GEOMETRY AWARE CONVOLUTIONAL FILTERS FOR OMNIDIRECTIONAL IMAGES REPRESENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Due to their wide field of view, omnidirectional cameras are frequently used by autonomous vehicles, drones and robots for navigation and other computer vision tasks. The images captured by such cameras, are often analyzed and classified with techniques designed for planar images that unfortunately fail to properly handle the native geometry of such images. That results in suboptimal performance, and lack of truly meaningful visual features. In this paper we aim at improving popular deep convolutional neural networks so that they can properly take into account the specific properties of omnidirectional data. In particular we propose an algorithm that adapts convolutional layers, which often serve as a core building block of a CNN, to the properties of omnidirectional images. Thus, our filters have a shape and size that adapts with the location on the omnidirectional image. We show that our method achieves better results compared to existing deep neural network techniques for omnidirectional image classification. Finally we show that our method is not limited to spherical surfaces and is able to incorporate the knowledge about any kind of omnidirectional geometry inside the deep learning network.

1 INTRODUCTION

Drone vision, autonomous cars and robot navigation systems often use omnidirectional cameras, as they allow recording the scene with wide field of view. Despite their obvious advantages, images obtained by such cameras have different statistics compared to planar images. Nevertheless, omnidirectional images are often processed with standard techniques, which are unfortunately poorly adapted to the specific geometry in such images.

In this paper we improve one of the most popular frameworks for image processing, namely convolutional neural network (CNN) for omnidirectional images. CNNs prove to be effective, as they permit to achieve very good performance on many different tasks like image classification, segmentation, generation and compression. In the context of omnidirectional cameras, CNNs are typically applied directly to the unwrapped and distorted spherical images. This approach, however, is suboptimal: due to specific geometry of these images, and in particular the change in the image statistics with the position in the image, the network is typically forced to learn different filters for different locations in the omnidirectional images (see Fig. 1).

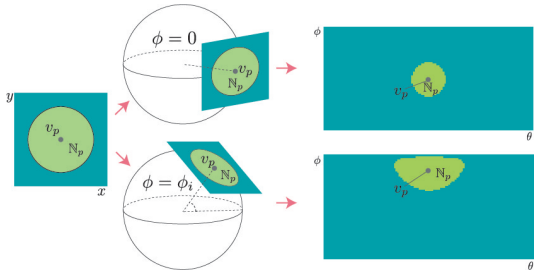

Figure 1: Our network allows adapting the size and shape of the filter with respect to the elevation.

To solve this issue, we replace ordinary convolutional filters with the graph-based ones that can adapt their size and shape depending on the position in the omnidirectional image thanks to the flexible graph structure. In order to overcome the common limitation of these graph-based filters being isotropic (i.e. invariant to the changes in the objects orientation in the image) we suggest to use multiple directed graphs instead of a single undirected one, as used in (Defferrard et al., 2016; Kipf & Welling, 2017; Khasanova & Frossard, 2017a). This permits our method to encode the orientation of the objects that appear in the images and, therefore, extract more meaningful features from them. This together with the inherent ability of our filters to adapt to the image projective geometry allows our method to reach state-of-the-art performance when working not just with spherical projections, as done by Khasanova & Frossard (2017b); Cohen et al. (2018), but virtually with any type of image projective geometry that can be encoded by a graph. In our experiments we show that apart from the state-of-the-art results on the regular omnidirectional images classification task, our approach outperforms the existing classification techniques when applied to the images projected on a randomly perturbed spherical surface or on a cube via the cube-map projection, which has recently become one of the popular ways to represent 360-images (Chen et al., 2018).

2 Related work

Classification and, generally, processing of omnidirectional images, have recently gained popularity with the development of self-driving cars and robot navigation systems. In this section we review some of the most recent approaches that have been developed for processing such images with the focus on deep neural networks, which are certainly the most popular architectures nowadays.

