DISCO: accurate Discrete Scale Convolutions Supplementary Materials

Ivan Sosnovik
i.sosnovik@uva.nl
Artem Moskalev
a.moskalev@uva.nl
Arnold Smeulders
a.w.m.smeulders@uva.nl

UvA-Bosch Delta Lab
University of Amsterdam
Netherlands

1 Proofs

We have shown that scale-convolution is indeed scale-equivariant only if the kernel κ and its up-scaled version $\kappa_{s^{-1}}$ satisfy the following relation

$$L_s[f] \star \kappa = L_s[f \star \kappa_{s^{-1}}], \quad \forall f, s \tag{1}$$

where L_s is an operator of downscaling.

1.1 Solutions in 1D

Let us consider an operator of downscaling L_s, which is represented as a rectangular interpolation matrix $\boldsymbol{L}$ of size $N_{\text{out}} \times N_{\text{in}}$. A convolution with a kernel κ is represented as a multiplication with a matrix $\boldsymbol{K}$ of size $N_{\text{out}} \times N_{\text{out}}$, and with a kernel $\kappa_{s^{-1}}$ written as a matrix $\boldsymbol{K}_{s^{-1}}$ of size $N_{\text{in}} \times N_{\text{in}}$. The equivariance constraint with respect to $\kappa_{s^{-1}}$ is written as follows:

$$\boldsymbol{K}\boldsymbol{L} = \boldsymbol{L}\boldsymbol{K}_{s^{-1}} \tag{2}$$

Lemma 1. *Equation 2 has non-trivial solutions with respect to $\boldsymbol{K}_{s^{-1}}$ only if L performs downscaling by an integer factor.*

Proof. Let us consider $\boldsymbol{P}_{\text{in}}$ and $\boldsymbol{P}_{\text{out}}$, matrices of circular shift of rows of sizes $N_{\text{in}} \times N_{\text{in}}$ and $N_{\text{out}} \times N_{\text{out}}$ correspondingly. With no loss of generality we assume circular boundary conditions for convolutions. Thus, matrices $\boldsymbol{K}, \boldsymbol{K}_{s^{-1}}$ are circulant, and therefore $\boldsymbol{K} = \boldsymbol{P}_{\text{out}}\boldsymbol{K}\boldsymbol{P}_{\text{out}}^T$ and $\boldsymbol{K}_{s^{-1}} = \boldsymbol{P}_{\text{in}}\boldsymbol{K}_{s^{-1}}\boldsymbol{P}_{\text{in}}^T$ [3]. If we substitute it into Equation 2 we have the following:

$$\boldsymbol{P}_{\text{out}}^i \boldsymbol{K} (\boldsymbol{P}_{\text{out}}^T)^i \boldsymbol{L} = \boldsymbol{L}\boldsymbol{P}_{\text{in}}^j \boldsymbol{K}_{s^{-1}} (\boldsymbol{P}_{\text{in}}^T)^j, \quad \forall i, j \in \mathbb{Z} \tag{3}$$

If we multiply it from the left by $(\boldsymbol{P}_{\text{out}}^T)^i$ and from the right by $\boldsymbol{P}_{\text{in}}^j$ we get the following equation:

$$\boldsymbol{K}(\boldsymbol{P}_{\text{out}}^T)^i\boldsymbol{L}\boldsymbol{P}_{\text{in}}^j = (\boldsymbol{P}_{\text{out}}^T)^i\boldsymbol{L}\boldsymbol{P}_{\text{in}}^j\boldsymbol{K}_{s^{-1}} \tag{4}$$

