Kernel Transformer Networks for Compact Spherical Convolution

Yu-Chuan Su
The University of Texas at Austin

Kristen Grauman
Facebook AI Research
The University of Texas at Austin

Abstract

Ideally, 360° imagery could inherit the convolutional neural networks (CNNs) already trained with great success on perspective projection images. However, existing methods to transfer CNNs from perspective to spherical images introduce significant computational costs and/or degradations in accuracy. We present the Kernel Transformer Network (KTN) to efficiently transfer convolution kernels from perspective images to the equirectangular projection of 360° images. Given a source CNN for perspective images as input, the KTN produces a function parameterized by a polar angle and kernel as output. Given a novel 360° image, that function in turn can compute convolutions for arbitrary layers and kernels as would the source CNN on the corresponding tangent plane projections. Distinct from all existing methods, KTNs allow model transfer: the same model can be applied to different source CNNs with the same base architecture. KTNs successfully preserve the source CNN's accuracy, while offering transferability, scalability to typical image resolutions, and, in many cases, a substantially lower memory footprint.

1. Introduction

The 360° camera is an increasingly popular technology gadget. As a result, the amount of 360° data is increasing rapidly. For example, users uploaded more than a million 360° videos to Facebook in less than 3 years [1]. Because almost any application depends on semantic visual features, this rising trend prompts an unprecedented need for visual recognition algorithms on 360° images.

Today's wildly successful recognition CNNs are the result of tremendous data curation and annotation effort [2, 7, 9, 13, 15, 19], but they all assume perspective projection imagery. How can they be repurposed for 360° data? Existing methods often take an off-the-shelf model trained on perspective images and either 1) apply it repeatedly to multiple perspective projections of the 360° image [4, 17, 18, 20] or 2) apply it once to a single equirectangular projection [11, 12]. See Fig. 1(A,B). These two strategies, however, have severe limitations. The first is expensive because it has to project

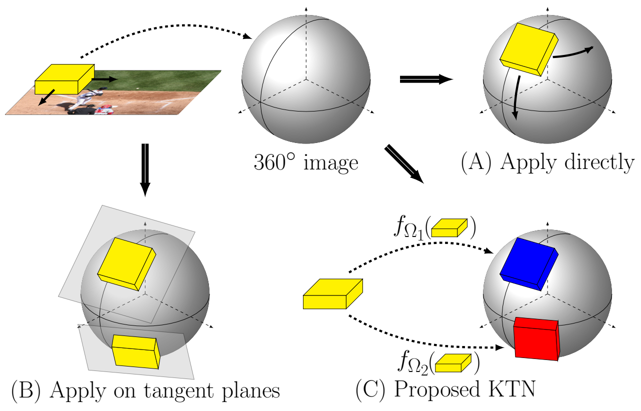

Figure 1: Our goal is to transfer CNNs trained on planar images to 360° images. Common approaches either (A) apply CNNs directly on the equirectangular projection of a 360° image or (B) project the content to tangent planes and apply the models on the tangent planes. In contrast, Kernel Transformer Network (KTN) adapts the kernels in CNNs to account for the distortion in 360° images.

the image and apply the recognition model repeatedly. The second is inaccurate because the visual content is distorted in equirectangular projection.

To overcome these challenges, recent work designs CNN models specifically for spherical data [5, 6, 8, 16, 21]. However, these approaches either introduce significant memory overhead or reduce the accuracy of the model. These shortcomings limit the applicability of the models to real-world data and applications. Furthermore, all of them require retraining to handle a new recognition task, which incurs a significant overhead for learning recognition models on 360° imagery.

In light of these shortcomings, we propose the Kernel Transformer Network (KTN). Instead of learning a new CNN on 360° images for a specific task, KTN learns a *function* that takes a kernel in the source CNN as input and transforms it to be applicable to a 360° image in its equirectangular projection. See Fig. 1 (C). The function accounts for the distortion in 360° images, returning different transformations depending on both the polar angle θ and the source kernel. The model is trained to reproduce the outputs of the source CNN on the perspective projection for each tangent plane on an arbitrary 360° image. Hence, KTN learns to behave similarly to the source CNN while avoiding repeated projection of the image.

