Local Reference Frames on Geometric Graphs

Abstract

In this project we study the benefits and drawbacks of projecting the positional vectors of the nodes in geometric graphs onto a *Local Reference Frame*, a construction taken from the world of computer vision and graphics. We implement different methods for constructing an LRF and propose *LRF-MPNN*, a general framework for deep learning on geometric graphs which leverages the properties of nodes' positional vectors being projected onto the space defined by an LRF. We conduct experiments on the framework by using standard MPNNs and carry out ablation studies on the results and comparisons between the framework and existing models invariant to rotations and translations of the nodes.

1 Introduction

3D shape matching is an important task in computer graphics and vision, which is concerned with computing similarities between meshes and point clouds in 3D settings. Methods addressing the task rely on representing the local geometry of the objects through *local feature descriptors*. Most modern methods are deep learning-based which attempt to learn those descriptors, yet the traditional methods' design relies on the construction of a *Local Reference Frame (LRF)* of the object, an independent coordinate system which offers rotational invariance and 3D spatial description robust to noise [12].

More precisely, LRFs are orthogonal bases which are used to project the data on in order to obtain new features that are invariant to point cloud rotations and translations. There are two main types of methods for computing the LRF, namely those which rely on the computation of the *covariance matrix* of the point cloud and those concerned with *geometric attributes* of the object. We refer to the former methods as being based on *covariance analysis*, or *CA-based*. The covariance matrix can be defined in several ways and computing its eigenvectors yields a full PCA (Principal Component Analysis) of the whole point cloud, which is used to choose LRF.

Geometric graphs are graphs embedded in Euclidean space and are used to model systems from various domains such as biochemistry [7] or materials science [2]. Traditional GNNs struggle with capturing 3D information of the graph because of the lack of invariance to rotations and translations within its design, which make them ill-suited for the task of node or graph classification of geometric graphs [1]. Meanwhile, several well-performing GNN extensions which are roto-translation invariant have been designed, including *SchNet* [11] and *DimeNet* [3].

In this project, we leverage the properties of LRFs constructed via 2 CA-based methods by projecting the positional information of graph nodes onto them to obtain new positional features invariant to roto-translations which are then processed by an MPNN, which yields a new framework for learning on geometric graphs. We conduct experiments of these models on property prediction tasks found in the *QM9* benchmark [4], a dataset containing small organic molecules modelled as geometric graphs. We additionally analyse the LRF construction methods in the context of geometric graphs.

2 Background

Geometric Graphs and Graph Neural Networks. A geometric graph $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \vec{\mathbf{S}}, \vec{\mathbf{V}})$ is an attributed graph (where $\mathbf{X}$ denotes the node features matrix) equipped with a coordinate matrix $\vec{\mathbf{S}}$

and a geometric features matrix $\vec{\mathbf{V}}$. In this project we disregard the $\vec{\mathbf{V}}$ matrix and only focus on implementing GNN models by using the geometric positions of the nodes.

The t-th layer of an MPNN is given by the following equations:

$$\mathbf{h}_v^{(t)} = \mathcal{UPD}^{(t)}(\mathbf{h}_v^{(t-1)}, \mathbf{m}_v^{(t-1)})$$
$$\mathbf{m}_v^{(t)} = \mathcal{AGG}^{(t)}(\{\{(\mathbf{h}_v^{(t-1)}, \mathbf{h}_u^{(t-1)}) : (u, v) \in E\}\})$$

The $\mathcal{UPD}^{(t)}$ is the update function (usually represented by an *MLP*) and $\mathcal{AGG}^{(t)}$ is the aggregation function (an arbitrary function on multisets).

Local Reference Frames and Covariance Matrix. For a given point cloud $\mathcal{P}$, an LRF $\mathcal{L}(p)$ at point $p \in \mathcal{P}$ is defined as $\mathcal{L}(p) = \{\mathbf{x}(p), \mathbf{y}(p), \mathbf{z}(p)\}$, where $\mathbf{x}(p), \mathbf{y}(p), \mathbf{z}(p)$ are an orthogonal set of unit vectors satisfying the right-hand rule $\mathbf{y} = \mathbf{z} \times \mathbf{x}$.

The covariance matrix of a point cloud $\mathcal{P}$ is defined in [8] as:

$$\mathbf{C} = (\mathbf{S} - \mathbf{m}) \cdot (\mathbf{S} - \mathbf{m})^T$$

Here, $\mathbf{S} \in \mathbb{R}$ is the matrix of feature vectors where each column represents the coordinate vectors of each point $p \in \mathcal{P}$ and $\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{S}_i$ is the mean of the feature vectors found in $\mathcal{P}$ and is being subtracted column-wise from $\mathbf{S}$.

