Les Cahiers du GERAD ISSN: 0711–2440

Improving classification performance on sparse data: Augmenting a convolutional neural net with generalized angular orientations of edges

A. A. Haji Abolhassani,
R. Dimitrakopoulos,
F. P. Ferrie, P. Lala

G–2019–98

December 2019

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2020
– Bibliothèque et Archives Canada, 2020

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2020
– Library and Archives Canada, 2020

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Improving classification performance on sparse data: Augmenting a convolutional neural net with generalized angular orientations of edges

Amir Abbas Haji Abolhassani [a,b]

Roussos Dimitrakopoulos [a,b]

Frank P. Ferrie [c]

Prasun Lala [c]

[a] *GERAD Montréal, Montréal (Québec), Canada, H3T 2A7*

[b] *COSMO – Stochastic Mine Planning Laboratory & Department of Mining and Materials Engineering, McGill University, Montréal (Québec), Canada, H3A 0E8*

[c] *Department of Electrical and Computer Engineering & the Centre for Intelligent Machines (CIM), McGill University, Montréal (Québec), Canada, H3A 0E9*

amir.hajiabolhassani@mcgill.ca
roussos.dimitrakopoulos@mcgill.ca
frank.ferrie@mcgill.ca
prasun.lala@cim.mcgill.ca

December 2019
Les Cahiers du GERAD
G–2019–98

Abstract: With sufficient layers, enough training data, enough time, and often a custom tailored architecture, modern deep learning methods can be extremely successful in classification tasks. However, with real-world practical applications, a lack of training data can impede the success of such techniques, which often over-fit the sparse data. A particular task hindered by sparse data, may take advantage of an existing pre-trained network of a separate but related task trained on sufficient data, to achieve high accuracy with minimal additional training. However, such transfer-learning does not generalize well to new tasks that have little relation between their data and the pre-trained network's data. To overcome this hurdle, we introduce a novel feature (AngOri) kernel that leverages the generalized inherent richness of curvature and gradient in image edges, and that can augment any type of convolutional neural network with any arbitrary number of channels to quickly achieve accurate classification with minimal training on sparse data. The AngOri kernels can be pre-computed and thus directly implemented in a convolutional layer of a network, which is not possible with other gradient based features. Testing on the MNIST, CIFAR-10, and a satellite image database, we consistently found AngOri to aid a network to achieve more accurate classification on a small dataset when compared to the same network without the AngOri layers. Such a generalizable, lightweight kernel holds promise for using neural networks to tackle real-world problems with limited resources, such as embedded systems examining sparse data.

Keywords: Texture flow, image gradient, angular orientation, classification, neural network, convolution

1 Introduction

A deep neural network is the most commonly used method for solving classification problems in computer vision, *e.g.* Object detection, Face recognition, etc. (Szegedy and Vanhoucke, 2016) because of its strength in learning the specific features that characterize the part of the underlying structure of data that is important for the target task. Deeper and wider networks result in higher classification accuracy but depend on large training datasets and longer training times. However, real-world practical applications often present sparse datasets (and limited resources) for learning the relevant features, thus causing traditional deep learning methods to perform inadequately. Transfer-learning (West and Ventura, 2007; Torrey and Shavlik, 2010, Kaboli, 2017) tackles the limitations of sparse data by leveraging the learned features from a pre-trained large-scale neural network *e.g.* VGG16/19 (Simonyan and Zisserman, 2014), INCEPTION (Szegedy and Liu, 2015), etc. These pre-trained, convolutional networks (ConvNets) are trained over large, accessible, labeled datasets for thousands of classes. In transfer-learning we keep (and freeze) all ConvNet layers, but substitute the Softmax layer, which is then fine-tuned by training on the new sparse dataset.

Transfer-learning is applicable only if the target training task is related to the source tasks trained in the pre-trained model, and if the target sparse training data is similar to the training data used for the pre-trained model. Thus, transfer-learning from one pre-trained network is not easily generalized to all kinds of tasks with sparse data.

One way to process data in a more generalized way is to use a more primal feature such as the Histogram of oriented gradients (HOG) (Dalal and Triggs, 2005) for computer vision tasks such as classification or detection. When looking at an environment with limited resources such as an embedded system, a HOG based system can be more efficient, but a ConvNet will still prove more accurate with enough data (Suleiman and Chen, 2017). Ideally when dealing with sparse data and limited resources, one would strive to combine the generalized and efficient traits of a feature like HOG with the potential accuracy of a ConvNet, as some have tried to do (Lipetski and Sidla, 2017). Unfortunately, one cannot directly augment a ConvNet with a feature like HOG as it cannot be added to a convolution layer; rather one must preprocess the data with the HOG feature in a separate step. The work in this paper aims to show how we can combine these benefits, but first we must outline how to measure performance, using transfer-learning as an example.

