Linearized Multi-Sampling for Differentiable Image Transformation

Wei Jiang[1] Weiwei Sun[1] Andrea Tagliasacchi[1,2] Eduard Trulls[2,3] Kwang Moo Yi[1]

[1]Visual Computing Group, University of Victoria [2]Google
[3]Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne (EPFL)
{jiangwei, weiweisun, kyi}@uvic.ca, atagliasacchi@google.com, eduard.trulls@epfl.ch

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 using their intensity values. We then use this approximation as a differentiable formula for the transformed image. However, we observe that these auxiliary samples may collapse to a single pixel under severe image transformations, and propose to address it by adding constraints to the distance between the center pixel and the auxiliary samples. 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 image registration and classification tasks, particularly under large downsampling.

1. Introduction

The seminal work of [12] introduced Spatial Transformer Networks (STN), a differentiable component that allows for spatial manipulation of image data by deep networks. It has since become widespread to include attention mechanisms in the form of image transformation operations in deep architectures. STNs have been applied to object detection [8], segmentation [10, 19], dense image captioning [14], local correspondence for image classification [1], learning local features [33, 23, 22], and as a tool for local hard attention [16]. 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 is computed by considering 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 using it degrades when scale changes are severe, as for example shown in Fig. 1. Note that in most applications the resolution of the transformed image is much smaller than the original—*e.g.* extracting a 32×32 patch from a 256×256 image—so that in practice we are downsampling by a significant factor, even while zooming in to the region of interest. As noted in the original paper [12], this drawback is exacerbated in the optimization

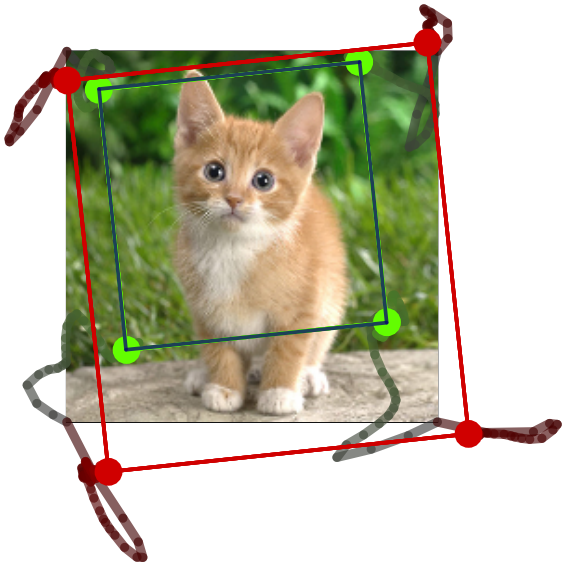


Figure 1: An example of image alignment via gradient descent with different sampling strategies, where the ground truth region is shown in blue (blue) and the initial transformation is identity. When our linearized multi-sampling is used (green), the optimization finds the global optimum without any problems. Conversely, when bilinear sampling (red) is used, the iterative alignment gets stuck in a local minima as the gradients are too noisy and local.

of deep networks, as local gradients are also computed by first-order approximations, with the same bilinear interpolation procedure.

Several methods have been proposed to improve the stability and robustness of networks that employ bilinear sampling. For example, Lin *et al*. [18] introduced a recurrent architecture to compensate for smaller perturbations, Jia *et al*. [13] proposed to warp features rather than images, while Shu *et al*. [27] 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 [21, 2], and linearize the interpolation process by building a suitable first-order approximation via 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 capable to capture how the image changes according to the transformation.

However, we observe that this process alone can cause sample locations to collapse, as the transformation can warp them all to a single pixel, for example, when zooming in to a small area of the image. We therefore propose a technique to constrain the location of the auxiliary samples with respect to the central pixel after the transformation, pushing them away from the center. Although this process is non-differentiable, our linearization formulation allows the gradients to back-propagate through it. Combining these two techniques, our approach is able to deal with *any* type of 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 networks with embedded attention mechanisms, *i.e*. anywhere a STN is used, should perform better when substituting simple bilinear interpolation with our approach.

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 [21]. 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) [21] algorithm 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 idea has proved greatly successful in many applications such as optical flow estimation [3]. The idea of linearization is widely utilized to improve the consistency of the pixel values [11].

Sampling techniques. Multisampling is the *de facto* standard to enhance the reliability of sampling strategies, by incorporating the corresponding neighboring pixels. Some methods such as [35, 7] have demonstrated improvements in pixel classification for hyperspectral images by sampling multiple nearby pixels before feeding it to the classifier, jointly smoothing the scores from the center pixel and its neighbors. Alternatively, Tan *et al*. [29] introduced a gradient-based technique to fit hand models to depth observations, taking finite differences with a large stencil in order to jump over image discontinuities due to occlusions.

