Linearized Multi-Sampling for Differentiable Image Transformation

Wei Jiang¹  Weiwei Sun¹  Andrea Tagliasacchi¹²  Eduard Trulls²  Kwang Moo Yi¹

¹Visual Computing Group, University of Victoria  ²Google Research
{jiangwei, weiweisun, kyi}@uvic.ca, {taglia, trulls}@google.com

Figure 1:  Image alignment example - We iteratively find the image transformation parameters that produce the downsampled target image when applied to the source image using different sampling strategies. We visualize the transformed source image at convergence when using bilinear sampling and the proposed linearized sampling. We further display the optimization path – iterations – for each method. The proposed linearized sampling provides improved gradients, leading to better convergence.

Abstract

We propose a novel image sampling method for differentiable image transformation in deep neural networks. The sampling schemes currently used in deep learning, such as Spatial Transformer Networks, rely on bilinear interpolation, which performs poorly under severe scale changes, and more importantly, results in poor gradient propagation. This is due to their strict reliance on direct neighbors. Instead, we propose to generate random auxiliary samples in the vicinity of each pixel in the sampled image, and create a linear approximation with their intensity values. We then use this approximation as a differentiable formula for the transformed image.

We demonstrate that our approach produces more representative gradients with a wider basin of convergence for image alignment, which leads to considerable performance improvements when training networks for classification tasks. This is not only true under large downscaling, but also when there are no scale changes. We compare our approach with multi-scale sampling and show that we outperform it. We then demonstrate that our improvements to the sampler are compatible with other tangential improvements to Spatial Transformer Networks and that it further improves their performance.

1. Introduction

The seminal work of [17] introduced Spatial Transformer Networks (STN), a differentiable component that allows for spatial manipulation of image data by deep networks. It has since become commonplace to include attention mechanisms in deep architectures in the form of image transformation operations. STNs have been applied to object detection [10], segmentation [15, 24], dense image captioning [19], local correspondence for image classification [1], local features [38, 28, 27], and as a tool for local hard attention [21]. Regardless of application and architecture, all of these methods rely on bilinear interpolation.

A major drawback of bilinear interpolation is that it is extremely local – it considers only the four closest pixel neighbors of the query. As this sampling does not account for the magnitude of the applied transformation, the performance of networks that rely on it degrades when scale changes are severe, as for example shown in Fig. 1. This shortcoming of the differentiable sampler was already hinted at in the original STN paper [17], but never fully investigated. Note that this is a problem in practice, as applications that leverage attention mechanisms often transform the image to a resolution much lower than the original [37, 28, 15]. Furthermore, STNs are typically used as pre-alignment networks for classification [17, 23]. The inability of the bilinear sampler in the STN to cope with large downscaling results in increased capacity requirements for the classification net-

---

¹ Code and models are available at https://github.com/vcg-uvic/linearized_multisampling_release.
work – this extra capacity will be needed to learn invariance to transformations the STN was not able to capture. We show that this is indeed the case by demonstrating that a better classification accuracy can be achieved with the same network by replacing the sampling operation with ours.

Several methods have been proposed to improve the stability and robustness of networks that employ bilinear sampling. For example, Lin et al. [23] introduced inverse compositional spatial transformer networks (ICSTN), which decompose transformations into smaller ones. Jia et al. [18] proposed to warp features rather than images. Shu et al. [32] proposed a hierarchical strategy incorporating flow-fields. However, all of these methods still rely on heavily localized bilinear interpolation when it comes to computing the values of individual pixels.

Instead, we take our inspiration from the well-known Lucas-Kanade (LK) optical flow algorithm [26, 2], and linearize the interpolation process by building a suitable first-order approximation by randomly sampling the neighborhood of the query. These auxiliary samples are created in the transformed image domain, so that their spatial locations take into account the transformation applied to the image. In other words, the resulting gradients are transformation-aware, and able to capture how the image changes according to the transformation.

However, this process can cause sample locations to collapse, as the transformation may warp them all to a single pixel – for example, when the transformer zooms into a small area of the image. To overcome this issue, we propose an effective solution that perturbs the auxiliary samples after warping. This two-stage strategy allows our approach to deal with any type of affine transformation.