Informally, covariance measures how strongly correlated two or more variables (feature vectors) are, or how the variables vary from the mean with respect to each other. Since $\mathbf{C}$ is a 3×3 symmetric matrix, it has an orthonormal eigenbasis which represents the full set of Principal Components of $\mathcal{P}$ and can be chosen as our LRF.

Translating our point cloud does not change the direction of its PCs since we first center is around 0 before computing the LRF. Moreover, rotating the point in $\mathcal{P}$ by the a degree will rotate its PCs by the same degree. Informally speaking, projecting the rotated points onto our chosen LRF will not change the coordinates they have in the LRF basis.

3 LRF Message Passing Neural Networks

In this section, we describe the two methods used to construct the LRF of the point cloud corresponding to geometric graphs and then extend the class of MPNNs to geometric graphs embedded into its LRF space, refering to this generalization as *LRF-MPNN*.

CA-based methods for constructing LRFs. In the world of 3D shape recognition, constructing LRFs poses significant challenges inherent to image processing such as clutter, occlusion, noise or partial overlapping. Different LRF methods are thus benchmarked based on their repeatability and robustness to these issues.

Another problem inherent to the construction of LRFs using CA methods is the ambiguity of the sign of the resulting vectors. Calculating the PCs of a point cloud will result in an unique roto-invariant LRF up to the sense of vectors. Methods for computing the LRF use various sign disambiguation algorithms, yet they do not achieve perfect sign disambiguation.

The various methods in the literature have been proposed within local descriptors for tackling different challenges, some of the popular ones being point feature histograms [10], *RoPS* [5] and *SHOT*[13]. Fortunately, in the geometric graphs settings, we don't encounter most of the aforementioned challenges besides noise - i.e. small deviations in the position of the nodes. We are, thus, interested in methods which construct repeatable LRFs that are robust to the noise inherent in graphs. In this project we implement two methods. The former computes the covariance matrix as defined in [8] of the whole graph and applies it globally, while the latter computes the *SHOT* LRF at each point. The choice of the methods is based on the study conducted in [14] on the comparisons

between the different existing methods in the context of different tasks and challenges, which found that *SHOT* is the superior LRF method when it comes to disambiguating the sign and robustness to positional noise.

The *SHOT* method computer the covariance matrix at a point $p \in \mathcal{P}$ as follows:

$$\mathbf{C}(\mathbf{p}) = \frac{1}{\sum_{q \in \mathcal{N}(\mathbf{p})} w_q} \sum_{q \in \mathcal{N}(\mathbf{p})} w_q (\mathbf{q} - \mathbf{p})(\mathbf{q} - \mathbf{p})^T$$

Here, $\mathcal{N}(\mathbf{p})$ denotes the set of nodes that are inside the sphere of radius R centered at $\mathbf{p}$ and $w_q = R - \|q - p\|_2$. The weight w_q assigned to each point in $\mathcal{N}(\mathbf{p})$ get smaller the farther $\mathbf{q}$ is, which enhances the robustness under deviations of node positions by treating them based on how close to $\mathbf{p}$ they occur. Moreover, the method uses $\mathbf{p}$ as the barycenter to reduce computational complexity, yet our experiments show that the method performs significantly better when we use the mean of the feature vectors found in $\mathcal{N}(\mathbf{p})$, so our implementation uses this as the barycenter - that is because the former choice induces a lot of noise into the graph after being embedded into the new space.

Both methods we implement use the same sign disambiguation algorithm, which was proposed in [13] along with *SHOT* and works as follows:

Suppose the set of eigenvectors of $\mathbf{C}(\mathbf{p})$ are $\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ in decreasing order of their corresponding eigenvalues. The algorithm disambiguates the $\mathbf{x}$ as follows:

$$\mathbf{x} = \begin{cases} \mathbf{x} & |S_{\mathbf{x}}^+| \geq |S_{\mathbf{x}}^-| \\ -\mathbf{x} & \text{otherwise} \end{cases}, \text{where :}$$

$$S_{\mathbf{x}}^+ = \{\mathbf{q} \in \mathcal{N}(\mathbf{p}) \wedge (\mathbf{q} - \mathbf{p}) \cdot \mathbf{x} \geq 0\}$$
$$S_{\mathbf{x}}^- = \{\mathbf{q} \in \mathcal{N}(\mathbf{p}) \wedge (\mathbf{q} - \mathbf{p}) \cdot \mathbf{x} < 0\}$$

The $\mathbf{z}$ axis is disambiguated similarly and, finally, $\mathbf{y}$ is computed as $\mathbf{z} \times \mathbf{x}$.

