Molecular Mechanics-Driven Graph Neural Network with Multiplex Graph for Molecular Structures

Shuo Zhang[1,2], **Yang Liu**[2], **Lei Xie**[1,2,3]
{sz780, yl1708, lei.xie}@hunter.cuny.edu
[1]The Graduate Center, CUNY [2]Hunter College, CUNY
[3]Weill Cornell Medicine, Cornell University

Abstract

When predicting molecular properties, previous Graph Neural Networks (GNNs) improve their expressive power by incorporating auxiliary information in molecules while inevitably increase their computational complexity. Motivated by this we design a powerful and efficient GNN for molecules. Inspired by molecular mechanics, our approach first represents each molecule as a multiplex graph. Then a message passing module is proposed to balance the trade-off of expression power and computational complexity. The performance of the resulting Multiplex Molecular Graph Neural Network (MXMNet) is successfully validated on the QM9 dataset.

1 Introduction

Recently, Graph Neural Networks (GNNs) have shown superior performance on predicting molecular properties by treating the molecules as graphs and performing the message passing scheme [1]. To increase the expressive power, previous GNNs have adopted auxiliary information such as chemical properties, pairwise distances between atoms, and angular information [2, 3, 4, 5, 6, 7, 8]. However, adopting such information in GNNs will inevitably increase the computational complexity. For example, when passing messages on a molecular graph that has N nodes with an average of k nearest neighbors for each node, $O(Nk^2)$ messages are required for the previous state-of-the-art GNNs [7, 8] to capture the angular information. With restricted memory resources, those GNNs could exhibit limited expressive power or even fail when applied to macromolecules like proteins or RNAs.

To address such limitation, we aim to propose a powerful and efficient GNN for molecules. Inspired by molecular mechanics [9], we will use the angular information to model only the local connections to avoid using expensive computations on all connections. To do so, we represent a molecule as a two-layer multiplex graph G as shown in Figure 1. In G, one layer G_l only contains the local connections that capture covalent interactions, and another layer G_g contains the global connections that cover the non-covalent interactions. Then a novel angle-aware message passing with $O(Nk^2)$ messages is operated on G_l. For G_g, we propose an efficient message passing with $O(Nk)$ messages. With the message passing schemes, we integrate them into a Multiplex Molecular (MXM) module and then construct Multiplex Molecular Graph Neural Network (MXMNet). To evaluate the performance of MXMNet, we conduct experiments on the QM9 dataset [10]. Our model can outperform the state-of-the-art models and require significantly less memory than DimeNet as shown in Figure 2.

2 Preliminaries

Let $G = (V, E)$ be a graph with $N = |V|$ nodes and $M = |E|$ edges. The nearest neighbors of node i are defined as $\mathcal{N}(i) = \{j|d(i, j) = 1\}$, where $d(i, j)$ is the shortest distance between node i and j. The average number of the nearest neighbors of each node is $k = 2M/N$. Next we provide the definition of a multiplex graph:

Machine Learning for Molecules Workshop at NeurIPS 2020. https://ml4molecules.github.io

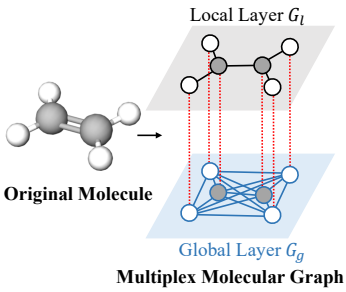

Figure 1: Example of representing a molecule as a multiplex molecular graph.

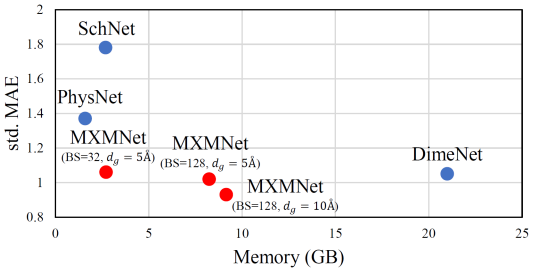

Figure 2: Mean std. MAE vs. memory consumption on QM9 dataset [10]. When compared with SchNet [5], PhysNet [6] and DimeNet [7], our MXMNet can get stat-of-the-art performance and is memory-efficient.