In the context of deep learning. While deep learning has revolutionized computer vision, early efforts were limited in their inability to manipulate the input data, which is crucial for spatial invariance. Jaderberg *et al*. proposed to address this shortcoming with Spatial Transformer Networks [12], 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 bilinear sampling operation that can be differentiated through, making it possible to propagate the loss in subsequent tasks, such as image alignment, onto the parameters of the predicted transformation.

Consequently, STN is widely used in many challenging tasks [31, 27, 34, 4], and has proved very successful wherever sampling patches is essential. Modern methods, such as Wang *et al*. [30] improve the patch sampling through an in-network transformation. Qi *et al*. introduced Pointnet [25], 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.

By contrast, several methods rely on the bilinear sampler without explicitly learning a transformation. LIFT [32] and LF-Net [23] use STN to warp image patches and learn local features in an end-to-end manner, with the transformation parameters given by specially tailored networks for *e.g*. keypoint detection. Affnet [22] used similar techniques to learn affine-covariant regions. These methods are currently the state of the art in feature-based wide-baseline matching.

Bilinear sampling has also been used in a different context for image upsampling, particularly for image segmen-

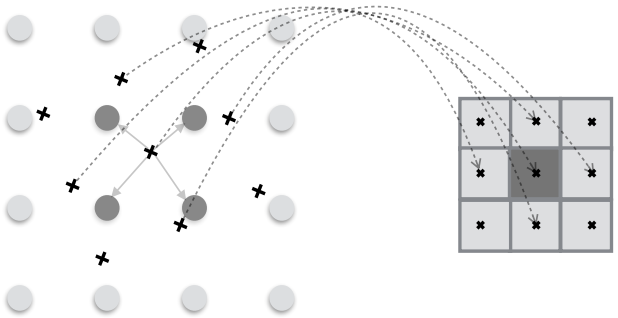


Figure 2: Bilinear sampling. To sample a 3×3 image on the right from the image on the left, each pixel is queried, to find the exact corresponding point in the left image, according to a given transformation. As illustrated by the central pixel, the intensities are computed through the nearest neighbors by bilinear interpolation. Note that, even when the pixels on the right fall into further away regions, the interpolation will *always* be performed from their direct neighbors.

tation [6, 10, 20, 26]. This is typically achieved by deconvolution layers that can learn a non-linear upsampling, but remain confined to a pre-determined neighborhood.

Current limitations of Spatial Transformers. Despite its popularity, the bilinear sampling utilized in the STN framework is inherently unreliable and not robust. In view of this, several variants have been recently proposed in order to cope with its shortcomings.

Jia *et al*. [13] proposed to take advantage of patterns in the feature map deformations throughout the network layers that are insensitive to the bilinear interpolation, thus boosting the accuracy of STN. Chang *et al*. [5] train a deep network with a Lucas-Kanade layer to perform coarse-to-fine image alignment. The inverse compositional network proposed in [18] 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 [27], by incorporating a U-Net [26] for richer transformation cues. However, all of these strategies still rely on bilinear sampling to transform the image.

3. Differentiating through interpolation

For the sake of completeness, we first briefly review how bilinear interpolation [12] is used to provide differentiable sampling operations for deep learning. Fig. 2 illustrates how bilinear interpolation is used to sample a new image according to a transformation using a sampling grid.

The main reason bilinear interpolation is used for implementing image transformations is that the image transformation operations require indexing operations in practice, which is non-differentiable. More specifically, if we denote the image coordinates as $\mathbf{x} \in \mathbb{R}^2$, and the image intensity at this coordinate as $\mathbf{I}(\mathbf{x}) \in \mathbb{R}^C$, where C is the number of channels in this image, and the coordinate transformation with parameters $\boldsymbol{\theta}$ as $\tilde{\mathbf{x}} = T_{\boldsymbol{\theta}}(\mathbf{x})$, we write the image transformation operation as

$$\tilde{\mathbf{I}}(\tilde{\mathbf{x}}) = \sum_{\mathbf{x}} \mathbf{I}(T_{\boldsymbol{\theta}}(\mathbf{x})) * K(\tilde{\mathbf{x}}, T_{\boldsymbol{\theta}}(\mathbf{x})), \qquad (1)$$

where $K(\cdot, \cdot)$ is the kernel which defines the influence of each pixel to the final transform result. Note here that the image indexing operation $\mathbf{I}(T_{\boldsymbol{\theta}}(\mathbf{x}))$ is non differentiable, as it is a selection operation, and the way gradients flow through is purely based on the kernel.

