Orientation-Aware Semantic Segmentation on Icosahedron Spheres

Chao Zhang*[1] Stephan Liwicki*[1] William Smith[2] Roberto Cipolla[1,3]

[1]Toshiba Research Europe Limited, Cambridge, United Kingdom
[2]University of York, United Kingdom [3]University of Cambridge, United Kingdom

Abstract

We address semantic segmentation on omnidirectional images, to leverage a holistic understanding of the surrounding scene for applications like autonomous driving systems. For the spherical domain, several methods recently adopt an icosahedron mesh, but systems are typically rotation invariant or require significant memory and parameters, thus enabling execution only at very low resolutions. In our work, we propose an orientation-aware CNN framework for the icosahedron mesh. Our representation allows for fast network operations, as our design simplifies to standard network operations of classical CNNs, but under consideration of north-aligned kernel convolutions for features on the sphere. We implement our representation and demonstrate its memory efficiency up-to a level-8 resolution mesh (equivalent to 640×1024 *equirectangular images). Finally, since our kernels operate on the tangent of the sphere, standard feature weights, pretrained on perspective data, can be directly transferred with only small need for weight refinement. In our evaluation our orientation-aware CNN becomes a new state of the art for the recent 2D3DS dataset, and our Omni-SYNTHIA version of SYNTHIA. Rotation invariant classification and segmentation tasks are additionally presented for comparison to prior art.*

1. Introduction

We address the problem of spherical semantic segmentation on omnidirectional images. Accurate semantic segmentation is useful for many applications including scene understanding, robotics, and medical image processing. It is also a key component for autonomous driving technology. Deep convolutional neural networks (CNNs) have pushed the performance on a wide array of high-level tasks, including image classification, object detection and semantic segmentation. In particular, most research on CNNs for semantic segmentation [18, 22, 30, 4] thus far has focused on perspective images. In our work, we focus on omnidirectional images, as such data provides a holistic understanding of the surrounding scene with a large field of view. The complete receptive field is especially important for autonomous driving systems. Furthermore, recent popularity in omnidirectional capturing devices and the increasing number of datasets with omnidirectional signals make omnidirectional processing very relevant for modern technology.

While spherical input could be represented as planar equirectangular images where standard CNNs are directly applied, such choice is inferior due to latitude dependent distortions and boundaries. In [25] a perspective network is distilled to work on equirectangular input. The main drawback is that weight sharing is only enabled on longitudes. Therefore, the model requires more parameters than a per-

*Equal contribution: {chao.zhang, stephan.liwicki}@crl.toshiba.co.uk

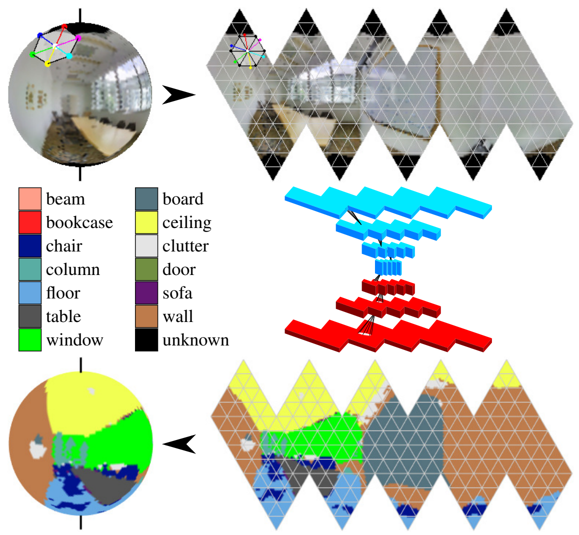


Figure 1. Given spherical input, we convert it to an unfolded icosahedron mesh. Hexagonal filters are then applied under consideration of north alignment, as we efficiently interpolate vertices. Our approach is suited to most classical CNN architectures, *e.g.* U-Net [22]. Since we work with spherical data, final segmentation results provide a holistic labeling of the environment.

spective one. SphereNet [9] projects equirectangular input onto a latitude-longitude grid. A constant grid kernel is convolved with each vertex on the sphere by sampling on the tangent plane. However, it is not straightforward to implement pooling and up-sampling for dense prediction tasks.

In 3D shape analysis, one of the challenges in applying CNNs is how to define a natural convolution operator on non-euclidean surfaces. Several works [3, 20, 2] have focused on networks for manifolds or graphs. Unlike general 3D shapes, omnidirectional images are orientable with the existence of north and south poles. Therefore, the lack for shift-invariance on surfaces or graphs could be overcome with an orientation-aware representation.

Most recently, several works propose to use an icosahedron mesh as the underlying spherical data representation. The base icosahedron is the most regular polyhedron, consisting of 12 vertices and 20 faces. It also provides a simple way of resolution increase *via* subdivision. In [14], UGSCNN is proposed to use the linear combinations of differential operators weighted by learnable parameters. Since the operators are precomputed, the number of parameters is reduced to 4 per kernel. The main issue of this approach, as observed in our experiments, is that it requires a lot of memory for their mesh convolution if the resolution is raised for better input/output quality. Similar to our method in the use of icosahedron, [7] proposes a gauge equivariant CNN. Here, filter weights are shared across multiple orientations. While rotation covariance and invariance is essential in applications such as 3D shape classification and climate pattern prediction, it might be undesired for semantic segmentation which we consider here. On the contrary, we argue that the orientation information of cameras attached to vehicles or drones is an important cue and should be exploited.

Therefore, we propose and investigate a novel framework for the application of CNNs to omnidirectional input, targeting semantic segmentation. We take advantage of both, the icosahedron representation for efficiency and orientation information to improve accuracy in orientation-aware tasks (Fig. 1). Our hypothesis is that aligning all learnable filters to the north pole is essential for omnidirectional semantic segmentation. We also argue that high resolution meshes (*i.e.* a level-8 icosahedron mesh) are needed for detailed segmentation. Due to memory restrictions, CNN operations need to be implemented efficiently to reach such high resolution.

