Segmentation-Aware Convolutional Networks Using Local Attention Masks

Adam W. Harley
Carnegie Mellon University
aharley@cmu.edu

Konstantinos G. Derpanis
Ryerson University
kosta@ryerson.ca

Iasonas Kokkinos
Facebook AI Research
iasonask@fb.com

Abstract

We introduce an approach to integrate segmentation information within a convolutional neural network (CNN). This counter-acts the tendency of CNNs to smooth information across regions and increases their spatial precision. To obtain segmentation information, we set up a CNN to provide an embedding space where region co-membership can be estimated based on Euclidean distance. We use these embeddings to compute a local attention mask relative to every neuron position. We incorporate such masks in CNNs and replace the convolution operation with a "segmentation-aware" variant that allows a neuron to selectively attend to inputs coming from its own region. We call the resulting network a segmentation-aware CNN because it adapts its filters at each image point according to local segmentation cues. We demonstrate the merit of our method on two widely different dense prediction tasks, that involve classification (semantic segmentation) and regression (optical flow). Our results show that in semantic segmentation we can match the performance of DenseCRFs while being faster and simpler, and in optical flow we obtain clearly sharper responses than networks that do not use local attention masks. In both cases, segmentation-aware convolution yields systematic improvements over strong baselines. Source code for this work is available online at http://cs.cmu.edu/~aharley/segaware.

1. Introduction

Convolutional neural networks (CNNs) have recently made rapid progress in pixel-wise prediction tasks, including depth prediction [15], optical flow estimation [14], and semantic segmentation [47, 9, 34]. This progress has been built on the remarkable success of CNNs in image classification tasks [29, 50] – indeed, most dense prediction models are based closely on architectures that were successful in object recognition. While this strategy facilitates transfer learning, it also brings design elements that are incompatible with dense prediction.

By design CNNs typically produce feature maps and predictions that are smooth and low-resolution, resulting from the repeated pooling and subsampling stages in the network architecture, respectively. These stages play an important role in the hierarchical consolidation of features, and widen the higher layer effective receptive fields. The low-resolution issue has received substantial attention: for instance methods have been proposed for replacing the subsampling layers with resolution-preserving alternatives such as atrous convolution [9, 58, 43], or restoring the lost resolution via upsampling stages [39, 34]. However, the issue of smoothness has remained relatively unexplored. Smooth neuron outputs result from the spatial pooling (*i.e.*, abstraction) of information across different regions. This can be useful in high-level tasks, but can degrade accuracy on per-pixel prediction tasks where rapid changes in activation may be required, *e.g.*, around region boundaries or motion discontinuities.

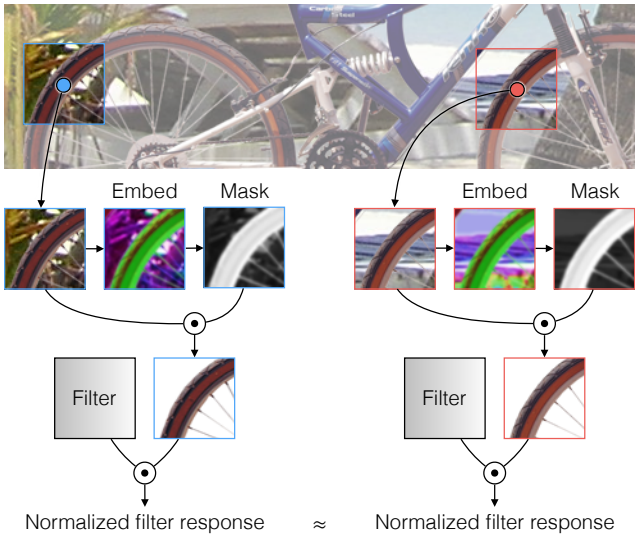

Figure 1: Segmentation-aware convolution filters are invariant to backgrounds. We achieve this in three steps: (i) compute segmentation cues for each pixel (*i.e.*, "embeddings"), (ii) create a foreground mask for each patch, and (iii) combine the masks with convolution, so that the filters only process the local foreground in each image patch.

To address the issue of smoothness, we propose *segmentation-aware* convolutional networks, which operate as illustrated in Figure 1. These networks adjust their behavior on a *per-pixel* basis according to segmentation cues, so that the filters can selectively "attend" to information coming from the region containing the neuron, and treat it differently from background signals. To achieve this, we complement each image patch with a local foreground-background segmentation mask that acts like a gating mechanism for the information feeding into the neuron. This avoids feature blurring, by reducing the extent to which foreground and contextual information is mixed, and allows neuron activation levels to change rapidly, by dynamically adapting the neuron's behavior to the image content. This goes beyond sharpening the network outputs post-hoc, as is currently common practice; it fixes the blurring problem "before the damage is done", since it can be integrated at both early and later stages of a CNN.

The general idea of combining filtering with segmentation to enhance sharpness dates back to nonlinear image processing [42, 53] and segmentation-aware feature extraction [54, 55]. Apart from showing that this technique successfully carries over to CNNs, another contribution of our work consists in using the network itself to obtain segmentation information, rather than relying on hand-crafted pipelines. In particular, as in an earlier version of this work [23], we use a constrastive side loss to train the "segmentation embedding" branch of our network, so that we can then construct segmentation masks using embedding distances.