Key highlights of the proposed KTN are its *transferability* and *compactness*—both of which owe to our function-based design. Once trained for a base architecture, the same KTN can transfer multiple source CNNs to 360° images. For example, having trained a KTN for VGG [15] on ImageNet classification, we can transfer the same KTN to run a VGG-based Pascal object detector on 360° panoramas. This is possible because the KTN takes the source CNN as input rather than embed the CNN kernels into its own parameters (unlike [5, 6, 8, 16, 21]). Furthermore, since the KTN factorizes source kernels from transformations, it is implementable with a lightweight network (e.g., increasing the footprint of a VGG network by only 25%). Finally, KTN is much more data efficient compared with other Spherical CNN methods [5, 6, 8, 21], because it does not require any annotated 360° images for training and is more accurate.

2. Approach

In this section, we first introduce the KTN module, which can replace the ordinary convolution operation in vanilla CNNs. We then describe the architecture and objective function of KTN.

2.1. KTN for Spherical Convolution

Our KTN can be considered as an generalization of ordinary convolutions in CNNs. In the convolution layers of vanilla CNNs, the same kernel is applied to the entire input feature map to generate the output feature map. The assumption underlying the convolution operation is that the feature patterns, i.e., the kernels, are translation invariant and should remain the same over the entire feature map. This assumption, however, does not hold in 360° images. A 360° image is defined by the visual content projected on the sphere centered at the camera's optical center. To represent the image in digital format, the sphere has to be unwrapped into a 2D pixel array, e.g., with equirectangular projection or cubemaps. Because all sphere-to-plane projections introduce distortion, the feature patterns are not translation invariant in the pixel space, and ordinary CNNs trained for perspective images do not perform well on 360° images.

To overcome this challenge, we propose the Kernel Transformer Network, which can generate kernels that account for the distortion. Assume an input feature map $I \in \mathbf{R}^{H\times W\times C}$ and a source kernel $K \in \mathbf{R}^{k\times k\times C}$ defined in undistorted images (i.e., perspective projection). Instead of applying the source kernel directly

$$F[x,y] = \Sigma_{i,j} K[i,j] * I[x-i, y-j], \tag{1}$$

we learn the KTN (f) that generates different kernels for different distortions:

$$K_\Omega = f(K, \Omega) \tag{2}$$

$$F[x,y] = \Sigma_{i,j} K_\Omega[i,j] * I[x-i, y-j] \tag{3}$$

where the distortion is parameterized by Ω. Because the distortion in 360° images is location dependent, we can define Ω as a function on the sphere

$$\Omega = g(\theta, \phi), \tag{4}$$

where θ and ϕ are the polar and azimuthal angle in spherical coordinates, respectively. Given the KTNs and the new definition of convolution, our approach permits applying an ordinary CNN to 360° images by replacing the convolution operation in Eq. 1 with Eq. 3.

KTNs make it possible to take a CNN trained for some target task (recognition, detection, segmentation, etc.) on ordinary perspective images and apply it directly to 360 panoramas. Critically, KTNs do so without using any annotated 360° images. Furthermore, because the information required to solve the target task is encoded in the source kernel, which is fed into the KTN as an *input* rather than part of the model, the same KTN can be applied to another CNN having the same base architecture but trained for a different target task. For example, we could train the KTN according to a VGG network trained for ImageNet classification, then apply the same KTN to transfer a VGG network trained for Pascal object detection; with the same KTN, both tasks can be translated to 360° images.

2.2. KTN Architecture

In this work, we consider 360° images that are unwrapped into 2D rectangular images using equirectangular projection. Equirectangular projection is the most popular format for 360° images and is part of the 360° video compression standard [3]. The main benefit of equirectangular projection for KTNs is that the distortion depends only on the polar angle. Because the polar angle has an one-to-one correspondence with the image row ($y{=}\theta H/\pi$) in the equirectangular projection pixel space, the distortion can be parameterized easily using $\Omega = g(\theta, \phi) = y$.

The architecture for KTN is in Fig. 2. We use a Residual Network [10]-like architecture. For both the residual and shortcut branches, we first apply a row dependent projection on the kernels. The projection consists of h projection matrices P_i, for $i \in [1, h]$, where h is the number of rows in the 360° image. The row dependent projection helps to generate different kernels at different polar angle θ. The residual branch then applies depthwise separable convolution twice. We use depthwise separable convolution to reduce the model size. The output kernel is then applied to a 360° feature map as in Eq. 3. Note that while the KTN can be applied to different kernels, the structure of a KTN depends on P_i, which is determined by the receptive field of the source kernel. Therefore, we need one KTN for each layer of a source CNN.