In our work, we first map the spherical data to an icosahedron mesh, which we unfold along the equator, similarly to cube maps [19, 5] and [17, 7]. In the icosahedron, vertices have at most 6 neighbors. Therefore, we propose to use a hexagonal filter that is applied to each vertex's neighborhood. After simple manipulation of the unfolded mesh, standard planar CNN operations compute our hexagonal convolutions, pooling and up-sampling layers. Finally we emphasize, since our filters are similar to standard 3×3 kernels applied to the tangent of the sphere, weight transfer from pretrained perspective CNNs is possible.

To validate our approach we use the omnidirectional 2D3DS dataset [1] and additionally prepare our Omni-SYNTHIA dataset, which is produced from SYNTHIA data [23]. Qualitative as well as quantitative results demonstrate that our method outperforms previous state-of-the-art approaches in both scenarios. Performance on spherical MNIST classification [6] and climate pattern segmentation [21] is also shown in comparison with previous methods in literature. In summary, our contributions are:

1. We propose and implement a memory efficient icosahedron-based CNN framework for spherical data.
2. We introduce fast interpolation for orientation-aware filter convolutions on the sphere.
3. We present weight transfer from kernels learned through classical CNNs, applied to perspective data.
4. We evaluate our method on both non-orientation-aware and orientation-aware, publicly available datasets.

2. Related Work

CNNs on Equirectangular Images Although classical CNNs are not designed for omnidirectional data, they could still be used for spherical input if the data are converted to equirectangular form. Conversion from spherical coordinates to equirectangular images is a linear one-to-one mapping, but spherical inputs are distorted drastically especially in polar regions. Another artifact is that north and south poles are stretched to lines. Lai *et al*. [15] apply this method in the application of converting panoramic video to normal perspective. Another method along this line is to project spherical data onto multiple faces of convex polygons, such as a cube. In [19], omnidirectional images are mapped to 6 faces of a cube, and then trained with normal CNNs. However, distortions still exist and discontinuities between faces have to be carefully handled.

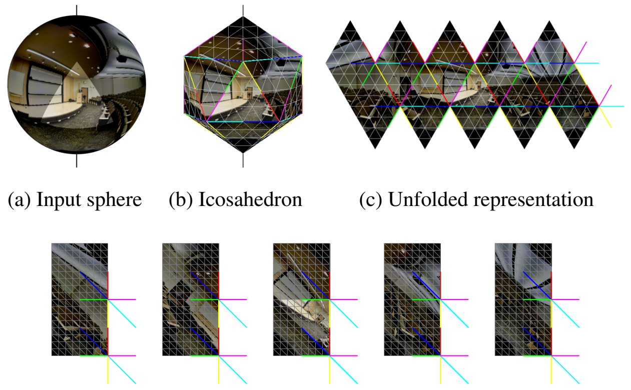
Spherical CNNs In order to generalize convolution from planar images to spherical signals, the most natural idea is to replace shifts of the plane by rotations of the sphere. Cohen *et al*. [6] propose a spherical CNN which is invariant in the $SO(3)$ group. Esteves *et al*. [11] use spherical harmonic basis to achieve similar results. Zhou *et al*. [31] propose to extend normal CNNs to extract rotation-dependent features by including an additional orientation channel.

CNNs with Deformable Kernels Some works [10, 13] consider adapting the sampling locations of convolutional kernels. Dai *et al*. [10] propose to learn the deformable convolution which samples the input features through learned offsets. An Active Convolutional Unit is introduced in [13] to provide more freedom to a conventional convolution by using position parameters. These methods requires additional model parameters and training steps to learn the sampling locations. In our work, we adapt the kernel shape to fit icosahedron geometry. Unlike deformable methods, our sampling locations can be precomputed and reused without the need of training.

CNNs with Grid Kernels Another line of works aim to adapt the regular grid kernel to work on omnidirectional images. Su and Grauman [25] propose to process equirectangular images as perspective ones by adapting the weights according to the elevation angles. Weight sharing is only enabled along longitudes. To reduce the computational cost and degradation in accuracy, a Kernel Transformer Network [26] is applied to transfer convolution kernels from perspective images to equirectangular inputs. Coors *et al*. [9] present SphereNet to minimize the distortions introduced by applying grid kernels on equirectangular images. Here, a kernel of fixed shape is used to sample on the tangent plane according to the location on the sphere. Wrapping the kernel around the sphere avoids cuts and discontinuities.

CNNs with Reparameterized Kernels For the efficiency of CNNs, several works are proposed to use parameterized convolution kernels. Boscani *et al*. [2] introduce oriented anisotropic diffusion kernels to estimate dense shape correspondence. Cohen and Welling [8] employ a linear combination of filters to achieve equivariant convolution filters. In [28], 3D steerable CNNs using linear combination of filter banks are developed. Recently, Jiang *et al*. [14] utilized parameterized differential operators as spherical convolution for unstructured grid data. Here, a convolution operation is a linear combination of four differential operators with learnable weights. However, these methods are limited to the chosen kernel types and are not maximally flexible.

CNNs on Icosahedron Related to our approach in discrete representation, several works utilize an icosahedron for spherical image representation. As the most uniform and accurate discretization of the sphere, the icosahedron is the regular convex polyhedron with the most faces. A spherical mesh can be generated by progressively subdividing each face into four equal triangles and reprojecting each node to unit length. Lee *et al*. [16] is one of the first to suggest the use of icosahedrons for CNNs on omnidirectional images. Here, convolution filters are defined in terms of triangle faces. In [14], UGSCNN is proposed to efficiently train a convolutional network with spherical data mapped to an icosahedron mesh. Liu *et al*. [17] uses the icosahedron based spherical grid as the discrete representation of the spherical images and proposes an azimuth-zenith


(a) Input sphere (b) Icosahedron (c) Unfolded representation

(d) Image-grid-aligned representation of spherical data

Figure 2. Spherical input data (a) is represented by an icosahedron-based geodesic grid (b). Similar to cubes [19, 5], we unfold our mesh (c) and align its 5 components to the standard image grid (d) for efficient computation of convolution, pooling and up-sampling.

