μ	Dipole Moment
α	Isotropic polarizability
ϵ_{HOMO}	Highest occupied molecular orbital energy
ϵ_{LUMO}	Lowest unoccupied molecular orbital energy
$\Delta\epsilon$	Gap between ϵ_{HOMO} and ϵ_{LUMO}
$\langle R^2 \rangle$	Electronic spatial extent
$ZPVE$	Zero point vibrational energy
U_0	Internal energy at 0K
U	Internal energy at 298.15K
H	Enthalpy at 298.15K
G	Free energy at 298.15K
c_v	Heat capavity at 298.15K
U_0^{ATOM}	Atomization energy at 0K
U^{ATOM}	Atomization energy at 298.15K
H^{ATOM}	Atomization enthalpy at 298.15K
G^{ATOM}	Atomization free energy at 298.15K
A	Rotational constant
B	Rotational constant
C	Rotational constant

Table 1. Table with physical properties calculated via MD *ab initio* [9]

implemented in the package. In this case, the prediction will be supervised, that is, the error measure will be based on the difference between the expected result and the result of the model output (RMSE). Also, the activation function that will be used in the GCN will be the *Linear Rectifier* (ReLu). For the training phase 70% of the data will be allocated and 30% for the testing phase. The methodology flow can be seen in Figure **??**.

To generalize the behavior of the data present in the QM9 dataset, it was necessary to evaluate which combinations of hyperparameters fit best to describe them. The evaluated hypermeters were:

– Number of convolutional layers ranging between 1 and 4.
– Number of neurons present in convolutional beds, ranging from 2 to 128.
– Number of fully connected layers, ranging from 1 to 4.
– Number of neurons present in fully connected layers, ranging from 2 to 128.
– Size of the batch, ranging from 1 to 1024 graphs of molecules.
– Learning rate, ranging from 1e-5 to 1e-1.

For this case, around 1800 models will be evaluated. This type of hyperparameter optimization via brute force has an extensive evaluation time due to the large number of models, so we use the *Optuna* [2] package for the Python programming language as a way to parallelize the process model training, in addition to selecting the best models and recording all the information generated. The metric used for evaluation is the combination of three factors (Runtime, Validation Error and Training Error), those models that manage to minimize

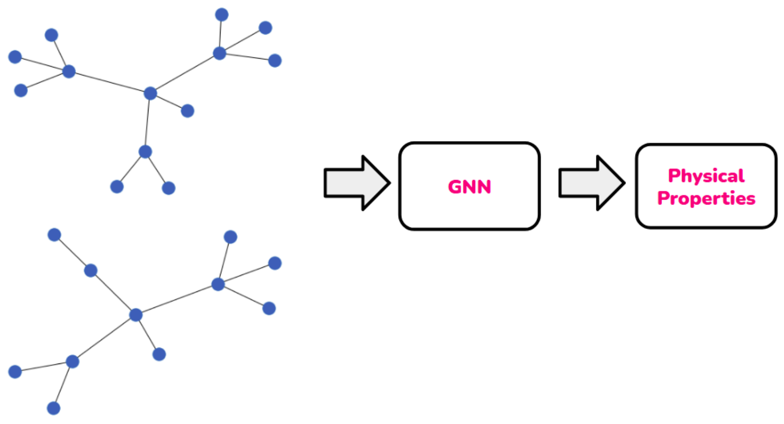


Fig. 2. The methodology is outlined starting from the graph of molecules present in the QM9 dataset, then the structural and characteristic information will be used in the learning process of GNN using the NMP strategy. At the end, the RMSE responsible for indicating the learning process of the GNN in extracting the characteristics of the graphs in order to predict the 19 physical properties of the molecules will be computed.

