Towards Interpretable Sparse Graph Representation Learning with Laplacian Pooling

Emmanuel Noutahi
InVivo AI
emmanuel@invivoai.com

Dominique Beani
InVivo AI
dominique@invivoai.com

Julien Horwood
InVivo AI, Mila
julien@invivoai.com

Prudencio Tossou
InVivo AI
prudencio@invivoai.com

Abstract

Recent work in graph neural networks (GNNs) has lead to improvements in molecular activity and property prediction tasks. However, GNNs lack interpretability as they fail to capture the relative importance of various molecular substructures due to the absence of efficient intermediate pooling steps for sparse graphs. To address this issue, we propose LaPool (Laplacian Pooling), a novel, data-driven, and interpretable graph pooling method that takes into account the node features and graph structure to improve molecular understanding. Inspired by theories in graph signal processing, LaPool performs a feature-driven hierarchical segmentation of molecules by selecting a set of centroid nodes from a graph as cluster representatives. It then learns a sparse assignment of remaining nodes into these clusters using an attention mechanism. We benchmark our model by showing that it outperforms recent graph pooling layers on molecular graph understanding and prediction tasks. We then demonstrate improved interpretability by identifying important molecular substructures and generating novel and valid molecules, with important applications in drug discovery and pharmacology.

1 Introduction

Following the recent rise of deep learning for image and speech processing, there has been great interest in generalizing convolutional neural networks to arbitrary graph-structured data [10, 14, 43]. To this end, graph neural networks (GNN), which fall into either spectral-based or spatial-based approaches, have been proposed. Spectral methods define the graph convolution (GC) as a filtering operator of the graph signal [7], while spatial methods define the GC as a message passing and aggregation across nodes [14, 15, 43]. In drug discovery, GNNs have been very successful across several molecular graph classification and generation tasks. In particular, they outperform predetermined molecular fingerprints and string-based approaches for molecular property prediction and *de novo* generation of drug-like compounds [15, 24].

However, the node feature update performed by most GNNs introduces some important limitations. Indeed, experimental results indicate a performance decrease for deeper GNNs due to the signal smoothing effect of each GC layer [23]. This limits the network's depth and restricts the receptive field of the vertices in the graph to a few-hop neighbourhood, which is insufficient to properly capture local structures, relationships between nodes, and subgraph importance in sparse graphs such as molecules. For example, at least three consecutive GC layers are needed for atoms at the opposite side of a benzene ring to exchange information. This issue is exacerbated by the single global pooling step performed at the end of most GNNs that ignores any hierarchical structure within the graph.

Preprint. Under review.

To cope with these limitations, graph coarsening (pooling) methods have been proposed to reduce graph size and enable long distance interaction between nodes. The first proposed methods relies solely on deterministic clustering of the graphs, making them non-differentiable and task-independent [5, 15, 27, 39]. In contrast, more recent methods use node features but are uninterpretable and, as we will show, are unable to preserve the structure of sparse graphs after pooling [8, 44].

Building on theory in graph signal processing, we propose LaPool (Laplacian Pooling), a differentiable pooling method that takes into account both the graph structure and its node features. LaPool performs a dynamic and hierarchical segmentation of graphs by selecting a set of centroid nodes as cluster representatives (leaders), then learns a sparse assignment of the remaining nodes (followers) into these clusters using an attention mechanism. LaPool is compared to other state-of-the-art methods in Table 1, with the primary contributions of this paper summarized below:

- Using established tools from graph signal processing (GSP), we propose a novel and differentiable pooling module (LaPool) that can be incorporated into existing GNNs to yield more expressive networks.
- We show that LaPool outperforms recently proposed graph pooling layers on discriminative and generative learning benchmarks for sparse molecular graphs.
- We performed a qualitative assessment of the pooling performed by LaPool to highlight its improved interpretability.

As shown in Figure 1, LaPool enables a better representation of molecular graphs given that the data-driven dynamic segmentation is closely linked to chemical fragmentation [12]. It is also the first GNN method to directly address the issue of model interpretability for sparse graphs.

Table 1: Properties of the proposed graph pooling method compared to state-of-the-art methods

Property	Spectral-Based Pooling *EigenPool*	Junction Tree	Graph U-Net	DiffPool	LaPool
Uses graph structure	✓	✓			✓
Data-driven			✓	✓	✓
Dynamic nb. of clusters		✓			✓
Interpretable	✓	✓			✓

1.1 Related Work

In this section, we introduce related work on graph convolutions (GC) and graph pooling, then provide an overview of techniques used in molecular screening and generation.

Increased interest in Graph Neural Networks (GNNs) has resulted in a variety of networks being proposed recently [10, 13, 14, 20, 43]. As our focus herein is on graph pooling, we refer the readers to [42] which reviews recent progress in the field and provides further connection with graph signal processing.

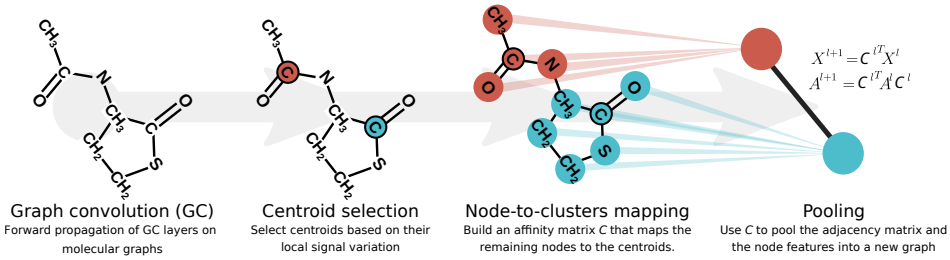
Virtual High-Throughput Screening (V-HTS) aims to accurately predict molecular properties directly from molecular structure. It can thus play an important role in the early stages of drug discovery by rapidly triaging the most promising compounds for any given indication, and can further assist during lead compound optimization [37]. Importantly, data-driven V-HTS approaches that leverage recent advances in deep learning rather than pre-determined features such as molecular fingerprints [33] and string representations have been shown to dramatically improve prediction accuracy [17, 41].

