Spatial Graph Convolutional Networks

Tomasz Danel[1][0000−0001−6053−0028], Przemysław Spurek[1][0000−0003−0097−5521], Jacek Tabor[1][0000−0001−6652−7727], Marek Śmieja[1][0000−0003−2027−4132], Łukasz Struski[1][0000−0003−4006−356X], Agnieszka Słowik[2][0000−0001−7113−4098], and Łukasz Maziarka[1][0000−0001−6947−8131]

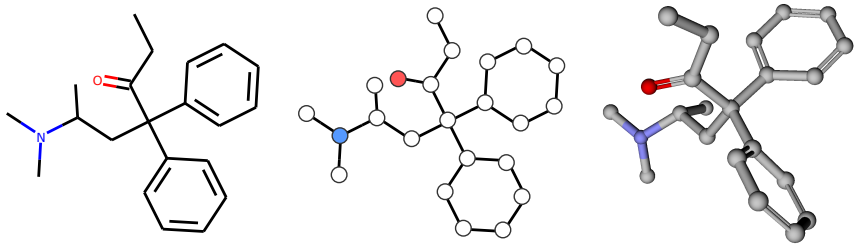
[1] Faculty of Mathematics and Computer Science, Jagiellonian University, Łojasiewicza 6, 30-428 Krakow, Poland
[2] Department of Computer Science and Technology, University of Cambridge, 15 JJ Thomson Ave, Cambridge CB3 0FD, UK

Abstract. Graph Convolutional Networks (GCNs) have recently become the primary choice for learning from graph-structured data, superseding hash fingerprints in representing chemical compounds. However, GCNs lack the ability to take into account the ordering of node neighbors, even when there is a geometric interpretation of the graph vertices that provides an order based on their spatial positions. To remedy this issue, we propose Spatial Graph Convolutional Network (SGCN) which uses spatial features to efficiently learn from graphs that can be naturally located in space. Our contribution is threefold: we propose a GCN-inspired architecture which (i) leverages node positions, (ii) is a proper generalization of both GCNs and Convolutional Neural Networks (CNNs), (iii) benefits from augmentation which further improves the performance and assures invariance with respect to the desired properties. Empirically, SGCN outperforms state-of-the-art graph-based methods on image classification and chemical tasks.

Keywords: Graph convolutional networks · Convolutional neural networks · Chemoinformatics.

1 Introduction

Convolutional Neural Network (CNNs) use trainable filters to process images or grid-like objects in general. They have quickly overridden feed-forward networks in computer vision tasks, and later also excelled in parsing text data [6], thanks to the small number of parameters and the locality of the extracted features. The convolutional architectures were applied to numerous visual learning tasks on which they outperformed humans, e.g. image classification [8], object detection [12] or image captioning [16]. However, CNNs can only be used to analyze tensor data in which local patterns are anticipated, e.g. images, text, and time series. One of the common data structures that does not conform to this requirement is graph, which can be used to represent, e.g. social networks, neural networks, city maps, and chemical compounds. In these applications, local patters may also play a key role in processing big graph structures. Borrowing from CNNs,



(a) Structural formula (b) Molecular graph (c) Exemplary conformation

Fig. 1: Representation of small molecules. (a) shows the structural formula of a compound (methadone). This notation is commonly used by chemists. (b) presents the molecular graph constructed from the structural formula. The vertices denote atoms, and bonds are represented by the undirected edges. (c) depicts a molecular conformation (one of the energetic minima), which is a 3D embedding of the graph.

Graph Convolutionl Networks (GCNs) use local filters to aggregate information from neighboring nodes [2,1]. However, most of these networks do not distinguish node neighbors and apply the same weights to each of them, sometimes modified by node degrees [7], edge attributes, or trainable attention weights [14].