In theory, one can use a kernel that involves the entire image, thus making the gradient all pixel values affect the optimization. However, in practice, this is too computationally intensive to implement, as one would require for each pixel to collect information from the entire image. In the case of bilinear interpolation the sampling kernel is set so that $K(\mathbf{x}, \mathbf{y}) = 0$ when $\mathbf{x}$ and $\mathbf{y}$ are not nearest neighbors. Therefore, in this case, the gradient only flows through what we will refer to as *sub-pixel gradients*, the intensity difference between neighboring pixels on the *original* image.

This can be quite harmful. For example, 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 the kernel K needs to be chosen in a way that is invariant to these deformations, and carefully so that it reflects how the image actually would change in case of deformations. In the following section we present our strategy to solve this through linearization and multi-sampling.

4. Linearized multi-sampling

Fig. 3 illustrates our method, step by step. Given a pixel location, we apply random Gaussian noise to generate K nearby auxiliary sample locations. We then perform bilinear sampling on these locations, including the desired sample location, and the transformation parameter. After sampling, we use the intensities of these points and their coordinates to perform a linear approximation at the sample location. Finally, this linear approximation is used as the mathematical representation for each pixel of the transformed image. This provides a larger context for the local transformation and increases robustness under downsampling.

Formally, given the parameterization of the transformation $\boldsymbol{\theta}$, a desired sample point as $\mathbf{x}_i$, where i is the index for this sample, and the image $\mathbf{I}(\cdot)$, we write the linear approx-

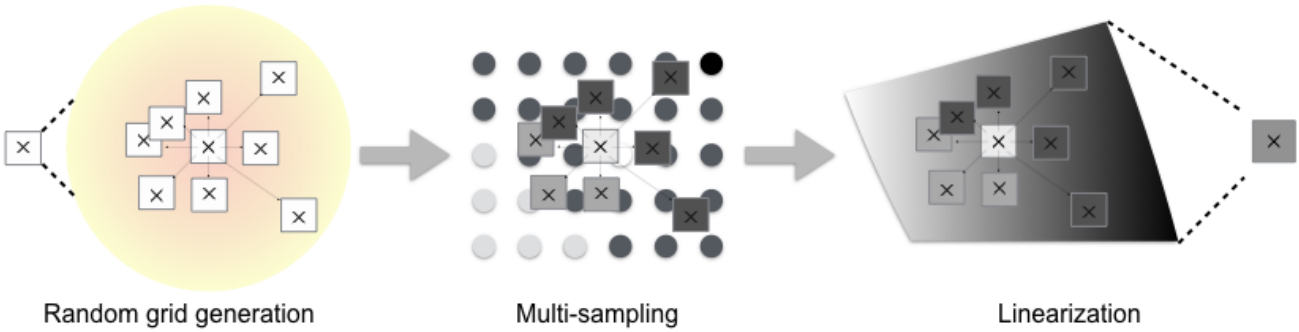


Figure 3: Our linearized multi-sampling. For each pixel that we query, we generate a set of random auxiliary samples, whose intensities we extract through bilinear sampling. We then use these intensities to form a linear approximation, which we use as the differentiable representation for the intensity of the queried pixel.

imation $\hat{\mathbf{I}}(\mathbf{x})$ at this location as

$$\hat{\mathbf{I}}(\mathbf{x}) = \mathbf{A}_i \left(T_{\boldsymbol{\theta}}(\mathbf{x}) - T_{\boldsymbol{\theta}}(\mathbf{x}_i) \right) + \mathbf{I}\left(T_{\boldsymbol{\theta}}(\mathbf{x}_i) \right), \quad (2)$$

where $\mathbf{A}_i$ is a matrix that defines the linearization that we would like 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 $T_{\boldsymbol{\theta}}(\mathbf{x}_i)$, thus treating everything except for $T_{\boldsymbol{\theta}}(\mathbf{x})$ as a constant. To make sure this is the case, in our implementation we explicitly stop the gradient from back-propagating for all variables except for $T_{\boldsymbol{\theta}}(\mathbf{x})$. Therefore, $\mathbf{A}_i$ in Eq. (2) corresponds to the gradient of $\hat{\mathbf{I}}(\mathbf{x})$ with respect to $\mathbf{x}$.