Transfer-learning is judged to improve the training process of a new task only if, at least, one out of three common measures are satisfied: first, the initial performance of the target task improves using the pre-trained model in early training steps compared to the randomly initialized network; second, the performance improves faster in transfer-learning compared to training from the randomly initialized network; and third, a higher maximum performance is achieved after many training steps in the transfer-learning case compared to the randomly initialized network. Otherwise, if transfer-learning decreases the performance, negative-transfer occurs (Olivas and Guerrero, 2010). (Figure 1) represents these three measures on a typical training plot. A simple example that has a potential for negative-transfer is training a classifier for the make of a car (training target task) given a pre-trained model that is trained only to detect general labels such as a car, a truck, a human, etc. The pre-trained model is never exposed to data pertaining to car makes; hence, the extracted features from the pre-trained model are more likely to suppress the useful information that discriminates one make from another.

Taking these measures and the above-stated goals into account, we propose a novel analytic method for generating a generalized kernel-based convolutional layer, that can be stacked as a building block of a neural network. Three set of experiments are provided that each validate all three measures of transfer-learning (Figure 1) (Olivas and Guerrero, 2010).

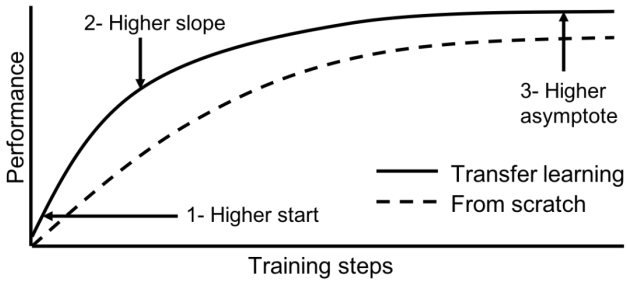


Figure 1: Three measures that indicate, the transfer-learning could improve the training process of a target task, e.g. classification task

2 Generalized angular orientations of edges to augment convolutional neural networks

We extend previous work (Ben-Shahar and Zucker, 2003; Abolhassani, Haji and Dimitrakopoulos, 2016) to propose a new method that uses Stochastic Angular Orientation (AngOri) of texture flows of intensity images, to initialize convolutional layers with an arbitrary number of nodes and channels. However, unlike a simple initialization step, we want to preserve the useful features in these layers, and so use a technique from transfer-learning, namely to freeze these layers during the first part of the learning. Texture flows comprise "locally parallel dense patterns" that define perceptual coherence for perceptual grouping in a manner useful for many feature-based computer vision tasks (Ben-Shahar and Zucker, 2003). (Figure 2) shows an 8-layered AngOri kernel of size 20x20 at two scales, each of which could be used for initializing the weights of any convolutional layer with size 20x20 and 8 channels.

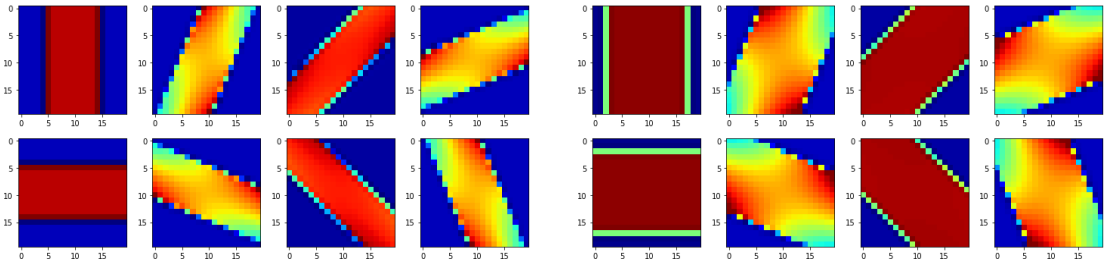
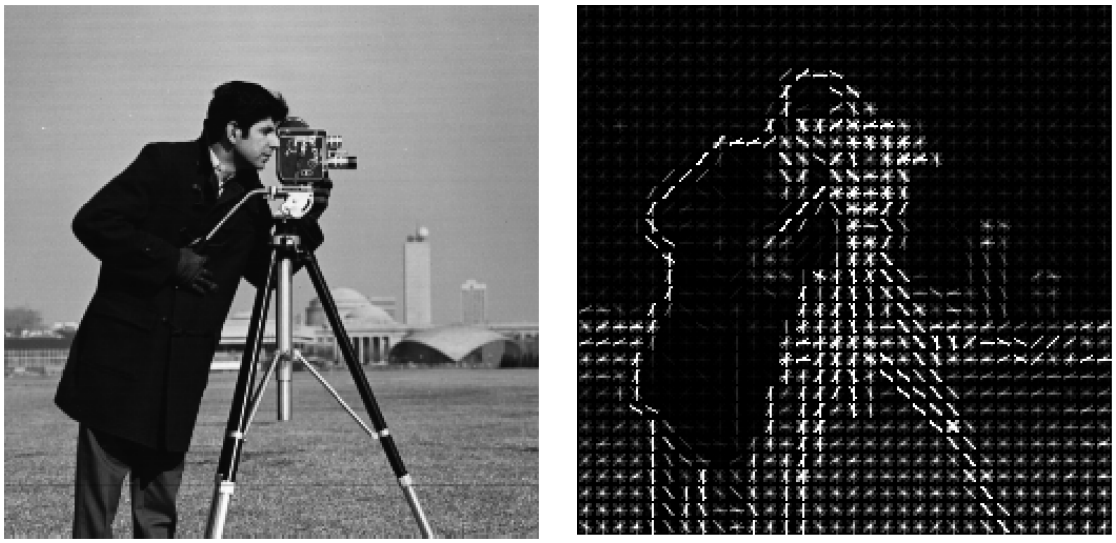


Figure 2: 8-layered AngOri kernel with size 20x20: (a) Scale-1 with 50% coverage of pixels the convolution window, (b) Scale-2 with 75% coverage of pixels the convolution window.