In many cases, graphs are coupled with spatial information embedded in their nodes. For example, images can be transformed to graphs where nodes correspond to image pixels (color channels). In this case, each pixel has 2-dimensional coordinates, which define its position in the image. In chemical applications, molecules can be represented as graphs constructed from their structural formulas (Figure 1). Additionally, atoms (nodes in a molecular graph) organize themselves in the 3-dimensional space to reach the minimum energy state, and the shape of a compound is called a **molecular conformation**. Standard GCNs do not take spatial positions of the nodes into account, which is a considerable difference between GCNs and CNNs. Moreover, in the case of images, geometric features allow to augment data with translation or rotation and significantly enlarge the given dataset, which is crucial when the number of examples is limited.

In this paper, we propose Spatial Graph Convolutional Networks (SGCN), a variant of GCNs, which is a proper generalization of CNNs to the case of graphs. In contrast to existing GCNs, SGCN uses spatial features of nodes to aggregate information from the neighbors. On one hand, this geometric interpretation is useful to model many real examples of graphs, such as graphs of chemical compounds. In this case, it is possible to perform data augmentation by rotating a given graph in a spatial domain and, in consequence, improve network generalization when the amount of data is limited. Since typical graph convolutions do not take spatial features into account, data augmentation has not been possible in that case. On the other hand, a single layer of SGCN can be parametrized so

that it returns an output identical to a standard convolutional layer on grid-like objects, such as images (see Theorem 1).

The proposed method was evaluated on various datasets and compared with the state-of-the-art methods. SGCN was applied to classify images represented as graphs. The proposed method was also tested on chemical benchmark datasets. Experiments demonstrate that combining spatial information with data augmentation leads to more accurate predictions.

Our contributions can be summarized as follows:

- SGCN is proposed as a novel architecture of GCN that can effectively use additional geometric/spatial features to enhance a graph structure.
- The proposed architecture is a proper generalization of GCNs and CNNs, which is formally proven in Section 3.
- In contrast to the existing approaches, SGCN gives more control over the geometric structure, allowing to exploit the spatial arrangement of graph nodes and to use data augmentation to achieve better performance through the selective invariance of spatial properties, such as node ordering, rotations, or translations. Note that SGCN surpasses classical GCNs when at least one spatial property, e.g. node ordering, is relevant for the given task.

2 Spatial graph convolutional network

In this section, we introduce SGCN. First, we recall a basic construction of standard GCNs. Next, we present the intuition behind our approach and formally introduce SGCN. Finally, we briefly discuss practical advantages of spatial graph convolutions.

We use the following notation throughout this paper: let $\mathcal{G} = (V, \boldsymbol{A})$ be a graph, where $V = \{v_1, \ldots, v_n\}$ denotes a set of nodes (vertices) and $\boldsymbol{A} = [a_{ij}]_{i,j=1}^{n}$ represents edges. Let $a_{ij} = 1$ if there is a directed edge from v_i to v_j, and $a_{ij} = 0$ otherwise. Each node v_i is represented by a d-dimensional feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$. Typically, graph convolutional neural networks transform these feature vectors over multiple subsequent layers to produce the final prediction.

2.1 Graph convolutions

Let $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n]$ denote the matrix of node features being an input to a convolutional layer, where $\boldsymbol{h}_i \in \mathbb{R}^{d_{in}}$ are column vectors. The dimension of $\boldsymbol{h}_i$ is determined by the number of filters used in the previous layer. We denote as $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ the input representation for the first layer.

A typical graph convolution is defined by combining two operations. For each node v_i, feature vectors of its neighbors $N_i = \{j : a_{ij} = 1\}$ are first aggregated:

$$\bar{\boldsymbol{h}}_i = \sum_{j \in N_i} u_{ij} \boldsymbol{h}_j, \tag{1}$$

which could be also written in a matrix form as $\bar{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{H}^T$. Where the weights $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ are either trainable (e.g. [14] applied attention mechanism) or determined by adjacency matrix $\boldsymbol{A}$ (e.g. [7] motivated their selection using spectral graph theory).