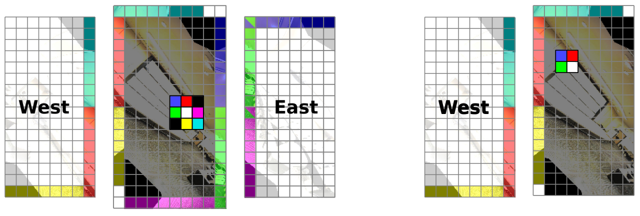
anisotropic CNN for 3D shape analysis. Cohen *et al*. [7] employ an icosahedron mesh to present a gauge equivariant CNN. Equivariance is ensured by enforcing filter weight sharing across multiple orientations.

3. Proposed Spherical Representation

We represent the spherical input through vertices on an icosahedron mesh (Fig. 2). The mapping is based on the vertices' azimuth and zenith angles – *e.g*. the input color is obtained from an equirectangular input through interpolation. Similar to cube maps [19, 5], the icosahedron simplifies the sphere into a set of planar regions. While the cube represents the sphere only with 6 planar regions, the icosahedral representation is the convex geodesic grid with the largest number of regular faces. In total, our grid consists of 20 faces and 12 vertices at the lowest resolution, and $f_r = 20 * 4^r$ faces and $n_r = 2 + 10 * 4^r$ vertices at resolution level $r \geq 0$. Note, a resolution increase is achieved by subdivision of the triangular faces at $r = 0$ into 4^r equal regular triangular parts. In the following, we present an efficient orientation-aware implementation of convolutions in §3.1, and our down- and up-sampling techniques in §3.2. Finally, weight transfer from trained kernels of standard perspective CNNs is discussed in §3.3.

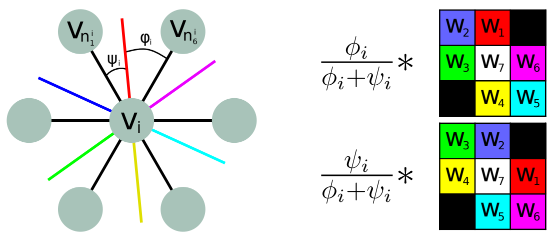
3.1. Orientation-aware Convolutions

If a camera is attached to a vehicle, the orientation and location of objects such as sky, buildings, sidewalks or roads are likely similar across the dataset. Therefore, we believe an orientation-aware system can be beneficial, while tasks with arbitrary rotations may benefit from rotation invariance [6] or weight sharing across rotated filters [29, 7].



(a) Convolution (b) Up-sampling

Figure 3. Convolution with our hexagonal filters (a) and up-sampling (b) reduce to standard CNN operations after padding the sphere component with features from neighboring sphere parts. Pooling is computed with a standard 2x2 kernel with stride 2.



(a) North-alignment (b) Interpolated filters

Figure 4. Given arc-based interpolation of the neighborhood for north-alignment (a), our convolution is computed with 2 weighted filters (b). The weights are precomputed for all vertices.

Efficient Convolutions through Padding We first define the north and south pole as any two vertices that have maximum distance on the icosahedron mesh. Similar to [17, 7], the mesh is then converted to a planar representation by unfolding it along the equator (Fig. 2). Finally, we split the surface into five components, and align the vertices with a regular image grid through a simple affine transformation.

Notice, vertices have a neighborhood of either 5 or 6 points. Hence we employ hexagonal filters in our work, instead of regular 3×3 kernels. Let us ignore the vertices at the poles (*e.g*. through reasoning of dropout), and adjust the neighborhood cardinality to 6 for all vertices with 5 neighbors through simple repetition. Now, our planar representation of the icosahedron simplifies the convolution with hexagonal filters to standard 2D convolution with a masked kernel, after padding as shown in Fig. 3.

North-alignment through Interpolation In its natural implementation, our filters are aligned to the icosahedron mesh. Consequently, the filter orientation is inconsistent, since the surfaces near the north and south poles are stitched. We reduce the effect of such distortions by aligning filters vertically through interpolation (Fig. 4).

The naïve convolution with weights $\{w_j\}_{j=1}^7$ at vertex $\mathbf{v}_i$ and its neighbors $\{\mathbf{v}_{n_j^i}\}_{j=1}^6$, is computed as $\sum_{j=1}^6 w_j \mathbf{v}_{n_j^i} + w_7 \mathbf{v}_i$, where n_j^i holds the neighborhood indices of $\mathbf{v}_i$. Instead, we north-align the neighborhood with interpolations using arc-based weights $\{\theta_j^i\}_{j=1}^6$ as follows:

$$\sum_{j=2}^{6} w_j(\theta_j^i \mathbf{v}_{n_j^i} + (1-\theta_j^i)\mathbf{v}_{n_{j-1}^i}) + w_1(\theta_1^i \mathbf{v}_{n_1^i} + (1-\theta_1^i)\mathbf{v}_{n_6^i}) + w_7 \mathbf{v}_i. \quad (1)$$

Since the hexagonal neighborhood is approximately symmetric, we further simplify (1) by introducing a unified weight α_i, such that $\{\alpha_i \approx \theta_j^i\}_{j=1}^6$ holds. Hence we write

$$\alpha_i \left(\sum_{j=1}^{6} w_j \mathbf{v}_{n_j^i} + w_7 \mathbf{v}_i \right) + (1-\alpha_i) \left(\sum_{j=2}^{6} w_j \mathbf{v}_{n_{j-1}^i} + w_1 \mathbf{v}_{n_6^i} + w_7 \mathbf{v}_i \right). \quad (2)$$

Thus, north-aligned filters can be achieved through 2 standard convolutions, which are then weighted based on the vertices' interpolations α_i.