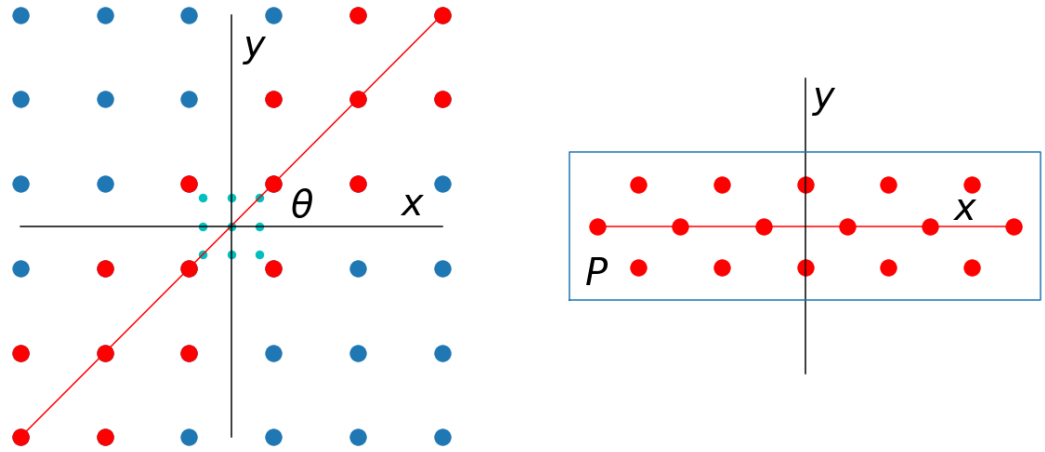
At each orientation θ_l, only the influence of n_s number of points is considered by the kernel. These n_s points are the closest points to the orientation line and are selected by the selection tensor S_θ with the shape $n_k \times n_k \times n_s$, (Figure 4).

The AngOri kernel calculates the strength of the edges along different directions, 0 to π, at every pixel of the intensity images. (Figure 3b) shows the normalized results of convolving an 8-directional AngOri kernel of size 16x16 with the image shown at left in (Figure 3a). At each pixel the intensities of the overlaid line along each direction encode the strength of the edge in that direction. A similar type of output is produced by the HOG method (Dalal and Triggs, 2005). The key advantage of AngOri when compared to HOG is that AngOri is implemented as a kernel that can be pre-computed and applied as a single convolution that produces the directional edge likelihood for the whole image. By comparison, HOG calculates the edge likelihood of each block by binning the local gradients and calculating the histogram of the orientations. The HOG process cannot be pre-computed and also needs to be calculated for each image separately. This computational advantage makes the AngOri kernel suitable for the initialization of ConvNets within the training process, while HOG cannot be similarly adapted.



(a) (b)

Figure 3: (a) is convolved with an 8-directional AngOri kernel of size 16x16 to produce (b). In (b), 8 orientations are presented by directional lines with gray-scale intensities representing the likelihood of an edge along that orientation.



(a) (b)

Figure 4: (a) The blue dots are the image pixels within the kernel window. Red dots are the first n_s number closest points to the orientation line at angle θ. The green dots in the fine grid are where the angular edge likelihoods are calculated. In this example, the size of the kernel, number of selected nodes and the size of the up-sampling fine grid are $k_n = 6,\ s_n = 16,\ n_{up} = 3$ respectively. (b) The selected points are rotated $-\theta$ to align with the x axis.

3 Calculating the AngOri kernel

The shape of the desired AngOri kernel $\hat{K}$ is $n_k \times n_k \times c_{in} \times c_{out}$ with equal height and width, n_k, and input and output channels c_{in} and c_{out}, respectively. This AngOri kernel consists of one lower-dimensional kernel, K, per input channel of shape $\mathrm{n_k \times n_k \times c_{out}}$.

$$K_{\{i,j,k,l\}} = K_{\{i,j,l\}} \text{ for all } i,j \in [0,\ \ldots,\ n_k - 1], k \in [0,\ \ldots,\ c_{in} - 1],\ \text{and } l \in [0,\ \ldots,\ c_{out} - 1].$$

We consider $\mathrm{c_{out}}$ to be the number of discrete orientations $c_{out} = n_\theta$. Hence, $l \in [0,\ \ldots,\ n_\theta]$ indexes the orientation parameter $\theta_l \in 0,\ \ldots,\ 2\pi$.

Each oriented kernel K_θ is multi-part and constructed by inner product of three θ oriented tensors.

$$K_\theta^{n_k \times n_k} = S_\theta^{n_k \times n_k \times n_s} \cdot E_\theta^{ns \ \times \ (n_{up} \times n_{up})} \cdot D_\theta^{(n_{up} \times n_{up})} \tag{1}$$

where S is the indicator tensor for selecting n_s number of points on the image grid closest to the orientation line with angle θ with x axis. E is a tensor for calculating the edge likelihood at the fine grid points, green points in (Figure 4), along direction θ. D is the anti-aliasing down-sampling tensor. Note, $(\cdot)$ indicates a matrix or vector dot product. Each tensor is defined in the following subsections.