LRF-MPNN To this end, we introduce *LRF-MPNN*. Crucially, we first project the positional data of the geometric graph into its corresponding LRF computed using any of the two methods above and feed the resulted data into a Geometric GNN which additionally takes a matrix $\mathbf{S}$ of coordinate vectors of the nodes.

The t-th layer of an LRF-MPNN is, thus, given by the following equations:

$$\mathbf{h}_v^{(t)}, \mathbf{s}_v^{(t)} = \mathcal{UPD}^{(t)}(\mathbf{h}_v^{(t-1)}, \mathbf{s}_v^{(t-1)}, \mathbf{m}_v^{(t-1)})$$
$$\mathbf{m}_v^{(t)} = \mathcal{AGG}^{(t)}(\{\{(\mathbf{h}_v^{(t-1)}, \mathbf{s}_v^{(t-1)}, \mathbf{h}_u^{(t-1)}, \mathbf{s}_u^{(t-1)}) : (u, v) \in E\}\})$$

Just like in the standard MPNN framework, $\mathcal{UPD}^{(t)}$ is the update function and $\mathcal{AGG}^{(t)}$ is the aggregation function.

4 Experiments

4.1 Setup

The setup is inspired from [6]. We conduct our experiments on LRF-MPNN using different methods by training and evaluating our model on a random subset of 10.000 graphs for the Dipole moment (μ) and Internal energy at 0K (U_0) regression tasks of *QM9*. We measure the accuracy using the mean absolute error (MAE) metric. At the end of the training process, we determine the model accuracy as being the test error corresponding with the smallest validation error computed throughout the training process. We compare LRF-MPNN when using the *SHOT* method and *Global* method to LRF-MPNN without any LRF computation and *SchNet*. For the *SHOT* method, we additionally consider different values of R. Testing on the value of R shows that for:

- $R = 5$: each node's neighbourhood is the whole graph
- $R = 3$: neighbourhoods start lacking some of the graph nodes
- $R = 1$: each node's neighbourhood is almost empty

See 1 for results.

As for hyperparameters, we train the model on 100 epochs with batch size of 256, using the Adam optimizer with a learning rate of 10^{-3}. We set the number of MPNN layers to 4, hidden layer dimension to 64. We use sum as aggregation function and a 2-layer MLP for the update and message functions. We additionally use residual connections after each MPNN layer, perform mean graph pooling at the end and apply a linear map to the result mapping to a single value.

Since the graphs in *QM9* are very sparse, we may encounter problems such as over-squashing. Thus, we convert all input graphs to being fully connected, where an edge has an edge embedding indicating the bond type if it is physically existing in the graph and has a zero edge embedding otherwise. Experiments validate that considering the graphs as fully-connected improves on the performance.

4.2 Ablation studies

In this subsection, we compare the two different methods in terms of hypotheses on their capability to transform the data original coordinates into a system that invariant to rotations and translations. We start by observing that both methods seem to be performing just as well but not to the level of *SchNet*.

Global method. The LRF is computed based on all nodes and applied to each node's positional vectors. It, thus, preserves their relative distances, which leads to nodes close to each other in the original space being close in the projected space.

Because of the sign ambiguity problem, some rotated and translated graph might get different features which may lead to a prediction different to the original graph's. This happens whenever $|S_{\mathbf{x}}^{+}| = |S_{\mathbf{x}}^{-}|$ for one of the basis vectors $\mathbf{x}$ in our LRF. We've conducted additional tests on the graphs from the *QM9* dataset by taking each graph, rotating it by a random degree and translating it by a random vector and checking whether projecting it to its LRF yields the same new features as the projected features of the original graph. Results show that the sign disambiguation algorithm selects a unique LRF for 81.54% of the graphs.

Local method (SHOT). Since the LRF is computed at each node in terms of its surroundings, two nodes have similar local structure will get the same new features up to sign. Note that, by local structure, we strictly mean the structure in the embedding space - that is, we don't take into account any graph-related structure. Since we are fully connecting all input graphs, nodes similar in this regard get to exchange information with one another even after the first MPNN layer, thus enhancing the overall performance.

Because of the sign ambiguity, in practice, this method is expected to provide less invariance to roto-translations than the Global method, since it is prone to calculate dissimilar new features for similar nodes and thus induce noise into the graph.

Moreover, as we decrease the radius R, the model starts underfitting the data, since the similar nodes will have more and more dissimilar neighbourhoods. We found that having R big enough to include for each node all the graph as its neighbourhood has the best performance.

