Seminar Report: Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs

Tobias Valinski and Supervisor Benjamin Busam

Technical University Munich

1 Introduction

While Convolutional Neural Networks (CNNs) are currently state-of-the-art in the Machine Learning community when it comes to classifying images or audio with near-perfect results, they lack the ability to be applied to other structured data like graphs and manifolds. The reason is that graphs and manifolds are non-euclidean. Therefore standard, grid-like convolutional filters cannot be applied to those kinds of data.
But since applying CNNs to non-euclidean data has many useful applications, a few teams of researchers tried to find solutions for this problem. The following section will give a short overview over the most promising approaches.

2 Existing Approaches

2.1 Graphs

Spectral CNN Bruna et al. tried to find a way to apply standard CNNs to graphs by using a Fourier Transformation. By transforming the input signals to the frequency domain, it is possible to apply standard frequency filters before transforming the result back to get the desired outcome.

$$f_l^{out} = \xi \left(\sum_{l'=1}^{p} \Phi_k \hat{G}_{l,l'} \Phi_k^\top f_{l'}^{in} \right). \tag{1}$$

This formula shows how the output signals are computed. $f_{l'}^{in}$ represents the input signals on the vertices. This input gets transformed to the frequency domain by applying the forward Fourier Transformation $\Phi_k^\top$ before applying the learnable filter $\hat{G_{l,l'}}$ and transforming everything back to "graph space" with Φ_k.
Several approaches, that research teams came up with after Spectral CNN was published, are based on this framework. Nevertheless this framework has two big disadvantages:
Since $\hat{G}$ is basis dependent, Spectral CNN cannot be trained on one graph and applied to another. Additionally the Fourier transforms are expensive in terms of computation time ($\mathcal{O}(n^2)$)

ChebNet In order to improve the computational effort of Spectral CNN Defferard et al. defined the spectral filters in a recursive form:

$$g_\alpha(\Delta) = \sum_{j=0}^{r-1} \alpha_j T_j(\hat{\Delta}) = \sum_{j=0}^{r-1} \alpha \Phi T_j(\hat{\Lambda}) \Phi^\top \tag{2}$$

where T_j is defined as

$$T_j(\lambda) = 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda). \tag{3}$$

By using this filter, the computational effort changes from explicitly applying two Fourier Transforms to applying the Laplacian r times, where r is the dimensionality of the filter's parameter α. As a result this reduces the cost of calculating such a convolution to $\mathcal{O}(rn)$.

Conclusion ChebNet was an improvement compared to Spectral CNN but the problem of not being able to generalize between different graphs is still not solved. This is an important property to have for classifying growing graphs (e.g. social networks)

2.2 Manifolds

GCNN Masci et al. developed a framework where patches of a manifold are extracted before applying a convolutional kernel to those patches. The patches are created by a patch operator of the form

$$(D(x)f) = \int_\chi w_{\rho,\theta}(x,y) f(y) dy \tag{4}$$

where w describes a non-learnable weighting function, x and y are points on the manifold and $f(y)$ is a function on the manifold. These weighted patches are then used to do a geodesic convolution:

$$(f \star g)(x) = \max_{\Delta\theta \in [0,2\pi)} \int_0^{2\pi} \int_0^{\rho_{max}} g(\rho, \theta + \Delta\theta)(D(x)f)(\rho,\theta) d\rho d\theta. \tag{5}$$

The learned filter $g(\rho,\theta)$ is applied to the extracted patch $(D(x)f)$ multiple times with different rotations. In the end the maximum of all rotations is taken to resolve the ambiguity between them.

ACNN A follow-up work tried to improve the used patch operator by allowing anisotropic patches. Therefore they use the anisotropic diffusion equation

$$f_t(x,t) = -div_\chi(A(x)\nabla_\chi f(x,t)) \tag{6}$$

where the conductivity tensor A is defined as

$$A_{\alpha\theta}(x) = R_\theta(x) \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} R_\theta^\top(x). \tag{7}$$

Using a anisotropic diffusion as the weighting function of a patch results in an anisotropic patch whose shape can be controlled by varying α for the patch's elongation (where $\alpha = 1$ results in an isotropic patch), θ for the rotation of the patch and t for its scale.

Conclusion Both GCNN and ACNN do not suffer from the problem of the presented graph approaches, where the trained CNN cannot be used on different domains. Even though both methods showed good results when compared to handcrafted approaches, their accuracy can be further improved.

3 The new approach

3.1 The Framework

All presented existing approaches use predefined patch operators. Monti et al. tried to improve those operators by declaring them in a new way:

$$D_j(x)f = \sum_{y \in \mathcal{N}(x)} w_j(u(x,y))f(y), \quad j = 1, ..., J. \tag{8}$$

This new patch operator uses a parameterized weighting function $w_\Theta(u)$ where Θ represents some learnable parameters and $u(x, y)$ are pseudo-coordinates between points x and y. This results in a learnable patch operator so that the non-euclidean convolution

$$(f \star g)(x) = \sum_{j=1}^{J} g_j D_j(x) f \tag{9}$$

consists of both a learnable kernel g_j and a learnable patch D_j.
The importance of this framework becomes clear when defining w_j (including $u(x, y)$) in certain ways. Table 1 shows, that by doing so all presented existing approaches and even a standard euclidean CNN can be generated with this new framework.