Our experiments demonstrate that this allows for a wider basin of convergence for image alignment, which helps the underlying optimization task. Most importantly, any network with embedded attention mechanisms, i.e., anywhere differentiable image transformation is required, should perform better when substituting standard bilinear interpolation with our approach. We demonstrate this by adding our sampling method to ICSTN [23], and show that we are able to not only reduce its error rate by 14.6% (relative) even without down sampling, but also that we effectively eliminate the harming effect of 4x downsampling on the performance of the classifier (error at 1x is 4.85% and at 4x is 4.86%). This means that we can use a classification network that is 11 times smaller than the original, without any performance loss.

2. Related work

There is a vast amount of work in the literature on estimating spatial transformations, with key applications to image registration problems, going back to the pioneering work of Lucas and Kanade [26]. Here we review previous efforts on image alignment and sampling techniques, particularly with regards to their use in deep learning.

**Linearization.** The Lucas & Kanade (LK) algorithm [26] predicts the transformation parameters using linear regression. It mathematically linearizes the relationship between pixel intensities and pixel locations by taking the first-order Taylor approximation. In this manner, the sampling process can be enhanced by enforcing linear relationships during the individual pixel sampling. This approach has proved greatly successful in many applications such as optical flow estimation [5]. Linearization is also widely utilized to improve the consistency of the pixel values in image filtering [16].

**Multisampling.** Multisampling is a common approach to enhance the reliability of sampling strategies. For example, one can sample multiple nearby pixels before feeding them to a classifier, jointly smoothing the scores from a pixel and its neighbors [40, 9]. Non-local means [5] computation can be sped up by Markov Chain Monte Carlo (MCMC) sampling [8]. Sampling can also be employed to compute finite differences at large stencils to produce gradient estimates that are less sensitive to noise and discontinuities [34].

In the context of deep learning. Early efforts in deep learning for computer vision were limited by their inability to manipulate input data, a crucial requirement in achieving spatial invariance. Jaderberg et al. proposed to address this shortcoming with Spatial Transformer Networks [17], which introduced the concept of a differentiable transformation to actively manipulate the input image or the feature maps. This effectively enabled learning hard attention mechanisms in and end-to-end fashion. To achieve this, they introduced a differentiable sampling operation, making it possible to propagate the loss in subsequent tasks with respect to the parameters of the predicted transformation.

STNs are widely used in applications that operate on image patches [36, 32, 39, 4]. Modern methods, such as Wang et al. [35], improve the patch sampling through an in-network transformation. Qi et al. introduced PointNet [30], a deep network for 3D point cloud segmentation which relies on STN to learn to transform the data into a canonical form before giving it to the network. Nevertheless, the paper reports only marginal performance improvements with the use of the STN, which demonstrates there is potential for further research in this area.

Several methods use bilinear sampling without explicitly learning a transformation. LIFT [37] and LF-Net [28] relied on STN to warp image patches and learn local features in an end-to-end manner, with the transformation parameters given by specially-tailored networks (e.g. keypoint detection). AffNet [27] applied a similar strategy to learn affine-covariant regions. Polar Transformer Networks [12] were used by [11] to build scale-invariant descriptors by transforming the input patch into log-polar space.
Bilinear sampling has also been used in the context of image upsampling, depth estimation, and segmentation [14, 7, 15, 25, 31]. For example, in Mask R-CNN [15], the region of interest alignment layer mainly relies on bilinear sampling/interpolation. These methods could also potentially benefit from enhanced sampling.

Current limitations of Spatial Transformers. Despite its popularity, in this paper we show how the bilinear sampler utilized in the STN framework is inherently unreliable and lacks robustness. Several variants have been recently proposed in order to cope with these shortcomings.

Jia et al. [18] showed how to exploit patterns found in the feature map deformations throughout the network layers that are insensitive to bilinear interpolation, thus boosting the accuracy of STN. Chang et al. [6] trained a deep network with a Lucas-Kanade layer to perform coarse-to-fine image alignment. The inverse compositional network proposed in [23] passes the transformation parameters instead of the transformed image in the forward pass in order to mitigate errors in the bilinear sampling. Recently, a hierarchical strategy was proposed to deal with transformations at multiple levels [32], by incorporating a U-Net [31], providing richer transformation cues. However, all of these still rely on bilinear sampling to transform the image.