Definition 1. ***Multiplex Graph.*** *A multiplex graph* $G = \{G^1, G^2, ..., G^L\}$ *is a set of graphs, where* $G^l = (V, E^l), l \in \{1, 2, ..., L\}$ *is a layer.* V *is the node set and* E^l *is the edge set in the* l*-th layer.*

In our later formulations, we will use $\boldsymbol{e}_{ji}$ as the edge feature between node i and j, which embeds the pairwise distance, MLP as the multi-layer perceptron, $\|$ as the concatenation operation, $\odot$ as the element wise production and $\boldsymbol{W}$ as the weight matrix.

3 Approach

In this section, we introduce our approach including the multiplex molecular graphs, the Multiplex Molecular (MXM) module, and the Multiplex Molecular Graph Neural Network (MXMNet).

3.1 Multiplex Molecular Graphs

In molecular mechanics methods [9], the molecular energy E is modeled as $E = E_{\text{local}} + E_{\text{nonlocal}}$, where $E_{\text{local}} = E_{\text{bond}} + E_{\text{angle}} + E_{\text{dihedral}}$ models the local, covalent contributions including E_{bond} that depends on bond lengths, E_{angle} on bond angles, and E_{dihedral} on the dihedral angles. E_{nonlocal} models the non-local, non-covalent contributions between atom pairs. In such computations, **only** the E_{angle} and E_{dihedral} in the E_{local} term needs the angular information. As a contrast, DimeNet uses the angular information in the modeling of **both** local and non-local interactions and is inefficient. Inspired by molecular mechanics, the angular information will be used to model **only** the local interactions in our proposed model to reduce the computational complexity. To achieve our goal, we propose to represent the molecule as a *multiplex graph* $G = \{G_l, G_g\}$, which consists of a **local layer** G_l and a **global layer** G_g. The atoms are the nodes in both layers. In G_l, we can create edges by using either chemical bonds or finding the neighbors of each node within a small cutoff distance. In G_g, we create the edges by defining the neighbors of each node within a relatively large cutoff distance.

3.2 Multiplex Molecular (MXM) Module

With the multiplex molecular graph, we propose a Multiplex Molecular (MXM) module as shown in Figure 3(a) that performs massage passing on it to model the interactions in molecules. For G_g, we propose the **global layer message passing**. For G_l, we propose the **local layer message passing**. To transfer the information between different layers, we use a **cross layer mapping**.

Global Layer Message Passing Module By taking advantage of addressing high-order neighbors in GNNs [11, 12, 13], we here propose a message passing that captures *both* one-hop and two-hop neighbors per iteration. As illustrated in Figure 3(b), our global layer message passing module consists of two *same* message passing operations and an update function f_{u} using multiple residual modules (Figure 3(d)) between them. Each message passing operation is formulated as follows:

$$\boldsymbol{m}_{ji} = \text{MLP}([\boldsymbol{h}_j^{input} \,\|\, \boldsymbol{h}_i^{input} \,\|\, \boldsymbol{e}_{ji}]) \odot (\boldsymbol{e}_{ji}\boldsymbol{W}), \qquad \boldsymbol{h}_i^{output} = \boldsymbol{h}_i^{input} + \sum\nolimits_{j \in \mathcal{N}(i)} \boldsymbol{m}_{ji},$$

where $i, j \in G_{global}$, the superscripts denote the state of $\boldsymbol{h}$ in the operation. By performing the message passing operation twice in the module, each central node i will receive the messages from up to its two-hop neighbors. The resulting operation only needs $O(2Nk)$ messages.

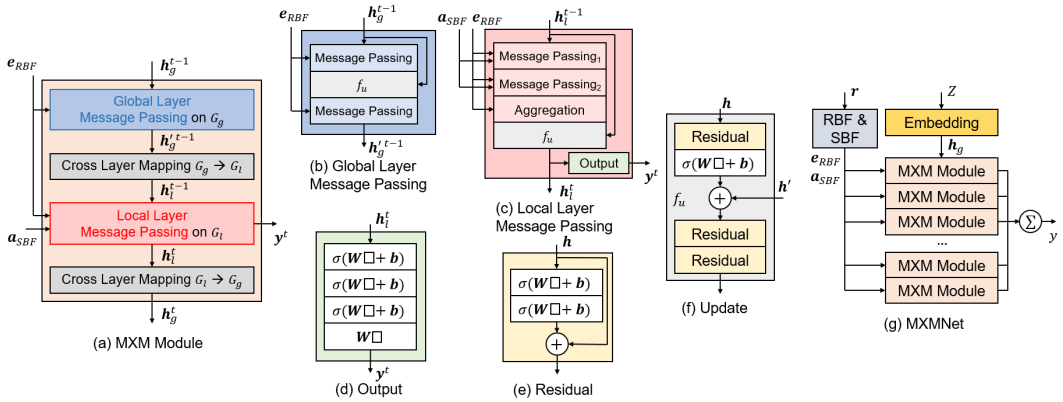

Figure 3: Overview of the architecture of the MXM module and the MXMNet. In the illustrations, σ denotes the non-linear transformation, $\square$ denotes the input for the layer.