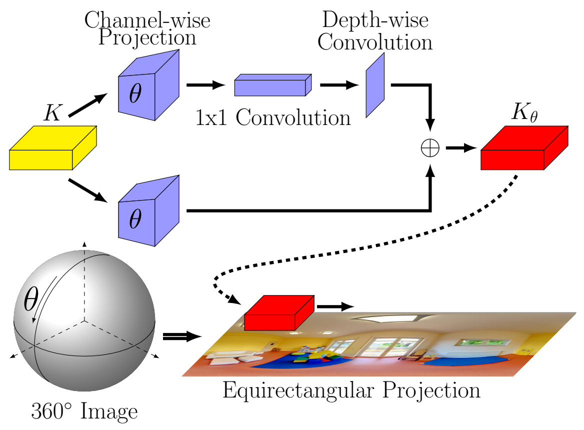

Figure 2: KTN consists of row dependent projections and depth separable convolution blocks. It takes a source kernel K and θ as input and generates an output kernel K_Ω. K_Ω is then applied to the 360° image in its equirectangular projection at row $y{=}\theta H/\pi$.

2.3. KTN Objective and Training Process

Having introduced the KTN module and how to apply it for CNNs on 360° images, we now describe the KTN objective function and training process. The goal of the KTN is to adapt the source kernel to the 360° domain. Therefore, we train the model to reproduce the outputs of the source kernels. Let $F^l \in \mathbf{R}^{H\times W\times C^l}$ and $F^{l+1} \in \mathbf{R}^{H\times W\times C^{l+1}}$ be the feature maps generated by the l-th and $(l{+}1)$-th layer of a source CNN respectively. Our goal is to minimize the difference between the feature map generated by the source kernels K^l and that generated by the KTN module:

$$\mathcal{L} = \|F^{l+1} - f^l(K^l, \Omega) * F^l\|^2 \qquad (5)$$

for any 360° image. Note that during training the feature maps F^l are not generated by applying the source CNN directly on the equirectangular projection of the 360° images. Instead, for each point (x, y) in the 360° image, we project the image content to the tangent plane of the sphere at

$$(\theta, \phi) = (\frac{\pi \times y}{H}, \frac{2\pi \times x}{W}) \qquad (6)$$

and apply the source CNN on the tangent plane. This ensures that the target training values are accurately computed on undistorted image content.

The objective function depends only on the source CNN and does not require any annotated data for training. In fact, it does not require image data specific to the target task, because the loss is defined over all 360° images. For example, in experiments we train a KTN on YouTube video frames and then apply it for a Pascal object detection task. Our goal is to fully reproduce the behavior of the source kernel. Therefore, even if the training images do not contain the same objects, scenes, etc. as are seen in the target task, the KTN should still minimize the loss in Eq. 5.

3. Experiments

We evaluate KTN on multiple datasets and source CNNs.

Datasets Our experiments make use of both unannotated 360° videos and 360° images with annotation.

Spherical MNIST is constructed from the MNIST dataset by back projecting the digits into equirectangular projection with 160×80 resolution. Classification accuracy on the 360°-ified test set is used as the evaluation metric.

Pano2Vid is a real world 360° video dataset [18]. We sample frames from non-overlapping videos for training and testing, and the frames are resized to 640×320 resolution. The models are trained to reproduce the convolution outputs of the source model, so no labels are required for training. The root-mean-square error (RMSE) of the final convolution outputs is used as the evaluation metric.

Pascal VOC 2007 is a perspective image dataset with object annotations. We backproject the object bounding boxes to equirectangular projection with 640×320 resolution. We evaluate the accuracy of the detector network on the validation set. This dataset is used for evaluation only.

Source Models For *Spherical MNIST*, we train the source CNN on the MNIST training set. The model consists of three convolution layers followed by one fully connected layer. For *Pano2Vid* and *Pascal VOC*, we take off-the-shelf Faster R-CNN [14] models with VGG architecture [15] as the source model. The Faster R-CNN is trained on Pascal VOC if not mentioned specifically.

