Linearized Multi-Sampling for Differentiable Image Transformation

https://tinyurl.com/linearized

Wei Jiang[1], Weiwei Sun[1], Andrea Tagliasacchi[1,2], Eduard Trulls[2], Kwang Moo Yi[1]

[1]University of Victoria [2]Google Research

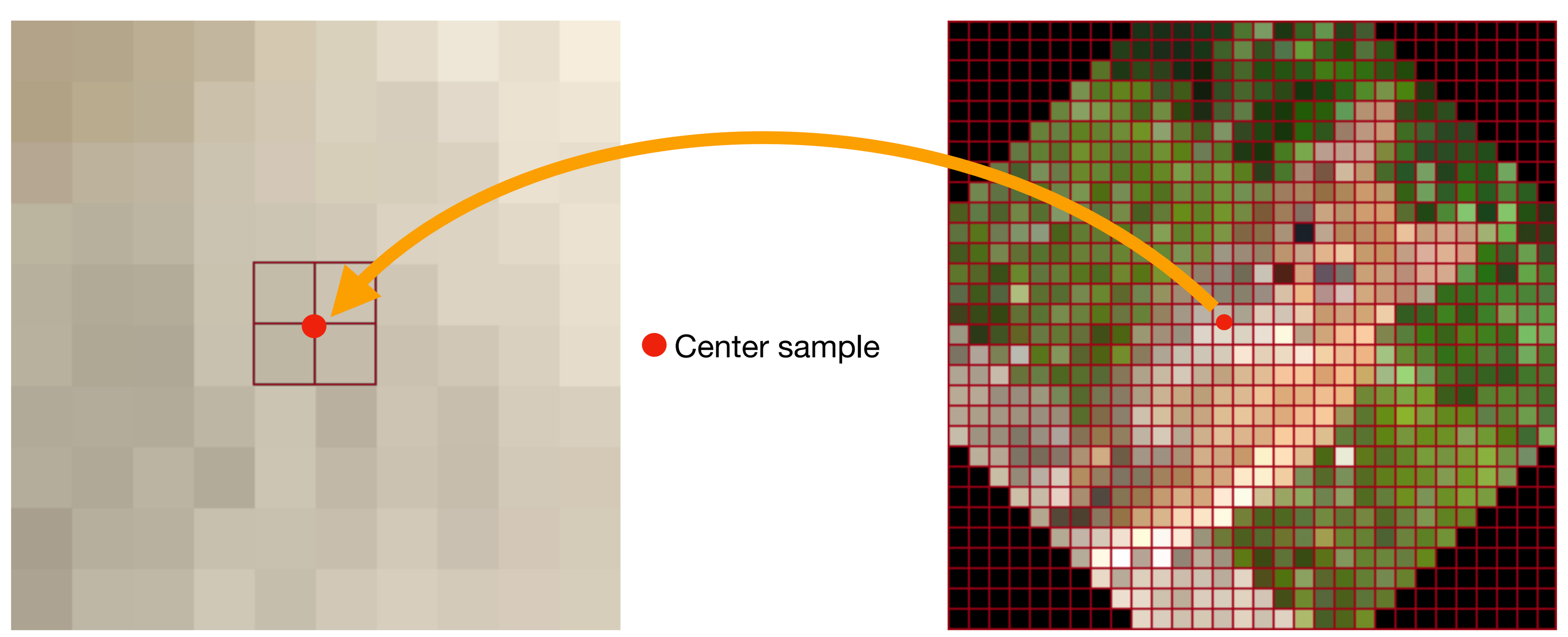
ICCV 2019 Seoul, Korea

Contribution

- A differentiable sampling method as an alternative to the prevalent *bilinear sampling,* which is the default option for the spatial transformer (STN), and all works based on STN.
- Sampling methods have significant impact on gradient-based optimization, but have largely been *overlooked*.
- Our sampling method improves bilinear STN on *multiple* experiments, and is also *compatible* with many other improvements as it only touches the sampler.