Indicator tensor, S_θ:

Consider $Z \in$ input image, to be a part of the image overlapping with the kernel, where X and Y indicate the 2D position of the points relative to the kernel, all with shape $n_k \times n_k$. At each orientation θ, only a set of closest points to the orientation line is considered for calculating the likelihood of an edge in that direction, this selection is done by the indicator tensor S_θ. The points are first sorted based on the distance to the orientation line and then n_s number of closest points are selected:

$$\begin{aligned} \overline{X}_\theta^T &= X \bigotimes S_\theta = [x_0, \ \ldots, \ x_{n_s-1}] \\ \overline{Y}_\theta^T &= Y \bigotimes S_\theta = [y_0, \ \ldots, \ y_{n_s-1}] \\ \overline{Z}_\theta^T &= Z \bigotimes S_\theta = [z_0, \ \ldots, \ z_{n_s-1}] \end{aligned} \tag{2}$$

where $\bigotimes$ denotes the 2D covolution operator.

$\overline{X}_\theta$ and $\overline{Y}_\theta$ are then rotated $-\theta$ to align with axis x, denoted by $\overline{X}_\theta^{'}$ and $\overline{Y}_\theta^{'}$ and shown in Figure 4b. An example of rotation $-\theta$ applied to a point $(x, \ y)$ is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3}$$

The indicator tensor is sparse with only 1 element equal to 1 at each section k, $S_{\theta;i,j,k} = 1$ at $i = i_k, \ j = j_k$ and $S_{\theta;i,j,k} = 0$ otherwise.

Edge likelihood tensor, E_θ:

This is the main part of the kernel for calculating the likelihood of an edge to be along direction at the center of an overlapping part of the image with a sliding window of the kernel. (Figure 5a) shows a sample of the image exposed to the kernel in a 3D view. The red dots are the selected image points closest to the orientation line θ. (b) A red Quadratic-linear surface is fit to the red points. (c) The green plane is parallel to the xy plane and has a value $z = mean(Z)$. (d) The side view shows the plane curves at the intersection of the vertical plane, passing through the orientation line θ, the fitted surface (yellow) and the mean plane (blue). d is the distance between the fitted surface and the mean plane. The curvature of the fitted surface at $(x = 0, \ y = 0)$ is used for calculating the edge likelihood along θ and at the center of the patch. The likelihood of the edge increases proportional to the absolute value of $|d|$ and inversely proportional to the absolute value of the curvature κ at $(x = 0, \ y = 0)$.

$$\text{Likelihood of edge along } \theta = d(0,0) - \alpha \ \kappa(0,0) \tag{4}$$

with $\alpha > 0$. Qualitatively, d represents the edginess and κ represents the non-linearity of the edge. Next, we calculate both d and κ at the origin to re-factor the edge likelihood kernel.

Edginess, d:

d, as stated in (Section3), is actually the difference between value of the fitted surface at the origin and the mean value of the patch.

$$d(0,0) = Z^{fit}(0,0) - mean(Z) \tag{5}$$

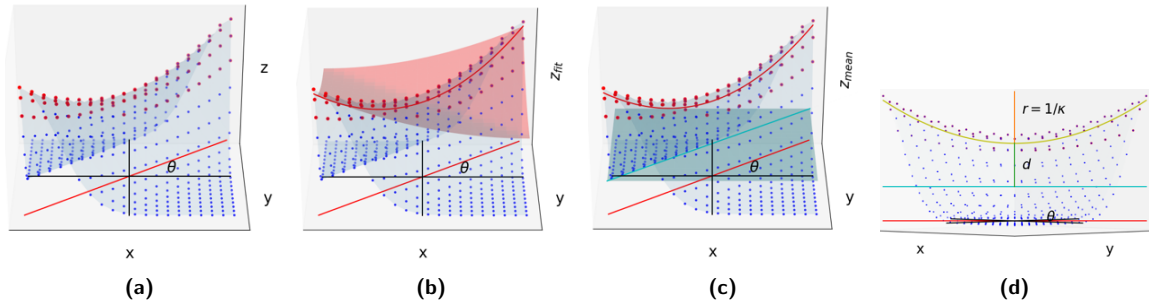


Figure 5: (a) part of image windowed by the kernel in 3D view. (b) Fitted surface to the selected points along orientation line at θ. (c) A plane parallel to the xy plane at $z = mean(Z)$. (d) side view of the image patch and the cross-section plane curves used for calculating the edge likelihood.

The mean term is trivial to re-factor:

$$mean(Z) = Z \bigotimes I^{n_k \times n_k}\left(\frac{1}{n_k^2}\right) \tag{6}$$

where I is the identity matrix and $\bigotimes$ is the 2D convolution operator.