To obtain $\mathbf{A}_i$ we first sample multiple points nearby the desired sample points $\mathbf{x}_i$ and find a least-square fit for the sample results. Specifically, we take $K-1$ samples

$$\mathbf{x}_i^k \sim \mathcal{N}(\mathbf{x}_i, \boldsymbol{\sigma}), \quad \forall k \in \{1, 2, ..., K-1\}, \quad (3)$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ denotes a Gaussian distribution centered at $\boldsymbol{\mu}$ with standard deviation $\boldsymbol{\sigma}$, and $\mathbf{x}_i^0 = \mathbf{x}_i$. In our experiments we set $\boldsymbol{\sigma}$ 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 $\mathbf{A}_i$ through least-squares fitting. If we simplify the notation for $\tilde{\mathbf{I}}\left(T_{\boldsymbol{\theta}}\left(\mathbf{x}_i^k\right)\right)$ as $\tilde{\mathbf{I}}_i^k$, and for $T_{\boldsymbol{\theta}}\left(\mathbf{x}_i^k\right)$ as $\tilde{\mathbf{x}}_i^k$, where $\tilde{\mathbf{x}}_i^k = \left[\tilde{u}_i^k, \tilde{v}_i^k\right]$, we form two data matrices $\mathbf{Y}_i$ and $\mathbf{X}_i$, where

$$\mathbf{Y}_i = \mathbf{X}_i \mathbf{A}_i, \quad (4)$$

$$\mathbf{Y}_i = \begin{bmatrix} \tilde{\mathbf{I}}_i^1 - \tilde{\mathbf{I}}_i^0 & \tilde{\mathbf{I}}_i^2 - \tilde{\mathbf{I}}_i^0 & \cdots & \tilde{\mathbf{I}}_i^{K-1} - \tilde{\mathbf{I}}_i^0 \end{bmatrix}^\top, \quad (5)$$

$$\mathbf{X}_i = \begin{bmatrix} \tilde{u}_i^1 - \tilde{u}_i^0 & \tilde{u}_i^2 - \tilde{u}_i^0 & & \tilde{u}_i^{K-1} - \tilde{u}_i^0 \\ \tilde{v}_i^1 - \tilde{v}_i^0 & \tilde{v}_i^2 - \tilde{v}_i^0 & \cdots & \tilde{v}_i^{K-1} - \tilde{v}_i^0 \\ 1 & 1 & & 1 \end{bmatrix}^\top, \quad (6)$$

and we therefore write

$$\mathbf{A}_i = \left(\mathbf{X}_i^\top \mathbf{X}_i \right)^{-1} \mathbf{X}_i \mathbf{Y}_i. \quad (7)$$

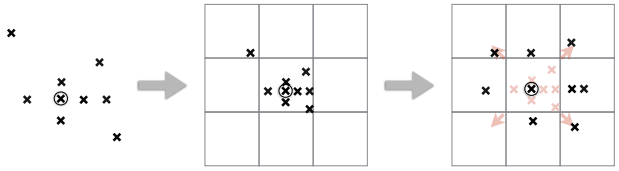


Figure 4: Sample collapse prevention. (**Left**) Auxiliary samples can fall onto a single pixel (**middle**), depending on the transformation, *e.g.* when upsampling. (**Right**) We apply additional noise to the *transformed* auxiliary sample locations, thus preventing sample collapse.

For numerical stability, we add a small epsilon to the inversion in Eq. (7).

$$\mathbf{A}_i = \left(\mathbf{X}_i^\top \mathbf{X}_i + \epsilon \mathbf{E} \right)^{-1} \mathbf{X}_i \mathbf{Y}_i, \quad (8)$$

where $\mathbf{E}$ is the 3×3 identity matrix.

Note that an alternative approach would be to use multi-scale alignment with auxiliary samples distributed over predefined 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 global structure in an efficient manner.

5. Preventing sample collapse

While in theory Eq. (7) is straightforward to implement and compute, in practice, we need to take special care that the random samples do not collapse into a single pixel. As illustrated in Fig. 3, we generate the auxiliaries in the sampled image coordinates, *i.e.* the grid used to sample and ultimately form our output image. Because of this, when the transformation is shrinking the coordinates, which happens when zooming into a specific region or performing upsampling, there is a chance for all samples to fall into a single

Figure 5: Selected images from our datasets. **(Top-left)** MNIST. **(Top-right)** GTSRB. **(Bottom)** Pascal VOC 2012.

pixel. This can cause numerical instability in Eq. (8), even with $\epsilon\mathbf{E}$, as $\mathbf{X}_i$—the data matrix generated from coordinate differences of transformed auxiliary samples—will be composed of very small numbers. Furthermore, $\mathbf{Y}_i$ in Eq. (8)—the data matrix generated from difference in intensities—may also become zero in this case, leading to zero gradients.

To avoid this sample collapse from happening, as illustrated in Fig. 4, we add additional noise to the auxiliary samples, once their coordinates are transformed. Mathematically, if we denote the modified coordinates for the k-th auxiliary sample for pixel i with $\tilde{u}_i^k$ and $\tilde{v}_i^k$, for $k \in \{1, 2, ..., K-1\}$ we then apply

$$\tilde{u}_i^k \leftarrow \tilde{u}_i^k + \mathcal{N}(0, \delta_u), \tag{9}$$

$$\tilde{v}_i^k \leftarrow \tilde{v}_i^k + \mathcal{N}(0, \delta_v), \tag{10}$$

where δ_u and δ_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 inspect the gradients produced by bilinear sampling and our linearized multi-sampling approach visually, and demonstrate how the two methods differ when directly optimizing through either sampling technique. We then show that this leads to significant improvements in the performance of deep networks when downsampling is involved. Finally, we study the effect of the number of auxiliary samples and the magnitude of the noise signal used to generate said auxiliary samples.