The pitfalls of sampling

- Bilinear sampling works fine if the output is at the same resolution as the input. However, when severe down sapling is applied, the gradients are no longer representative.
- The gradients will only back propagate through the **nearest 4 pixels** in the original input, and the gradients with respect to the warping parameters would only come from those, as shown below.



Input image (zoom in) Output image

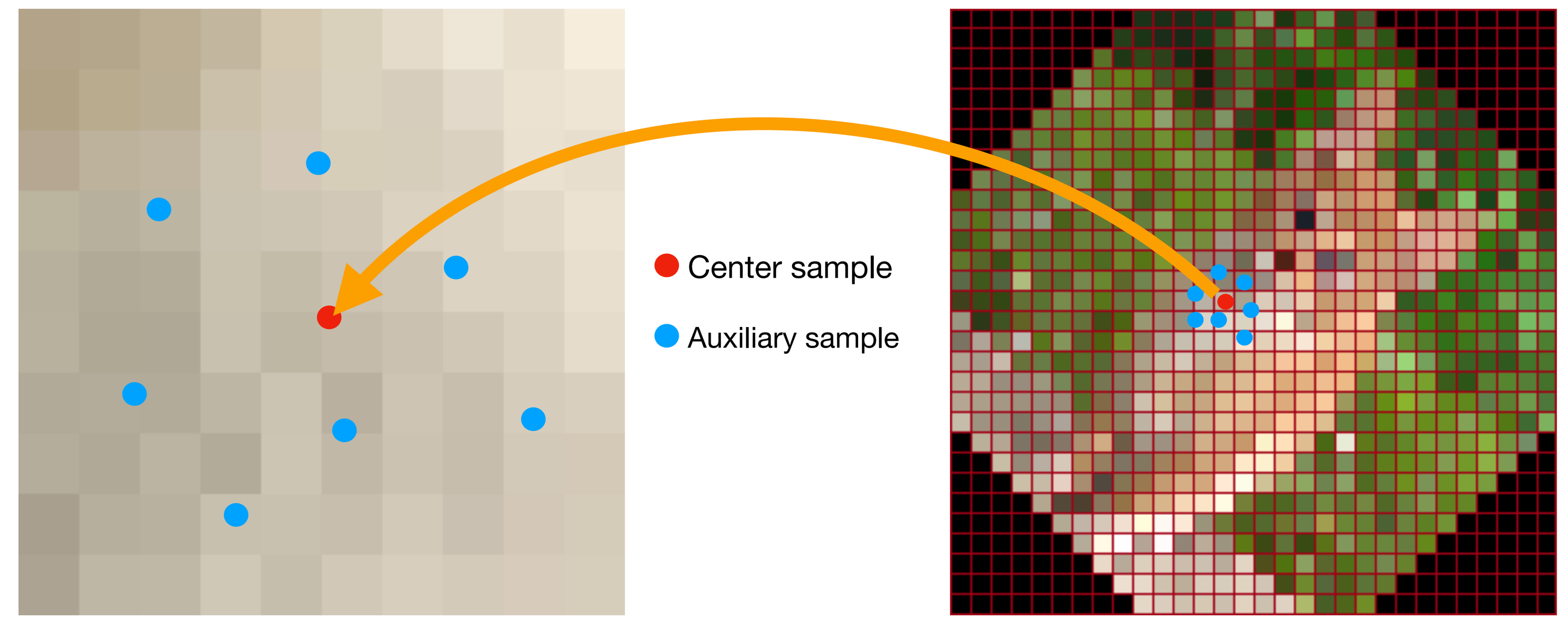
Try our sampling method — PyTorch:

- Code is available at: https://tinyurl.com/linearized
- The function signature is the same as the one in PyTorch:

```
def grid_sample(input,grid,mode='linearized',padding_mode='zeros')
```

- We provide a single self-contained python file which only depends on PyTorch, you can copy and paste into your own project!