Z^{fit} is a quadratic-linear surface P to a set of rotated selected points $\left(\overline{X'}_\theta,\ \overline{Y'}_\theta,\ \overline{Z}_\theta\right)$, (2) and (3).

$$\begin{aligned} P:\ Z^{fit} &= a^T \cdot \Phi \\ \Phi^T &= \left[\overline{X'}_\theta^{\,2}\ \overline{Y}'_\theta \quad \overline{X'}_\theta^{\,2} \quad \overline{X}'_\theta\ \overline{Y}'_\theta \quad \overline{Y}'_\theta \quad 1\ \right] \\ a^T &= \left[a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4\ \right] \end{aligned} \tag{7}$$

P is quadratic in x and linear in y direction. $\overline{X'}_\theta^{\,n}\overline{Y'}_\theta^{\,m}$ are all piece-wise vector operations. a is the vector of parameters of P, fitted to the set of selected points. The result of multi-linear least-square fit to the $\left(\overline{X'}_\theta,\ \overline{Y'}_\theta,\ \ \overline{Z}_\theta\right)$:

$$a^T = \overline{Z}_\theta^T \cdot\ Pinv\left(\Phi\right) \tag{8}$$

where $Pinv(.)$ is the pseudo inverse operator of a matrix. The Z^{fit} is now is calculated by multiplying (8) by $\phi = \left[x^2y, x^2, x\ y, y, 1\right]^T$ and substituting $\overline{Z}_\theta$ from (2):

$$Z^{fit(x,\ y)} = a^T \cdot \phi = Z \bigotimes\ S_\theta \cdot Pinv\left(\Phi\right) \cdot \phi \tag{9}$$

Furthermore,

$$Z^{fit(0,\ 0)} = a_4 = Z \bigotimes S_\theta \cdot Pinv\left(\Phi\right) \cdot \left[0,\ 0,\ 0,\ 0,\ 1\right]^T \tag{10}$$

We now substitute (10) and (6) into (5), Z could be re-factored, leaving the edginess part of the kernel:

$$d(0,0)\ = Z \bigotimes \left(I^{n_k \times n_k}\left(\frac{1}{n_k^2}\right) +\ S_\theta \cdot Pinv\left(\Phi\right) \cdot \left[0,\ 0,\ 0,\ 0,\ 1\right]^T \right) \tag{11}$$

Non-linearity of the edge, κ:

The non-linearity of the edge introduced in (4) is the curvature of the yellow plane curve in (Figure 5d), the cross-section of the fitted surface at $(0,\ 0)$ along the orientation line θ. By rotating the selected points by the axis z for $-\theta$, (Figure 5b), this cross-sectioning plane curve will be along axis x. Hence by substituting $\phi(x,\ y = 0)\ =\ \left[0, x^2, 0, 0, 1\right]^T$ in (9) we obtain:

$$Z^{fit}\left(x,\ 0\right) = a_1x^2\ +\ 1 = Z \bigotimes S_{\backslash theta} \cdot Pinv\left(\Phi\right) \cdot \left[0, x^2, 0, 0, 1\right]^T \tag{12}$$

The Curvature of the graph (12) is calculated by:

$$\begin{aligned}\kappa(x, y=0) &= \frac{\left|\frac{\partial^2 Z^{fit(x,\ y=0)}}{\partial x^2}\right|}{\left\{\left(1 + \left(\frac{\partial^2 Z^{fit(x,\ y=0)}}{\partial\ x^2}\right)^2\right)^{\frac{3}{2}}\right\}} \\ &= \frac{2a_1}{\left(1 + (2\ a_1 x)^2\right)^{\frac{3}{2}}} \rightarrow \\ \kappa(x=0, y=0) &= 2a_1 = Z \bigotimes S_\theta \cdot Pinv\left(\Phi\right) \cdot [0, 2, 0, 0, 0]^T\end{aligned} \tag{13}$$

while $a_1 > 0$.

Eventually, plugging (11) and (13) into (4), we arrive at the edge likelihood tensor in convolutional kernel format, the first two terms of the (1):

$$S_\theta \cdot E_\theta = Z \bigotimes (\ I^{n_k \times n_k}\left(\frac{1}{n_k^2}\right) + \ S_\theta \cdot Pinv\left(\Phi\right) \cdot [0,\ -2\alpha,\ 0,\ 0,\ 1]^T) \tag{14}$$

We have set $\alpha = \frac{8}{(n_k - 1)^2}$ for the experiments by matching the maximum of κ and d. For each Convolutional window, the edge likelihood E_θ is calculated at the sub-pixel $n_{up} \times n_{up}$ fine shape grid, the green dots in (Figure 5), for a smoother and more accurate result.