Advances in generative models for molecular graphs were enabled by deep generative techniques such as variational autoencoders (VAE) [19], generative adversarial networks (GAN) [11], and adversarial autoencoders (AAE) [28]. The first molecular generative models (e.g. Grammar-VAE [21]) resorted to generating string representations of molecules (via SMILES), which resulted in many invalid structures due to the complex syntax of SMILES. Graph generative models have since been developed (e.g. JT-VAE [15], GraphVAE [36], MolGAN [6], MolMP [24], etc.) and have been shown to improve the validity and novelty of generated molecules. In addition, these methods allow conditional molecule generation via Bayesian optimization or reinforcement learning [1, 15, 26, 30, 45].

Following the recent success of graph neural networks, Graph Pooling (GP) methods have been proposed to reduce graph size and increase the receptive field of nodes without increasing network depth. Contrary to the regular structure of images, graphs are irregular and complex, making it challenging to properly pool together nodes. Some graph pooling methods therefore rely on deterministic and non-differentiable clustering to segment the graph [7, 15]. In contrast, a differentiable pooling layer (DiffPool) was proposed in [44] to perform a similarity-based node clustering using an affinity matrix learned by GNN, while [8] proposed Graph U-net, a sampling method that retains a subset of the nodes at each pooling step but remains differentiable.

2 Graph Laplacian Pooling

Figure 1: Overview of the proposed LaPool method



A reliable pooling operator should maintain the overall structure and connectivity of a graph. LaPool achieves this by taking into account the local structure defined by the neighborhood of each node. As shown in Figure 1, the method uses a standard GC layer with a centroid selection and a follower selection step. First, the centroids of the graph are selected based on the local signal variation (see Section 2.2). Next, LaPool learns an affinity matrix C using a distance normalized attention mechanism to assign all nodes of the graph to the centroids (see Section 2.3). Finally, the affinity matrix allows for coarsening the graph into a smaller one. These steps are detailed below.

2.1 Preliminaries

Notation Let $G = \langle V, A, X \rangle$ be an undirected graph, where $V = \{v_1, \dots v_n\}$ is its vertex set, $A \in \{0,1\}^{n \times n}$ denotes its adjacency matrix, and $X = \left[\mathbf{x_1}, \dots \mathbf{x_n}\right]^\intercal \in \mathbf{R^{n \times d}}$ is the node feature matrix with each node v_i having d-dimensional feature x_i. The features X can also be viewed as a d-dimensional signal on G [35]. Without loss of generality we may assume a fixed ordering of the nodes that is respected in V, A, and X.

Graph Signal For any graph G, its unnormalized graph Laplacian matrix L is defined as $L = D - A$, where D is a diagonal matrix with D_{ii} being the degree of node v_i. The graph Laplacian is a difference operator and can be used to define the smoothness $s(X)$ (the extent at which the signal changes between connected nodes) of a signal X on G. For a 1-dimensional signal $\mathbf{f}$:

$$s(\mathbf{f}) = \mathbf{f}^\intercal \mathbf{L} \mathbf{f} = \frac{\mathbf{1}}{\mathbf{2}} \sum_{\mathbf{i,j}}^{\mathbf{n}} \mathbf{A_{i,j}} (\mathbf{f_i} - \mathbf{f_j})^\mathbf{2} \tag{1}$$

Graph Neural Networks We consider GNNs that act in the graph spatial domain as message passing [10]. We focus on the Graph Isomorphism Network (GIN) [43], which uses a SUM-aggregator on messages received by each node to achieve a better understanding of the graph structure :

$$x_i^l = \mathrm{M}_\Theta^l \left(x_i^{(l-1)} + \sum_{v_j \in \mathcal{N}(v_i)} A_{i,j}^{(l-1)} x_j^{(l-1)} \right) \tag{2}$$

where M_Θ^l is a neural network with trainable parameters Θ, x_i is the feature vector for node v_i, $v_j \in \mathcal{N}(v_i)$ are the neighbours of v_i and l is the layer number. Notice the term $A_{i,j}$ that takes into account the edge weight between nodes v_i and v_j when A is not a binary.

In this work, we focus on molecular graphs and mostly place ourselves in a supervised setting where, given a molecule m and its corresponding molecular graph G_m, we aim to predict some properties of m. Molecular graphs present two particularities: (1) they are often sparse and (2) there is no regularity in the graph signal (non-smooth variation) as adjacent nodes tend not to have similar features.

2.2 Graph Downsampling via Band-Pass Filtering

This section details how LaPool downsamples the original graph by selecting a set V_C of nodes as centroids after l consecutive GC layers.

Centroid Selection For any given vertex v_i, we can define a local measure s_i of signal intensity variation around v_i. As s_i measures how different a node is from the average of its neighbours, we are interested in the set V_C of nodes with the highest s_i, corresponding to the high frequencies of the signal.

$$s_i = \sum_{j \in \mathcal{N}(v_i)} A_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_2, \qquad S = \big[s_1, \dots s_n\big]^\intercal = \|L \cdot X\|_{\mathbf{R}^d} \qquad V_C = \text{top}_S(V, k) \tag{3}$$

Observe that the GC layers preceding each pooling step perform a smoothing of the graph signal and thus act as a low-pass filter. Combined with the high-pass filter of Eq. (3), it results in a band-pass filtering of X that attenuates low and high frequency noise, but retains the important signal in the medium frequencies. The intuition of using the Laplacian maxima for selecting the centroids is that a smooth signal can be very well approximated using a linear interpolation between its local maxima and minima. This is in contrast with most approaches in GSP that use the lower frequencies for signal conservation, but requires the signal to be k-bandlimited [2, 3, 27]. For a 1D signal, LaPool selects points, usually near the maxima/minima, where the derivative changes the most and is hardest to interpolate linearly (see Appendix C for further details). For molecular graphs, this corresponds to sampling a subset of nodes that are critical for reconstructing the original molecule.

Dynamic Selection of the Centroids The method presented in Eq. (3) implies the selection of k centroids. In contrast with other methods [8, 44], we do not use a fixed value of k because the optimal value can be graph-dependant and might result in densely located centroids. Instead, we dynamically choose k by selecting the nodes where the signal variation s_i is greater than its neighbours s_j:

$$V_C = \{v_i \in V \mid \forall\, v_j \in \mathcal{N}(v_i)\, s_i > s_j\} \tag{4}$$

2.3 Learning the Node-to-Cluster Assignment Matrix

Once the set V_C of centroid nodes is determined, we compute a mapping of the remaining "follower" nodes $V_F = V \setminus V_C$ into the new clusters formed by the nodes in V_C. This mapping gives the cluster assignment $C = \big[\mathbf{c}_1, ...\mathbf{c}_n\big]^\intercal \in [0,1]^{n \times m}$ s.t. $\forall i,\ \mathbf{1}\mathbf{c}_i^\intercal = 1$, where each row $\mathbf{c}_i$ corresponds to the affinity of node v_i towards each of the m clusters in V_C.