The most typical way of dealing with images taken by omnidirectional cameras is to apply standard image processing techniques directly on the equirectangular projection images (which is one of the most common representations for the omnidirectional images (De Simone et al., 2016)). However, due to the strong distortion effects introduced by the process of unwrapping of the projection surface to a plane, standard techniques lose much of their efficiency, as the appearance of the same object may change depending on its location in the image. To overcome this problem the recent work of (Khasanova & Frossard, 2017b) introduced special convolutional filers for omnidirectional images based on graphs, where pixels and intensities are respectively encoded by the nodes of the graph and the values of the signal defined on them. A different approach was suggested by Cohen et al. (2018) who proposed a CNN that is designed for spherical shapes and define filters directly on its surface. Both methods rely on the convolutional operation defined in spectral domain, with isotropic convolutional filters. The resulting architectures are invariant to rotation and translation transformations. However, there exist various tasks, where the orientation of the object plays an important role (e.g. classification and segmentation of natural images) and, therefore, introduction of the aforementioned invariance may harm the performance. On the other hand, we propose to build anisotropic graphs-based convolutional filters that do not have this limitation.

Further, Su & Grauman (2017) suggest adapting the size of the convolutional kernel to the elevation of the equirectangular image. This approach, however, requires a significantly larger number of parameters than the competing techniques, as it does not have the weight sharing property of CNNs. It rather requires learning different convolutional filters for different elevations of the equirectangular image. A different approach was suggested by Jeon & Kim (2017), who propose to learn the shape of the convolutional filter with back-propagation, by learning the sampling locations (position offsets), where the elements of the filter are evaluated. This allows the network to adapt the filter size to the problem at hand; however, the size of the filter will be the same for all image locations and therefore these filters will not be able to adapt to different positions and in particular different elevations in omnidirectional images. This idea was later extended by Dai et al. (2017), who suggest to learn the dynamic offsets of the filter elements depending on the image, which relaxes the constraint of the previous method and allows the filters to adapt to different parts of the image. Though this method was not specifically designed for omnidirectional images it can potentially adapt to their specific geometry. However, it will require an extensive training set, as the network needs to learn how it should react to various objects appearing at any possible elevation value. In our work we rather take advantage of the knowledge of the image projective geometry and we use this knowledge in the design of our architecture to adapt the size and shape of convolutional filters.

A different approach to compensate for the distortion of the appearance of the object depending on the spatial location in the image was proposed in the works of (Coors et al., 2018; Tateno et al.,

2018). These approaches suggest adapting the sampling locations of the convolutional filters to the geometry of the omnidirectional lens by projecting kernels to the sphere and using interpolated pixel values on the projected locations for implementing the convolutional filters. These works are the closest to ours. However, we propose a more general architecture, which permits to adapt the shape and size of the convolutional kernel to the location of omnidirectional image, and therefore to use the information about all the pixels and not only of a subset of them.

Finally, the authors in (Monroy et al., 2018; Ruder et al., 2018) suggest a completely different approach to tackle the distortion of omnidirectional images. Instead of working with equirectangular images, they propose to project an omnidirectional image to a cube, where each of its faces represents an image that would have been seen though a regular perspective camera, with the optical center located in the center of the cube (Chen et al., 2018). Representing an omnidirectional image in this way allows having less noticeable distortion effects as compared to equirectangular images. However, this representation suffers from another type of distortion that appears due to discontinuity effect on the borders between the faces of the cube. To mitigate this issue, Monroy et al. (2018) propose to apply a smoothing filter as a post-processing step and Ruder et al. (2018) suggest an algorithm that enforces consistency between the neighboring facets of the cube. Contrary to the mentioned approaches, as we model cube surface as a graph, our algorithm does not suffer from discontinuity problem and can be easily adapt to very different projection surfaces, including cube mapping one, and compensate for its distortion.

3 GEOMETRY-AWARE CNN

In this section we describe our algorithm, which adapts convolutional filters to the distortion of omnidirectional images. We start with the introduction of the equirectangular projection, as it is one of common ways to represent images from the omnidirecitonal cameras (De Simone et al., 2016; Coors et al., 2018). We then describe our graph-based representation learning framework.