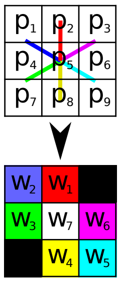
The arc-interpolation α_i is based on the angle distance between the direction towards the first and sixth neighbors (*i.e.* $\mathbf{v}_{n_1^i}$ and $\mathbf{v}_{n_6^i}$ respectively) and the north-south axis when projected onto the surface of the sphere. In particular, we first find the projective plane of the north-south axis $\mathbf{a} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^{\mathsf{T}}$ towards vector $\mathbf{v}_i$ as the plane with normal $\mathbf{n}_i = \frac{\mathbf{v}_i \times \mathbf{a}}{|\mathbf{v}_i \times \mathbf{a}|}$. Since the spherical surface is approximated by the plane of vectors $\mathbf{v}_i - \mathbf{v}_{n_1^i}$ and $\mathbf{v}_i - \mathbf{v}_{n_6^i}$, we only require the angles between these vectors and the plane given by $\mathbf{n}_i$, to find interpolation $\alpha_i = \frac{\phi_i}{\phi_i + \psi_i}$ with

$$\psi_i = \arccos \frac{(\mathbf{v}_i - \mathbf{v}_{n_1^i})^{\mathsf{T}}(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^{\mathsf{T}})(\mathbf{v}_i - \mathbf{v}_{n_1^i})}{\left|(\mathbf{v}_i - \mathbf{v}_{n_1^i})\right| \left|(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^{\mathsf{T}})(\mathbf{v}_i - \mathbf{v}_{n_1^i})\right|}$$
$$\phi_i = \arccos \frac{(\mathbf{v}_i - \mathbf{v}_{n_6^i})^{\mathsf{T}}(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^{\mathsf{T}})(\mathbf{v}_i - \mathbf{v}_{n_6^i})}{\left|(\mathbf{v}_i - \mathbf{v}_{n_6^i})\right| \left|(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^{\mathsf{T}})(\mathbf{v}_i - \mathbf{v}_{n_6^i})\right|}. \quad (3)$$

3.2. Pooling and Up-sampling

Down-sampling through pooling and bi-linear up-sampling are important building blocks of CNNs, and are frequently employed in the encoder-decoder framework of semantic segmentation (*e.g*. [22]). Pooling is aimed at summarising the neighborhood of features to introduce robustness towards image translations and omissions. Typically, a very small and non-overlapping neighborhood of 2×2 pixels is considered in standard images, to balance detail and redundancy. Bi-linear up-sampling is used in the decoder to increase sub-sampled feature-maps to larger resolutions.

We note, in our icosahedron mesh the number of vertices increases by a factor of 4 for each resolution (excluding poles). Therefore during down-sampling from resolution r to $r-1$, we summarize a neighborhood of 4 at r



$$\begin{aligned}
w_1 &= p_2 \\
w_2 &= \sin\tfrac{\pi}{3}\tfrac{p_1+p_4}{2} + (1-\sin\tfrac{\pi}{3})\tfrac{p_2+p_5}{2} \\
w_3 &= \sin\tfrac{\pi}{3}\tfrac{p_4+p_7}{2} + (1-\sin\tfrac{\pi}{3})\tfrac{p_5+p_8}{2} \\
w_4 &= p_8 \\
w_5 &= \sin\tfrac{\pi}{3}\tfrac{p_6+p_9}{2} + (1-\sin\tfrac{\pi}{3})\tfrac{p_5+p_8}{2} \\
w_6 &= \sin\tfrac{\pi}{3}\tfrac{p_3+p_6}{2} + (1-\sin\tfrac{\pi}{3})\tfrac{p_2+p_5}{2} \\
w_7 &= p_5
\end{aligned}$$

Figure 5. The weights of conventional 3×3 kernels trained on perspective data can be transferred to our model *via* simple interpolation as our filters operate on the sphere's tangent planes.

with 1 vertex at $r - 1$. A natural choice is to pool over $\{\mathbf{v}_i, \mathbf{v}_{n_1^i}, \mathbf{v}_{n_2^i}, \mathbf{v}_{n_3^i}\}$ for vertices $\mathbf{v}_i$ that are represented in both resolutions. Thus, we apply a simple standard 2×2 strided pooling with kernel 2×2 on each icosahedron part.

Analogously, bi-linear up-sampling or transposed convolutions are applied by padding the icosahedron parts at left and top followed by up-sampling by a factor of 2 in height and width (Fig. 3). Due to padding, this results in a 1-pixel border at each size which we simply remove to provide the expected up-sampling result. Finally we emphasize, methods like pyramid pooling [?] can be computed by combining our pooling and up-sampling techniques.

3.3. Weight Transfer from Perspective Networks

Similar to SphereNet [9], our network applies an oriented filter at the local tangent plane of each vertex on the sphere. Consequently, the transfer of pretrained perspective network weights is naturally possible in our setup. Since we apply hexagonal filters with 7 weights, we interpolate from the standard 3×3 kernels as shown in Fig. 5. Specifically, we align north and south of the hexagon with the second and eighth weight of the standard convolution kernel respectively. Bi-linear interpolation provides the remaining values for our filter. After transfer, weight refinement is necessary, but can be computed on a much smaller dataset (as done in [9]), or reduced learning iterations. Alternatively, but left for future work, it should be possible to learn hexagonal filter weights directly on perspective datasets [27, 12].

4. Evaluation

The main focus of this paper is omnidirectional semantic segmentation. Both synthetic urban scene and real indoor environments are evaluated. For completeness, we also include our model in comparison with previous state-of-the-art methods on spherical MNIST classification in §4.1 and a climate pattern prediction task in §4.2. In §4.3 and §4.4, performance on omnidirectional semantic segmentation tasks are summarized and analysed.

Method	N/N	N/R	R/R
Spherical CNN [6]	96.__	**94.__**	95.__
Gauge Net [7]	99.43	69.99	**99.31**
UGSCNN [14]	99.23	35.60	94.92
HexRUNet-C	**99.45**	29.84	97.05

Table 1. Spherical MNIST with non-rotated (N) and rotated (R) training and test data. Orientation-aware HexRUNet-C is competitive only when training and test data match (*i.e.* N/N and R/R).