Let $X^{(l)}$ be the node embedding matrix at an arbitrary layer and $X_C^{(l)}$ the embedding of the "centroids". We compute C using a soft-attention mechanism [40] measured by the cosine similarity between $X^{(l)}$ and $X_C^{(l)}$:

$$\mathbf{c_i} = \begin{cases} \delta_{i,j} & \text{if } v_i \in V_C \\ \text{sparsemax}\Big(\beta_i \frac{\mathbf{x_i}^{(l)} \cdot X_C^{(l)}}{\|\mathbf{x_i}^{(l)}\| \|X_C^{(l)}\|}\Big) & \text{otherwise} \end{cases} \tag{5}$$

where $\delta_{i,j}$ is the Kronecker delta and sparsemax is an alternative to the softmax operator [22, 29], which ensures the sparsity of the attention coefficients and encourages the assignment of each node to a single centroid. This alleviates the need for entropy minimization as done by DiffPool.

Eq. (5) also prevents the selected centroid nodes from being assigned to other clusters. Moreover, notice the term β_i that regularizes the value of the attention for each node. We can define $\beta_i = \frac{1}{\mathbf{d_{i,V_C}}}$, where $\mathbf{d_{i,V_C}}$ is the shortest path distance between each node $v_i \in V_F$ and centroids in V_C. Although

this regularization incurs an additional $\mathcal{O}(|V|^2|V_C|)$ cost, it will strengthen the affinity to closer centroids.

We explored alternatives without this regularization or by restricting the mapping of each follower to centroids within a fixed k-hop neighborhood. The results for the various alternatives are presented jointly in Section 3.1.

Finally, after C^l is computed at layer l, the coarsened graph $G^{(l+1)} = \langle V^{(l+1)}, A^{(l+1)}, X^{(l+1)} \rangle$ is computed using Eq. (6), as in [44]. In these equations, M_Ψ is a neural network with trainable parameters Ψ that is used to update the embedding of nodes in $G^{(l+1)}$ after the mapping.

$$A^{(l+1)} = {C^{(l)}}^\intercal A^{(l)} C^{(l)} \in \mathbb{R}^{|V_C^{(l)}| \times |V_C^{(l)}|}, \qquad X^{(l+1)} = M_\Psi\left({C^{(l)}}^\intercal X^{(l)}\right) \tag{6}$$

This process can be repeated by feeding the new graph $G^{(l+1)}$ into another GNN layer.

2.4 Properties of the LaPool Method

Permutation Invariance It is trivial to show that LaPool is permutation invariant as long as the GNN used as its basis is permutation invariant, since both the graph downsampling (Eq. 3,4) and the node mapping (Eq. 5,6) are not affected by any permutation on the vertices set.

Information Preservation The centroid selection and the sparse distance-regularized node mapping ensure an appropriate segmentation of the graph and prevent the coarsened graph from being fully connected. Although LaPool enforces a hierarchical structure on the graph, it still preserves the information of the original graph after a global sum-pooling (GSUM-Pool).

Proposition 1. *Define, the structure-aware feature content of a graph $G = \langle V, A, X \rangle$ as $I(G) = AX$. Ignoring the feature update performed by LaPool, GSUM-Pool($I(LaPool(G))$) = GSUM-Pool($I(G)$).*

Proof. GSUM-Pool$(I(LaPool(G))) =$ GSUM-Pool$({C}^\intercal AC \cdot {C}^\intercal X) =$ GSUM-Pool$({C}^\intercal AX) = \mathbf{1}{C}^\intercal AX = \mathbf{1}AX =$ GSUM-Pool(I_G) □

Emphasizing the *Strong* Features Similar to how most CNNs implement a max-pooling layer to emphasize the *strong* features, LaPool does so by selecting the high Laplacian as leaders. For molecular graphs, the leaders are biased towards high degree nodes and atoms different than their neighbours (e.g. a Nitrogen in a Carbon ring).

3 Results and Discussion

A fundamental objective of LaPool is to learn an interpretable representation of sparse graphs, notably molecular substructures. We argue that this is an essential step towards building neural network models that adequately represent the distribution of molecular data. Indeed, beyond purely discriminative ability, a generative graph network should be able to re-construct molecular graphs from semantically important substructure components. This stems from the intuition that molecular validity and functional properties derive more from chemical fragments than individual atoms.

Our experimental results thus aim to empirically demonstrate the following properties of LaPool, as benchmarked against current state-of-the-art pooling models and the Graph Isomorphism Network.

- LaPool's consideration of semantically important information such as node distance translates to improved performance on molecule substructure prediction tasks
- Visualization of LaPool's behaviour at the pooling layer demonstrates its ability to identify coherent and meaningful molecular substructures
- A more coherent pooling layer may lead to better results for supervised tasks such as molecule toxicity prediction
- Learning meaningful substructures can be leveraged to construct a generative model which leads to more realistic and feasible molecules

We use the architecture depicted in Appendix A throughout our experiments (see Encoder's architecture). Furthermore, we note that minimal architectural tuning was performed given that the objective of our experiments is to maintain an even comparison across pooling models. We thus maintained an even network capacity across models, instead performing hyper-parameter tuning on pooling-specific variables. Specifically, we optimized over the number of clusters for DiffPool and Graph U-Net and over the Laplacian regularization and node neighbourhood parameters (β) for LaPool.

3.1 Substructure Prediction

While DiffPool and Graph U-Net models outperform standard graph convolution networks for supervised tasks on dense graphs [8, 44], we expect them to be ineffective at identifying important substructures on sparse graphs as they do not explicitly consider structural relationships. We wish to demonstrate this empirically by extracting known molecular substructure information from the publicly available [1]Tox21 and ChEMBL datasets [4, 9] and evaluating performance in identifying these structures. For the ChEMBL dataset, we use the same subset of approximately 17,000 molecules previously used in [25] for kinase activity prediction.