Next, a standard MLP is applied to transform the intermediate representation $\bar{\boldsymbol{H}} = [\bar{\boldsymbol{h}}_1, \ldots, \bar{\boldsymbol{h}}_n]$ into the final output of a given layer:

$$\mathrm{MLP}(\bar{\boldsymbol{H}}; \boldsymbol{W}) = \mathrm{ReLU}(\boldsymbol{W}^T \bar{\boldsymbol{H}} + \boldsymbol{b}), \tag{2}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_{in} \times d_{out}}$ is a trainable weight matrix and $\boldsymbol{b} \in \mathbb{R}^{d_{out}}$ is a trainable bias vector (added column-wise). A typical graph convolutional neural network is composed of a sequence of graph convolutional layers (described above), see Figure 2. Next, its output is aggregated to the final response depending on a given task, e.g. node or graph classification.

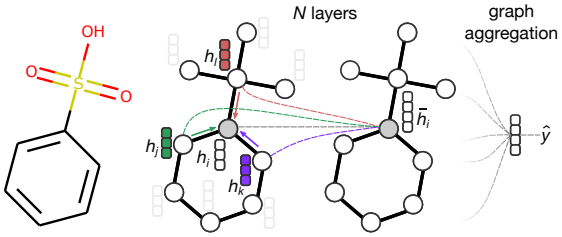


Fig. 2: An overview of the full network. A molecule is transformed to the graph representation and fed to the N consecutive (spatial) graph convolutional layers. In the figure, the convolution is demonstrated at the grey node – feature vectors of the adjacent nodes $\boldsymbol{h}_j$, $\boldsymbol{h}_k$, and $\boldsymbol{h}_l$ are aggregated together with the central node $\boldsymbol{h}_i$ to create a new feature vector $\bar{\boldsymbol{h}}_i$ for the grey node. In the proposed spatial variant, the relative positions of the neighbors are used in the aggregation (see Equation 3). At the end, all the node vectors are averaged, and the final prediction $\hat{y}$ is produced.

2.2 Spatial graph convolutions

In this section, the spatial graph convolutions are defined. The basic assumption is that each node v_i is additionally identified by its coordinates $\boldsymbol{p}_i \in \mathbb{R}^t$. In the case of images, $\boldsymbol{p}_i$ is the vector of two dimensional pixel coordinates, while for chemical compounds, it denotes location of the atom in thr two or three dimensional space (depending on the representation of chemical compound). In contrast to standard features $\boldsymbol{x}_i$, $\boldsymbol{p}_i$ is not changed across layers, but only used to construct a better graph representation. For this purpose, (1) is replaced by:

$$\bar{\boldsymbol{h}}_i(\boldsymbol{U}, \boldsymbol{b}) = \sum_{j \in N_i} \mathrm{ReLU}(\boldsymbol{U}^T(\boldsymbol{p}_j - \boldsymbol{p}_i) + \boldsymbol{b}) \odot \boldsymbol{h}_j, \tag{3}$$

where $\boldsymbol{U} \in \mathbb{R}^{t \times d}, \boldsymbol{b} \in \mathbb{R}^d$ are trainable parameters, d is the dimension of $\boldsymbol{h_j}$ and $\odot$ is element-wise multiplication. The pair $\boldsymbol{U}, \boldsymbol{b}$ plays a role of a convolutional filter which operates on the neighborhood of v_i. The relative positions in the neighborhood are transformed using a linear operation combined with nonlinear ReLU function. This scalar is used to weigh the feature vectors $\boldsymbol{h}_j$ in a neighborhood.

By the analogy with classical convolution, this transformation can be extended to multiple filters. Let $\boldsymbol{U} = [\boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(k)}]$ and $\boldsymbol{B} = [\boldsymbol{b}^{(1)}, \ldots, \boldsymbol{b}^{(k)}]$ define k filters. The intermediate representation $\bar{\boldsymbol{h}}_i$ is then a vector defined by:

$$\bar{\boldsymbol{h}}_i(\boldsymbol{U}, \boldsymbol{B}) = \bar{\boldsymbol{h}}_i(\boldsymbol{U}^{(1)}, \boldsymbol{b}^{(1)}) \oplus \cdots \oplus \bar{\boldsymbol{h}}_i(\boldsymbol{U}^{(k)}, \boldsymbol{b}^{(k)}),$$

where $\oplus$ denotes the vector concatenation. Finally, MLP transformation is applied in the same manner as in (2) to transform these feature vectors into new representation.