3.1 EQUIRECTANGULAR PROJECTION

Omnidirectional visual content can be represented as a sphere with radius r, where the user is assumed to be located at the center. Each 3D point can be projected to a point on the surface of this sphere, which is described by spherical coordinates, namely a longitude $\theta \in [-\pi, \pi]$ and a latitude $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.

Omnidirectional images are generally not processed directly in their native geometry, but they are first projected to the 2D plane where classical image processing techniques can be activated. One of the popular projections involves sampling the data on the sphere with equal steps $\Delta\theta$ and $\Delta\phi$, which results in an equrectangular image on the 2D plane. Thus, each point of equrectangular image is defined by its spherical coordinates. To describe this projection let us introduce the tangent plane T, which is tangent to the sphere in the point (θ_0, ϕ_0). Thus, each point (x, y) on the tangent plane is projected to the sphere surface (θ, ϕ) as follows (Coors et al., 2018):

$$\begin{array}{rl} \phi(x,y) = & \sin^{-1}(\cos\nu \sin\phi_0 + \frac{y \sin\nu \cos\phi_0}{\rho}) \\ \theta(x,y) = & \theta_0 + \tan^{-1}(\frac{x \sin\nu}{\rho \cos\phi_0 \cos\nu - y \sin\phi_0 \sin\nu}) \end{array} . \tag{1}$$

where $\rho = \sqrt{(x^2 + y^2)}$ and $\nu = \tan^{-1} \rho$.

In order to have similar filter response regardless of the position of the object we model distortion of the applied filter. Thus, similarly to the works of (Khasanova & Frossard, 2017b) and (Coors et al., 2018), we define the filter kernel on this tangent plane T. Fig. 1 illustrates a sample equirectangular image with various kernels corresponding to tangent plane at different positions on the sphere. As we can see the projected area is different for various tangent plane locations. This projected area defines the support of our geometry-aware features, as described in the next section.

3.2 GEOMETRY-ADAPTIVE FILTERS

In the context of omnidirectional cameras, the main drawback of the classical convolutional approach is that it applies the same rectangular filters to different image positions. However, as we

mentioned before, equirectangular images have different statistics at various elevations. Therefore, we propose to adapt the size and shape of the filter to the elevation on the spherical image.

To do so we propose to use a graph-based approach, which has recently become popular. The nodes of this graph $v_i \in \mathcal{G}$ and signal $y(v_i)$ defined on the nodes represent pixels and their intensity values respectively. Based on this graph $\mathcal{G}$ we can then use the Laplacian polynomial filters $\mathcal{F}$ as proposed in (Defferrard et al., 2016; Khasanova & Frossard, 2017a). Briefly $\mathcal{F}$ has the following form:

$$\mathcal{F} = \sum_{l=1}^{M} \alpha_l \mathcal{L}^l, \tag{2}$$

where the α_l are the trainable parameters, $\mathcal{L}$ is a Laplacian matrix and M is the degree of $\mathcal{F}$.

The Laplacian matrix is based on the adjacency matrix of the graph $\mathcal{G}$, which defines a neighborhood $\mathbb{N}_p$ for each node v_p of the graph. Using these filters we can construct a Deep learning architecture and use it for various tasks, e.g., classification of omnidirectional images. The main advantage of our approach is that by appropriately constructing the graph $\mathcal{G}$ we make the Laplacian polynomial filters $\mathcal{F}$ react similarly to the same object seen at different elevation of the equirectangular image regardless of geometric distortion due to the spherical geometry. Here we call those filters geometry-aware (GA).

In order to adapt the GA-filter $\mathcal{F}$ to the elevation we build a graph $\mathcal{G}$ in such a way that the neighbourhood of each node is different for different elevations of the omnidirectional image. In the following section we describe two approaches that rely on undirected and directed graphs respectively, which consequently define isotropic and anisotropic filters. Then we describe in more details the polynomial anisotropic filter $\mathcal{F}$ that is used for directed graphs..