As shown in Tables 2 and 3, capturing these structural relationships translates to superior performance of LaPool, as measured across standard metrics on various substructure prediction tasks. We benchmark on different types of substructures and across datasets to verify the robustness of this comparison. We find that for predicting the presence of both 86 molecular fragments arising purely from structural information, as well as 55 structural alerts associated with molecule toxicity, LaPool globally outperforms other pooling models and a baseline GIN for the F1 (micro/macro averaged) and ROC-AUC metrics. The different versions of LaPool depicted in the results correspond to the regularization options for the cluster assignment described in 2.3. We note that the distance-regularized version of LaPool yields consistent high performance, suggesting that stronger affinity towards closer centroids often translates into an improved graph representation.

Table 2: Fragment prediction results.

	Tox21			ChEMBL		
	F1-macro	F1-micro	ROC-AUC	F1-macro	F1-micro	ROC-AUC
GIN	79.6	83.5	94.5	88.2	96.8	91.8
DiffPool	79.3	80.9	93.9	86.6	95.6	90.9
Graph U-net	72.1	72.3	88.3	77.3	87.0	82.9
LaPool$_{distance}$	**81.6**	**85.4**	95.0	**89.0**	**97.1**	91.9
LaPool$_{unregularized}$	80.3	84.2	**95.1**	88.8	96.6	**92.4**
LaPool$_{3-hop}$	80.7	86.1	**95.1**	87.7	96.4	92.2

Table 3: Structural alert prediction results.

	Tox21			ChEMBL		
	F1-macro	F1-micro	ROC-AUC	F1-macro	F1-micro	ROC-AUC
GIN	78.9	68.3	72.6	93.6	76.7	59.2
DiffPool	79.2	68.0	**75.6**	94.5	**83.3**	59.3
Graph U-net	71.1	47.6	67.9	92.9	68.1	59.3
LaPool$_{distance}$	80.6	**74.2**	73.5	**95.2**	81.3	**59.5**
LaPool$_{unregularized}$	**81.3**	72.8	74.1	94.1	75.8	58.9
LaPool$_{3-hop}$	79.1	71.6	74.8	93.8	75.0	59.1

3.2 Model Interpretability

To better understand the insights provided by LaPool, we investigate the behaviour of the network by plotting the clustering made at the pooling layer level. We believe this provides further insight by

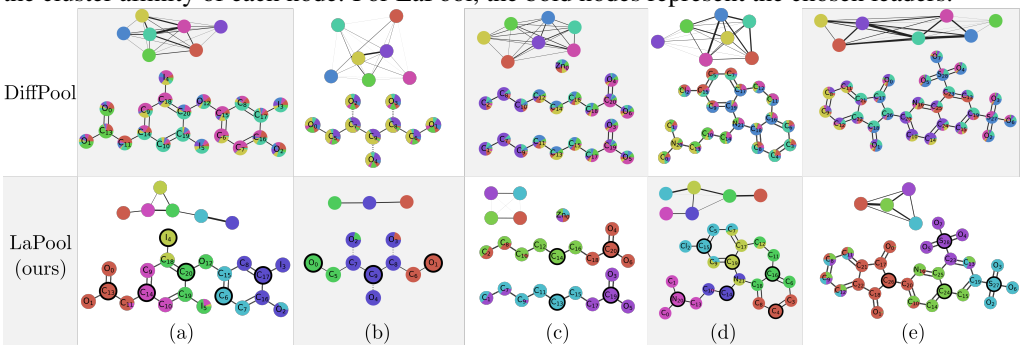
[1]These datasets and the full source code for LaPool and all experiments are available at (URL to appear for camera-ready version)

highlighting the improved explainability of LaPool compared to other pooling models on fragment prediction tasks.

By analyzing the relationship between the pooling layer used and the vertex-cluster attention, we may better understand why LaPool's pooling is preferable to current methods for identifying *meaningful* clusters. While defining what is meaningful is inherently subjective, we attempt to shed light on these models by observing their behaviour in the chemical domain, using our understanding of chemical structure as reference. We focus on DiffPool, since its the most similar method, and also because the node sampling performed by Graph U-net ignore the graph structure, usually disconnecting it.

In general, we show in Figure 2 that LaPool is able to coarsen the molecular graphs into robust, sparsely connected graphs, which can be interpreted as the *skeleton* of the molecules. In contrast, DiffPool's cluster assignment is much more uniform across the graph, leading to densely connected coarsened graphs which are less interpretable from a chemical viewpoint. Example (c) shows how DiffPool creates a fully connected graph from an originally disconnected graph, and example (b) shows that symmetric elements, despite being far from each other, are assigned identically. Such failures are not present for the proposed LaPool model. A typical failure case for LaPool is seen in (e) and corresponds to a missing leader node in a given region of the graph, which results in a soft assignment of the region to multiple clusters. However, this behaviour is inherent to most DiffPool samples since the fixed number of clusters and the inability to consider node distance cannot account for the diversity of the molecular datasets.

Figure 2: Visualization of the DiffPool and LaPool clustering for structural alert prediction. The top graph is the pooled graph. The bottom graph is the original molecule, with the pie-charts representing the cluster affinity of each node. For LaPool, the bold nodes represent the chosen leaders.



3.3 Toxicity Prediction

In addition to evaluating structural understanding of the pooling models, we benchmark our model on molecular toxicity prediction using the Tox21 dataset. As shown in Table 4, we demonstrate that the improved structural interpretation of LaPool's pooling mechanism may also lead to an improvement in molecular property prediction, a key performance metric for molecular optimization.