Equation 3 can be easily parametrized to obtain graph convolution presented in Equation 1. If all spatial features $\boldsymbol{p}_i$ are put to 0, then (3) reduces to:

$$\bar{\boldsymbol{h}}_i(\boldsymbol{U}, \boldsymbol{b}) = \sum_{j \in N_i} \mathrm{ReLU}(\boldsymbol{b}) \boldsymbol{h}_j.$$

This gives a vanilla graph convolution, where the aggregation over neighbors does not contain parameters. Different $\boldsymbol{b} = \boldsymbol{u}_{ij}$ can also be used for each pair of neighbors, which allows to mimic many types of graph convolutions.

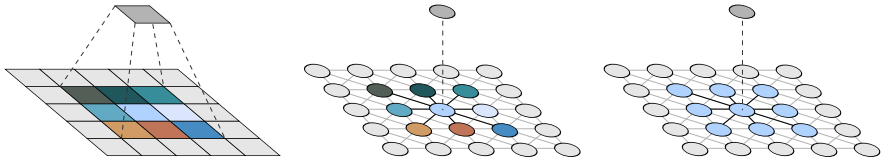
2.3 Data augmentation

In practice, the number of training data is usually too small to provide sufficient generalization. To overcome this problem, one can perform data augmentation to produce more representative examples. In computer vision, data augmentation is straightforward and relies on rotating or translating the image. Nevertheless, in the case of classical graph structures, analogical procedure is difficult to apply. This is a serious problem in medicinal chemistry, where the goal is to predict biological activity based only on a small amount of verified compounds. The introduction of spatial features and our spatial graph convolutions allow us to perform data augmentation in a natural way, which is not possible using only the graph adjacency matrix.

The formula (3) is invariant to the translation of spatial features, but its value depends on rotation of graph. In consequence, the rotation of the geometrical graph leads to different values of (3). Since in most domains the rotation does not affect the interpretation of the object described by such a graph (e.g. rotation does not change the chemical compound although one particular orientation may be useful when considering binding affinity, i.e. how well a given compound binds to the target protein), this property can be used to produce more instances of the same graph. This reasoning is exactly the same as in the classical view of image processing.

In addition, chemical compounds can be represented in many conformations. In a molecule, single bonds can rotate freely. Each molecule seeks to reach the minimum energy state, and thus some conformations are more probable to be found in nature than the other ones. Because there are multiple stable conformations, augmentation helps to learn only meaningful spatial relations. In some tasks, conformations may be included in the dataset, e.g. in binding affinity prediction, active conformations are those formed inside the binding pocket of a protein. Such a conformation can be discovered experimentally, e.g., through crystallization.

3 Relation between SGCN and CNNs



(a) CNN (b) SGCN (c) GCN

Fig. 3: Comparison of different neural convolutional filters. Each color denotes different trainable weights. (a) shows convolutional filters for images, (b) shows our spatial graph convolutions, and (c) depicts graph convolutions.

In contrast to typical GCNs, which consider graphs as relational structures, SGCN assumes that a graph can be coupled with a spatial structure, e.g. chemical compounds is a graph determined by atoms and bonds, which are embedded in 3D space. In particular, if we represent an image as a graph, where neighbor pixels are connected with edges, SGCN is capable of imitating the behavior of any CNNs operating on analogical image (see Figure 3). In other words, the formula (3) is constructed so that to parametrize any convolutional filter defined by classical CNNs. In this section, we first give a formal proof of this fact and next confirm this observation in an experimental study.