Method	BG	TC	AR	Mean	mAP
Gauge Net[7]	**97.4**	**97.9**	**97.8**	**97.7**	**0.759**
UGSCNN[14]	97._	94._	93._	94.7	-
HexRUNet-8	95.71	95.57	95.19	95.49	0.518
HexRUNet-32	97.31	96.31	97.45	97.02	0.555

Table 2. Climate pattern segmentation results. We include mean class accuracy and mean average precision (mAP) where available. (The background class is denoted BG.)

4.1. Spherical MNIST

We follow [6] in the preparation of the spherical MNIST dataset, as we prepare non-rotated training and testing (N/N), non-rotated training with rotated testing (N/R) and rotated training and testing (R/R) tasks. Both non-rotated and rotated versions are generated using public source code provided by UGSCNN [14].[1] Training set and test set include 60,000 and 10,000 digits, respectively. Input signals for this experiment are on a level-4 mesh (*i.e.* $r = 4$). The residual U-Net architecture of [14], including the necessary modifications to adapt to the classification task, is used in our experiments. We call this network "HexRUNet-C".

As shown in Table 1, our method outperforms previous methods for N/N, achieving 99.45% accuracy. In R/R, our method performs better than competing Spherical CNN and UGSCNN. Gauge Net benefits from weight sharing across differently oriented filters, and achieves best accuracy for this task amongst all approaches. Similar to [14], our method is orientation-aware by design and thus not rotation-invariant. Therefore, it is expected to not generalize well to randomly rotated test data in the N/R setting, while Spherical CNN performs best in this case.

4.2. Climate Pattern Segmentation

We further evaluate our method on the task of climate pattern segmentation. The task is first proposed by Mudigonda *et al.* [21], and the goal is to predict extreme weather events, *i.e.* Tropical Cyclones (TC) and Atomospheric Rivers (AT), from simulated global climate data. The training set consists of 43,916 patterns, and 6,274 samples are used for validation. Evaluation results on the validation set are shown in Table 2 and Fig. 6. Here, we use the same residual U-Net architecture as UGSCNN [14].

[1] https://github.com/maxjiang93/ugscnn

Method	mIoU	beam	board	bookcase	ceiling	chair	clutter	column	door	floor	sofa	table	wall	window
UNet	35.9	8.5	27.2	30.7	78.6	35.3	28.8	4.9	33.8	89.1	8.2	38.5	58.8	23.9
Gauge Net	39.4	–	–	–	–	–	–	–	–	–	–	–	–	–
UGSCNN	38.3	8.7	32.7	33.4	82.2	42.0	25.6	10.1	41.6	87.0	7.6	41.7	61.7	23.5
HexRUNet	**43.3**	**10.9**	**39.7**	**37.2**	**84.8**	**50.5**	**29.2**	**11.5**	**45.3**	**92.9**	**19.1**	**49.1**	**63.8**	**29.4**

Table 3. Mean intersection over union (IoU) comparison on 2D3DS dataset. Per-class IoU is shown when available.

Method	mAcc	beam	board	bookcase	ceiling	chair	clutter	column	door	floor	sofa	table	wall	window
UNet	50.8	17.8	40.4	59.1	91.8	50.9	**46.0**	8.7	44.0	94.8	26.2	68.6	77.2	34.8
Gauge Net	55.9	–	–	–	–	–	–	–	–	–	–	–	–	–
UGSCNN	54.7	19.6	48.6	49.6	93.6	63.8	43.1	**28.0**	63.2	**96.4**	21.0	70.0	74.6	39.0
HexRUNet	**58.6**	**23.2**	**56.5**	**62.1**	**94.6**	**66.7**	41.5	18.3	**64.5**	96.2	**41.1**	**79.7**	**77.2**	**41.1**

Table 4. Mean class accuracy (mAcc) comparison on 2D3DS dataset. Per-class accuracy is shown when available.

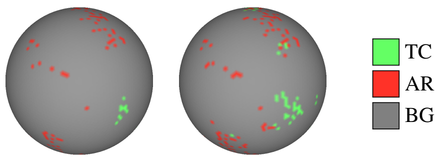


Figure 6. Semantic segmentation results of HexRUNet-32 on climate pattern (right) in comparison to ground truth (left).

We include two variants using different numbers of parameters: HexRUNet-8 and HexRUNet-32 use 8 and 32 as output channels for the first convolution layer, respectively. As is shown, both versions outperform UGSCNN in terms of mean accuracy. With 32 features, HexRUNet-32's mean accuracy is similar to best performing Gauge Net. However, our method does not match Gauge Net in terms of mean average precision (mAP). We attribute this to the fact that there is no direct orientation information to exploit in this climate data. In contrast, Gauge Net shows its advantage of weight sharing across orientations.

4.3. Stanford 2D3DS

For our first omnidirectional semantic segmentation experiment, we evaluate our method on the 2D3DS dataset [1], which consists of 1413 equirectangular RGB-D images. The groundtruth attributes each pixel to one of 13 classes. Following [14], we convert the depth data to be in meter unit and clip to between 0 and 4 meters. RGB data is converted to be in the range of [0, 1] by dividing 255. Finally, all data is mean subtracted and standard deviation normalized. The preprocessed signals are sampled on a level-5 mesh ($r = 5$) using bi-linear interpolation for images and nearest-neighbors for labels. Class-wise weighted cross-entropy loss is used to balance the class examples.

Using our proposed network operators, we employ the residual U-Net architecture of [14], which we call HexRUNet (see Sup. Mat. for details). We evaluate our method following the 3-fold splits, and show both qualitative and quantitative results in Fig. 7 and Table 3 and 4. Our method outperforms orientation-aware UGSCNN [14], rotation-equivariant Gauge Net [7] and the U-Net baseline [22] on equirectangular images that have been subsampled to mach level-5 mesh resolution. As for per-class evaluations, our method achieves best performance in most classes. This demonstrates that semantic segmentation indeed benefits from orientation-aware network with more expressive filters than [14].