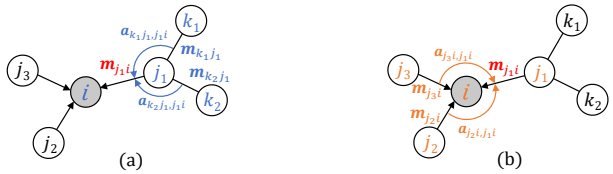

Figure 4: Illustration of Message Passing 1 and 2 used in the Local Layer Message Passing module. (a) Message Passing 1 can capture the angles like $\angle ij_1k_1$ and $\angle ij_1k_2$ when updating $\boldsymbol{m}_{j_1i}$. (b) Message Passing 2 can capture the angles like $\angle j_1ij_3$ and $\angle j_1ij_2$ when updating $\boldsymbol{m}_{j_1i}$.

Local Layer Message Passing Module In this module that performs message passing on the local layer, we propose a 3-step scheme: Step 1 contains Message Passing 1 that can capture the angles between 1-hop and 2-hop neighbors to update $\{\boldsymbol{m}_{ji}\}$ (see Figure 4(a)). Step 2 contains Message Passing 2 that can capture the angles between 1-hop neighbors to update $\{\boldsymbol{m}_{ji}\}$ (see Figure 4(b)). Step 3 finally aggregates $\{\boldsymbol{m}_{ji}\}$ to update $\boldsymbol{h}_i$. The t-th iteration can be formulated as follows:

$$\text{Step 1:} \quad \boldsymbol{m}_{kj}^{t-1} = \text{MLP}_{\text{kj}}([\boldsymbol{h}_k^{t-1} \,\|\, \boldsymbol{h}_j^{t-1} \,\|\, \boldsymbol{e}_{kj}]) \odot (\boldsymbol{e}_{kj}\boldsymbol{W}_{e1}) \odot \text{MLP}_{\text{a1}}(\boldsymbol{a}_{kj,ji}), \tag{1}$$

$$\boldsymbol{m}_{ji}^{t-1} = \text{MLP}_{\text{ji}}([\boldsymbol{h}_j^{t-1} \,\|\, \boldsymbol{h}_i^{t-1} \,\|\, \boldsymbol{e}_{ji}]) + \sum\nolimits_{k\in\mathcal{N}(j)\setminus\{i\}} \boldsymbol{m}_{kj}^{t-1}, \tag{2}$$

$$\text{Step 2:} \quad \boldsymbol{m}_{j'i}^{'t-1} = \text{MLP}_{\text{j'i}}(\boldsymbol{m}_{j'i}^{t-1}) \odot (\boldsymbol{e}_{j'i}\boldsymbol{W}_{e2}) \odot \text{MLP}_{\text{a2}}(\boldsymbol{a}_{j'i,ji}), \tag{3}$$

$$\boldsymbol{m}_{ji}^{'t-1} = \text{MLP}_{\text{ji}}^{'}(\boldsymbol{m}_{ji}^{t-1}) + \sum\nolimits_{j'\in\mathcal{N}(i)\setminus\{j\}} \boldsymbol{m}_{j'i}^{'t-1}, \tag{4}$$

$$\text{Step 3:} \quad \boldsymbol{h}_i^t = f_{\text{u}}(\sum\nolimits_{j\in\mathcal{N}(i)} \boldsymbol{m}_{ji}^{'t-1} \odot (\boldsymbol{e}_{ji}\boldsymbol{W}_{e3})), \tag{5}$$

where $i, j, k \in G_{local}$, $\boldsymbol{a}_{kj,ji}$ is the feature for angle $\alpha_{kj,ji} = \angle \boldsymbol{h}_k\boldsymbol{h}_j\boldsymbol{h}_i$. We define f_{u} using the same form as in the global layer message passing. These steps need $O(2Nk^2 + Nk)$ messages in total.