We can now multiply Equation 4 by a coefficient α_{ij} and then the following holds true:

$$\boldsymbol{K}\sum_{i=1}^{N_{\text{out}}}\sum_{j=1}^{N_{\text{in}}}\alpha_{ij}\boldsymbol{Q}_{ij} = \sum_{i=1}^{N_{\text{out}}}\sum_{j=1}^{N_{\text{in}}}\alpha_{ij}\boldsymbol{Q}_{ij}\boldsymbol{K}_{s^{-1}}, \forall\alpha_{ij} \tag{5}$$

where $\boldsymbol{Q}_{ij} = (\boldsymbol{P}_{\text{out}}^T)^i\boldsymbol{L}\boldsymbol{P}_{\text{in}}^j$. Equation 5 holds true for all α_{ij}. Which gives us the following system of equations:

$$\begin{cases} \boldsymbol{K}(\boldsymbol{Q}_{00} - \boldsymbol{Q}_{N_{\text{out}}N_{\text{in}}}) = (\boldsymbol{Q}_{00} - \boldsymbol{Q}_{N_{\text{out}}N_{\text{in}}})\boldsymbol{K}_{s^{-1}} \\ \boldsymbol{K}(\boldsymbol{Q}_{01} - \boldsymbol{Q}_{N_{\text{out}},N_{\text{in}}+1}) = (\boldsymbol{Q}_{01} - \boldsymbol{Q}_{N_{\text{out}},N_{\text{in}}+1})\boldsymbol{K}_{s^{-1}} \\ \dots \\ \boldsymbol{K}(\boldsymbol{Q}_{10} - \boldsymbol{Q}_{N_{\text{out}}+1,N_{\text{in}}}) = (\boldsymbol{Q}_{10} - \boldsymbol{Q}_{N_{\text{out}}+1,N_{\text{in}}})\boldsymbol{K}_{s^{-1}} \\ \dots \end{cases} \tag{6}$$

Which has non-trivial solutions if the expressions in all brackets are equal to zero. Thus, $\boldsymbol{Q}_{00} = \boldsymbol{Q}_{N_{\text{out}}N_{\text{in}}}$. In other words, $\boldsymbol{L}$ is a row-circulant matrix and N_{in} is divisible by N_{out}. Therefore, the downscaling is performed by an integer factor $N_{\text{in}}/N_{\text{out}}$ □

In order to obtain the solution of Equation 2 we represent convolutional matrices by using their eigendecompositions.

$$\begin{aligned} \boldsymbol{K} &= \boldsymbol{F}_{\text{out}}\text{diag}(\boldsymbol{F}_{\text{out}}\boldsymbol{\kappa})\boldsymbol{F}_{\text{out}}^* \\ \boldsymbol{K}_{s^{-1}} &= \boldsymbol{F}_{\text{in}}\text{diag}(\boldsymbol{F}_{\text{in}}\boldsymbol{\kappa}_{s^{-1}})\boldsymbol{F}_{\text{in}}^* \end{aligned} \tag{7}$$

where $\boldsymbol{F}_{\text{in}}, \boldsymbol{F}_{\text{out}}$ are matrices of the Discrete Fourier Transform of appropriate sizes and $\boldsymbol{\kappa}, \boldsymbol{\kappa}_{s^{-1}}$ are vector representations of convolutional kernels. After substituting the second part of Equation 7 into Equation 2 we obtain:

$$\boldsymbol{K}\boldsymbol{L} = \boldsymbol{L}\boldsymbol{F}_{\text{in}}\text{diag}(\boldsymbol{F}_{\text{in}}\boldsymbol{\kappa}_{s^{-1}})\boldsymbol{F}_{\text{in}}^* \tag{8}$$

We then multiply both sides of the equation with $\boldsymbol{F}_{\text{in}}$ from the right.

$$(\boldsymbol{K}\boldsymbol{L}\boldsymbol{F}_{\text{in}})_{ij} = \sum_k (\boldsymbol{L}\boldsymbol{F}_{\text{in}})_{ik}\text{diag}(\boldsymbol{F}_{\text{in}}\boldsymbol{\kappa}_{s^{-1}})_{kj} \tag{9}$$

As the left hand side is per-column proportional to $\boldsymbol{L}\boldsymbol{F}_{\text{in}}$, we can calculate the solution just by using the first row of each matrix.