3. Differentiating through interpolation

We now briefly review bilinear interpolation [17]; see Fig. 2. The main reason that a form of interpolation – typically bilinear interpolation – is used to implement differentiable image transformations is that transformations require indexing operations, which are not differentiable by themselves. More specifically, denoting the image coordinates as \( \mathbf{x} \in \mathbb{R}^2 \), the image intensity at this coordinate as \( \mathbf{I}(\mathbf{x}) \in \mathbb{R}^C \), where \( C \) is the number of channels in the image, and the coordinate transformation with parameters \( \theta \) as \( T_\theta(\mathbf{x}) \), then, the transformed image \( \mathbf{I} \) evaluated at \( \tilde{\mathbf{x}} \) is

\[
\hat{\mathbf{I}}(\tilde{\mathbf{x}}) = \sum_x \mathbf{I}(T_\theta(\mathbf{x})) \odot K(\tilde{\mathbf{x}}, T_\theta(\mathbf{x})),
\]

where \( K(\cdot, \cdot) \) is the kernel which defines the influence of each pixel. Note here that the image indexing operation \( \mathbf{I}(T_\theta(\mathbf{x})) \) is non-differentiable, as it is a selection operation, and the way gradients propagate through the network depends on the kernel. In theory, this kernel could leverage the entire image, thus making the gradient for all pixel values affect the optimization. However, this would require back-propagating through every pixel in the original image for every pixel in the transformed image, which is prohibitively expensive. In the case of bilinear interpolation, the kernel is set so that \( K(\mathbf{x}, \mathbf{y}) = 0 \) when \( \mathbf{x} \) and \( \mathbf{y} \) are not direct neighbors. Therefore, the gradient only flows through what we will refer to as sub-pixel gradients, i.e., the intensity difference between neighboring pixels in the original image.

This can be quite harmful under significant downsampling. The sub-pixel gradients will not correspond to the large-scale changes that occur when the transformation parameter changes, as these cannot be captured by the immediate neighbourhood of a point. In other words, the intensity values with non-zero gradients from kernel \( K \) need to be chosen in a way that is invariant to these deformations, and carefully so that it reflects how the image would actually change under deformations. We will now introduce a novel strategy to overcome this problem.

4. Linearized multi-sampling

Fig. 3 illustrates our method, step by step. Given a pixel location, we apply Gaussian noise to generate \( K \) nearby auxiliary sample locations. We then perform bilinear sampling at these locations. After sampling, we use the intensities of these points and their coordinates to perform a linear approximation at the sample location. Finally, these linear approximations are used as the differentiable representation of each pixel in the transformed image. This provides a larger context to the local transformation, and increases robustness under downsampling.

Formally, given the parameterization of the transformation \( \theta \), a desired sample point as \( \mathbf{x}_i \), where \( i \) is the index for this sample, we write the linear approximation \( \hat{\mathbf{I}}(\mathbf{x}) \) as

\[
\hat{\mathbf{I}}(\mathbf{x}) = \mathbf{I}(T_\theta(\mathbf{x}_i)) + \mathbf{A}_i (T_\theta(\mathbf{x}) - T_\theta(\mathbf{x}_i)),
\]

where \( \mathbf{A}_i \) is a matrix that defines the linearization that we seek to find. For the sake of explanation, let us assume for the moment that we have \( \mathbf{A}_i \). Then, notice here that we...
are linearizing at the transformed coordinate \( T_{\theta}(x_i) \), thus
treating everything except for \( T_{\theta}(x) \) as a constant. Therefore, \( A_i \) in Eq. (2)
corresponds to the gradient of \( \hat{I}(x) \) with respect to \( x \).

To obtain \( A_i \) we first sample multiple points near the
desired sample points \( x_i \) and find a least-square fit for the
sample results. Specifically, we take \( K \) samples

\[
x^k_i \sim \mathcal{N}(x_i, \sigma), \quad \forall k \in \{1, 2, \ldots, K\},
\]

where \( \mathcal{N}(\mu, \sigma) \) denotes a Gaussian distribution centered at
\( \mu \) with standard deviation \( \sigma \), and \( x^0_i = x_i \). In our experiments
we set \( \sigma \) to match the pixel width and height of the
sample output. Note that by using Gaussian noise, we are
effectively assuming a Gaussian point-spread function for
each pixel when sampling.