6.1. Implementation details

We implement our method[1] with PyTorch [24], and use its default bilinear sampling to get the intensity values for each perturbed point. However, this sampling process is not differentiated through, as we use these intensities for linearization only (Eq. (5)).

To further prevent manifestation from out-of-bound samples, we simply mask the sample points that do not fall into the original image. Specifically, for each pixel i, if $T_{\boldsymbol{\theta}}\left(\mathbf{x}_i^k\right)$ is out of bounds, we exclude it from Eq. (5) and Eq. (6). This can be easily done mathematically, by multiplying zero on these elements. Then, when all pixels are out of bounds, $\mathbf{A}_i$ becomes a zero matrix, thus providing a zero gradient.

Throughout the entirety of our experiments, we use eight auxiliary samples per pixel, thus $K = 8$.

6.2. Datasets

Our experiments feature three datasets: MNIST [17], GTSRB [28], and Pascal VOC 2012 [9]. We use MNIST to demonstrate that the improvements of our approach over bilinear interpolation persist even on simple datasets. On GTSRB, we display the benefits our method brings to image alignment for natural images. We use Pascal VOC 2012 to showcase that the advantages of our method are most noticeable on natural images. Example images from these datasets are pictured in Fig. 5.

6.3. Gradient analysis

We first show how the gradients produced by our approach differ from bilinear sampling. We compare how the gradients flow when we artificially set the sample parameters at certain locations, with the ground truth at the center. Specifically, we use natural images from the Pascal VOC dataset, crop the central two-thirds of the image, downsam-

[1] We will release the code and the experimental setup upon acceptance.

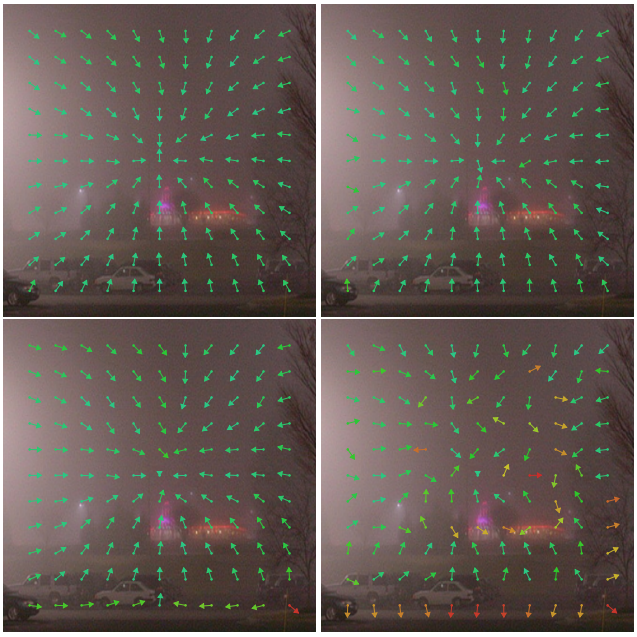


Figure 6: Gradient flow for **(top)** our method and **(bottom)** bilinear sampling, downsampling by a factor of **(left)** 1x and **(right)** 8x. Notice how our method shows a wider basin of convergence than bilinear sampling. The difference becomes more prominent as the downsampling rate increases.

ple it at different rates with bilinear interpolation, and use this image as the image sampled with ground truth parameters. We then place our sampling grid at different locations within the image, and investigate the gradients with respect to the ℓ_2 loss between the image sampled with the ground truth, and the current grid locations.

In Fig. 6 we visualize the negative gradient flows, *i.e.* the direction of steepest descent, on top of the image, on which the gradients were computed. As shown, and especially distinctive for the case of downsampling by 8x, our method provides a wider basin of convergence, having the gradients point towards the center, *i.e.* the ground truth locations, from points further away than bilinear sampling. The advantages of our linearized formulation become more prominent as the downsampling rates become more severe.

6.4. Optimizing for image alignment

Next, we demonstrate that this difference in the gradients produced by our approach leads to better outcomes when aligning images. In order to isolate the effect of the sampling only, we do not train a network to align an image, but we directly optimize the parameters of a Spatial Transformer Network (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 for 80 times, on randomly selected images from the

Figure 7: From left to right: the original image, the target, our result, and the output of the bilinear-based STN. Both are down sampled by 8x and optimized for 80 iterations.

datasets, and with random transfomations.