$$(\boldsymbol{F}_{\text{in}}\boldsymbol{\kappa}_{s^{-1}})_j = \frac{(\boldsymbol{K}\boldsymbol{L}\boldsymbol{F}_{\text{in}})_{1j}}{(\boldsymbol{L}\boldsymbol{F}_{\text{in}})_{1j}} \tag{10}$$

The first row of $\boldsymbol{F}_{\text{in}}$ consists of ones so as the first row of $\boldsymbol{L}\boldsymbol{F}_{\text{in}}$. Additionally, $(\boldsymbol{K}\boldsymbol{L}\boldsymbol{F}_{\text{in}})_{1j} = s^{-1}[\boldsymbol{\kappa}, \boldsymbol{\kappa}]_j$. As the discrete Fourier image of the solution is a scaled concatenated image of the source, the solution is just a dilation of the original kernel [3].

1.2 Solutions in 2D

We are interested in solving Equation 1 with respect to $\kappa_{s^{-1}}$ for any set of κ's which forms a complete basis in the space of square matrices of a certain, fixed size. If the solution exists for any basis, then it exists for a basis of 2-dimensional separable kernels. As the rank of the set of solutions is less or equal to the rank of the initial basis, the solution is separable as well. Let us consider an image $\boldsymbol{F}$ of size $N_{\text{in}} \times N_{\text{in}}$. Taking into account that its rescaling is a separable operation, the matrix form of Equation 1 is:

$$\boldsymbol{K}'\boldsymbol{L}\boldsymbol{F}\boldsymbol{L}^T\boldsymbol{K}^T = \boldsymbol{L}\boldsymbol{K}'_{s^{-1}}\boldsymbol{F}\boldsymbol{K}^T_{s^{-1}}\boldsymbol{L}^T, \quad \forall \boldsymbol{F} \tag{11}$$

where $\boldsymbol{K}'$ and $\boldsymbol{K}$ are matrix representations of 1-dimensional components of a separable kernel. As Equation 11 holds true for all images, it satisfies $\boldsymbol{F} = \boldsymbol{f}\boldsymbol{c}^T$ and $\boldsymbol{F} = \boldsymbol{c}\boldsymbol{f}^T$ where $\boldsymbol{c}$ is a vector of constants and $\boldsymbol{f}$ is an arbitrary vector. After substituting these functions into Equation 11 it degenerates into a system of two independent equations up to a multiplication constant:

$$\begin{cases} \boldsymbol{K}\boldsymbol{L} = \boldsymbol{L}\boldsymbol{K}_{s^{-1}} \\ \boldsymbol{K}'\boldsymbol{L} = \boldsymbol{L}\boldsymbol{K}'_{s^{-1}} \end{cases} \tag{12}$$

Thus, if a solution exists for 2-dimensional discrete signals it also exists for the 1-dimensional case.

2 Implementation Details

2.1 Scale Convolution

Let us consider a scale-convolutional layer defined on scales $\{1, \sqrt{2}, 2, 2\sqrt{2}, 4, \}$. The kernel on the smallest scale is of size 3×3. As it was noted, as soon as the kernel on the intermediate $\sqrt{2}$ scale is defined, all other kernel can be calculated via dilation.

In scale-convolutional layer the kernels κ are parametrized as follows:

$$\kappa_s = \sum_j \psi_{s,j} w_j \tag{13}$$

where $\psi_{s,j}$ is a j-th basis function defined on scale s, and w_j is the corresponding trainable coefficient.