We then obtain \( A_i \) by least-squares fitting. If we simplify
the notation for \( I(T_{\theta}(x^k_i)) \) as \( I^k_i \), and for \( T_{\theta}(x^0_i) \) as \( \bar{x}^k_i \),
where \( \bar{x}^k_i = [\bar{u}^k_i, \bar{v}^k_i] \), we form two data matrices \( Y_i \) and
\( X_i \), where

\[
Y_i = X_i A_i,
\]

\[
Y_i = \begin{bmatrix}
I^0_i - I^0_i \\
I^2_i - I^0_i \\
I^K_i - I^0_i
\end{bmatrix},
\]

\[
X_i = \begin{bmatrix}
\bar{u}^1_i - \bar{u}^0_i \\
\bar{u}^2_i - \bar{u}^0_i \\
\bar{u}^K_i - \bar{u}^0_i
\end{bmatrix},
\]

and then solve for \( A \) in a least square sense with Tikhonov
regularization for numerical stability

\[
A_i = (X_i^T X_i + \epsilon I)^{-1} X_i Y_i,
\]

where \( I \) is the \( 3 \times 3 \) identity matrix, and \( \epsilon \) a small scalar.

Multi-scale sampling  An alternative approach would be to
use multi-scale alignment with auxiliary samples distributed over pre-defined grids at varying levels of coarseness. This can scale up as much as desired – potentially up
to using the entire image as a neighborhood for each pixel –
but the increase in computational cost is linear with respect to
the number of auxiliary samples. Random selection allows us to capture both local and contextual structure in an
efficient manner.

5. Sample collapse prevention

While Eq. (7) is straightforward to compute, we need to
take special care that the random samples do not collapse into
a single sample. This can happen when the transformation is zooming into a specific region; see Fig. 4 (middle).
If this happens, the data matrix \( X_i \) generated from coordinate differences of transformed auxiliary samples will be
comprised of very small numbers, and the data matrix \( Y_i \),
generated from differences in intensities, may also become
zero. This leads to exploding gradients. To avoid them,
we perturb the auxiliary samples after the transformation; see Fig. 4 (right).
Denoting the modified coordinates for the \( k \)-th auxiliary sample for pixel \( i \) with \( \bar{u}^{k}_{i} \) and \( \bar{v}^{k}_{i} \), for
\( k \in \{1, 2, \ldots, K\} \), we then apply

\[
\bar{u}^k_i \leftarrow \bar{u}^k_i + \mathcal{N}(0, \sigma_u),
\]

\[
\bar{v}^k_i \leftarrow \bar{v}^k_i + \mathcal{N}(0, \sigma_v),
\]
Figure 5: Dataset image examples – (top-left) MNIST. (top-right) GTSRB. (bottom) Pascal VOC 2012.

where $\delta_u$ and $\delta_v$ correspond to the single pixel width and height in the image that we sample from – the magnitude of the transform in either dimension.

6. Results

To demonstrate the effectiveness of our method, we first present quantitative results on the performance of STN and ICSTN with different sampling methods. We show that the enhanced gradients from linearization-based sampling lead to significant improvements in the performance of deep networks, most prominently when downsampling is involved. We then visually inspect the gradients produced by bilinear sampling and our linearized multi-sampling approach, to demonstrate that the gradients produced by our method are more robust and lead to better convergence. We further show that this can be confirmed quantitatively by a simple image alignment task, performed with different sampling methods. Finally, we study the effect of the number of auxiliary samples and the impact of the magnitude of the sampling noise.

Datasets. Our experiments feature three datasets: MNIST [22], GTSRB [33], and Pascal VOC 2012 [13]; see Fig. 5. With MNIST, we show that our approach outperforms bilinear interpolation in one of the classical classification datasets. With GTSRB, we demonstrate that our method also leads to better performance in the more challenging case of classifying traffic sign images. Finally, we use Pascal VOC 2012 to showcase that the advantages of our method are even more prominent on natural images with richer textures.

Baselines. In addition to standard bilinear sampling, we also evaluate a multi-scale baseline where the image is sampled at multiple scales and then aggregated. This is similar to how multi-scale sampling with mipmaps is performed. To create a multi-scale representation, we create three scale levels, using Gaussian kernels with standard deviations $\{1, 5, 10\}$. We apply these sampling methods to STN [17] and ICSTN [23] for classification.