Cross Layer Mapping To address the relations between the same nodes across G_l and G_g, we use the following functions as in Figure 3(a): $\boldsymbol{h}_l = f_{cross}(\boldsymbol{h}_g)$ or $\boldsymbol{h}_g = f'_{cross}(\boldsymbol{h}_l)$, where $g \in G_{global}, l \in G_{local}$. We use multi-layer perceptrons to model the functions.

Complexity Analysis We denote the cutoff distance when creating the edges as d_g and d_l in G_g and G_l. The average number of the nearest neighbors per node is k_g in G_g and is k_l in G_l. For 3D molecules, we have $k_g \propto {d_g}^3$ and $k_l \propto {d_l}^3$. As $d_g > d_l$, we know that $k_g \gg k_l$. Our MXM module requires $O(2Nk_g + 2N{k_l}^2 + Nk_l)$ messages, which is much smaller than $O(N{k_g}^2)$ when performing DimeNet on G_g in the original work.

3.3 Multiplex Molecular Graph Neural Network (MXMNet)

With MXM module, we build Multiplex Molecular Graph Neural Network (MXMNet) for the prediction of molecular properties as shown in Figure 3(g). In the **Embedding module**, the atomic

Table 1: Comparison of MAEs of targets on QM9 for different models.

Target	SchNet	PhysNet	MEGNet-f	Cormorant	MGCN	DimeNet	**MXMNet** BS=32 d_g=5Å	**MXMNet** BS=128 d_g=5Å	**MXMNet** BS=128 d_g=10Å
μ (D)	**0.021**	0.0529	0.040	0.038	0.056	0.0286	0.0396	0.0382	0.0255
$\alpha(a_0^3)$	0.124	0.0615	0.083	0.085	**0.030**	0.0469	0.0447	0.0482	0.0465
ϵ_{HOMO} (meV)	47	32.9	38	34	42.1	27.8	24.7	23.0	**22.8**
ϵ_{LUMO} (meV)	39	24.7	31	38	57.4	19.7	19.7	19.5	**18.9**
$\Delta\epsilon$ (meV)	74	42.5	61	61	64.2	34.8	32.6	31.2	**30.6**
$\langle R^2 \rangle (a_0^2)$	0.158	0.765	0.265	0.961	0.11	0.331	0.512	0.506	**0.088**
ZPVE (meV)	1.616	1.39	1.40	2.027	**1.12**	1.29	1.15	1.16	1.19
U_0 (meV)	12	8.15	9	22	12.9	8.02	**5.90**	6.10	6.59
U (meV)	12	8.34	10	21	14.4	7.89	**5.94**	6.09	6.64
H (meV)	12	8.42	10	21	16.2	8.11	**6.09**	6.21	6.67
G (meV)	13	9.40	10	20	14.6	8.98	**7.17**	7.30	7.81
$c_v(\frac{\text{cal}}{\text{molK}})$	0.034	0.0280	0.030	0.026	0.038	0.0249	**0.0224**	0.0228	0.0233
std. MAE (%)	1.78	1.37	1.57	1.61	1.89	1.05	1.06	1.02	**0.93**

numbers Z are represented with randomly initialized, trainable embeddings to be the input node embeddings. In the **RBF & SBF module**, the Cartesian coordinates $\boldsymbol{r}$ of atoms are used to compute the pairwise distances and angles. We use the basis functions proposed in [7] to construct the representations of $\boldsymbol{e}_{RBF}$ and $\boldsymbol{a}_{SBF}$. Then we stack **MXM modules** to perform message passings. In each MXM module, we use an **Output module** to get the node-level output. The final prediction y is computed by summing all outputs together among all nodes and all layers.

4 Experiments

We examine our model on the QM9 dataset [10], which is a benchmark for the prediction of physical properties of molecules and consists of around 130k small organic molecules. We use the same way to split the data as in [7] and use the mean absolute error (MAE) to evaluate our model. We use the chemical bonds as the edges in the local layer, and a cutoff distance to create the edges in the global layer. In our experiments, we use the following state-of-the-art models as baselines: SchNet [5], PhysNet [6], MEGNet-full [14], Cormorant [15], MGCN [16] and DimeNet [7]. On QM9, we use the results reported in the original works for the baselines.