Table 4: Tox21 Prediction Results

	Tox21		
	F1-macro	F1-micro	ROC-AUC
GIN	59.0	24.5	76.6
DiffPool	59.7	24.3	79.9
Graph U-net	56.7	18.1	75.3
LaPool$_{distance}$	**61.7**	**31.0**	80.7
LaPool$_{unregularized}$	59.9	25.4	80.7
LaPool$_{3-hop}$	59.9	26.2	**81.8**

3.4 Molecular Generation

We showcase LaPool's utility in drug discovery by demonstrating that it can be leveraged to generate molecules. In previous work, GANs and VAEs were used to generate either string representations or molecular graphs. Here, we use the GAN-based Wasserstein Auto-Encoder recently proposed in [38] to model the data distribution of molecules (see Figure B.4 in Appendix). For the encoder, we use a similar network architecture as in our supervised experiments. The decoder and discriminator are simple MLPs, with complete architecture details provided in Appendix A.4. Even though the encoder is permutation invariant, the decoding process might not be. In our particular case, we use a canonicalization algorithm [34] that reorders atoms to ensure a unique graph for each molecule, thus forcing the decoder to learn a single graph ordering. Nevertheless, we further improve the robustness of our generative model to node permutations by computing the reconstruction loss using a permutation-invariant embedding, parameterized by a GIN, on both the input and reconstructed graphs (see Appendix A.4.2). We find that such a formulation improves the reconstruction loss and increases the ratio of valid molecules generated.

Dataset and Baseline Models Following previous work on molecular generation, we evaluate our generative model with an encoder enhanced by the LaPool layer (referred to as WAE-LaP) on the QM9 molecular dataset [32]. This dataset contains 133,885 small drug-like organic compounds with up to 9 heavy atoms (C, O, N, F). We compare WAE-LaP to alternatives within our WAE framework where either no pooling is used (WAE-GNN) or DiffPool is used as the pooling layer (WAE-Diff). Our results are also compared to previous results on the same dataset, including Grammar-VAE, GraphVAE, and MolGAN.

Evaluation Metrics We measure the performance of the generative model using metrics standard in the field: validity (proportion of valid molecules from generated samples), uniqueness (proportion of unique molecules generated), and novelty (proportion of generated samples not found in the training set). All metrics were computed on a set of 10,000 generated molecules.

Table 5: Performance comparison of the generative models on QM9. Values are reported in percentages and baseline results are taken from [6].

	Grammar-VAE	GraphVAE	MolGAN	WAE-GNN	WAE-Diff	WAE-LaP
% Valid	60.2	91.0	98.1	96.8	97.2	**98.8**
% Unique	9.3	24.1	10.4	50.0	29.3	**65.5**
% Novel	80.9	61.0	**94.2**	78.9	78.9	78.4

As shown in Table 5, WAE-LaP generated the most valid and unique molecules while MolGAN performed best on the novelty metric. Moreover, as we show in Appendix A.4, LaPool enables the generation of a more diverse set of molecules compared to DiffPool and a plain GNN. We observe that all WAE-based methods produced similar proportions of novel molecules, suggesting that combining LaPool with other generative approaches could improve the uniqueness and validity of generated compounds. We therefore argue that the pooling performed by LaPool can improve molecular graph representation, which is crucial in a generative setting.

4 Conclusion

Building on the literature for graph signal processing, we have derived LaPool, a novel, differentiable, and robust pooling operator for sparse molecular graphs. We have shown that LaPool considers both node information and graph structure during the graph coarsening process. By incorporating the proposed pooling layer into existing graph neural networks, we have demonstrated that the enforced hierarchization allows the resulting network to capture a richer and more relevant set of features at the graph-level representation. We discussed the performance of LaPool relative to existing graph pooling layers and demonstrated on three molecular classification benchmarks that LaPool outperforms existing graph pooling modules and produces more interpretable results. In particular, we argue that the molecular graph segmentation performed by LaPool provides greater insight into molecular activity and the associated properties that can be leveraged in drug discovery. Finally, we

briefly highlight how this new pooling layer can be used to facilitate *de novo* molecular design. In future work, we aim to further investigate the link between the steps performed by LaPool and the spectral domain of the graph, and how additional sources of information such as edge features could be incorporated into the process. Moreover, although we focused on molecular graphs, it would be of interest to evaluate the performance of LaPool on dense or highly structured graphs.

References

[1] Rim Assouel, Mohamed Ahmed, Marwin H. Segler, Amir Saffari, and Yoshua Bengio. DEFactor: Differentiable Edge Factorization-based Probabilistic Graph Generation. *arXiv:1811.09766 [cs]*, November 2018. URL `http://arxiv.org/abs/1811.09766`. arXiv: 1811.09766.

[2] Siheng Chen, Rohan Varma, Aliaksei Sandryhaila, and Jelena Kovačević. Discrete Signal Processing on Graphs: Sampling Theory. *IEEE Transactions on Signal Processing*, 63(24):6510–6523, December 2015. ISSN 1053-587X, 1941-0476. doi: 10.1109/TSP.2015.2469645. URL `http://arxiv.org/abs/1503.05432`. arXiv: 1503.05432.

[3] Siheng Chen, Rohan Varma, Aarti Singh, and Jelena Kovačević. Signal Representations on Graphs: Tools and Applications. *arXiv:1512.05406 [cs, math]*, December 2015. URL `http://arxiv.org/abs/1512.05406`. arXiv: 1512.05406.

[4] National Research Council et al. *Toxicity testing in the 21st century: a vision and a strategy*. National Academies Press, 2007.

[5] Shahaf Dafna and Carlos Guestrin. Learning Thin Junction Trees via Graph Cuts. In *Artificial Intelligence and Statistics*, pages 113–120, April 2009. URL `http://proceedings.mlr.press/v5/dafna09a.html`.

[6] Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[8] Hongyang Gao and Shuiwang Ji. Graph U-Net. *under review*, September 2018. URL `https://openreview.net/forum?id=HJePRoAct7`.

[9] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2011.

[10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. *arXiv:1704.01212 [cs]*, April 2017. URL `http://arxiv.org/abs/1704.01212`. arXiv: 1704.01212.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] Mark S Gordon, Dmitri G Fedorov, Spencer R Pruitt, and Lyudmila V Slipchenko. Fragmentation methods: A route to accurate calculations on large systems. *Chemical reviews*, 112(1):632–672, 2011.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[14] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep Convolutional Networks on Graph-Structured Data. *arXiv:1506.05163 [cs]*, June 2015. URL `http://arxiv.org/abs/1506.05163`. arXiv: 1506.05163.

[15] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction Tree Variational Autoencoder for Molecular Graph Generation. *arXiv:1802.04364 [cs, stat]*, February 2018. URL `http://arxiv.org/abs/1802.04364`. arXiv: 1802.04364.

[16] Artur Kadurin, Sergey Nikolenko, Kuzma Khrabrov, Alex Aliper, and Alex Zhavoronkov. drugan: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular pharmaceutics*, 14(9):3098–3104, 2017.