Technical details



Input image (zoom in) Output image

1. Generate auxiliary samples around the center sample.
2. Transform all samples (center and auxiliary) to the input.
3. Retrieve the intensity of all samples with bilinear interpolation.
4. Linear approximation at the center point.

Transformed coordinate

$$\hat{\mathbf{I}}(\mathbf{x}) = \mathbf{I}(T_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \mathbf{A}_i\left(T_{\boldsymbol{\theta}}(\mathbf{x}) - T_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$$

Linear approximation at x_i Intensity at x_i Linear approximation coefficients

5. Solve for matrix A.

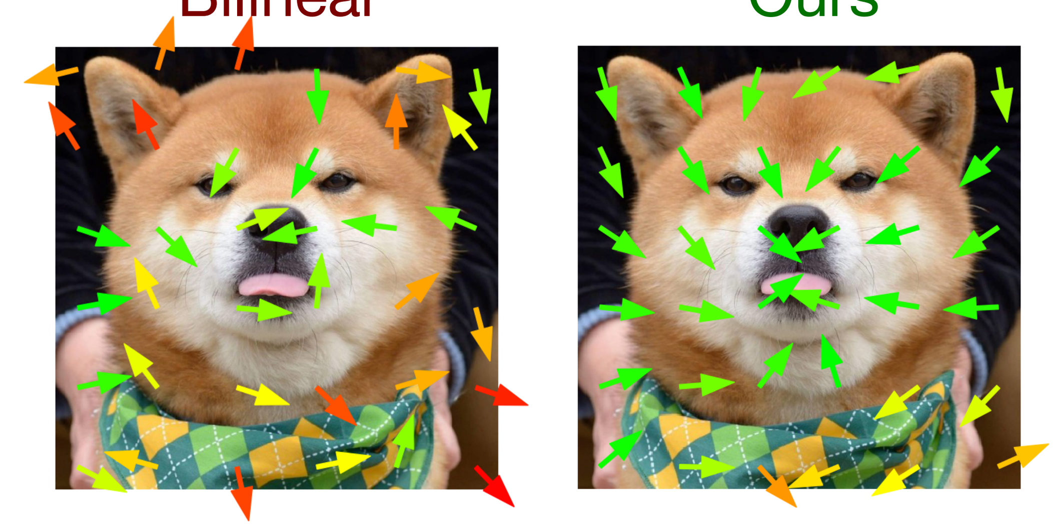
$$\mathbf{A}_i = \left(\mathbf{X}_i^\top \mathbf{X}_i + \epsilon \mathbf{E}\right)^{-1} \mathbf{X}_i^\top \mathbf{Y}_i$$