Theoretical findings. Let us introduce a notation concerning convolutions in the case of images. For simplicity only convolutions without pooling and with odd mask size are considered. In general, given a filter $\boldsymbol{F} = [f_{i'j'}]_{i',j' \in \{-k..k\}}$ its result on the image $\boldsymbol{H} = [h_{ij}]_{i \in \{1..N\}, j \in \{1..K\}}$ is given by

$$\boldsymbol{F} * \boldsymbol{H} = \boldsymbol{G} = [g_{ij}]_{i \in \{1..N\}, j \in \{1..K\}},$$

where

$$g_{ij} = \sum_{\substack{i'=-k..k:\, i+i' \in \{1..N\}, \\ j'=-k..k:\, j+j' \in \{1..K\}}} f_{i'j'} h_{i+i', j+j'}.$$

Theorem 1. *Let $\boldsymbol{H} \in \mathbb{R}^{N\times K}$ be an image. Let $\boldsymbol{F} = [f_{i'j'}]_{i',j'\in\{-k..k\}}$ be a given convolutional filter, and let $n = (2k+1)^2$ (number of elements of $\boldsymbol{F}$). Then there exist SGCN parameters: $\boldsymbol{U} \in \mathbb{R}^{2\times 1}$, $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}$, and $\boldsymbol{w} \in \mathbb{R}^n$ such that the image convolution can be represented as SGCN, i.e.*

$$\boldsymbol{F} * \boldsymbol{H} = \sum_{i=1}^{n} \boldsymbol{w}_i \bar{\boldsymbol{H}}(\boldsymbol{U}, \boldsymbol{b}_i).$$

Proof. Let $P \subset \mathbb{R}^2$ denote all possible positions in the convolutional filter $\boldsymbol{F}$, i.e. $P = \big\{[i', j']^T : i', j' \in \{-k, \ldots, k\}\big\}$. Let $\boldsymbol{u} \in \mathbb{R}^2$ denote an arbitrary vector which is not orthogonal to any element from $(P - P)$. Let also consider that $\boldsymbol{U} = [\boldsymbol{u}]^T$ and $\boldsymbol{b}_i = b_i$. Then:

$$\boldsymbol{u}^T\boldsymbol{p} \neq \boldsymbol{u}^T\boldsymbol{q}, \quad \text{for } \boldsymbol{p}, \boldsymbol{q} \in P : \ \boldsymbol{p} \neq \boldsymbol{q}.$$

Consequently, the elements of P may be ordered so that $\boldsymbol{u}^T\boldsymbol{p}_1 > \ldots > \boldsymbol{u}^T\boldsymbol{p}_n$. Let $\boldsymbol{F}_i$ denote the convolutional filter, which has value one at the position $\boldsymbol{p}_i$, and zero otherwise.

Let now choose b_i, such that:

$$b_i \in (-\boldsymbol{u}^T\boldsymbol{p}_i, -\boldsymbol{u}^T\boldsymbol{p}_{i+1}) \ \text{ for } i < n; \quad b_n > -\boldsymbol{u}^T\boldsymbol{p}_n.$$

For example one may take:

$$b_i = -\boldsymbol{u}^T\frac{\boldsymbol{p}_i + \boldsymbol{p}_{i+1}}{2} \ \text{ for } i < n; \quad b_n = -\boldsymbol{u}^T\boldsymbol{p}_n + 1.$$

Then observe that

$$\begin{aligned}\bar{\boldsymbol{H}}(\boldsymbol{U}, \boldsymbol{b}_1) &= (\boldsymbol{u}^T\boldsymbol{p}_1 + b_1) \cdot \boldsymbol{F}_1 * \boldsymbol{H},\\ \bar{\boldsymbol{H}}(\boldsymbol{U}, \boldsymbol{b}_2) &= \big[(\boldsymbol{u}^T\boldsymbol{p}_1 + b_2) \cdot \boldsymbol{F}_1 + (\boldsymbol{u}^T\boldsymbol{p}_2 + b_2) \cdot \boldsymbol{F}_2\big] * \boldsymbol{H},\end{aligned}$$

and generally for every $k = 1..n$ we get

$$\bar{\boldsymbol{H}}(\boldsymbol{U}, \boldsymbol{b}_k) = \left[\sum_{i=1}^{k} (\boldsymbol{u}^T\boldsymbol{p}_i + b_k)\boldsymbol{F}_i\right] * \boldsymbol{H},$$

where all the coefficients in the above sum are strictly positive.