In more detail, for the STN we use a simple network. If we denote the convolution layers with c channels as C(c), Relu activation as R, max-pooling layer as P, our network is C(4)RC(8)RPC(16)RPC(32)RPC(1024). For the perturbation we apply random in-plane rotation, scale changes in the horizontal and vertical directions, independently, and translation. We sample Gaussian noise with varying levels of standard deviation, and convert them. If we denote this noise as n, for rotation we use $\frac{\pi}{4}n$, for scale change in x and y we use $2^{0.5n}$, and for translation $0.2n$, where the image coordinates are normalized to be between -1 and 1.

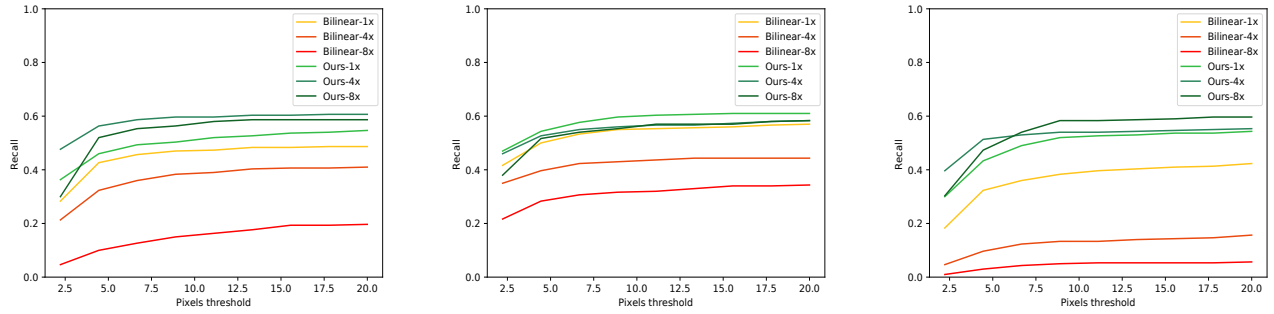
Fig. 7 shows some examples of alignments recovered with our method, and with bilinear sampling. Even at small downsampling rates, the bilinear sampler does not align the images well in case of large perturbations. This is especially prominent in natural images, where the local sub-pixel gradients do not necessarily represent how the image changes. Our method on the other hand, successfully aligns the images even for large transformations.

We further quantitatively summarize our image alignment results in Fig. 8. As shown, our results are significantly better than those of bilinear sampling. Again, the gap is largest under high downsampling rates, and on natural images. We further see that the effects of downsampling become quite severe at a factor of 8x, effectively breaking the bilinear sampler.

6.5. Training Spatial Transformer Networks

To evaluate the difference our approach makes on learning image transformations when compared to bilinear sampling, we train a classifier network with an embedded STN and evaluate the classification accuracy, on the GTSRB dataset. Specifically, emulating a typical setup for using STN in large networks, we keep the image as-is and simply ask the STN to give an output which is one-eighth the size of the original. Thus, the task of the STN is to focus on a single region that helps to classify traffic signs.

We randomly split the training set of GTSRB into two parts, 35309 images for training and 3900 for validation. There is a separate dataset with 12630 images, which we use for testing. We crop and resize all images to 50×50.



(a) MNIST (b) GTSRB (c) Pascal VOC 2012

Figure 8: Recall vs threshold for the image alignment experiments. We threshold based on the average reprojection error of the four corners of the sampled region. Notice that our method does not degrade as the downsampling increases, whereas the performance with bilinear sampling drops drastically. Note also that for natural images our method performs significantly better, even without downsampling.

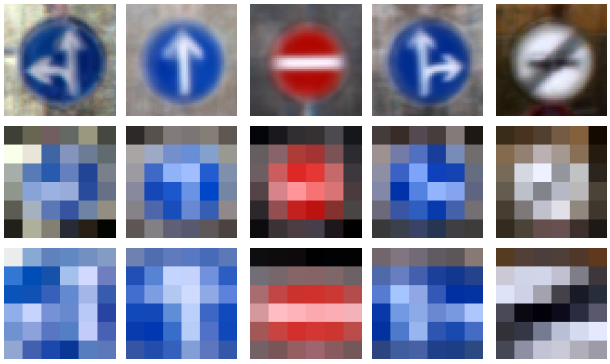
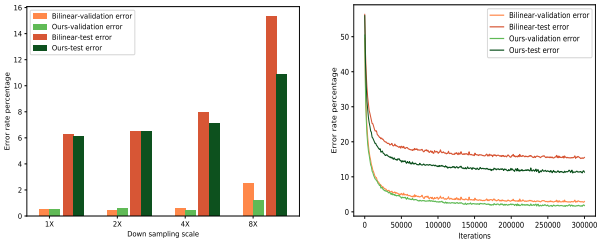


Figure 9: Outputs of two STNs trained with our approach and with bilinear sampling, with 8x downsampling, on GTSRB. **(Top)** input image. **(Middle)** bilinear sampling. **(Bottom)** ours. The STN trained with our linearized sampling scheme learns to zoom-in for better classification accuracy.