Baselines We compare to the following existing methods:

- S^2CNN [5]—We train S^2CNN using the authors' implementation. Input resolution is reduced to 64×64 for *Pano2Vid* and *Pascal VOC* due to memory limits.
- SPHERICAL CNN [8]—We train SPHERICAL CNN using the authors' implementation. Input resolution is reduced to 80×80 for *Pano2Vid* and *Pascal VOC* due to memory limits.
- SPHERICAL U-NET [21]—We use the spherical convolution layer in Spherical U-Net to replace ordinary convolution in CNN. Input resolution is reduced to 160×80 for *Pano2Vid* and *Pascal VOC* due to memory limits.
- SPHERENET [6]—We implement SPHERENET using row dependent projection.[1] We derive the weights of the projection matrices using the feature projection operation and train the source kernels.
- SPHCONV [16]—We use the authors' implementation.
- PROJECTED—Similar to SPHERENET, except that it uses the source kernels without training.

For all methods, the number of layers and kernels are the same as the source model. Note that the resolution reductions specified above were necessary to even run those base-

[1] The authors' code and data were not available.

Table 1: Model accuracy.

	MNIST (Acc.↑)	*Pano2Vid* (RMSE ↓)	*Pascal VOC* (Acc.↑)
S^2CNN [5]	95.79	2.37	4.32
SPHERICAL CNN [8]	97.48	2.36	6.06
SPHERICAL U-NET [21]	98.43	2.54	24.98
SPHERENET [6]	87.20	2.46	46.68
SPHCONV [16]	**98.72**	**1.50**	63.54
PROJECTED	10.70	4.24	6.15
KTN	97.94	1.53	**69.48**

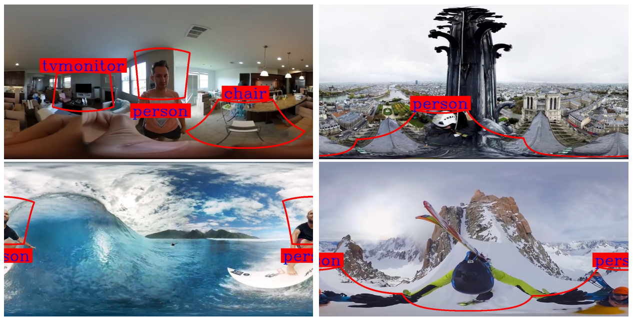

Figure 3: KTN object detection examples on *Pano2Vid*.

line models on the non-MNIST datasets, even with state-of-the-art GPUs. The fact that KTN scales to higher resolutions is precisely one of its technical advantages.

3.1. Model Accuracy

Table 1 summarizes the methods' accuracy on all three 360° datasets. KTN performs on par with the best baseline method (SPHCONV) on *Spherical MNIST*. Furthermore, KTN and SPHCONV perform significantly better than the other baselines on the high resolution datasets, i.e., *Pano2Vid* and *Pascal VOC*. The result verifies that KTN can transfer the source kernels to the entire sphere by learning to reproduce the feature maps, and it can match the accuracy of existing models trained with annotated 360° images. Note that the performance of PROJECTED and SPHERENET suggests that the transformation f cannot be modeled by a tangent plane-to-sphere projection. The accuracy of using a projection only transformation is significantly worse than KTN even if the model has a much larger number of learnable parameters (i.e. SPHERENET).

Fig. 3 shows example outputs of KTN with a Faster R-CNN source model. The detector successfully detects objects despite the distortion. On the other hand, KTN can fail when a very close object cannot be captured in the field-of-view of perspective images.

3.2. Transferability

Next, we evaluate the transferability of KTN across different source models on *Pano2Vid*. In particular, we evaluate whether KTNs trained with a COCO trained Faster R-CNN can be applied to a Pascal VOC trained Faster R-CNN (both using VGG architecture) and vice versa. We

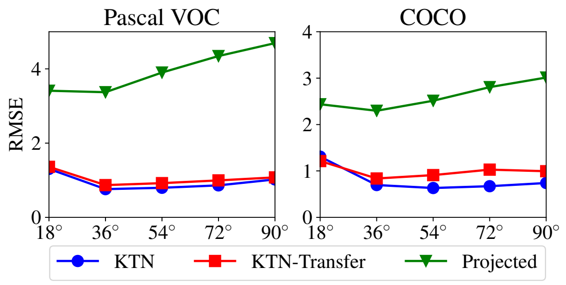

Figure 4: Model transferability. The title indicates the source CNN being tested.