Method	Pseudo-coordinates	$\mathbf{u}(x,y)$	Weight function $w_j(\mathbf{u}), j = 1, \ldots, J$
CNN [23]	Local Euclidean	$\mathbf{x}(x,y) = \mathbf{x}(y) - \mathbf{x}(x)$	$\delta(\mathbf{u} - \bar{\mathbf{u}}_j)$
GCNN [26]	Local polar geodesic	$\rho(x,y), \theta(x,y)$	$\exp(-\frac{1}{2}(\mathbf{u} - \bar{\mathbf{u}}_j)^\top \begin{pmatrix} \bar{\sigma}_\rho^2 & \\ & \bar{\sigma}_\theta^2 \end{pmatrix}^{-1} (\mathbf{u} - \bar{\mathbf{u}}_j))$
ACNN [7]	Local polar geodesic	$\rho(x,y), \theta(x,y)$	$\exp(-\frac{1}{2}\mathbf{u}^\top \mathbf{R}_{\bar{\theta}_j} \begin{pmatrix} \bar{\alpha} & \\ & 1 \end{pmatrix} \mathbf{R}_{\bar{\theta}_j}^\top \mathbf{u})$
GCN [21]	Vertex degree	$\deg(x), \deg(y)$	$\left(1 - \lvert 1 - \frac{1}{\sqrt{u_1}} \rvert\right)\left(1 - \lvert 1 - \frac{1}{\sqrt{u_2}} \rvert\right)$
DCNN [3]	Transition probability in r hops	$p^0(x,y), \ldots, p^{r-1}(x,y)$	$\mathrm{id}(u_j)$

Table 1. Source: [Paper]

3.2 MoNet

Monti et al. defined $w_j(u)$ as

$$w_j(u) = exp(-\tfrac{1}{2}(u-\mu_j)^\top \Sigma_j^{-1}(u-\mu_j)) \tag{10}$$

where Σ_j is the covariance matrix and μ_j is the mean vector of a Gaussian kernel (both of which are learned parameters). Therefore the patch operator D_j using this weighting function uses J gaussian distributions which results in $(f \star g)$ being a Gaussian Mixture Model.

4 Results

4.1 MNIST

They first tested their new CNN model in a well known environment: The MNIST dataset of handwritten digits and their classifications. In order to apply MoNet to a grid-like structure, they interpreted the image as a regular graph as seen in Figure 1.

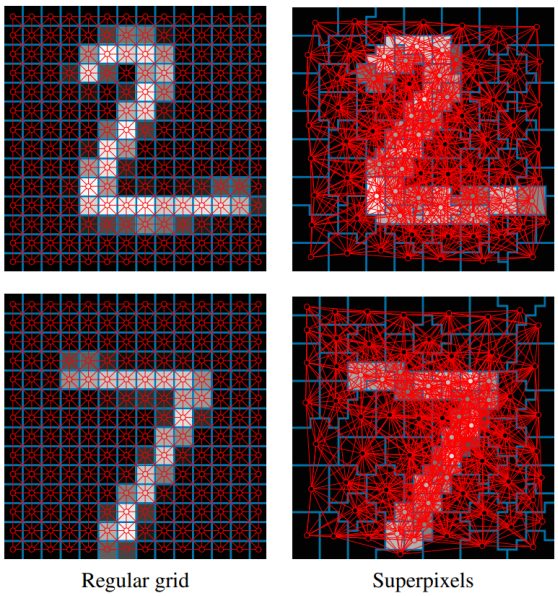

Figure 1. Source: [Paper]

To demonstrate the weakness of other graph CNNs like ChebNet, they also used a superpixel representation of the images. This causes the underlying graphs to be irregular and vastly different between different images.

Dataset	LeNet5	ChebNet	**MoNet**
Full grid	99.33%	99.14%	99.19%
$\frac{1}{4}$ grid	98.59%	97.70%	98.16%
300 Superpixels	-	88.05%	**97.30%**
150 Superpixels	-	80.94%	**96.75%**
75 Superpixels	-	75.62%	**91.11%**

Table 2. Source: [Paper]

The result of this image classification experiment can be seen in Table 2. LeNet is a state-of-the-art approach for euclidean data. On regular grid-like images, all tested approaches show similar results only with minor differences between them. When tested on superpixel images though, ChebNet suffers when applied to images with less pixels more than MoNet. This is due to the fact, that the underlying graphs differ more between images the less superpixel there are.

4.2 Cora and PubMed

Their second experiment was done on the datasets "Cora" and "PubMed" both of which consists of classified citation graphs. PubMed includes 19717 vertices representing scientific papers and 44338 edges between them representing citations. The applied CNNs have to classify each vertex into one of 3 groundtruth classes. The Cora dataset is smaller, but has 7 different classes for classification. The results on these datasets (Table 3) were not as impressive as for the MNIST dataset.