6.1. Implementation details

We implement our method with PyTorch [29], and use its default bilinear sampling to fetch the intensity corresponding to each of our random samples. Note that this sampling process is not differentiated through, and we use these intensities only to compute the linearization. To ensure this, our implementation explicitly stops the gradient from backpropagating for all variables except for $T_\theta(x)$.

To prevent manifestation from out-of-bound samples, we simply mask the sample points that fall outside the original image. Specifically, for each pixel $i$, if $T_\theta(x_k^i)$ is out of bounds, we exclude it from Eq. (5) and Eq. (6). This can be easily achieved by multiplying the corresponding entries with zero. When all pixels are out of bounds, $A_i$ becomes a zero matrix, thus providing a zero gradient.

Throughout our experiments, we use eight auxiliary samples per pixel ($K = 8$), a choice we will substantiate in Section 6.5.

6.2. Classification

To demonstrate that sampling plays a critical role in networks that contain image transformations, we train a network for image classification with STN/ICSTN prepended to it on the GTSRB dataset. We emulate a standard setup used to leverage attention mechanisms by using STN/ICSTN to produce a transformation of a smaller resolution than the input image, which is then given to the classifier – the task of STN/ICSTN is thus to focus on a single region that helps to classify traffic signs. We evaluate the accuracy of the classifier under different downsampling
Table 1: Test error of models trained with various sampling methods, using STN and ICSTN. Best results are marked in bold. Our method gives best performance even when there is no downsampling, and the gap widens at higher downsampling rates.

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

Network architecture and training setup. To train the network we randomly split the training set of GTSRB into two sets, 35309 images for training and 3900 for validation. For testing we use the provided test set that holds 12630 images. We crop and resize all images to 50×50.

For the STN/ICSTN module, denoting convolution layers with c channels as C(c), ReLU activations as R, and max-pooling layers as P, our network is: C(4)R(C(8)R(C(16)R(C(32)R(C(1024)). All convolutional layers use 7×7 kernels. We apply max-pooling over each channel of the last feature map to produce one feature vector of size 1024. We then apply a fully connected layer with 48 neurons and ReLU activations, followed by another fully connected layer that maps to transformation parameters – translation, scale, and rotation in our experiments.

For the classification network, we use a simple network with a single hidden layer with 128 neurons and ReLU activations. We choose a simple architecture on purpose in order to prevent the network relying on the increased capacity of a large classification network and learn spatial invariance and effectively ignoring the STN.

We use ADAM [20] as the optimizer, and choose 10^{-5} and 10^{-3} as the learning rates for the STN/ICSTN modules and the classifier respectively. We train the models from scratch with a batch size of 64, and set the maximum number of iterations to 300k. We use early stopping if the model shows no improvement on the validation split within the last 80k iterations.

Results. In Table 1 we show the test accuracy of our approach compared to bilinear sampling and multi-scale sampling, with both STN and ICSTN. Unsurprisingly, the performance of both degrades as downsampling becomes more severe, but significantly less so with our method. Notably, the network trained with our sampling method performs the best for all downsampling rates – even without any downsampling. Moreover, with ICSTN and our method, there is no noticeable performance degradation even with 4x downsampling, allowing the classification network to be 11 times smaller compared to when no downsampling is used.

Finally, we show the average transformed image for a subset of the classes in the test set in Fig. 6. Note how the outputs of our network are zoomed-in with respect to the original input image, and more so than the results with bilinear sampling. This shows that with our linearized sampling strategy the network learns to focus on important regions more effectively.

6.3. Gradient analysis

To understand where such a drastic difference comes from, we show how the gradients produced by our approach differ from those of bilinear sampling. We compare how the gradients flow when we artificially crop the central region of an image, move it to a different location, and ask the STN to move it back to the original position. We use natural images from the Pascal VOC dataset for this purpose. Specifically, we crop the central two-thirds of the image, downsample them at different rates with bilinear interpolation, and use this image as our guidance for the STN with an \ell_2 loss.

In Fig. 7 we visualize the negative gradient flows, i.e. the direction of steepest descent, extracted on a coarse grid on top of the image for which the gradients were computed.

Since the target area is the central region, the gradients should all point towards the center. As shown, our method provides a wider basin of convergence, having the gradients consistently point towards the center, i.e., the ground truth location, from points further away than with bilinear sampling. Note that while this has a drastic effect for the case.
of downsampling by 8x, it holds true even when there is no downsampling (Fig. 8, left). This is particularly distinctive around the bottom of the image, which has richer structures than the rest, which is blurry and thus rather smooth. This causes bilinear sampling to provide poor gradients, whereas our method is not susceptible to this.