[17] 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.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. 2018.

[21] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1945–1954. JMLR. org, 2017.

[22] Anirban Laha, Saneem Ahmed Chemmengath, Priyanka Agrawal, Mitesh Khapra, Karthik Sankaranarayanan, and Harish G Ramaswamy. On Controllable Sparse Alternatives to Softmax. page 11, February 2016.

[23] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. *arXiv:1801.07606 [cs, stat]*, January 2018. URL `http://arxiv.org/abs/1801.07606`. arXiv: 1801.07606.

[24] Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of Cheminformatics*, 10(1):33, dec 2018. ISSN 1758-2946. doi: 10.1186/s13321-018-0287-6. URL `https://jcheminf.springeropen.com/articles/10.1186/s13321-018-0287-6`.

[25] Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 10(1):33, 2018.

[26] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning Deep Generative Models of Graphs. 2018. ISSN 2326-8298. doi: 10.1146/annurev-statistics-010814-020120. URL `http://arxiv.org/abs/1803.03324`.

[27] Yao Ma, Suhang Wang, Charu C. Aggarwal, and Jiliang Tang. Graph Convolutional Networks with EigenPooling. *arXiv:1904.13107 [cs, stat]*, April 2019. URL `http://arxiv.org/abs/1904.13107`. arXiv: 1904.13107.

[28] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[29] André F. T. Martins and Ramón Fernandez Astudillo. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. *arXiv:1602.02068 [cs, stat]*, February 2016. URL `http://arxiv.org/abs/1602.02068`. arXiv: 1602.02068.

[30] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1):48, September 2017. ISSN 1758-2946. doi: 10.1186/s13321-017-0235-x. URL `https://doi.org/10.1186/s13321-017-0235-x`.

[31] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Sergey Nikolenko, Alan Aspuru-Guzik, and Alex Zhavoronkov. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *arXiv preprint arXiv:1811.12823*, 2018.

[32] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1:140022, 2014.

[33] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[34] Nadine Schneider, Roger A Sayle, and Gregory A Landrum. Get your atoms in order - an open-source implementation of a novel and robust molecular canonicalization algorithm. *Journal of chemical information and modeling*, 55(10):2111–2120, 2015.

[35] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *arXiv preprint arXiv:1211.0053*, 2012.

[36] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[37] Sangeetha Subramaniam, Monica Mehrotra, and Dinesh Gupta. Virtual high throughput screening (vhts)-a perspective. *Bioinformation*, 3(1):14, 2008.

[38] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

[39] Ulrike von Luxburg. A Tutorial on Spectral Clustering. *arXiv:0711.0189 [cs]*, November 2007. URL `http://arxiv.org/abs/0711.0189`. arXiv: 0711.0189.

[40] Lilian Weng. Attention? Attention!, 2018. URL `https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html`.

[41] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[42] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? *arXiv:1810.00826 [cs, stat]*, October 2018. URL `http://arxiv.org/abs/1810.00826`. arXiv: 1810.00826.

[44] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4805–4815, USA, 2018. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=3327345.3327389`. event-place: Montréal, Canada.

[45] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, pages 6410–6421, 2018.

[46] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773*, 2018.

A Architecture search and hyperparameter selection

Below, we describe the network architecture and the training process used for the supervised and generative experiments.

A.1 Edge attributes

Some of the work presented assumes the absence of edge attributes in the graphs. However, in molecular graphs, the nature of a bond between two atoms plays an important role regarding activity and property. As such, edge types should be considered in the supervised models. To take this into consideration, we add to our network an initial Edge-GC layer that explicitly takes into account edge attributes. Let $G = \langle V, E, X \rangle$ be an undirected molecular graph, such that $E = [E_1, \dots E_k] \in \{0,1\}^{e \times n \times n}$ where n is the number of nodes in the graph and e is the number of possible edge. We have that

$$\sum_{1 \leq i \leq e} E_{::i} = A \in \{0,1\}^{n \times n} \tag{7}$$

where A is the adjacency matrix of the graph.

The Edge GC layer is defined as follows :

$$Y = M_{\Theta_1}(E_1, X) \| \dots \| M_{\Theta_e}(E_e, X) \tag{8}$$

where $\|$ is the concatenation operator on the node feature dimension and M_{Θ_1} are graph neural networks parameterized to learn different features for each edge type. A new graph defined as $G' = \langle V, A, Y \rangle$ can then be feed into the subsequent layers of the network.

A.2 Molecular node and edge attributes

In our experiments, the initial node feature tensor is represented by a one-hot encoding of 50 atoms (ignoring hydrogens which were removed) within the datasets and additional properties such as the atom implicit valence, its formal charge, number of radical electrons and whether it is in a molecular ring. This results in a 64-dimensional feature vector for each atom.

For edge attributes, we consider the single, double and triple bond, which were enough to cover all molecules of the database. Finally, we kekulize the molecules.

A.3 Supervised experiments

In all of our supervised experiments, we use a graph convolution module consisting of two graph convolutional layers of 64 channels each with identity connection and ReLU activation; followed by a graph pooling layer. This basic module is followed by two additional graph convolution layers and a global sum pooling to yield a graph-level representation (64). This is further followed two by fully connected layers (FCL) with 64 output channels; finalized by an FCL output layer for the task readouts. Except for the output layer for which a sigmoid activation function is used, we use the ReLU activation function for all other layers.

For DiffPool, we performed a hyperparameter search to find the optimal number of clusters (3, 5, 7, 9). Similarly, a search is also performed for the Graph-Unet pooling layer to determine the best number of node to retain (3, 5, 7, 9). The grid values were set in a way that appropriately reflects the size of the molecules in the datasets.

For LaPool, we performed a grid search over the window size k used as regularization to prevent nodes from mapping to centroids that are more than k-hop away and the regularization of the graph signal computed using the graph Laplacian. This latter regularization allows increasing the receptive field of each node when computing the signal variation. Because the signal variation at the nodes can be defined as $S = \|L \cdot X\|_d$, by taking $S = \|L^p \cdot X\|_d$, we can measure the signal variation at each node, considering nodes within its $p - hop$ neighborhood. The grid search was performed for $k \in \{0, 3\}$ (0 means no regularization) and $p \in \{1, 2, 3\}$.

For the supervised experiments, we use a batch size of 32 and train the networks for 100 epochs.