Results On the QM9 dataset, we test the performance of MXMNet under different configurations by changing the batch size BS and the cutoff distance d_g used in the global layer. As reported in Table 1, MXMNet variants get better results than the baselines on 9 targets. We also compute the mean standardized MAE (std. MAE) as used in [7] to evaluate the overall performance of the models. MXMNet (BS=128, $d_g = 10$Å) has the lowest std. MAE among all models and decreases the mean std. MAE by 13% compared to the previous best model DimeNet. By comparing the results of the MXMNet variants using different d_g, we find that different targets benefit from different d_g. Therefore, in practice, it is recommended to properly choose d_g for predicting different properties.

Efficiency Evaluation To evaluate the efficiency of MXMNet, we compare the memory consumption during the training on QM9 for SchNet, PhysNet, DimeNet and MXMNet. For the baselines, the model configurations are the same as those in their original papers. As illustrated in Figure 2, all of the three MXMNet variants use much smaller memory than DimeNet. Note that MXMNet can also benefit from large batch training with BZ=128 to achieve a speedup of the training to 2.6× against DimeNet that can only use BZ=32 on our GPU. For SchNet and PhysNet that consume less memory than MXMNet, they perform worse than MXMNet with higher mean std. MAEs.

5 Conclusion

In this paper, we propose a powerful and efficient GNN, MXMNet, for predicting the molecular properties. The novelty of MXMNet lies in its representation of molecules as a multiplex graph that is rooted in molecular mechanics. Experiments on the QM9 dataset have successfully demonstrated the power and efficiency of MXMNet compared with the state-of-the-art baselines.

References

[1] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[2] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[3] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[4] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272, 2017.

[5] KT Schutt, Pan Kessel, Michael Gastegger, KA Nicoli, Alexandre Tkatchenko, and K-R Muller. Schnetpack: A deep learning toolbox for atomistic systems. *Journal of chemical theory and computation*, 15(1):448–455, 2018.

[6] Oliver T Unke and Markus Meuwly. Physnet: A neural network for predicting energies, forces, dipole moments, and partial charges. *Journal of chemical theory and computation*, 15(6):3678–3693, 2019.

[7] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

[8] Zeren Shui and George Karypis. Heterogeneous molecular graph neural networks for predicting molecule properties. *arXiv preprint arXiv:2009.12710*, 2020.

[9] Tamar Schlick. *Molecular modeling and simulation: an interdisciplinary guide*, volume 21. Springer Science & Business Media, 2010.

[10] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[11] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4424–4431, 2019.

[12] Yiding Yang, Xinchao Wang, Mingli Song, Junsong Yuan, and Dacheng Tao. Spagan: Shortest path graph attention network. In *IJCAI*, pages 4099–4105, 2019.

[13] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. *arXiv preprint arXiv:1905.00067*, 2019.

[14] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, 2019.

[15] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pages 14537–14546, 2019.

[16] Chengqiang Lu, Qi Liu, Chao Wang, Zhenya Huang, Peize Lin, and Lixin He. Molecular property prediction: A multilevel quantum interactions modeling perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1052–1060, 2019.

A Appendix

B Details of the Experiments

B.1 Dataset Sources

For the QM9 dataset, we use the source[1] provided by [1]. Following the previous works [2, 3, 4, 5, 6], we process the QM9 dataset by removing about 3k molecules that fail a geometric consistency check or are difficult to converge [7]. For the properties U_0, U, H, and G, only the atomization energies are used by subtracting the atomic reference energies as in [8]. For the property $\Delta\epsilon$, we follow the same way as the DFT calculation and predict it by calculating $\epsilon_{\text{LUMO}} - \epsilon_{\text{HOMO}}$.

B.2 Baseline Sources

For the baselines used in the experiment on efficiency evaluation, we find their codes provided by the original papers are based on different frameworks: SchNet [3] is based on PyTorch [9], while PhysNet [4] and DimeNet [8] are based on Tensorflow [10]. To make fair comparisons and exclude the differences brought by different frameworks, we adopt the implementations of SchNet[2] and DimeNet[3] provided by the widely used PyTorch Geometric library [11] for graph representation learning. Since DimeNet is built based on PhysNet, by comparing their original implementations, we create the implementation of PhysNet based on[4] by changing the corresponding modules. Besides, the code of our MXMNet is also built based on[4].