Method	Cora	PubMed
ManiReg [4]	59.5%	70.7%
SemiEmb [42]	59.0%	71.1%
LP [45]	68.0%	63.0%
DeepWalk [28]	67.2%	65.3%
Planetoid [44]	75.7%	77.2%
DCNN [3]	$76.80 \pm 0.60\%$	$73.00 \pm 0.52\%$
GCN [21]	$81.59 \pm 0.42\%$	$78.72 \pm 0.25\%$
MoNet	$\mathbf{81.69} \pm 0.48\%$	$\mathbf{78.81} \pm 0.44\%$

Table 3. Source: [Paper]

Even though MoNet outperformed the other approaches, the authors explain that for their rather complex architecture, the training set was too small. When applied to larger and/or more complex data they expect MoNet to perform even better.

4.3 FAUST

The FAUST dataset was used to test MoNet on manifolds. The dataset consists of 100 3D bodyscans of 10 different persons in 10 different poses. The task is

for each point on a given reference bodyscan to find the corresponding point on another scan.

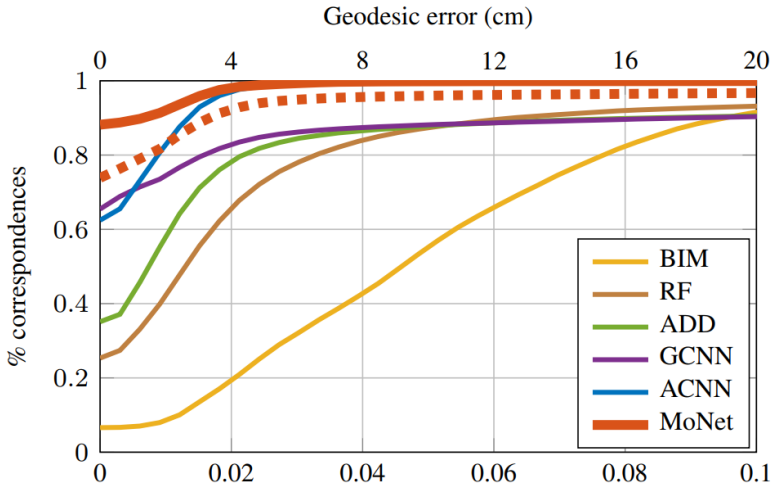

Figure 2. Source: [Paper]

Figure 2. shows the results for all tested architectures. MoNet clearly outperforms GCNN and ACNN. It finds groundtruth for almost 90% of the points and for 99% of the points the error is below 4cm. The distribution of those errors are depicted in Figure 3.

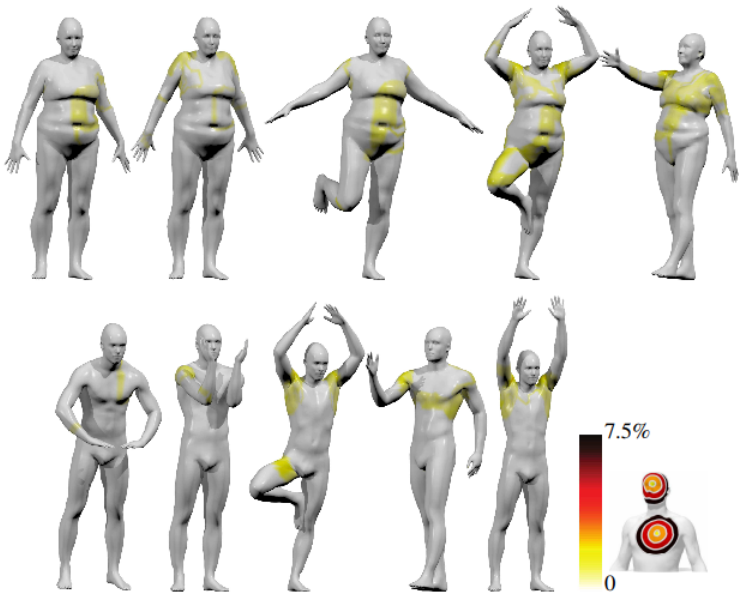

Figure 3. Source: [Paper]

With achieving almost perfect results MoNet was then used on the FAUST dataset to do a texture transform from the reference shape to the other shapes. The result can be seen in Figure 4.

Figure 4. Source: [Paper]

5 Conclusion

Even though the paper was mathematically heavy, most of the used formulas were well explained. Especially the created framework of using parameterized weighting functions for the patch operators is very promising and might be used by other research teams in the future. Since MoNet's weighting function is called a "convenient choice" by the authors, finding a weighting function that provides even better results seems possible.

6 References

[Paper]: http://openaccess.thecvf.com/content_cvpr_2017/papers/Monti_Geometric_Deep_Learning_CVPR_201
(Accessed on 16.12.2017)

All used formulae are taken from the paper.