We use a CNN as the regressor to output the parameters of the transformation for the STN. The convolutional part of the CNN consists of 7×7 kernels, and the number of channels grows from 3 (RGB) to 8, 32, 1024. We apply max-pooling over each channel of the last feature map and get one feature vector size 1024. This is followed by two fully-connected layers where the number of features goes 1024, 48, 43 (as there are 43 types of traffic signs). We use ADAM [15] as the optimizer, and choose 10^{-5} and 10^{-3} as the learning rates for the regressor and the classifier respectively. We trained the models from scratch with a batch size of 64, set the maximum number of iterations to 300k, and stop the training if there is no better validation error at 80k iterations. We take the model with the lowest validation error and report both the validation error and test error.



(a) Mean accuracy (b) Error evolution

Figure 10: Quantitative results for the GTSRB dataset. While our method performs similarly when the downsampling rate is small, it performs significantly better under severe downsampling rates (4x, 8x).

In Fig. 10 we show the convergence properties and the final accuracy results of our approach vs bilinear sampling. Unsurprisingly, the performance of both degrades as downsampling becomes more severe, but significantly less so with our method. This clearly shows that bilinear sampling is not suitable for these tasks. Furthermore, as evidenced by the gap between validation and test, STNs trained with bilinear transformation tend to suffer more from overfitting.

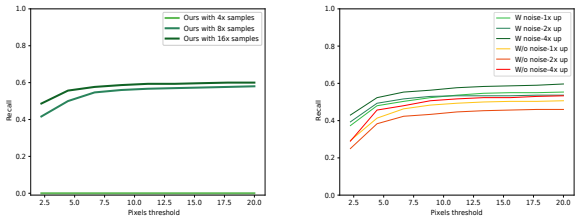
Finally, we show the average transformed image for a subset of the classes in Fig. 9. Note how the products of our network are more zoomed-in, compared to the original input image as well as the results from bilinear sampling. This shows that with our linearized sampling strategy the network successfully learns to focus on important regions.

6.6. Ablation tests

Auxiliary sample noise. We also examine the effect of the magnitude of the noise we apply to place the auxiliary

(a) (b) (c) (d)

Figure 11: Effect of the magnitude of the noise used to generate the auxiliary samples. (**a**) Quantitative results in terms of recall vs threshold. (**b-d**) The spatial context increases along with the noise (left to right), as the auxiliary samples spread out, as indicated by the blurring. In (**a**), more noise results in more spatial context and thus better image alignment results. However, as shown in (**b-d**), the image becomes more blurry as compromise.



(a) (b)

Figure 12: Effect of (**a**) number of samples on image alignment under 4x downsampling, and of (**b**) sample collapse prevention under 1x, 2x, and 4x upsampling. Both are in terms of recall vs threshold. We again use the average reprojection error of four corners of the sampled region. In (**a**) more number of samples provide better alignment results. As shown in (**b**), sample collapse prevention is essential when upsampling.

samples in Eq. (3). Fig. 11 (b)–(c) shows the sampling result under various $\boldsymbol{\sigma}$. Since the Gaussian noise emulates a point spread function, with a larger $\boldsymbol{\sigma}$ we obtain a blurrier image. While the outcome is blurry, this effectively allows the gradients to take into account a wider support region when computed, and thus be smoother.

As shown in Fig. 11 (a), larger spatial extent of the noise translates into better image alignment results in the Pascal VOC dataset. However, with larger spatial extent, the result of the image transformation becomes more blurry as a compromise.

Number of auxiliary samples. In Fig. 12a we evaluate the effect of number of samples per each pixel with the Pascal VOC dataset and 4x downsampling. As expected, more number of auxiliary samples lead to better alignment results. When only four auxiliary samples are used, the method reaches close to the minimum number of samples required for linearization, thus does not perform well.

Sample collapse avoidance. In Fig. 12b we demonstrate the importance of sample collapse prevention in Section 5. When sample collapse prevention is removed, as shown, it results in worse image alignment due to all auxiliary samples falling into a single pixel. With sample collapse avoidance, our method does not suffer from this problem.