Undirected graph construction for adaptive filtering. To adapt $\mathcal{F}$ to the elevation level we construct a graph $\mathcal{G}$ with nodes that have different neighborhoods depending on the elevation. To do so we define a circular area on a tangent plane $\mathcal{T}$, centered in the tangency point. Then we move $\mathcal{T}$ such that it becomes tangent to the sphere $\mathcal{S}$ in different positions v_p and project the circular area onto $\mathcal{S}$. For every point of $\mathcal{S}$ this creates a neighborhood $\mathbb{N}_p$, which changes its shape and size together with the elevation, as can be seen in Fig. 1.

Based on this geometry adaptive neighborhood, we then construct the graph $\mathcal{G}$ in the following way. We connect the node $v_p \in \mathcal{G}$, corresponding to a tangent point on the sphere, with the node $v_j \in \mathbb{N}_p$. The corresponding edge e_{pi} has a weight w_{pi} that is inversely proportional to the Euclidean distance between v_p and v_i, which are defined on the sphere:

$$\begin{aligned} w_{pi} &= ||v_p - v_i||_{L_2}^{-1}, \quad && v_i \in \mathbb{N}_p \ , \\ w_{pi} &= 0, && v_i \notin \mathbb{N}_p \ . \end{aligned} \tag{3}$$

This allows us to vary the size of the neighbourhood according to the geometry of the omnidirectional image for each node in $\mathcal{G}$, and weight the contribution of the nodes to the final filter response according to their distances to v_p. Therefore, depending on the elevation the filter is changing its shape and size.

While effective, these filters do not have a defined orientation in space as according to Eq. (2) filter applies the same weights α_l to all nodes in the l-hoop neighborhood, with the contribution of each node being weighted by the distance to v_p. This results in $\mathcal{F}$ being isotropic, which leads to suboptimal representation, as the network is not able to encode the orientation of the object.

Directed graphs construction. In order to overcome the limitations of isotropic filters, we propose to replace a single undirected graph $\mathcal{G}$ with multiple directed graphs $\mathcal{G}_k$, where each $\mathcal{G}_k$ defines its own orientation. Let us consider the case of a 3x3 classical convolutional filter. In this case the filter has 9 distinct elements. To mimic the same structure with our graph convolutional filters we employ the following algorithm. First we define 9 non-overlapping areas $s_k, k = 1..9$ on the tangent plane, which together form a rectangular region that is centered in the tangency point v_p of T as defined in the Fig. 2. This rectangular region effectively defines the receptive field of the filter on the tangent plane. Then we build a set of nine directed graphs $G_k, k = 1..9$ in the similar way, as mentioned in the previous section for undirected graph. In particular in order to build graph $\mathcal{G}_k$ we do as follows. For the area s_k and for every node v_p we move the tangent plane at point v_p and then

project s_k from T onto the sphere. This operation defines a specific neighborhood $\mathbb{N}_k(p)$ on the sphere that consists of the points that belong to the projection of the region s_k from the plane T. We then connect v_p with a directed edge to each of these points, where the weight of the edge is defined in Eq. (3). Note that the direction of the edge is very important, because connecting v_p and v_i with an undirected edge forces v_p to be part of the neighborhood $\mathbb{N}_k(i)$. This, however, is not possible, as the neighborhood $\mathbb{N}_k(i)$ is computed by projecting the area s_k from the plane T that is tangent to the sphere at point v_i and does not include v_p.

This results in construction of the directed graph $\mathcal{G}_k$, which corresponds to the kth region of the filter, illustrated in Fig. 2. We repeat this operation for all the areas $s_k, k = 1..9$ of our filter, which leads to creation of 9 directed graphs $\mathcal{G}_k, k = 1..9$. Given this set of graphs $\mathcal{G}_k$ we define the resulting convolutional operation F as follows:

$$F = \sum_{k=1}^{9} \mathcal{F}_k, \tag{4}$$

where $\mathcal{F}_k$ is the filtering operation defined on the graph $\mathcal{G}_k$. Note that this filtering operation is slightly different from the operation that is used when working with undirected graphs and is discussed in more details in the following section.