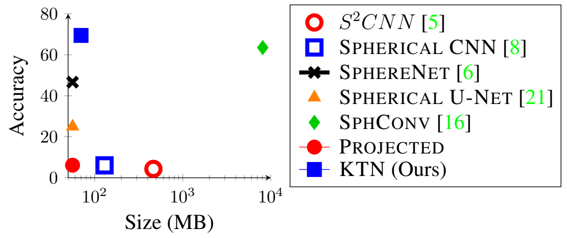

Figure 5: Model size vs. accuracy for VGG.

denote KTN trained on a different source CNN than it is being tested on as KTN-TRANSFER and KTN otherwise.

Fig. 4 shows the results. The accuracy of KTN and KTN-TRANSFER are almost identical. The results demonstrate that KTN indeed learns a task-independent transformation and can be applied to different source models with the same base architecture. None of the existing models [5,6,8,16,21] are equipped to perform this kind of transfer, because they learn fixed kernels for a specific task.

3.3. Model Size

Finally, we compare the model size of KTN versus the baseline methods. In particular, we measure the size for the convolution layers in the VGG architecture. Fig. 5 shows the results. The model size of KTN is only 25% (14MB) larger than the source model. At the same time, KTN achieves a much better accuracy compared with all the models that have a comparable size. Compared with SPHCONV, KTN not only achieves a higher accuracy but is also orders of magnitude smaller.

4. Conclusion

We propose the Kernel Transformer Network for transfering CNNs from perspective images to 360° images. KTN learns a function that transforms a kernel to account for the distortion in the equirectangular projection of 360° images. The same KTN model can transfer to multiple source CNNs with the same architecture. Our results show KTN outperforms existing methods while providing superior scalability and transferability.

Acknowledgement. We thank Carlos Esteves for the help on SPHERICAL CNN experiments. This research is supported in part by NSF IIS-1514118, an AWS gift, a Google PhD Fellowship, and a Google Faculty Research Award.

References

[1] Brent Ayrey and Christopher Wong. Introducing facebook 360 for gear vr. https://newsroom.fb.com/news/2017/03/introducing-facebook-360-for-gear-vr/, March 2017. 1

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 1

[3] Byeongdoo Choi, Ye-Kui Wang, and Miska M. Hannuksela. Wd on iso/iec 23000-20 omnidirectional media application format. ISO/IEC JTC1/SC29/WG11, 2017. 2

[4] Shih-Han Chou, Yi-Chun Chen, Kuo-Hao Zeng, Hou-Ning Hu, Jianlong Fu, and Min Sun. Self-view grounding given a narrated 360° video. In *AAAI*, 2018. 1

[5] Taco Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *ICLR*, 2018. 1, 2, 3, 4

[6] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *ECCV*, 2018. 1, 2, 3, 4

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: a large-scale hierarchical image database. In *CVPR*, 2009. 1

[8] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so(3) equivariant representations with spherical cnns. In *ECCV*, 2018. 1, 2, 3, 4

[9] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[11] Hou-Ning Hu, Yen-Chen Lin, Ming-Yu Liu, Hsien-Tzu Cheng, Yung-Ju Chang, and Min Sun. Deep 360 pilot: Learning a deep agent for piloting through 360° sports video. In *CVPR*, 2017. 1

[12] Wei-Sheng Lai, Yujia Huang, Neel Joshi, Chris Buehler, Ming-Hsuan Yang, and Sing Bing Kang. Semantic-driven generation of hyperlapse from 360° video. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2017. 1

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 1

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2, 3

[16] Yu-Chuan Su and Kristen Grauman. Learning spherical convolution for fast features from 360° imagery. In *NIPS*, 2017. 1, 2, 3, 4

[17] Yu-Chuan Su and Kristen Grauman. Making 360° video watchable in 2d: Learning videography for click free viewing. In *CVPR*, 2017. 1

[18] Yu-Chuan Su, Dinesh Jayaraman, and Kristen Grauman. Pano2vid: Automatic cinematography for watching 360° videos. In *ACCV*, 2016. 1, 3

[19] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1

[20] Youngjae Yu, Sangho Lee, Joonil Na, Jaeyun Kang, and Gunhee Kim. A deep ranking model for spatio-temporal highlight detection from a 360° video. In *AAAI*, 2018. 1

[21] Ziheng Zhang, Yanyu Xu, Jingyi Yu, and Shenghua Gao. Saliency detection in 360° videos. In *ECCV*, 2018. 1, 2, 3, 4