7. Conclusions

Attention mechanisms are a powerful tool allowing deep networks to manipulate the input to attain spatial invariance. However, they remain limited in practice to a narrow range of transformations, due to their reliance on bilinear interpolation for differentiation. This results in poorly representative gradients when downsampling, which hurts the ultimate task the network is optimizing for. We propose to address this with a multi-sampling technique based on auxiliary samples randomly selected around the neighborhood of each pixel. This approach allows the gradients backpropagated via interpolation to better capture the effect of the image transformation, which results in better models for alignment and classification.

Acknowledgements

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

References

[1] H. Altwaijry, E. Trulls, S. Belongie, J. Hays, and P. Fua. Learning to Match Aerial Images with Deep Attentive Architecture. In *CVPR*, 2016. 1

[2] S. Baker and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *IJCV*, pages 221–255, March 2004. 2

[3] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of Optical Flow Techniques. *IJCV*, 12:43–77, 1994. 2

[4] C. Bhagavatula, C. Zhu, K. Luu, and M. Savvides. Faster than real-time facial alignment: A 3d spatial transformer network approach in unconstrained poses. In *ICCV*, volume 2, page 7, 2017. 2

[5] C. Chang, C. Chou, and E. Chang. CLKN: Cascaded Lucas-Kanade networks for image alignment. In *CVPR*, 2017. 3

[6] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *PAMI*, 2018. 3

[7] Y. Chen, N. M. Nasrabadi, and T. D. Tran. Hyperspectral image classification using dictionary-based sparse representation. *TGRS*, 49(10):3973–3985, 2011. 2

[8] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017. 1

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 5

[10] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 3

[11] K. He, J. Sun, and X. Tang. Guided image filtering. *PAMI*, (6):1397–1409, 2013. 2

[12] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial Transformer Networks. In *NIPS*, pages 2017–2025, 2015. 1, 2, 3

[13] Z. Jia, H. Hong, S. Wang, and Z. Tu. Top-down flow transformer networks. *arXiv Preprint*, 2017. 2, 3

[14] J. Johnson, A. Karpathy, and L. Fei-fei. Densecap: Fully Convolutional Localization Networks for Dense Captioning. In *CVPR*, 2016. 1

[15] D. P. Kingma and B. Jimmy. Adam: A method for stochastic optimization. *arXiv Preprint*, 2014. 7

[16] J. Kuen, Z. Wang, and G. Wang. Recurrent attentional networks for saliency detection. In *CVPR*, 2016. 1

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 5

[18] C. Lin and S. Lucey. Inverse compositional spatial transformer networks. *arXiv Preprint*, 2016. 2, 3

[19] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *CVPR*, pages 8759–8768, 2018. 1

[20] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, 2015. 3

[21] B. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI*, pages 674–679, 1981. 2

[22] D. Mishkin, F. Radenovic, and J. Matas. Repeatability Is Not Enough: Learning Affine Regions via Discriminability. In *ECCV*, 2018. 1, 2

[23] Y. Ono, E. Trulls, P. Fua, and K. M. Yi. Lf-Net: Learning Local Features from Images. In *NIPS*, 2018. 1, 2

[24] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in Pytorch. In *NIPS*, 2017. 5

[25] C. Qi, H. Su, K. Mo, and L. Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. 2

[26] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 3

[27] C. Shu, X. Chen, Q. Xie, C. Xiao, and H. Han. Hierarchical spatial transformer network. *arXiv Preprint*, 2018. 2, 3

[28] J. Stallkamp, S. Marc, S. Jan, and I. Christian. The german traffic sign recognition benchmark: a multi-class classification competition. *IJCNN*, 2011. 5

[29] D. Tan, T. Cashman, J. Taylor, A. Fitzgibbon, D. Tarlow, S. Khamis, S. Izadi, and J. Shotton. Fits Like a Glove: Rapid and Reliable Hand Shape Personalization. In *CVPR*, 2016. 2

[30] F. Wang, L. Zhao, X. Li, X. Wang, and D. Tao. Geometry-Aware Scene Text Detection with Instance Transformation Network. In *CVPR*, 2018. 2

[31] W. Wu, M. Kan, X. Liu, Y. Yang, S. Shan, and X. Chen. Recursive spatial transformer (rest) for alignment-free face recognition. In *CVPR*, pages 3772–3780, 2017. 2

[32] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. In *ECCV*, 2016. 2

[33] K. M. Yi, Y. Verdie, P. Fua, and V. Lepetit. Learning to Assign Orientations to Feature Points. In *CVPR*, 2016. 1

[34] H. Zhang and X. He. Deep free-form deformation network for object-mask registration. In *CVPR*, pages 4251–4259, 2017. 2

[35] H. Zhang, J. Li, Y. Huang, and L. Zhang. A nonlocal weighted joint sparse representation classification method for hyperspectral imagery. *J-STARS*, 7(6):2056–2065, 2014. 2