Additional experiments. We have additionally attempted to leverage directionality as in [9] to enhance the results given the heterophily of molecular graphs. However, no performance boost was observed.

Table 1: Performance measured using test MAE on QM9 regression tasks

Model	μ	U_0
LRF-MPNN	0.543	116.208
LRF-MPNN + Global	0.441	37.476
LRF-MPNN + SHOT ($R = 5$)	0.461	35.946
LRF-MPNN + SHOT ($R = 1.5$)	0.707	46.864
SchNet	0.263	28.197

5 Conclusion

We introduced *LRF-MPNN*, a framework for geometric graphs where nodes are projected onto the LRF defined using two methods based on covariance analysis. We conducted experiments and found that LRF-MPNN performs significantly better than standard MPNNs on geometric graphs while being slightly worse than *SchNet*, while also providing an analysis on using these methods.

Future Work. For future work there should be more experiments performed. The framework has not been tested on many datasets in order to accurately assess its quality, but the results show promising applications of LRFs in the domain of graph neural networks. Additionally, we have not produced any theoretical result on when a method proves to be more beneficial than the other. LRFs retain positional information when using the Global method while capturing some form of structural information in the SHOT method. We have yet to investigate whether we can improve on these methods to also include graph-related information which could enhance deep learning on graphs.

Python code and setup. We provide the code written along with the project. In order to run the code, *python run.py* should be typed when inside the *source* file. The model being used and LRF method are hard coded into that file.

The recommended setup installations are the following:
conda create -n lrf python=3.10
conda activate lrf
conda install pytorch==2.1.0 pytorch-cuda=12.1 -c pytorch -c nvidia
pip install torch-scatter -f https://pytorch-geometric.com/whl/torch-2.1.0+cu121.html
pip install torch-sparse -f https://pytorch-geometric.com/whl/torch-2.1.0+cu121.html
pip install torch-geometric

References

[1] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. arXiv: 2104.13478 [cs.LG].

[2] Lowik Chanussot et al. "Open Catalyst 2020 (OC20) Dataset and Community Challenges". In: *ACS Catalysis* 11.10 (May 2021), pp. 6059–6072. ISSN: 2155-5435. DOI: 10.1021/acscatal.0c04525. URL: http://dx.doi.org/10.1021/acscatal.0c04525.

[3] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. *Directional Message Passing for Molecular Graphs*. 2022. arXiv: 2003.03123 [cs.LG].

[4] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. 2017. arXiv: 1704.01212 [cs.LG].

[5] Yulan Guo et al. "Rotational Projection Statistics for 3D Local Surface Description and Object Recognition". In: *International Journal of Computer Vision* 105.1 (Apr. 2013), pp. 63–86. ISSN: 1573-1405. DOI: 10.1007/s11263-013-0627-y. URL: http://dx.doi.org/10.1007/s11263-013-0627-y.

[6] *https://github.com/chaitjo/geometric-gnn-dojo/blob/main/geometric_gnn_101.ipynb*.

[7] Arian Jamasb, Pietro Lio, and Tom Blundell. *Graphein - a Python Library for Geometric Deep Learning and Network Analysis on Protein Structures*. July 2020. DOI: 10.1101/2020.07.15.204701.

[8] Ajmal Mian, Mohammed Bennamoun, and Robyn Owens. "On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes". In: *International Journal of Computer Vision* 89 (Sept. 2010), pp. 348–361. DOI: 10.1007/s11263-009-0296-z.

[9] Emanuele Rossi et al. *Edge Directionality Improves Learning on Heterophilic Graphs*. 2023.

[10] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. "Fast Point Feature Histograms (FPFH) for 3D registration". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[11] Kristof T. Schütt et al. *SchNet: A continuous-filter convolutional neural network for modeling quantum interactions*. 2017. arXiv: 1706.08566 [stat.ML].

[12] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. "A concise and provably informative multi-scale signature based on heat diffusion". In: *Proceedings of the Symposium on Geometry Processing*. SGP '09. Berlin, Germany: Eurographics Association, 2009, pp. 1383–1392.

[13] Federico Tombari, Samuele Salti, and Luigi Di Stefano. "Unique Signatures of Histograms for Local Surface Description". In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 356–369. ISBN: 978-3-642-15558-1.

[14] Jiaqi Yang, Yang Xiao, and Zhiguo Cao. "Toward the Repeatability and Robustness of the Local Reference Frame for 3D Shape Matching: An Evaluation". In: *IEEE Transactions on Image Processing* 27.8 (2018), pp. 3766–3781. DOI: 10.1109/TIP.2018.2827330.