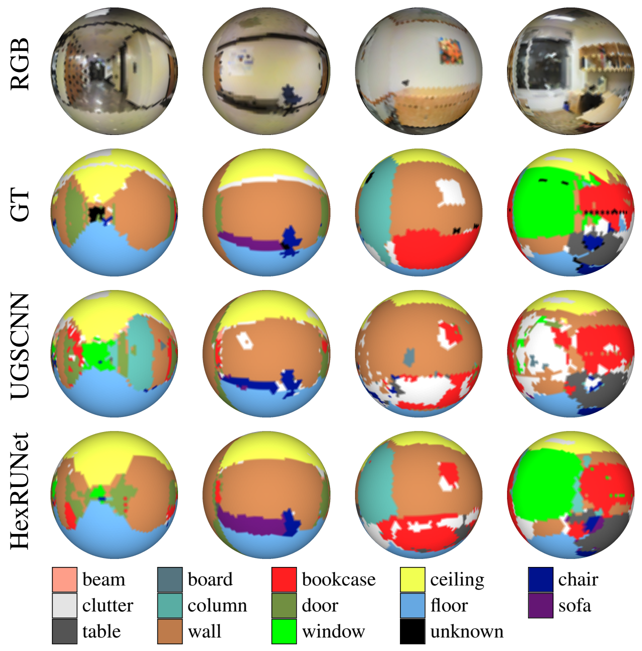


Figure 7. Qualitative segmentation results on 2D3DS dataset.

4.4. Omni-SYNTHIA

To further validate our method on omnidirectional semantic segmentation, we create an omnidirectional version from a subset of the SYNTHIA datset [23]. The SYNTHIA dataset consists of multi-viewpoint photo-realistic frames rendered from a virtual city and comes with pixel-level semantic annotations for 13 classes. We refer the readers to [23] for details. We select the "Summer" sequences of all five places (2×New York-like, 2×Highway and 1×European-like) to create our own omnidirectional dataset. We split the dataset into a training set of 1818 im-

Method	mIoU	building	car	cyclist	fence	marking	misc	pedestrian	pole	road	sidewalk	sign	sky	vegetation
UNet	38.8	80.8	59.4	0.0	0.3	54.3	12.1	4.8	16.4	74.3	58.2	0.2	90.4	49.6
UGSCNN	36.9	63.3	33.3	0.0	0.1	73.7	1.2	2.3	10.0	79.9	**69.3**	1.0	89.1	**56.3**
HexUNet-T	36.7	71.9	53.1	0.0	1.1	69.0	4.9	0.4	11.1	72.2	52.9	0.0	92.3	48.4
HexUNet-nI	42.4	77.1	64.8	0.0	2.4	**74.3**	10.4	2.0	23.6	**84.7**	68.6	1.0	93.1	48.7
HexUNet	**43.6**	**81.0**	**66.9**	0.0	**2.9**	71.0	**13.7**	**5.6**	**30.4**	83.1	67.0	**1.5**	**93.3**	50.2

Table 5. Mean IoU comparison at $r = 6$ on Omni-SYNTHIA dataset.

Method	mAcc	building	car	cyclist	fence	marking	misc	pedestrian	pole	road	sidewalk	sign	sky	vegetation
UNet	45.1	91.9	63.6	0.0	4.5	57.1	17.9	5.0	19.7	88.8	73.9	0.2	94.8	69.3
UGSCNN	50.7	**93.2**	**81.4**	0.0	**5.3**	83.2	33.7	2.5	14.9	90.8	82.7	1.3	96.1	74.0
HexUNet-T	44.8	80.0	60.9	0.0	1.6	74.7	26.9	0.4	13.0	80.0	75.2	0.0	**96.2**	73.4
HexUNet-nI	50.6	83.9	69.6	0.0	2.5	82.9	**39.1**	2.0	30.7	**91.8**	83.6	1.1	94.8	**76.5**
HexUNet	**52.2**	88.7	72.7	0.0	3.3	**85.9**	36.6	**6.2**	**42.5**	89.6	**83.7**	**1.6**	95.6	71.6

Table 6. Per-class accuracy comparison at $r = 6$ on Omni-SYNTHIA dataset.

ages (from New York-like and Highway sequences) and use 451 images of the European-like sequence for validation. Only RGB channels are used in our experiments. The icosahedron mesh is populated with data from equirectangular images using interpolation for RGB data and nearest neighbor for labels. Again, we report mIoU and mAcc. Here we use the standard U-Net architecture [22] to facilitate weight transfer from perspective U-Net in one of our experiments. We call this network “HexUNet”. For an ablation study, we also evaluate our method without north-alignment described in §3.1, denoted as “HexUNet-nI”.

Comparison with State-of-the-art We compare our method to UGSCNN [14] using data sampled at mesh level-6 ($r = 6$). We also include planar U-Net [22] using original perspective images, which have been sub-sampled to match the icosahedron resolution (see Sup. Mat. for details). Table 5 and 6 report mIoU and mAcc respectively, while Fig. 8 shows qualitative results. HexUNet outperforms previous state-of-the-art with significant margin across most classes. The performance on small objects, *e.g.* “pedestrian” and “sign”, is poor, while all methods fail for “cyclist”. We attribute this to an unbalanced dataset. Note here, class-wise weighted cross-entropy loss is not used. Finally we emphasize, HexUNet performs slightly better than HexUNet-nI, thus verifying the importance of orientation-aware filters in semantic segmentation.