Anti-aliasing down-sampling tensor, D_θ:

The last part of the kernel in (1) is the anti-aliasing down-sampling tensor, D_θ to up-sample E_θ to the image pixel grid resolution. A 2D discretized double Sinc kernel is produced by the vector outer product of two double Sinc 1D kernels.

$$D_\theta = G^{Sinc^2}\left(f_s,\ f_a\right) \times G^{Sinc^2}(f_s,\ f_a) \tag{15}$$

where $f_s = \frac{2}{n_{up}}$ and $f_a = \frac{1}{n_{up}}\$$ are the down-sampling and anti-aliasing frequencies, respectively. The double Sinc 1D kernel is defined by,

$$G^{Sinc^2}(f_s,\ f_a) = (Sinc(f_s\ x)\ Sinc(f_a\ x)\ |_{\left\{x = \left[-\left(\frac{3n_{up}}{2} - 1\right),\ ...,\ \left(3\frac{n_{up}}{2} - 1\right)\right]\right\}} \tag{16}$$

4 Experiments

We have run three sets of experiments. The first two are, CIFAR-10 (Krizhevsky and Hinton, 2009) and MNIST (LeCun, Bottou and Bengio, 1998), to confirm the advantage of initialization of the ConvNets with AngOri kernels. The third one shows the AngOri kernels in action from building and initializing to training a ConvNet for classification. It is subsequently, evaluated on sparse ships images in satellite imagery dataset (Ships in satellite imagery dataset, 2018). We have used the ALL-CNN network configuration and added dropout between layers, and made it fully connected at the last layers, (Figure 6). We have chosen this neural network configuration because first, it reaches a respectable top 10 validation accuracy among all methods both in CIFAR-10 and MNIST datasets. Second, the size of ALL-CNN architecture seems reasonable and tractable based on our available hardware and time resources for testing the AngOri kernel. The only differences in the models used for the three experiments are the shape and size of the input layer and number of outputs of the Softmax layer, CIFAR-10 32x32x3 input and 10 output, MNIST 28x28x1 input and 10 output, ship satellite imagery 80x80x3 input and 2 output.

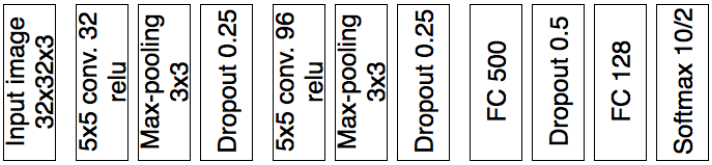


Figure 6: All-CNN-B model is used herein for all three set of experiments. Softmax layer has 10 outputs for CIFAR-10 and MNIST dataset, for ship satellite imagery it has 2 output.

CIFAR-10 and MNIST datasets:

The CIFAR dataset contains 60000 32×32 sized colored images of 10 different classes, 6000 per class, containing images of animals, cars, etc. The dataset is divided into mutually exclusive training set and test set of sizes 50000 and 10000, respectively. MNIST is a set of 70000 28x28 sized gray-scale images of single digit handwritten numbers, 0-9. 60000 samples are used for training and 10000 for testing.

We build two identical neural network models with ALL-CNN structures, (Figure 6). Hereafter, the first model is referred as AngOri-CNN and the second model is referred as CNN. The first two ConvNet layers of the AngOri-CNN are initialized by AngOri kernel but the remaining ConvNet layers in both models are initialized with the random uniform method. We use random uniform initialization method since it marginally outperforms other available initialization methods in our experiments.

The first experiment is on the CIFAR-10 dataset. We first randomly choose a subset of the original training set with different sizes, referred as reduced size training sets, ranging from 10 (1 image per class) to the full set of size 50000. The same form of reduced size training sets are produced for the second experiment on MNIST with the maximum size 60000 images per training set. In each experiment, for each reduced training set we initialize and train both models using identical batches of size 128 at each step randomly selected from the reduced training set. The training is continued for 50 epochs for CIFAR-10 and total of 40 epochs for MNIST. We freeze all the weights of two AngOri layers of the first model during the first half of training (mimicking the use-case of the pre-trained models in transfer-learning) and unfreeze them for the remaining epochs for fine tuning of the unfrozen weights. In contrast, the weights of the second model are unfrozen from beginning to end. We evaluate the trained models after every epoch by calculating the validation accuracy of each model over the whole test set, *i.e.* the mutually exclusive set of 10000 images for both CIFAR-10 and MNIST. (Figure 7a,b,d,e) represent the validation accuracy of both models for three examples of the reduced size training sets per experiment. A common feature in the results is the "head start" rise in the validation accuracy that the AngOri-CNN provides, thus satisfying the first described measure to validate the success of transfer-learning, (Figure 1) (Olivas and Guerrero, 2010). Another feature that is seen in most of the results, (Figure 7a,b,d,e), is the larger slope in the validation accuracy of the AngOri-CNN in first $\sim$20 epochs, thus satisfying the second described validation measure of transfer-learning, (Figure 1) (Olivas and Guerrero, 2010). Another common feature in the results is the short drop in validation accuracy of the AngOri-CNN quickly after unfreezing the weights. This stress is caused by the change in the back-propagation routine due to adding the new set of weights to be trained. Normally if the unfrozen weights are different from the optimum values, it disrupts the training process especially when the weights are in the lower layers and when we have dropout. This again supports the validity of our assumptions in the calculation of the AngOri kernels. Another interesting observation seen in almost all of the validation accuracy plots of the MNIST is the higher steady state validation accuracies achieved by AngOri-CNN especially in small size training sets, the third validation measure of the transfer-learning, (Figure 1) (Olivas and Guerrero, 2010).