As the basis is fixed during the training, it needs to be defined *a priori*. On the smallest scale all basis functions are just elements of the standard basis, i.e. if $\psi_{1,i}$ is the i-th basis function for the 3×3 filters on the first scale, then $\psi_{1,0}$ is a 3×3 matrix where the only non-zero element is a 1 in the top-left corner, and $\psi_{1,4}$ is a 3×3 matrix with 1 in the center. On the next integer scale 2, the basis is obtained according to Equation 12 of the main paper and computed as a dilation of $\psi_{1,i}$. To obtain non-integer scale bases we start by approximating the first intermediate $\sqrt{2}$ scale basis $\psi_{\sqrt{2},j}$ functions by minimizing the following objective function:

$$\|L[f] \star \psi_{1,j} - L[f \star \psi_{\sqrt{2},j}]\|_F^2 + \|L[f] \star \psi_{\sqrt{2},j} - L[f \star \psi_{2,j}]\|_F^2 \tag{14}$$

where f is a random sample from $\mathcal{N}(0,1)$ and L is an operation of downsampling by a factor of $\sqrt{2}$ by using bicubic interpolation. The basis the scale $\{2\sqrt{2}\}$ is calculated as a dilation of the approximated $\sqrt{2}$ basis. See Figure 1 for more details.

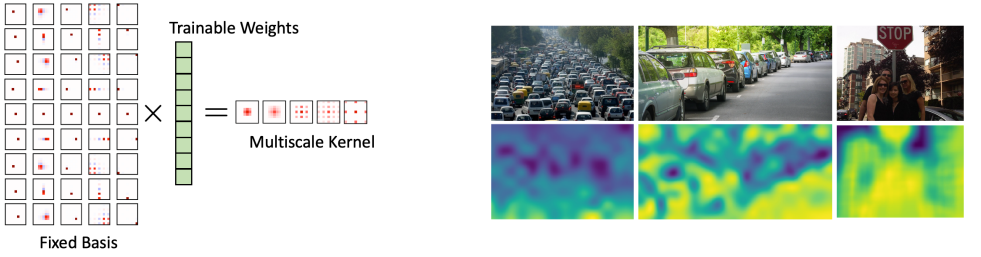

Figure 1: Left: kernels are computed via multiplying a fixed multi-scale basis with trainable weights. Right: images and their scale fields produced by the DISCO model trained to contrast scales.

After all basis functions are calculated, the basis is packed into a tensor of size:

$$\texttt{num\_functions} \times \texttt{num\_scales} \times \texttt{height} \times \texttt{width}$$

and used for runtime kernel calculations with the algorithm provided by [4].

2.2 Computational Complexity

Let us consider a scale-convolutional layer with a set of N_s scales with step $\sigma > 1$. The smallest kernel size is $W \times W$. The computational complexity for calculating the output for one spatial position for the state-of-the-art method from [4] can be estimated as follows:

$$O(SESN) \sim O(W^2(1+\sigma^2+\cdots+\sigma^{2N_s-2})) \sim O\left(W^2\frac{\sigma^{2N_s}-1}{\sigma^2-1}\right) \sim O(W^2\sigma^{2N_s}) \tag{15}$$

In contrast, for DISCO we arrive the following complexity:

$$O(DISCO) \sim O(N_s W^2) \tag{16}$$

Thus, where the state of the art SESN convolution grow exponentially in computational complexity with the number of scales, DISCO allow for linear growth.

When using a scale step of $\sqrt{2}$ we achieve a speedup of:

$$\frac{O(SESN)}{O(DISCO)} \sim \frac{2^{N_s}}{N_s}. \tag{17}$$

The main reason for the acceleration is that in SESN the filters are dense, as they are rescaled in the continuous domain by using Equation 4 of the main paper, while DISCO filters are sparse as the rescaling is performed by using dilation for the majority of scales. The actual speedup depends on the particular implementation of scale-convolution with such kernels. The current implementation is limited by the functionality of modern deep learning software which is not optimized for sparse filters of a big spatial extend.