B.3 Implementation Details

For the multi-layer perceptrons (MLPs) used in our MXMNet, they all have 2 layers to take advantage of the approximation capability of MLP [12]. For all activation functions, we use the self-gated Swish activation function [13]. For the basis functions $\boldsymbol{e}_{RBF}$ and $\boldsymbol{a}_{SBF}$, we use $N_{\text{SHBF}} = 7$, $N_{\text{SRBF}} = 6$ and $N_{\text{RBF}} = 16$. To initialize all learnable parameters, we use the default settings used in PyTorch without assigning specific initializations except the initialization for the input node embeddings $\boldsymbol{h}_g^{(0)}$ in the global layer: $\boldsymbol{h}_g^{(0)}$ are initialized with random values uniformly distributed between $-\sqrt{3}$ and $\sqrt{3}$.

In our experiment on QM9, we use the single-target training following [8] by using a separate model for each target instead of training a single shared model for all targets. The models are optimized by minimizing the mean absolute error (MAE) loss using the Adam optimizer [14]. We use a linear learning rate warm-up over 1 epoch and an exponential decay with ratio 0.1 every 600 epochs. The model parameter values for validation and test are kept using an exponential moving average with a decay rate of 0.999. To prevent overfitting, we use early stopping on the validation loss. Besides, we repeat our runs 3 times for each MXMNet variant following [15]. All of the experiments are done on a NVIDIA Tesla V100 GPU (32 GB).

In Table 1, we list the most important hyperparameters used in our experiments.

Table 1: List of hyperparameters used in our experiments on QM9.

Hyperparameters	Value / Range
Batch Size	32, 128
Hidden Dim.	12
Initial Learning Rate	1e-3, 1e-4
Number of Layers	6
Max. Number of Epochs	900
Dropout	0

[1] https://figshare.com/collections/Quantum_chemistry_structures_and_properties_of_134_kilo_molecules/978904

[2] https://github.com/rusty1s/pytorch_geometric/blob/73cfaf7e09/examples/qm9_schnet.py

[3] https://github.com/rusty1s/pytorch_geometric/blob/73cfaf7e09/examples/qm9_dimenet.py

C Ablation Study on QM9

To test whether our proposed two message passing modules will both contribute to the success of MXMNet, we conduct experiments by only using either of the two modules or even partial of a module. Table 2 shows that all those ablations will decrease the performance of MXMNet. These validate that both of the two message passing modules contribute to the power of MXMNet. Besides, when only using the global layer message passing module, the ablation with only one message passing operation performs worse than the ablation with two message passing operations. It shows the effectiveness of capturing the two-hop neighbors. When only using the local layer message passing module, the mean std. MAE increases significantly compared to the original MXMNet, suggesting that the local connections are not adequate for the task. The results also validate the power of our approach that captures more angles in Step 2 in the local layer message passing: The ablations with the Step 2 outperform the one without the Step 2, which contains the directional message passing that captures fewer angles.

Table 2: Comparison of mean std. MAEs of ablations that only contain parts of the MXM module. MXMNet cannot achieve the stat-of-the-art performance without any part of the MXM module.

Ablation		$\frac{\text{std. MAE}}{\text{std. MAE of MXMNet}}$
Only Global Layer Message Passing	One MP operation	116%
	Two MP operations	110%
Only Local Layer Message Passing	Step 1, 3	266%
	Step 2, 3	244%
	Step 1, 2, 3	224%

References

[1] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[2] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272, 2017.

[3] KT Schutt, Pan Kessel, Michael Gastegger, KA Nicoli, Alexandre Tkatchenko, and K-R Muller. Schnetpack: A deep learning toolbox for atomistic systems. *Journal of chemical theory and computation*, 15(1):448–455, 2018.

[4] Oliver T Unke and Markus Meuwly. Physnet: A neural network for predicting energies, forces, dipole moments, and partial charges. *Journal of chemical theory and computation*, 15(6):3678–3693, 2019.

[5] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, 2019.

[6] Chengqiang Lu, Qi Liu, Chao Wang, Zhenya Huang, Peize Lin, and Lixin He. Molecular property prediction: A multilevel quantum interactions modeling perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1052–1060, 2019.

[7] Felix A Faber, Luke Hutchison, Bing Huang, Justin Gilmer, Samuel S Schoenholz, George E Dahl, Oriol Vinyals, Steven Kearnes, Patrick F Riley, and O Anatole Von Lilienfeld. Prediction errors of molecular machine learning models lower than hybrid dft error. *Journal of chemical theory and computation*, 13(11):5255–5264, 2017.

[8] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[10] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[11] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[12] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[13] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2018.

[15] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pages 14537–14546, 2019.