Here the prior information afforded by the AngOri kernel makes up for the sparsity of the training data. This prior is rooted in the deep interpretation of the primal features, orientation of the edges in particular. (Figure 7c,f) represents all individual training sessions on every reduced size training set in two single graphs. These are semi-log plots of the validation accuracy of both models after 50 epochs (40 in MNIST case) as a discrete function of log of the size of reduced size training set. This shows

that a model with the same structure could achieve better validation accuracy after a fixed number of epochs if the weight of the ConvNet layers are initialized by AngOri kernel and frozen for some training epochs.

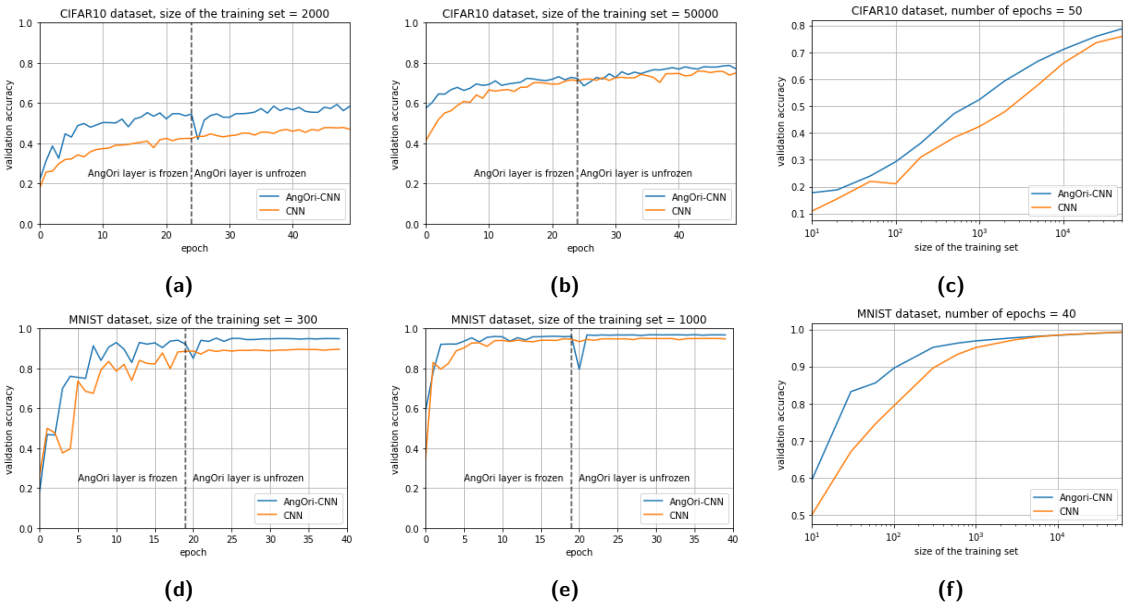


Figure 7: Experiment 1 and 2 on CIFAR-10 and MNIST dataset: (a), (b), (d) and (e) are the validation accuracy of AngOri-CNN and CNN models at every epoch. (c) and (f) represent maximum validation accuracy scores achieved after 100 training epochs plotted as a function of size of the training set.

Ships in satellite imagery:

This is a sparse set of 2800 colored images with size of 80x80, 2100 of which are non-ship and only 700 are ship images captured from satellite imagery, all of which are used for training. The model is validated visually by running the training classifier over a set of four test satellite images providing a full view of the San Francisco bay area. We adopted an ALL-CNN model, (Figure 6), for training a ship detector/classifier. Similar to previous experiments, the weights of two first ConvNet layers are initialized by AngOri kernel and the rest are initialized by the random uniform method. The training is continued for 40 epochs with a batch size of 32. The weights of the first two layers are frozen for 20 epochs. The trained model is then evaluated by four full satellite images, (Figure 8a,b,c,d). For evaluation we slide a search window of size 80x80 over the image with the stride of 10x10 and all generated 80x80 images are classified by the trained AngOri-CNN as ship/non-ship; the ship classified windows are then labeled by a green rectangle overlaid on the original full image. This is a challenging problem first because of the sparsity of the labeled training data. Second, this is an unbalanced classification problem both in the training dataset (700 ship *vs.* 2100 non-ship images) and evaluation (10 ship *vs.* 5M non-ship images). On the positive side of the problem, the shape of the ships are not very complex considering the background sea which is almost similar for all of the cases. Despite these conditions the AngOri-CNN detected all of the ships in the images with at most 3 false positives (non-ship images detected as ship).

5 Conclusions

Sparse and novel datasets should not impede the use of ConvNets to achieve quick and accurate classification training. The AngOri kernel characterizes curvature and gradient in image edges and can be pre-computed and directly implemented in a convolutional layer of a network (which is not possible with other gradient features).