Evaluation at Different Resolutions Most previous methods limit their mesh resolution to level $r = 5$ which consists of merely 2,562 vertices to represent omnidirectional input. In contrast, an icosahedron mesh at level $r = 8$ is required to match the pixel number of 640×1024 images, with $655,362 \approx 655,360$. Since we believe high resolution input/output is beneficial for the semantic segmentation task, we evaluate our method at different resolutions ($r = \{6, 7, 8\}$), shown in Table 7. Our method achieves best performance at $r = 7$, while $r = 7$ and $r = 8$ perform

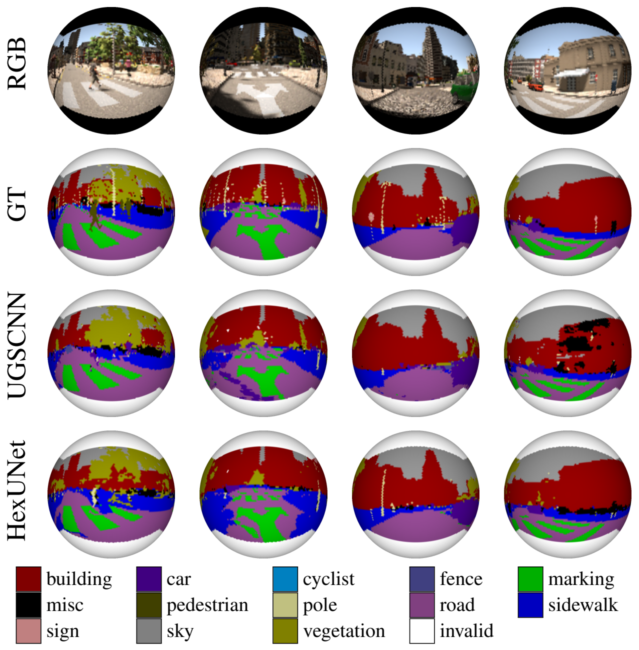


Figure 8. Segmentation results on Omni-SYNTHIA dataset.

Method	$r = 6$		$r = 7$		$r = 8$	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
UNet	38.8	45.1	44.6	52.6	43.8	52.4
UGSCNN	36.9	50.7	37.6	48.9	–	–
HexUNet-T	36.7	44.8	38.0	47.2	45.3	52.8
HexUNet-nI	42.4	50.6	45.1	53.4	45.4	53.2
HexUNet	**43.6**	**52.2**	**48.3**	**57.1**	**47.1**	**55.1**

Table 7. Evaluation at different resolution on Omni-SYNTHIA. (Current implementation of [14] could not fit data with resolution at $r = 8$. Note ground-truth at lower resolution is sub-sampled, thus evaluations of different resolutions are only indicative.)

similar. Since we use a standard U-Net structure consisting of only 4 encoder (and decoder) layers, perception of context is reduced at $r = 8$. This is further illustrated by the bottom-rightmost result in Fig. 9, where a car’s wheel

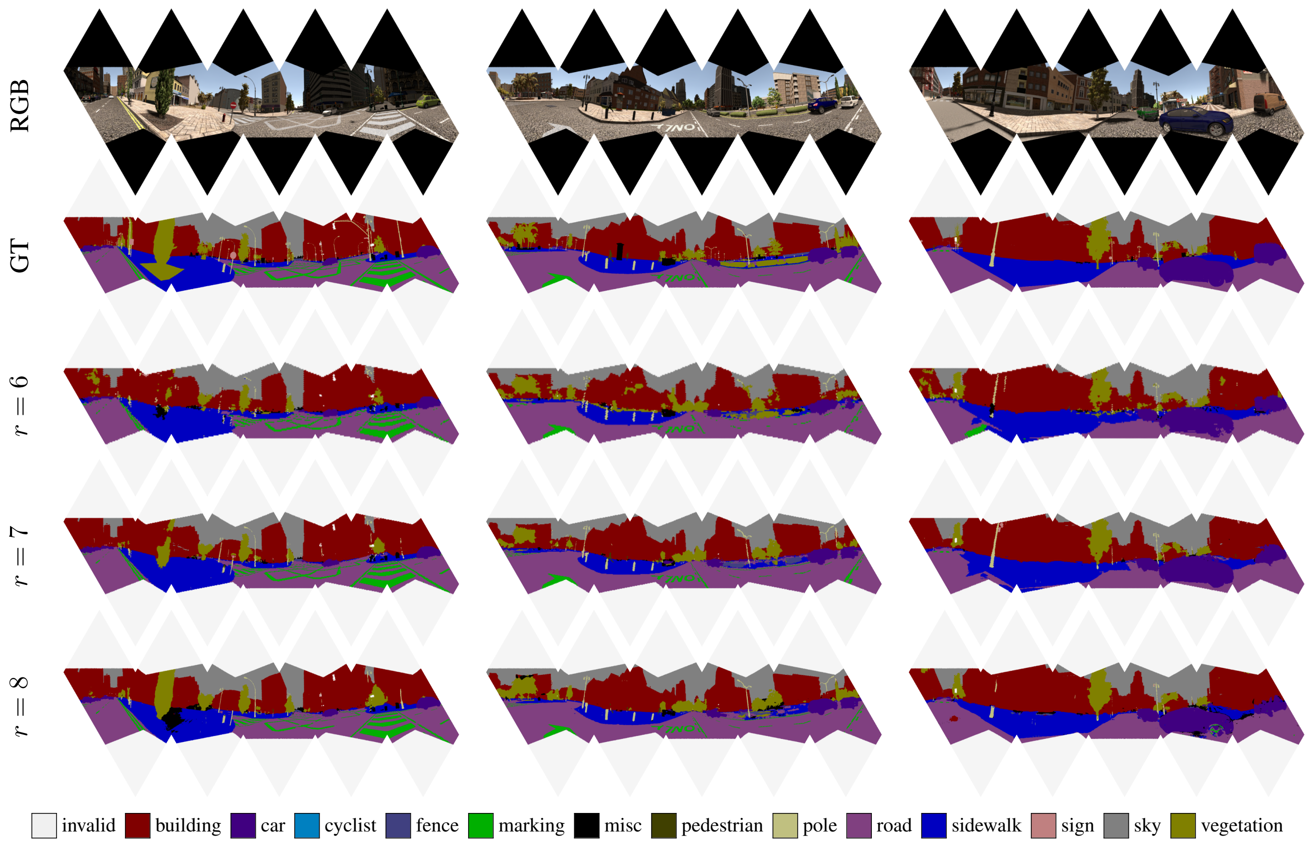


Figure 9. Unfolded visualization of semantic segmentation results at different resolutions on Omni-SYNTHIA dataset.

is misclassified as road-markings. Resolution $r = 6$ and $r = 7$ are able to adequately label this. Finally, network inference times are shown in Table 8.