Consequently,

$$\boldsymbol{F}_1 * \boldsymbol{H} = \frac{\bar{\boldsymbol{H}}(\boldsymbol{U}, \boldsymbol{b}_1)}{\boldsymbol{u}^T\boldsymbol{p}_1 + b_1},$$

and we obtain recursively that

$$\boldsymbol{F}_k * \boldsymbol{H} = \tfrac{1}{\boldsymbol{u}^T\boldsymbol{p}_k + b_k}\bar{\boldsymbol{H}}(\boldsymbol{U}, \boldsymbol{b}_k) - \tfrac{1}{\boldsymbol{u}^T\boldsymbol{p}_k + b_k}\sum_{i=1}^{k-1}(\boldsymbol{u}^T\boldsymbol{p}_i + b_k)\boldsymbol{F}_i * \boldsymbol{H},$$

which trivially implies that every convolution $\boldsymbol{F}_k * \boldsymbol{H}$ can be obtained as a linear combination of $(\bar{\boldsymbol{H}}(\boldsymbol{U}, \boldsymbol{b}_i))_{i=1..k}$.

Since an arbitrary convolution $\boldsymbol{F} = [f_{ij}]$ is given by $\boldsymbol{F} = \sum_{i=1}^{n} f_{p_i}\boldsymbol{F}_i$, we obtain the assertion of the theorem. □

Experimental verification. To experimentally demonstrate the correspondence between CNNs and SGCN, we consider the well-known MNIST dataset. To construct its graph representation, each pixel is mapped to a graph node, making a regular grid with connections between adjacent pixels. The node has 2-dimensional location, and it is characterized by a 1-dimensional pixel intensity. To show further capabilities of SGCN, we also consider an alternative representation [10], in which nodes are constructed from an irregular grid consisting of 75 superpixels. The edges are determined by spatial relations between nodes using k-nearest neighbors.

For a comparison, we report the results from the literature by state-of-the-art methods used to process geometrical shapes: ChebNet [1], MoNet [10], SplineCNN [3] and GAT [14]. In the first case of regular grid representation, SGCN is also compared to CNN with an analogical architecture, i.e. number of filters etc..

The results presented in Table 1 show that SGCN outperforms comparable methods on both variants on MNIST dataset. Its performance is slightly better than SplineCNN, which reports state-of-the-art results on this task. We also get higher accuracy than CNN, which confirms experimentally that SGCN is its proper generalization.

Table 1: Classification accuracy on two graph representations of MNIST.

Method	**Grid**	**Superpixels**
CNN	99.21%	-
ChebNet	99.14%	75.62%
MoNet	99.19%	91.11%
SplineCNN	99.22%	95.22%
GAT	99.26%	95.83%
SGCN	**99.61%**	**95.95%**

4 Experiments: a case study in predicting molecular properties of chemical compounds

In this section, we take into account graphs representing chemical compounds and perform a large scale experimental verification on real-life tasks.

Experimental setting. Three datasets were chosen from MoleculeNet [15], which is a benchmark for molecule-related tasks. Blood-Brain Barrier Permeability (BBBP) is a binary classification task of predicting whether or not a given compound is able to pass through the barrier between blood and the brain, allowing the drug to impact the central nervous system. Another two datasets, ESOL and FreeSolv, are solubility prediction tasks with continuous targets.

The datasets contain small molecules which are translated to the molecular graphs as it was presented in Figure 1. At each node, there is a constructed feature vector describing the atom, which includes the atom type, hybridization, formal charge, number of heavy neighbors, number of attached hydrogens, whether or not the atom is in a ring, and whether or not it is aromatic. To predict atom positions in 3D space, we use universal force field (UFF) method from the RDKit package, which finds such a conformation of compound that minimizes the energy of molecule. Since UFF is not deterministic we run it a few times (up to 30) and additionally augment the data with random rotations. Datasets are split into train, validation and test subsets according to the MoleculeNet standards. A random search is run for all models testing 100 hyperparameter sets for each of them. All runs are repeated 3 times.