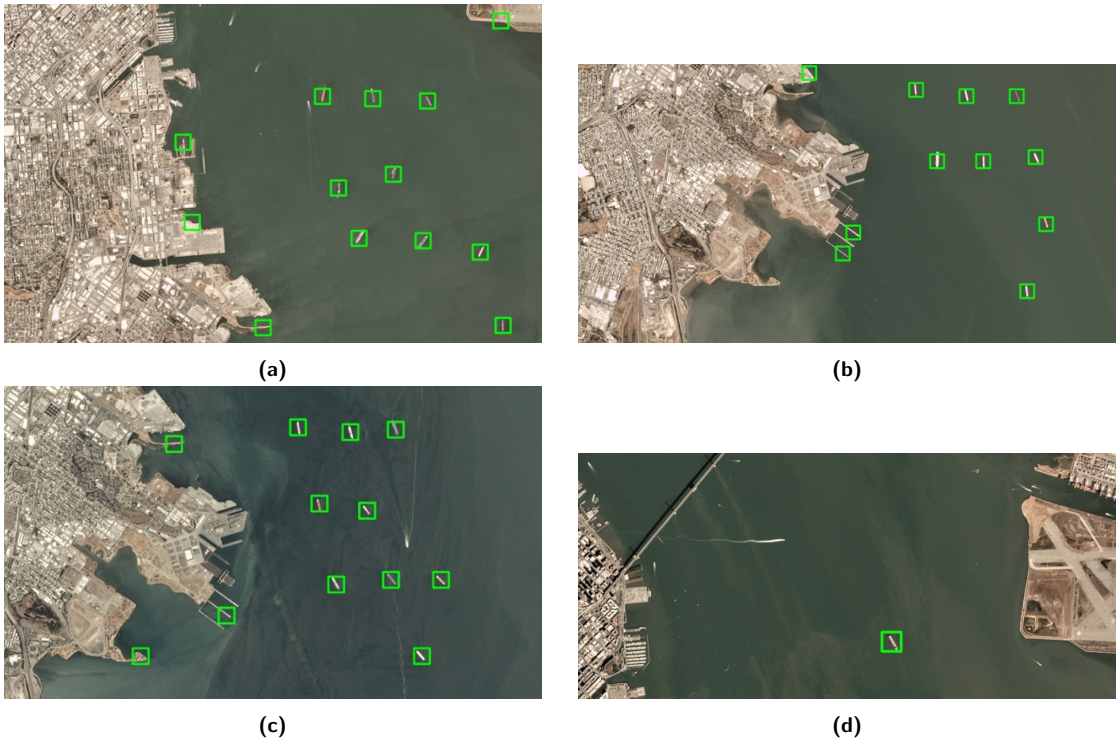


Figure 8: Experiment 3 on ships in satellite imagery: The trained AngOri-CNN is evaluated by classifying every 80x80 window of the image, the window slides with the stride of 10x10 pixel.

This method could lead to quick and computationally efficient ConvNets being implemented on systems lacking traditional computational resources *e.g.* embedded systems. Our results show that this new feature and methodology can lead to quick and accurate classification on small, diverse datasets. The methodology satisfies the three measures indicating the success of transfer-learning, while avoiding some pitfalls of transfer-learning *e.g.* inability to be easily generalized in particular sparse-data cases by replacing it with the AngOri kernel. In the future we plan to explore what training benefits the AngOri kernel can bring to large-scale networks. We also plan to extend the AngOri feature itself to capture even more complex edge shapes and explore the feature's use in other computer vision and general machine learning domains.

References

Szegedy C., Vanhoucke V., Ioffe S. , Shlens J. and Wojna Z. (2016) Rethinking the inception architecture for computer vision. CVPR, 2818–2826.

Schroff F., Kalenichenko D. and Philbin J. (2015) Facenet: A unified embedding for face recognition and clustering. CVPR, 815–823.

West J., Ventura D. and Warnick S. (2007) Spring research presentation: A theoretical foundation for inductive transfer. Brigham Young University, College of Physical and Mathematical Sciences, 1.

Torrey L., Shavlik J., Walker T. and Maclin R. (2010) Transfer learning via advice taking. ML, 147–170.

Kaboli M. (2017) A Review of Transfer Learning Algorithms. Technische Universität München.

Simonyan K. and Zisserman A. (2014) Very deep convolutional networks for large-scale image recognition. CoRR, 1409.1556.

Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V. and Rabinovich A. (2015) Going deeper with convolutions. CVPR.

Dalal N. and Triggs B. (2005) Histograms of oriented gradients for human detection. Computer Vision and Pattern Recognition. 1:886–893.

Suleiman A., Chen Y.-H., Emer J. and Sze V. (2017) Towards closing the energy gap between HOG and CNN features for embedded vision. IEEE ISCAS.

Lipetski Y. and Sidla O. A combined hog and deep convolution network cascade for pedestrian detection. (2017) Electronic Imaging, 4:11–17

Olivas E. S., Guerrero J. M. and Martinez-Sober M. (2010) Transfer Learning. Hershey, PA, USA: IGI Global, 242–264.

Ben-Shahar O. and Zucker S. W. (2003) The perceptual organization of texture flow: a contextual inference approach. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 25(4):401–417.

H. Abolhassani A. A., Dimitrakopoulos R. and Ferrie F. P. (2016) Anisotropic interpolation of sparse images. Computer and Robot Vision 440–447.

Krizhevsky A. and Hinton G. (2009) Learning multiple layers of features from tiny images. Techreport by Citeseer, Tech. Rep.

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

Ships in satellite imagery dataset. [Online]. Available: https://www.kaggle.com/rhammell/ships-in-satellite-imagery