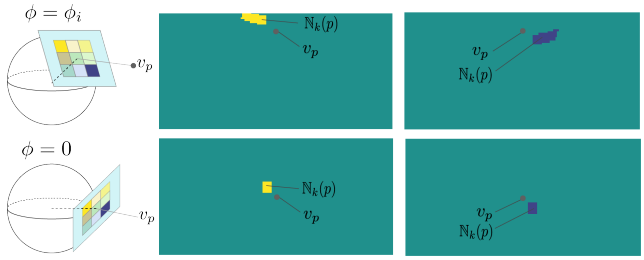

Figure 2: Illustration of 3×3 GA-filter kernel, defined on the tangent plane. Projection of area s_k forms the neighborhood $\mathbb{N}_k(p)$ of the node v_p, where the GA-filter is applied. The right part of the figure illustrates the change of the neighborhood $\mathbb{N}_k(p)$ depending on the location of v_p and the chosen filter area s_k.

To sum up, the introduced graph construction process allows having anisotropic filters F, defined in Eq. (4) that are capable of capturing the orientation of the object and therefore learn more meaningful feature representation for an image compared to the isotropic graph-based filters. It is important to note that in this paper we use the set of 9 non-overlapping rectangular areas defined on the tangent plane, as shown by Fig. 2, due to their rough correspondence to the elements of a 3×3 convolutional filter. However, our method can be easily extended to an arbitrary number of such areas with arbitrary shapes.

Geometry aware anisotropic filters. For directed graphs $\mathcal{G}_k$ Laplacian matrix is not defined, therefore, we use the polynomial filters proposed in (Sakiyama et al., 2017). Instead of the Laplacian matrix these filters rely on the normalized adjacency matrix $\mathcal{A}_k$, which is defined as follows:

$$\mathcal{A}_k = D_k^{-1} A_k, \tag{5}$$

where A_k and D_k are the weighted adjacency and the diagonal degree matrices of graph $\mathcal{G}_k$ respectively. The elements of D_k are computed as $D_k(m,m) = \sum_n A_k(m,n)$. Then, we define filters in the following way:

$$\mathcal{F}_k = \alpha_0^{(k)} + \alpha_1^{(k)} \mathcal{A}_k, \tag{6}$$

where $\alpha_0^{(k)}, \alpha_1^{(k)}$ are the training parameters of our filter. Here, we use polynomial filters of degree 1, as they achieve good balance between speed and performance.

Network architecture. The introduced approach focuses on the modification of the convolutional layer to incorporate the knowledge about the image projective geometry inside the neural network. Thus, it can be applied for a broad variety of tasks. In this paper we focus on the image classification

problem and therefore use a relatively standard architecture that consists of a sequence of convolutional layers with the introduced graph-based filters, followed by a sequence of the fully connected layers.

Discussion. Our method can be seen as a generalization of different approaches that have been developed for omnidirectional image. For example, if the node v_p at elevation ϕ_i has only one neighbor in each direction and the weight of the edges between nodes is always equal to one, it will be the standard CNN method (Lecun et al., 1998). Further, if these neighbors correspond to the projected points it becomes the recently proposed algorithm of (Coors et al., 2018). Finally, if we replace directed graphs with a single undirected one we get the same behavior of the polynomial filters as described in graph-based deep learning methods (Khasanova & Frossard, 2017a; Defferrard et al., 2016; Kipf & Welling, 2017; Khasanova & Frossard, 2017b).

4 Results

In this paper we propose GA-filters that adapt their shape to the geometry of an omnidirectional image. We evaluate our method on omnidirectional image classification tasks.

4.1 Spherical projection

To the best of our knowledge, there does not exist a publicly available dataset of labeled images of objects taken by omnidirectional cameras. Therefore, we created our own dataset, by projecting MNIST digits (LeCun & Cortes, 2010) at the random positions on the sphere, which results in a set of equirectangular images of size 64×128. We use 54000, 6000 and 10000 images for training, validation and testing of our method respectively. We will make this dataset publicly available.