SGCN is benchmarked against popular chemistry models: graph-based models (Graph Convolution [2], Weave Model [5], and Message Passing Neural Network [4]). We also use classical methods such as random forest and SVM, which do not operate on graph but rather they use a vector representation of chemical compound (ECFP fingerprint [11] of size 1024). In addition, EAGCN [13] and MAT [9] are included in the experiment, both of them using an attention mechanism. As for our method, the results are shown with train- and test-time augmentation of the data carried out in the manner described above[3]. In order to investigate the impact of the positional features, the atom representation of the classical graph convolutional network is also enriched with the predicted atom positions, and the same procedure of augmentation is applied. We name this enriched architecture pos-GCN and include it in the comparison.

Table 2: Performance on three chemical datasets measured with ROC AUC for BBBP and RMSE for ESOL and FreeSolv datasets. Best mean results and intervals overlapping with them are bolded. For the first column higher is better, for the second and the third lower is better.

Method	BBBP	ESOL	FreeSolv
SVM	0.603 ± 0.000	0.493 ± 0.000	0.391 ± 0.000
RF	0.551 ± 0.005	0.533 ± 0.003	0.550 ± 0.004
GC	0.690 ± 0.015	0.334 ± 0.017	0.336 ± 0.043
Weave	0.703 ± 0.012	0.389 ± 0.045	0.403 ± 0.035
MPNN	0.700 ± 0.019	0.303 ± 0.012	**0.299 ± 0.038**
EAGCN	0.664 ± 0.007	0.459 ± 0.019	0.410 ± 0.014
MAT	0.711 ± 0.007	0.330 ± 0.002	**0.269 ± 0.007**
pos-GCN	0.696 ± 0.008	0.301 ± 0.011	**0.278 ± 0.024**
SGCN	**0.743 ± 0.004**	**0.270 ± 0.005**	**0.299 ± 0.033**

[3] For all datasets, slight improvements can be observed with the augmented data.

Results. The results presented in Table 2 show that for the first two datasets, SGCN outperforms all tested models by a significant margin, i.e. the difference between SGCN and other methods is statistically significant. In the case of FreeSolv dataset the mean results obtained by SGCN is slightly worse than MAT and pos-GCN, but this difference is not statistically significant due to the relatively high variance. We emphasize that FreeSolv is extremely tiny dataset with only 513 examples, which makes it difficult to reliably compare the methods.

It is evident from the experiments that including positional features consistently improves the performance of the models across all tasks. For the smallest dataset, FreeSolv, pos-GCN even surpasses the score of SGCN. Nevertheless, learning from bigger datasets requires a better way of managing positional data, which can be noted for ESOL and BBBP datasets for which pos-GCN performs significantly worse than SGCN but still better than the vanilla GC.

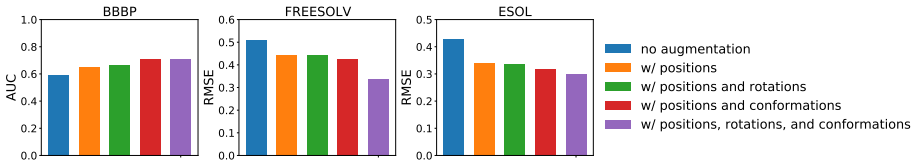


Fig. 4: Comparison of different augmentation strategies on three chemical datasets. No augmentation is a pure GCN without positions. In the conformation variant multiple conformations were precalculated and then sampled during training. Rotation augmentation randomly rotates molecules in batches. For the first bar-plot higher is better, for the second and the third lower is better.

Ablation study of the data augmentation. In the case of CNNs, data augmentation is often used to extend the dataset through random image transformations, but for GCNs there are no extensive studies on data augmentation as they would involve graph transformations of some sort. In contrast to classical GCNs, the proposed SGCN allows to introduce data augmentation to enlarge the given dataset by transforming the geometry of a graph. In this experiment, we investigate the influence of data augmentation on the SGCN performance.

First, we examined how removing predicted positions, and thus setting all positional vectors to zero in Equation 3, affects the scores achieved by our model on chemical tasks. The results are depicted in Figure 4. It clearly shows that even predicted node coordinates improve the performance of the method. On the same plot we also show the outcome of augmenting the data with random rotations and 30 predicted molecule conformations, which were calculated as described above. It occurs that the best performing model uses all types of position augmentation.