A.4 Generative models

A.4.1 WAE model

We use a Wasserstein Auto-Encoder (WAE) as our generative model (see Figure A.3. The WAE minimizes a penalized form of the Wasserstein distance between a model distribution and a target distribution. It allows using any reconstruction cost function and was shown to improve learning stability.

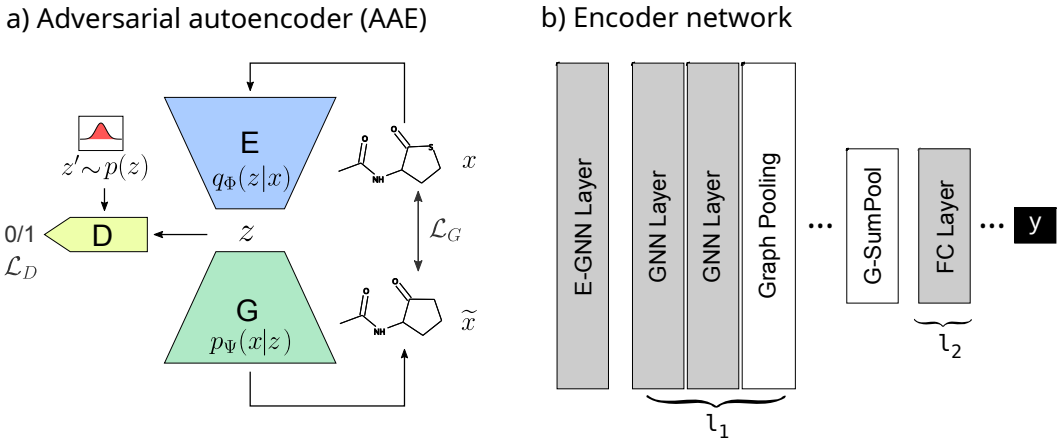


Figure A.3: Model architecture for the generative model. (a) We use a WAE, in which a generator (auto-encoder) progressively learn where the.

As described in [38], we aim to minimize the following objective:

$$\inf_{Q(Z|X)\in\mathcal{Q}} \mathbb{E}_{P_X}\mathbb{E}_{Q(Z|X)}[cost(X, G(Z)] + \lambda \mathcal{D}_Z(Q_Z, P_Z) \tag{9}$$

where $\mathcal{Q}$ is any nonparametric set of probabilistic encoders, D_Z is the Jensen-Shannon divergence between the learned latent distribution Q_Z and prior P_Z, and $\lambda > 0$ is a hyperparameter. D_Z is estimate using an adversarial training (discriminator).

For our generative model, the encoder follows a similar structure as the network used for our supervised experiments, with the exception being that the network now learns a continuous latent space $q_\Psi(z|\mathcal{G})$ given a set of input molecular graphs $\mathcal{G} = \{G_1, \cdots, G_n\}$. More precisely, it consists of one edge graph layer, followed by two GCs (32 channels each), an optional graph pooling, then two additional GC layers (64, 64), one global sum pooling step (128) and two FCLs (128), meaning the molecular graphs are embedded into a latent space of dimension 128. Following recent works for graph generation [1, 26, 46], we tried to model the nodes/edges decoding using an autoregressive framework, aiming to be better capture the interdependency between them. Given the latent code z, such decoding process will iteratively generate a continuous embedding of nodes using a recurrent neural network. However, our preliminary results suggested that this network converge slowly, and did not yield any substantial improvement in reconstruction accuracy compared to using a simple MLP. Therefore, during decoding, we use a simple MLP that takes the latent code z as input an pass it through two FCLs (128, 64). The output of those FCL will be used as shared embedding for two networks: one predicting the full edge tensor, and the second predicting the node features tensor. The network predicting the edge tensor (including edge types) E contains two stacked FCLs (64, 64) and an output layer that return the upper triangular entries of the tensor ($\frac{e \times n \times (n-1)}{2}$). The node feature network is a single FCL (32) followed by the output layer.

For the discriminator, we use a simple MLP that predicts whether the latent code comes from a normal prior distribution $z\, \mathcal{N}(0, 1)$. This MLP is constituted by two stacked FCLs (64, 32) followed by an output layer with sigmoid activation.

As in [16], we do not use batch-normalization, since it resulted in a mismatch between the discriminator and the generator.

All models use the same basic generative architecture, with the only difference being the presence of a pooling-layer and its associated parameters. For DiffPool, we fixed the number of cluster to three, while for LaPool, we use the distance-based regularization for the node-to-cluster mapping and no regularization when computing the node signal.

A.4.2 Reconstruction loss

For each input molecular graph $G = \langle V, E, X \rangle$, the decoder reconstruct a graph $\tilde{G} = \langle \tilde{V}, \tilde{E}, \tilde{X} \rangle$. We define the reconstruction loss $\mathcal{L}_{rec}$ as :

$$\mathcal{L}_{rec} = \mathcal{L}_E + \mathcal{L}_{\overline{E}} + \mathcal{L}_X \tag{10}$$

$$\mathcal{L}_E = -\frac{1}{|E|} \sum_{i,j|A_{ij}=1} E^{\intercal}_{:,i,j} \log\left(\tilde{E}_{:,i,j}\right) + (1 - E_{:,i,j})^{\intercal} \log\left(1 - \tilde{E}_{:,i,j}\right) \tag{11}$$

$$\mathcal{L}_{\overline{E}} = -\frac{1}{|\overline{E}|} \sum_{i,j|A_{ij}=0} \sum_{e} \log\left(1 - \tilde{E}_{e,i,j}\right) \tag{12}$$

$$\mathcal{L}_{\overline{X}} = -\frac{1}{|V|} \sum X^{\intercal} \log\left(\tilde{X}^{\intercal}\right) \tag{13}$$

where $\mathcal{L}_E$, $\mathcal{L}_{\overline{E}}$ and $\mathcal{L}_X$ are respectively the errors for reconstructing the edge type, properly prediction the absence of an edge, and reconstructing the node features.