Architecture and comparison methods. We compare our approach with standard ConvNets, the algorithm proposed in (Cohen et al., 2018) and other graph-based methods. We present our result in the Table 1[1]. For the graph-based methods we investigate 3 possible way of constructing $\mathcal{G}$:

- Regular grid-graph with 8 neighbors and all equal weights $w_{ij} = 1$;
- Regular grid-graph with 8 neighbors and weights that depend on the Euclidean distance d_{ij} between the nodes, as proposed in (Khasanova & Frossard, 2017b) $w_{ij} = d_{ij}^{-1}$;
- Irregular GA-graph $w_{ij} = d_{ij}^{-1}$ (isotropic filters from Section 3.2).

For all of them we build a normalized Laplacian matrix (Khasanova & Frossard, 2017b) and use polynomial filters of degree 1, which is equivalent to using 3×3 convolutional kernels. Therefore, for the standard ConvNet we similarly rely on filters of size 3×3.

All the competing approaches use the networks of roughly the same complexity. For all the methods we use the architectures with the same global structure that consists of two convolutional layers with stride two, followed by three fully-connected layers with $32, 16, 10$ neurons respectively. As (Coors et al., 2018) does not have fully connected layer, we used the architecture proposed in the paper with roughly the same number of parameters.

As it can be seen from Table 1, our approach significantly outperforms the standard ConvNets, as it is designed to use the geometry of the omnidirectional images. Further it shows a much higher accuracy then other graph-based techniques, which do not fully exploit the omnidirectional image geometry and rely on isotropic filters. Finally, our method achieves slightly better accuracy than (Cohen et al., 2018), however, our method is more general as it preserves the spatial relation between the pixels, which makes is applicable for a larger variety of tasks.

4.2 Different projection surfaces

Due to the flexibility offered by the graph-based representation of the input image, our approach is general and not limited to spherical surfaces that are frequently used in omnidirectional cameras.

[1]We were unable to compare our method to the recent work of (Coors et al., 2018) as there is no publicly available implementation as far as we know.

Table 1: Classification experiment on projected MNIST dataset to the sphere

Method	Accuracy
regular graph ($w_{ij} = 1$)	69.4
regular graph ($w_{ij} = 1/d_{ij}$)	69.8
GA graph ($w_{ij} = 1/d_{ij}$)	70.2
ConvNets	80.4
(Cohen et al., 2018)	95.2
Ours	**95.3**

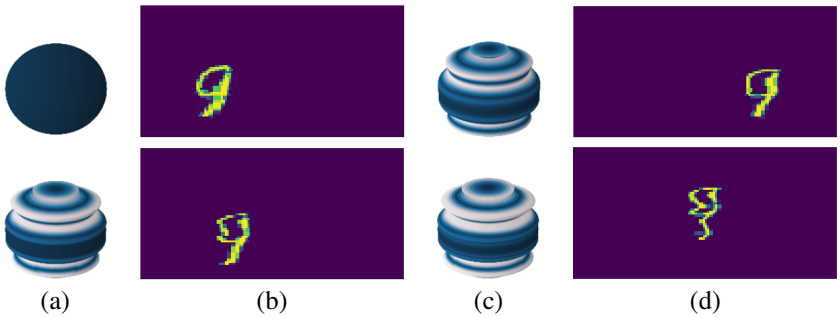

Figure 3: Illustration of surfaces (a,c) and corresponding equirectangular images (b,d) of the digit 9 projected to these surfaces at random positions. White color of surfaces highlights the furthest points to the spherical surface of the same radius in terms of Euclidean distance; blue color indicates the closest points.

In this section we show that our method is able to adapt the shape and size of the filters to various surfaces provided their geometry is known. To evaluate our method we use the same training, validation and test split of the MNIST dataset (LeCun & Cortes, 2010), as we used in the previous section and project these images to surfaces with different geometry.

Modified spherical surface. In order to evaluate the performance of our method as a function of deformation of the spherical surface, we have created a set of datasets by projecting the MNIST images to random locations of the surfaces, which have shapes, shown in Fig. 3 (a,c) and unwrap these spherical images to equirectangular ones. Each of the aforementioned surfaces is a modification of a spherical one, and the white color in Fig. 3 (a,c) denotes the areas of the generated surface that are the furthest from the spherical surface of the same radius. The Fig. 3 (b,d) illustrates sample images of digits projected onto the respective surfaces. In order to compute these images we use the version of Eq. (1) that is modified to reflect the change in the surface geometry[2]. We then use the same projection to construct the graph $\mathcal{G}$ that allows our method to adapt to the surface geometry and evaluate our method on each of the generated datasets.