	$r = 6$	$r = 7$	$r = 8$
UGSCNN [14]	458s	2755s	–
HexUNet	63s	65s	79s
UNet	34s	36s	40s

Table 8. Average evaluation time per validation (451 images) on Nvidia 1080Ti GPU with 11Gb memory. Planar UNet at equivalent resolution for front, back and side images (*i.e.* 1804 images) is also shown. HexUNet and UNet are implemented in Tensorflow, while the PyTorch implementation of [14] is used for comparison.

Evaluation of Perspective Weights Transfer As shown in §3.3, our method utilizes an orientation-aware hexagon convolution kernel which allows direct weight transfer from perspective networks. Initialized with the learned filters (3×3 kernels) from perspective U-Net, we perform weight refinement of only 10 epochs (in contrast to up-to 500 epochs otherwise), and report results as "HexUNet-T" in Table 5, 6 and 7. The proposed filter transfer obtains competitive results, especially at resolution level $r = 8$.

5. Conclusion

We introduced a novel method to perform CNN operations on spherical images, represented on an icosahedron mesh. Our method exploits orientation information, as we introduce an efficient interpolation of kernel convolutions, based on north-alignment. The proposed framework is simple to implement, and memory efficient execution is demonstrated for input meshes of level $r = 8$ (equivalent to a 640×1024 equirectangular image). In our evaluation on 2D3DS data [1] and our Omni-SYNTHIA version of SYNTHIA [23], our method becomes the new state of the art for the omnidirectional semantic segmentation task. Furthermore, weight transfer from pretrained standard perspective CNNs was illustrated in our work.

One limitation of the proposed approach is the poor segmentation accuracy for small objects (*e.g.* "pedestrian" and "cyclist") which we attribute to unbalanced dataset. Future work will incorporate better architectures such as [30, 24] for improved segmentation of small objects. Finally, we plan to exploit our framework for further orientation-aware learning tasks, such as localization and mapping.

References

[1] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[2] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *NIPS'16*, pages 3189–3197, 2016.

[3] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2018.

[5] H.-T. Cheng, C.-H. Chao, J.-D. Dong, H.-K. Wen, and T.-L. Liu. Cube padding for weakly-supervised salience prediction in 360° videos. In *CVPR'19*, 2019.

[6] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *ICLR'18*, 2018.

[7] Taco S. Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. *arXiv preprint arXiv:1902.04615*, 2019.

[8] Taco S. Cohen and Max Welling. Steerable CNNs. *arXiv preprint arXiv:1612.08498*, 2016.

[9] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. SphereNet: Learning spherical representations for detection and classification in omnidirectional images. In *ECCV'18*, pages 518–533, 2018.

[10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV'17*, pages 764–773, 2017.

[11] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *ECCV'18*, pages 54 – 70, 2018.

[12] Emiel Hoogeboom, Jorn WT Peters, Taco S Cohen, and Max Welling. Hexaconv. *arXiv preprint arXiv:1803.02108*, 2018.

[13] Yunho Jeon and Junmo Kim. Active convolution: Learning the shape of convolution for image classification. In *CVPR'17*, pages 4201–4209, 2017.

[14] Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Nießner. Spherical CNNs on unstructured grids. In *ICLR'19*, 2019.

[15] Wei-Sheng Lai, Yujia Huang, Neel Joshi, Christopher Buehler, Ming-Hsuan Yang, and Sing Bing Kang. Semantic-driven generation of hyperlapse from 360 degree video. *IEEE Trans. Visualization and Computer Graphics*, 24(9):2610–2621, 2018.

[16] Yeon Kun Lee, Jaeseok Jeong, Jong Seob Yun, Cho Won June, and Kuk-Jin Yoon. Spherephd: Applying cnns on a spherical polyhedron representation of 360 degree images. *arXiv preprint arXiv:1811.08196*, 2018.

[17] Min Liu, Fupin Yao, Chiho Choi, Sinha Ayan, and Karthik Ramani. Deep learning 3d shapes using alt-az anisotropic 2-sphere convolution. In *ICLR'19*, 2019.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR'15*, pages 3431–3440, 2015.

[19] Rafael Monroy, Sebastian Lutz, Tejo Chalasani, and Aljosa Smolic. Salnet360: Saliency maps for omni-directional images with cnn. *Signal Processing: Image Communication*, 2018.

[20] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR'17*, pages 5115–5124, 2017.

[21] Mayur Mudigonda, Sookyung Kim, Ankur Mahesh, Samira Kahou, Karthik Kashinath, Dean Williams, Vincen Michalski, Travis OBrien, and Mr Prabhat. Segmenting and tracking extreme climate events using neural networks. In *Deep Learning for Physical Sciences Workshop, held with NIPS'17*, 2017.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI'15*, pages 234–241, 2015.

[23] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR'16*, pages 3234–3243, 2016.

[24] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR'18*, pages 4510–4520, 2018.

[25] Yu-Chuan Su and Kristen Grauman. Learning spherical convolution for fast features from 360 imagery. In *NIPS'17*, pages 529–539, 2017.

[26] Yu-Chuan Su and Kristen Grauman. Kernel transformer networks for compact spherical convolution. *arXiv preprint arXiv:1812.03115*, 2018.

[27] Zhun Sun, Mete Ozay, and Takayuki Okatani. Design of kernels in convolutional neural networks for image classification. In *ECCV'16*, pages 51–66, 2016.

[28] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NIPS'18*, pages 10402–10413, 2018.

[29] Daniel Worrall and Gabriel Brostow. Cubenet: Equivariance to 3d rotation and translation. *arXiv preprint arXiv:1804.04458*, 2018.

[30] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[31] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *CVPR'17*, pages 4961–4970, 2017.