2.3 General Solution

While in many models which consider scale the scale-step is a root of some integer number, it is possible to build a DISCO model with arbitrary scale-steps. Let us consider a scale-

Interpolation	Nearest	Bilinear	Bicubic
Error	1.36 ± 0.06	1.37 ± 0.05	1.35 ± 0.05

Table 1: Classification accuracy on MNIST-scale for different interpolation methods which are used for approximate basis calculations.

convolutional layer defined on scales $\{s_0, as_0, a^2 s_0, \dots a^N s_0\}$ where $a > 1$. In order to construct kernels for such a layer it is first required to calculate a basis $\{\psi_{s_0,j}, \psi_{as_0,j}, \dots \psi_{a^N s_0,j}\}$ for all j. The basis can be calculated as a minimizer of the following objective:

$$\mathcal{L}(\psi_{s_0,j}, \psi_{as_0,j}, \dots \psi_{a^N s_0,j}) = \mathbb{E}_f \sum_{\substack{k,l=0 \\ k>l}}^{k,l=N} \|L_{a^{l-k}}[f] \star \psi_{a^l s_0,j} - L_{a^{l-k}}[f \star \psi_{a^k s_0,j}]\|_F^2 \tag{18}$$

3 Experiments

3.1 MNIST-scale

As a baseline model we use the SESN model [4]. It consists of 3 convolutional and 2 fully-connected layers. Each layer has filters of size 7×7. We keep the number of parameters the same for all SESN models and for DISCO. The main difference between the SESN and DISCO models is in the basis for scale-convolutions. We also discovered that average-pooling works slightly better for the DISCO, while for all other methods it either has no effect or worsens the performance. Both SESN and DISCO use the same set of scales in scale convolutions: $\{1, 2^{1/3}, 2^{2/3}, 2\}$

All models are trained with the Adam optimizer [7] for 60 epochs with a batch size of 128. We set the initial learning rate at 0.01 and divide it by 10 after 20 and once more after 40 epochs. We conduct the experiments with 2 different settings: without data augmentation and with scaling augmentation. We run the experiments on 6 different realizations of the MNIST-scale. We report the mean $\pm$ standard deviation over these runs.

We found in our experiments that the interpolation method which is used to calculate a basis by using equation 14 does not affect the final solution. The relative mean squared error between bases is less than percent. Moreover, DISCO model demonstrates almost the same results on MNIST-scale while various interpolation methods are used. See Table 1 for more results.

3.2 STL-10

As a baseline we use WideResNet [5] with 16 layers and a widening factor of 8. Scale-equivariant models are constructed according to [4]. All models have the same number of parameters. The scale factors in the scale convolutions are $\{1, \sqrt{2}, 2\}$.

The models are trained for 1000 epochs using the SGD optimizer with a Nesterov momentum of 0.9 and a weight decay of $5 \cdot 10^{-4}$. For DISCO, we increase the weight decay to $1 \cdot 10^{-4}$. Tuning weight decay for the other models did not bring any improvement. The learning rate is set to 0.1 at the start and decreased by a factor of 0.2 after the epochs 300, 400, 600 and 800. The batch size is set to 128. During training, we additionally augment the dataset with random crops, horizontal flips and cutout [1].

3.3 Scene Geometry by Contrasting Scales

For clarity we provide a PyTorch pseudo-code for DISCO scene geometry estimation (Listing 1). We utilize scale-equivariant ResNet as a backbone feature extractor. The produced feature map is reduced in a spatial domain. Then the `argmax` along the scale dimension is extracted and passed to the scale MLP regressor to produce a scale estimate. Additional qualitative results are presented in Figure 1.

Listing 1: PyTorch pseudo-code for DISCO scene geometry estimation.

```
import torch.nn as nn
import SE_ResNet

class ScaleEstimator(nn.Module):
    def __init__(self):
        super().__init__()
        self.backbone = SE_ResNet(pretrained=True)
        self.regressor = nn.Sequetial(
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Linear(256, 1),
            nn.ReLU()
        )

    def forward(self, x):
        # x.shape = B, 3, 64, 64
        y = self.backbone(x)
        # y.shape = B, 512, 9, 1, 1
        y = y.mean(-1).mean(-1)
        # y.shape = B, 512, 9
        y = y.argmax(-1)
        scale = self.regressor(y)
        return scale
```

References

[1] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] Nicholas Loehr. *Advanced linear algebra*. CRC Press, 2014.

[4] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. *arXiv preprint arXiv:1910.11093*, 2019.

[5] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.