As we can see from the Fig. 4 our approach is able to maintain relatively high performance independently of the complexity of the underlying projection surface. We further compared our method with the SphericalCNN of (Cohen et al., 2018) as it is the closest competitor of our method. This approach, however, is designed to only handle the distortion effects introduced by the lens of omnidirectional cameras and therefore by design is not able to easily adapt to other projections. The performance of SphericalCNN gradually decreases with the increase of complexity of the projection surface, as expected, while our method is able to adapt to the surface changes. For the sake of completeness we have also evaluated the classical CNN on these images, however, the latter is not able to adapt to such complex projective geometry and, therefore, achieves relatively low accuracy.

Cube mapping projection. We further evaluated our approach on a cube map projection that has recently gained popularity for handling omnidirectional images, due to its ability to reduce distortion

[2]Due to the space limitation the formulas for computing this projection are moved to the appendix.

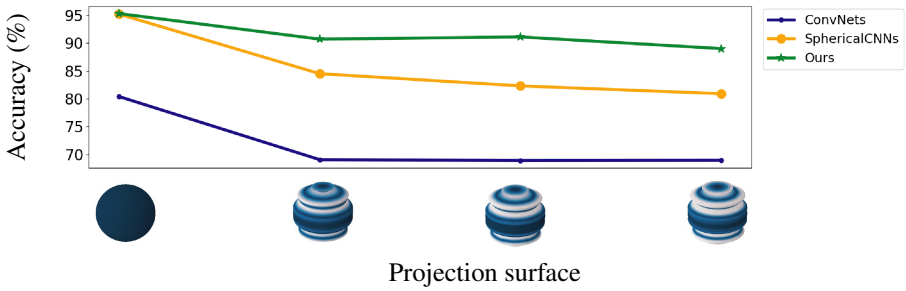

Figure 4: Illustration of the performance of our approach (green) and SphericalCNN (orange) on the classification tasks with gradually increasing complexity of the projection surface.

Table 2: Classification experiment on projected MNIST dataset to the cube

Method	Accuracy
ConvNets	79.41
Ours	**84.30**

artifacts that appear due to the spherical geometry. However, it introduces a new type of distortion when the cube surface is unwrapped to the rectangular image. Fig. 5 (a) schematically illustrates the process of unwrapping the surface of the cube onto a plane, which results in a rectangular image. Fig. 5 (b) further illustrates sample images from the MNIST dataset projected on the cube and unwrapped to their rectangular representation.

To efficiently process these images and take into account the knowledge about the geometry of the projection, we create a set of directed graphs, similarly to the method in Section 3.2 but using the cube map projective geometry instead of the spherical one. We then run the network with the same architecture, as described in the previous section with the created graph. For this type of projection the SphericalCNN method of (Cohen et al., 2018) is not defined, therefore we compared our approach only to classical ConvNets and show the results in Table 2. As we can see, our method outperforms the baseline approach as the latter has discontinuity on the edges, which is relatively hard to model using a simple ConvNet. Our method, on the other hand, does not suffer from this problem, as the relation between image pixels is modeled according to their geometrical relation.

5 Conclusion

In this paper we have presented a novel approach to incorporate the information about image geometry inside the neural network. We have introduced the graph-based geometry aware convolutional filters that adapt their shape and size to the geometry of the projection surface. In contrast to many existing graph-based filters, our filter is anisotropic, which allows it to better adjust to the specific properties of the problem. Our illustrative experiments show state-of-the-art performance of our approach applied to classification tasks for various types of projected surfaces.