6.4. Optimizing for image alignment

Next, we demonstrate that this improvement in the gradients produced by our approach leads to better outcomes when aligning images, as illustrated earlier in Fig. 1. In order to isolate the effect of the sampler, we exclude the classification network and directly optimize the parameters of a STN for one image at a time, with a single synthetic perturbation. We then evaluate the quality of the alignment after convergence. We repeat this experiment 80 times, on randomly selected images from the different datasets, and with random transformations.

For the perturbations, we apply random in-plane rotations, scale changes (represented in log-space, i.e. scale of 0.5 is represented as −1) in the horizontal and vertical directions (independently), and translations. We sample Gaussian noise with standard deviation of 1 and apply it as follows. With the image coordinates normalized to [−1, 1], we use a standard deviation of π/4 for rotations, √2 for scale changes in horizontal and vertical directions separately, and 0.2 for translations.

We summarize our image alignment experiments in Fig. 7. As shown, our results are significantly better than those of bilinear sampling, even when there is no downsampling. We also outperform the multi-scale if there is any downsampling at all. More importantly, the gap between methods is specially wide on natural images, where the image characteristics are much more complex than for the other two datasets. Finally, the effects of downsampling become quite severe at 8x, effectively breaking the bilinear sampler and the multi-scale approach in all datasets, and for PASCAL VOC in particular.
6.5. Ablation tests

**Auxiliary sample noise.** We also examine the effect of the magnitude of the noise we use to place the auxiliary samples in Eq. (3). Fig. 9 (b-c) shows the sampling result under various $\sigma$. Since the Gaussian noise emulates a point spread function, with a larger $\sigma$ we obtain a blurrier image via Eq. (1). While the outcome is blurry, this effectively allows our method to consider a wider support region in the computation of the gradients, which will therefore be smoother. As shown in Fig. 9 (a), the wider spatial extent covered by the random samples translates into better image alignment results in the Pascal VOC dataset.

**Number of auxiliary samples.** In Fig. 10 (a) we evaluate the effect of number of random samples taken for each pixel with the Pascal VOC dataset under 4x downsampling. As expected, increasing the number of auxiliary samples leads to better alignment results. When only four auxiliary samples are used, the linearization is quite ill-conditioned and thus accuracy of the method drops.

**Sample collapse avoidance.** In Fig. 10 (b) we demonstrate the importance of the sample collapse prevention scheme described in Section 5. When our technique for sample collapse prevention is removed, the performance in image alignment drops due to the numerical instability caused by all auxiliary samples falling in the same neighborhood. By contrast, our approach prevents this problem.

7. Conclusions and future work

We have proposed a linearization method based on multi-sampling to improve the poor gradients provided by bilinear sampling – a de-facto standard for differentiable image manipulation, typically within Spatial Transformers – for example, to implement hard attention. We have demonstrated empirically that our method can provide improved gradients, leading to enhanced performance on downstream tasks. Our approach simply swaps the sampler used by networks that rely on Spatial Transformers, and is thus compatible with any improvements that build on top of them.

Note that it is possible to replace the sampler with bilinear interpolation at inference time to preserve efficiency. As of now, this would result in a minor change in the sampling outcome and require fine tuning of the classifier. As immediate future work, we are currently investigating a bias-free formulation for Eq. (6), which would allow in-place replacement without further fine tuning.

Acknowledgements

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant “Deep Visual Geometry Machines” (RGPIN-2018-03788, DGECR-2018-00426), Google, and by systems supplied by Compute Canada.
References


[23] Chen-Hsuan Lin and Simon Lucey. Inverse Compositional Spatial Transformer Networks. CVPR, 2017. 1, 2, 3, 5, 11


[34] David Joseph Tan, Thomas Cashman, Jonathan Taylor, Andrew Fitzgibbon, Daniel Tarlow, Sameh Khamis, Shahram Izadi, and Jamie Shotton. Fits Like a Glove: Rapid and Reliable Hand Shape Personalization. In CVPR, 2016. 2


[36] Wanglong Wu, Meina Kan, Xin Liu, Yi Yang, Shiguang Shan, and Xilin Chen. Recursive Spatial Transformer...