Since we use a canonical ordering (available in rdkit) to construct G from the SMILES representation of molecules, the decoder is forced to learn how to generate a graph under this order. Therefore, the decoding process is not necessarily able to consider permutations on the vertices set, and generation of isomorphic graphs will be heavily penalized in the reconstruction loss. In [36], the authors use an expensive graph matching procedure to overcome that limitation. However, it suffices to compute the reconstruction loss on $\gamma(G)$ and $\gamma(\tilde{G})$, where γ is a permutation invariant embedding function. Since the Graph Isomorphism Network (GIN) was shown to be invariant to permutation in [43], we use an edge-aware GIN layer (see section A.1) with all weights initialized to 1 to embed both G and $\tilde{G}$. Then the reconstruction loss is defined as:

$$\mathcal{L}_{rec} = \frac{1}{|V|} \sum_{i} (\gamma(G)_i - \gamma(\tilde{G})_i)^2 \tag{14}$$

Our experiments show that this loss function was able to produce a higher number of valid molecules, although we speculate that such a function might prove harder to optimize on datasets with larger graphs.

A.4.3 Training procedure

The QM9 dataset was split into a train (60%), valid (20%) and a hold-out test dataset (20%). Only 25% of the training set sampled during each epoch, and in all experiments, we use a batch size of 32. The generator network (encoder-decoder) and the discriminator network are trained independently, using the Adam optimizer [18] with an initial learning rate of $1e-4$ for the generator and $1e-3$ for the discriminator. During training, we slowly reduce the learning rate by a factor of 0.5, for the generator, on plateau. To stabilize the learning process and prevent the discriminator from becoming "too good" at distinguishing the true data distribution from the prior, we train the generator two times more often.

B Molecule generation

Here we show how the graph pooling performed by LaPool (WAP-Lap) yields superior generative models compared to DiffPool (WAE-Diff) and no-pooling (WAE-GNN). We sample 5000 molecules for each generative model and use the MOSES benchmark [31] to access the overall quality of generated molecules by measuring the additional following metrics:

- Fragment similarity (Frag) and Scaffold similarity (Scaff) : the cosine distances between vectors of fragment/scaffold frequencies between the generated and the hold-out test sets.
- Nearest neighbor similarity (SNN): the average similarity of generated molecules to the nearest molecule from the test set.
- Internal diversity (IntDiv): the average pairwise similarity of generated molecules.

Table 6: Performance comparison of the generative models on the QM9 dataset.

	Valid	Unique	SNN/Test	Frag/Test	Scaf/Test	IntDiv
WAE-GNN	1.0	1.0	0.3292	0.474	**0.4761**	0.9203
WAE-Diff	1.0	1.0	0.3002	0.133	0.0531	0.9203
WAE-Lap	1.0	1.0	**0.3445**	**0.6356**	0.4393	0.9203

As shown in Table 6, WAE-Lap generates a more diverse set of molecules compared to WAE-Diff, suggesting that it learns a better hierarchical graph representation on molecular graphs. On Figure B.4 we highlight a few molecules generated by WAE-Lap.

C Signal preservation through Laplacian maxima

We illustrate here on a 1-d signal S, how using the Laplacian maxima serves to retain the most prominent regions of the graph signal, after smoothing (Figure C). We measure the energy conservation after downsampling:

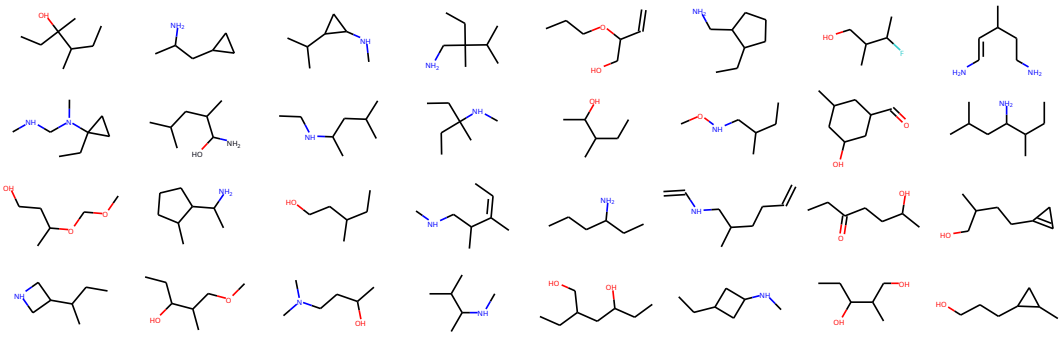


Figure B.4: Example of molecules generated by WAE-LaP. Hydrogen atoms are not shown for simplicity.

$\delta_E(S) = E(S) - E(S_{down})$ of the 1-d signal energy to highlight why selecting the Laplacian maxima allow reconstructing the signal with a low error when compared to the minimum Laplacian (which focuses on low frequencies). The energy E_S of a discrete signal y_i is defined in (15), and is similar to the energy of a wave in a physical system (without the constants).

$$E_S = \sum_i |y_i|^2 \tag{15}$$

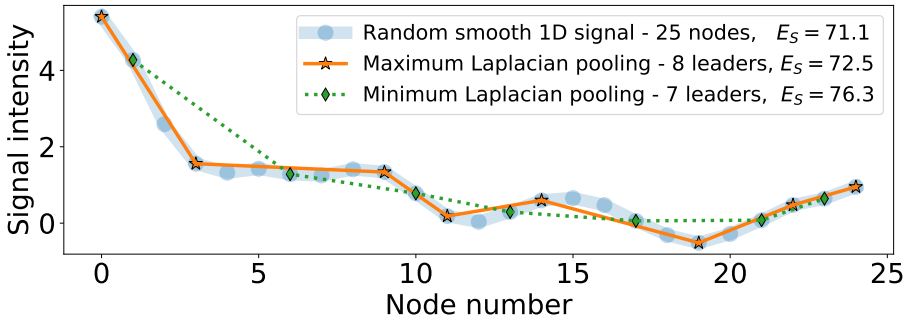


Figure C.5: Comparison of maximum/minimum Laplacian pooling for a random and smoothed signal on a 1D graph with 25 nodes. The graph energy E_S is indicated.

To mimic the molecular graph signal at the pooling stage, the given signal is built from an 8-terms random Fourier series with added Gaussian noise, then smoothed with 2 consecutive neighbor average smoothing. For the pooling methods, a linear interpolation is used to cover the same signal space before computing E_S. We observe that, for the given example, the maxima Laplacian selection seems to minimize the number of leaders required to preserve the signal and the energy, and significantly outperform minima selection.