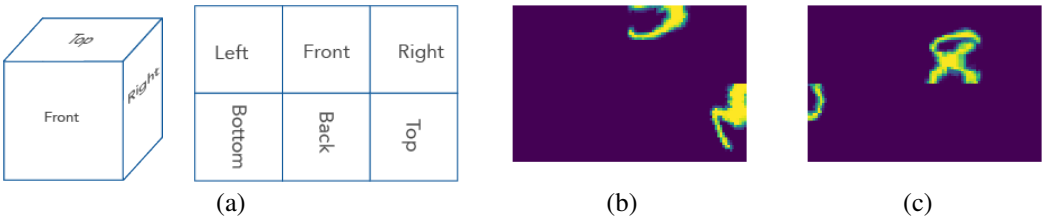

Figure 5: **Cube map projection**: (a) schematic illustration of the unwrapping process of the cube surface with a baseball arrangement (Chen et al., 2018) onto a planar surface; (b,c) sample projections of images from the MNIST dataset on the cube surface unwrapped into rectangular images.

REFERENCES

Z. Chen, Y. Li, and Y. Zhang. Recent advances in omnidirectional video coding for virtual reality: Projection and evaluation. 146, 2018.

T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical CNNs. In *International Conference on Learning Representations (ICLR)*, 2018.

B. Coors, A. P. Condurache, and A. Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *European Conference on Computer Vision (ECCV)*, 2018.

J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 764–773, 2017.

F. De Simone, P. Frossard, P. Wilkins, N. Birkbeck, and A. C. Kokaram. Geometry-driven quantization for omnidirectional image coding. In *Picture Coding Symposium*, pp. 1–5, 2016.

M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional Neural Networks on graphs with fast localized spectral filtering. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

Y. Jeon and J. Kim. Active convolution: Learning the shape of convolution for image classification. In *Conference on Computer Vision and Pattern Recognition, (CVPR)*, pp. 1846–1854, 2017.

R. Khasanova and P. Frossard. Graph-based isometry invariant representation learning. In *International Conference on Machine Learning (ICML)*, 2017a.

R. Khasanova and P. Frossard. Graph-based classification of omnidirectional images. In *Proceedings of the International Conference on Computer Vision (ICCV) Workshops*, 2017b.

T. N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Y. LeCun and C. Cortes. MNIST handwritten digit databaset. 2010.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

R. Monroy, S. Lutz, T. Chalasani, and A. Smolic. Salnet360: Saliency maps for omni-directional images with cnn. *Signal Processing: Image Communication*, 2018.

M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos and spherical images. *International Journal of Computer Vision*, pp. 1–21, 2018.

A. Sakiyama, T. Namiki, and Y. Tanaka. Design of polynomial approximated filters for signals on directed graphs. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 633–637, 2017.

Y.-C. Su and K. Grauman. Learning spherical convolution for fast features from 360°imagery. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 529–539. 2017.

K. Tateno, N. Navab, and F. Tombari. Distortion-aware convolutional filters for dense prediction in panoramic images. In *The European Conference on Computer Vision (ECCV)*, 2018.

APPENDIX

A MODIFIED SPHERICAL SURFACE PROJECTION

In order to compute the projections of the MNIST digits on the modified spherical surfaces we have used the following projective mapping, which is a modification of the mapping defined in Eq. (1) of the main paper:

$$\begin{aligned} x &= \cos(\phi_i)\sin(\theta_i - \theta_0) \\ y &= (\cos(\phi_0)\sin(\phi_i + p(\phi_i, r, l)) - \sin(\phi_0 + p(\phi_0, r, l))\cos(\phi_i)\cos(\theta_i - \theta_0))/c, \\ c &= \sin(\phi_i + p(\phi_i, r, l))\sin(\phi_0 + p(\phi_0, r, l)) + \cos(\phi_i)\cos(\phi_0)\cos(\theta_i - \theta_0), \end{aligned} \tag{7}$$

where (x, y) are the coordinates on the tangent plane and $p(\phi, r, l)$ is the perturbation function that can be written as

$$p(\phi, r, l) = r\sin^{-1}(\sin(l\phi)), \tag{8}$$

where ϕ is the elevation level; r is the parameter that regulates the perturbation magnitude and l defines frequency of the perturbation signal. In our experiments we have set $l = 10$. Note that for a specific case of $r = 0$ we get the ordinary spherical surface.