Eventually, we study the impact of various levels of augmentation. For this purpose, we precalculate 20 molecular conformations on the BBBP dataset using

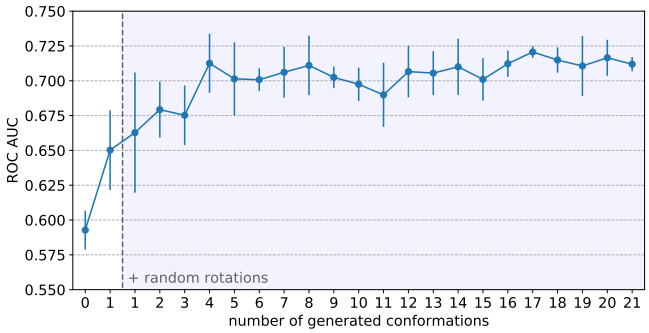


Fig. 5: ROC AUC scores achieved on the BBBP dataset by models using a different quantity of data augmentation. The first two models on the left (before the dashed line) are a standard GCN without node positions and SGCN with only one conformation (without any data augmentation) respectively. The models on the right of the dashed line are augmented with random rotations of a molecule. The amount of augmentation increases from left to right.

the UFF method and use them to augment the dataset. To test the importance of conformation variety, each run the number of available conformations to sample from is increased. The results are presented in Figure 5. One can see that including a bigger number of conformations helps the model to achieve better results. Also, the curve flattens out after a few conformations, which may be caused by the limited flexibility of small compounds and high similarity of the predicted shapes. It should be noted that data augmentation brings a huge improvement to the model, with more than 0.06 increase of the ROC AUC, which is enough to beat other models in the benchmark presented in the previous section.

5 Conclusion

We proposed SGCN which is a general model for processing graph-structured data with spatial features. Node positions are integrated into our convolution operation to create a layer which generalizes both GCNs and CNNs. In contrast to the majority of other approaches, our method can effectively use added information about location to construct self-taught feature masking, which can be augmented to achieve invariance with respect to the desired properties. Furthermore, we provide a theoretical analysis of our spatial graph convolutions. The experiments confirm the strong performance of our method.

It is also apparent in the benchmarks that both the local spatial features and graph-induced dependencies are needed to fully capture the nature of data in the presented setups. In the experiments, it was presented that not only does the information about conformations benefit the training, but also the means of

processing the spatial information of local neighborhood are crucial to a strong performance.

References

1. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems. pp. 3844–3852 (2016)
2. Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. In: Advances in neural information processing systems. pp. 2224–2232 (2015)
3. Fey, M., Eric Lenssen, J., Weichert, F., Müller, H.: Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 869–877 (2018)
4. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1263–1272. JMLR. org (2017)
5. Kearnes, S., McCloskey, K., Berndl, M., Pande, V., Riley, P.: Molecular graph convolutions: moving beyond fingerprints. Journal of computer-aided molecular design **30**(8), 595–608 (2016)
6. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
7. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
9. Maziarka, Ł., Danel, T., Mucha, S., Rataj, K., Tabor, J., Jastrzębski, S.: Molecule attention transformer. Workshop on Graph Representation Learning, Neural Information Processing Systems 2019 (2019)
10. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5115–5124 (2017)
11. Rogers, D., Hahn, M.: Extended-connectivity fingerprints. Journal of chemical information and modeling **50**(5), 742–754 (2010)
12. Seferbekov, S.S., Iglovikov, V., Buslaev, A., Shvets, A.: Feature pyramid network for multi-class land segmentation. In: CVPR Workshops. pp. 272–275 (2018)
13. Shang, C., Liu, Q., Chen, K.S., Sun, J., Lu, J., Yi, J., Bi, J.: Edge attention-based multi-relational graph convolutional networks. arXiv preprint arXiv:1802.04944 (2018)
14. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
15. Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., Leswing, K., Pande, V.: Moleculenet: a benchmark for molecular machine learning. Chemical science **9**(2), 513–530 (2018)
16. Yang, L., Tang, K., Yang, J., Li, L.J.: Dense captioning with joint inference and visual context. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2193–2202 (2017)