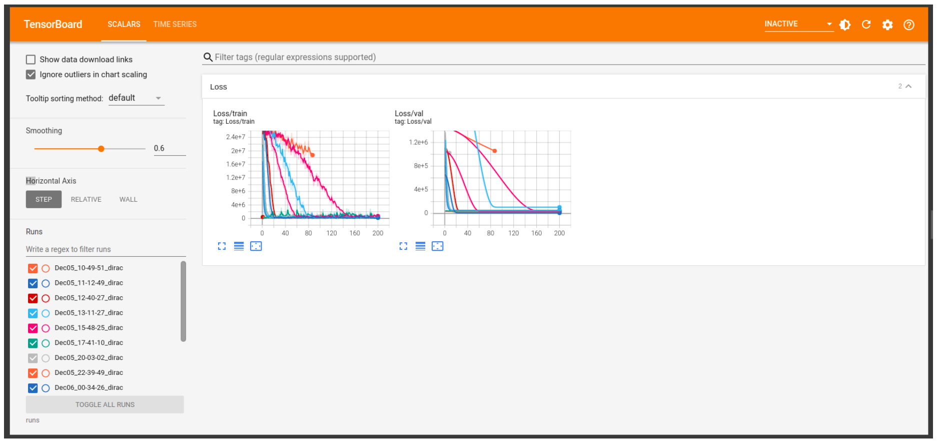
these three parameters will be selected as promising models. To compile in a single panel all the results generated as the learning curve and hyperparameters of the model, the *Tensorboard* [1] was used. In addition to the tools, a device with an Intel(R) Xeon(R) Gold 5118 processor @ 2.3GHz was used, with 755GB of RAM memory and equipped with 4 NVIDIA Tesla V100 16GB GDDR5 video cards.

5 Results

Performing the hyper parameter optimization resulted in the following configuration:

- Number of convolutional layers: 4.
- Number of neurons present in convolutional beds: 64.
- Number of fully connected layers: 4.
- Number of neurons present in fully connected layers: 64 .
- Batch Size: 128
- Learning rate: 0.001

This selected model contained an error in the validation phase of 3724.4 and in the training phase of 2538.0, resulting in a runtime of 1 hour and 53 minutes in 200 training epochs. Furthermore, the 1800 hyperparameter combinations were compiled by *Tensorboard*, Figure 3. Within these assessments, the disappearance of the gradient was observed several times, mainly related to the depth of the networks. We saw this behavior from the very learning curves that contained the behavior of a straight line.



☐ Show data download links
☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

☑ Dec05_10-49-51_dirac
☑ Dec05_11-12-49_dirac
☑ Dec05_12-40-27_dirac
☑ Dec05_13-11-27_dirac
☑ Dec05_15-48-25_dirac
☑ Dec05_17-41-10_dirac
☑ Dec05_20-03-02_dirac
☑ Dec05_22-39-49_dirac
☑ Dec06_00-34-26_dirac

Fig. 3. The *Tensorboard* compiles the learning curves in both the training and validation phases.

6 Final Considerations

The process of obtaining an architecture of neural networks is always a challenge for machine learning. In this work we evaluated 1800 models which, in addition to the delay in evaluating all of them, brought an unsatisfactory result, since the error associated with the validation phase was 3724.4. In addition to the poor result, recurring problems with underfitting were found, indicating that the architecture is still simple to generalize the problem, however, as we left the network deeper, the behavior known as gradient disappearance often occurred.

Understanding all the challenges found in the hyper parameter optimization process, we can point out future goals that will help us to evaluate the models even better. The first point is the use of a training stop criterion directed not only on the error variation within the validation data, but also using the gradient response criterion as an important criterion as well. The second point has a bias to optimize the training execution time, the training time of the models is extensive and it is also totally dependent on the size of the molecules, so it is essential to make use of optimized libraries for this type of problem ie. *cuDNN* [3], *cuGraph* and *Deep Graph Library* [8].

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng,

X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org

2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. CoRR **abs/1907.10902** (2019), http://arxiv.org/abs/1907.10902
3. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E.: cudnn: Efficient primitives for deep learning. CoRR **abs/1410.0759** (2014), http://arxiv.org/abs/1410.0759
4. Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. CoRR **abs/1509.09292** (2015), http://arxiv.org/abs/1509.09292
5. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
6. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry (4 2017), http://arxiv.org/abs/1704.01212
7. Hamilton, W.L.: Graph representation learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **14**(3), 1–159
8. Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., Zhang, Z.: Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv preprint arXiv:1909.01315 (2019)
9. Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., Leswing, K., Pande, V.S.: Moleculenet: A benchmark for molecular machine learning. CoRR **abs/1703.00564** (2017), http://arxiv.org/abs/1703.00564