Intensities

Linear approximation coefficients Pseudo inverse Coordinates

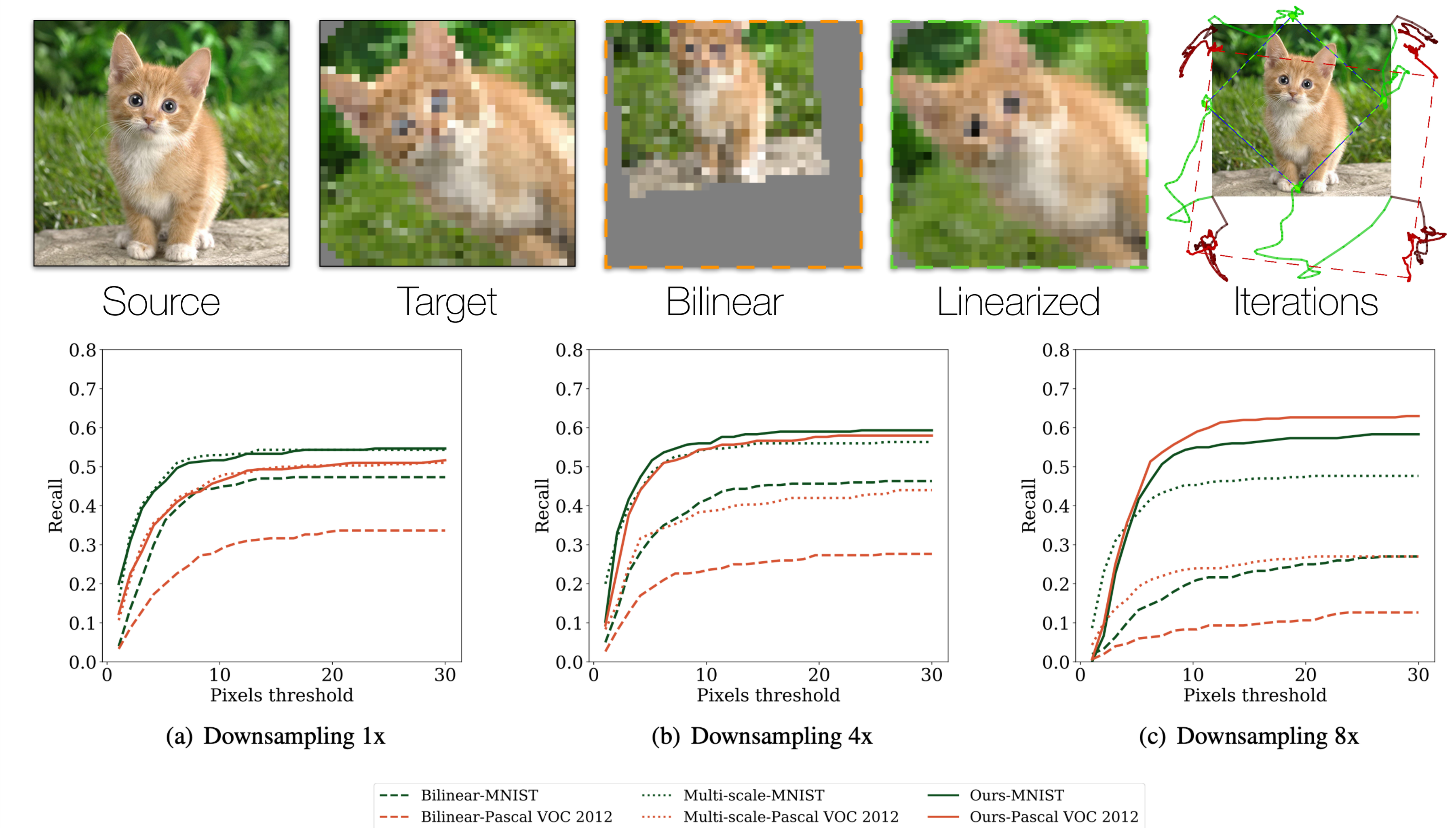
Qualitative Results

Gradient direction:
We translate the dog image to different locations and ask the optimizer to drag the translated back to the center using pixel-level L2 loss. The ideal gradient directions should always point towards the center.



Quantitative Results

Image alignment:



We try to find the the warping parameters of a transformation given source and target image. We directly optimize the warping parameters by minimizing the L2 loss between the target and current output. Our method improves recall by a large margin comparing to bilinear sampling.

Traffic sign classification:

We train a STN to classify traffic sign using different sampling methods. Our sampling method improves the performance at all downsampling rates.

Downsampling rate # classif. network param.	1x 966k	2x 246k	4x 61k	8x 19k
Baseline w/o STN	12.37	12.88	20.85	45.88
STN + Bilinear	6.29	6.50	7.95	15.31
STN + Multi-scale	6.83	6.70	8.30	15.00
STN + Ours	**6.08**	**6.48**	**7.13**	**10.89**
ICSTN + Bilinear	5.68	5.00	6.52	9.80
ICSTN + Multi-scale	5.40	5.95	6.06	10.19
ICSTN + Ours	**4.85**	**4.68**	**4.86**	**6.10**

We visualize the output of bilinear and our sampling method of a trained STN at 8X down sample rate. Our method tends to zoom in more, thus increases the accuracy of the classifier.