There are three steps to creating segmentation-aware convolutional nets, described in Sections 3.1-3.4: (i) learn segmentation cues, (ii) use the cues to create local foreground masks, and (iii) use the masks together with convolution, to create foreground-focused convolution. Our approach realizes each of these steps in a unified manner that is at once general (*i.e.*, applicable to both discrete and continuous prediction tasks), differentiable (*i.e.*, end-to-end trainable as a neural network), and fast (*i.e.*, implemented as GPU-optimized variants of convolution).

Experiments show that minimally modifying existing CNN architectures to use segmentation-aware convolution yields substantial gains in two widely different task settings: dense discrete labelling (*i.e.*, semantic segmentation), and dense regression (*i.e.*, optical flow estimation). Source code for this work is available online at `http://cs.cmu.edu/~aharley/segaware`.

2. Related work

This work builds on a wide range of research topics. The first is *metric learning*. The goal of metric learning is to produce features from which one can estimate the similarity between pixels or regions in the input [18]. Bromley *et al*. [5] influentially proposed learning these descriptors in a convolutional network, for signature verification. Subsequent related work has yielded compelling results for tasks such as wide-baseline stereo correspondence [20, 59, 60], and face verification [11]. Recently, the topic of metric learning has been studied extensively in conjunction with image descriptors, such as SIFT and SID [54, 49, 3], improving the applicability of those descriptors to patch-matching problems. Most prior work in metric learning has been concerned with the task of finding one-to-one correspondences between pixels seen from different viewpoints. In contrast, the focus of our work is (as in our prior work [23]) to bring a given point close to all of the other points that lie in the same object. This requires a higher degree of invariance than before – not only to rotation, scale, and partial occlusion, but also to the interior appearance details of objects. Concurrent work has targeted a similar goal, for body joints [38] and instance segmentation [17]. We refer to the features that produce these invariances as *embeddings*, as they embed pixels into a space where the quality of correspondences can be measured as a distance.

The embeddings in our work are used to generate local *attention masks* to obtain *segmentation-aware* feature maps. The resulting features are meant to capture the appearance of the foreground (relative to a given point), while being invariant to changes in the background or occlusions. To date, related work has focused on developing handcrafted descriptors that have this property. For instance, soft segmentation masks [41, 32] and boundary cues [36, 48] have been used to develop segmentation-aware variants of handcrafted features, like SIFT and HOG, effectively suppressing contributions from pixels likely to come from the background [54, 55]. More in line with the current paper are recent works that incorporate segmentation cues into CNNs, by sharpening or masking intermediate feature maps with the help of superpixels [12, 19]. This technique adds spatial structure to multiple stages of the pipeline. In all of these works, the affinities are defined in a handcrafted manner, and are typically pre-computed in a separate process. In contrast, we learn the cues directly from image data, and compute the affinities densely and "on the fly" within a CNN. Additionally, we combine the masking filters with arbitrary convolutional filters, allowing any layer (or even all layers) to perform segmentation-aware convolution.

Concurrent work in language modelling [13] and image generation [40] has also emphasized the importance of locally masked (or "gated") convolutions. Unlike these works, our approach uniquely makes use of embeddings to measure context relevance, which lends interpretability to the masks, and allows for task-agnostic pre-training. Similar attention mechanisms are being used in visual [35] and non-visual [52] question answering tasks. These works use a question to construct a single or a limited sequence of globally-supported attention signals. Instead, we use con-

volutional embeddings, and efficiently construct local attention masks in "batch mode" around the region of any given neuron.

Another relevant thread of works relates to efforts on mitigating the low-resolution and spatially-imprecise predictions of CNNs. Approaches to counter the spatial imprecision weakness can be grouped into preventions (*i.e.*, methods integrated early in the CNN), and cures (*i.e.*, post-processes). A popular preventative method is atrous convolution (also known as "dilated" convolution) [9, 58], which allows neurons to cover a wider field of view with the same number of parameters. Our approach also adjusts neurons' field of view, but focuses it toward the local foreground, rather than widening it in general. The "cures" aim to restore resolution or sharpness after it has been lost. For example, one effective approach is to add trainable upsampling stages to the network, via "deconvolution" layers [39, 34]. A complementary approach is to stack features from multiple resolutions near the end of the network, so that the final stages have access to both high-resolution (shallow) features and low-resolution (deep) features [22, 37, 14]. Sharpening can be done outside of the CNN, *e.g.*, using edges found in the image [8, 4], or using a dense conditional random field (CRF) [28, 9, 58]. Recently, the CRF approach has been integrated more closely with the CNN, by framing the CRF as a recurrent network, and chaining it to the backpropagation of the underlying CNN [61]. We make connections and extensions to CRFs in Section 3.3 and provide comparisons in Section 5.1.

3. Technical approach

The following subsections describe the main components of our approach. We begin by learning segmentation cues (Sec. 3.1). We formulate this as a task of finding "segmentation embeddings" for the pixels. This step yields features that allow region similarity to be measured as a distance in feature-space. That is, if two pixels have nearby embeddings, then they likely come from the same region. We next create soft segmentation masks from the embeddings (Sec. 3.2). Our approach generalizes the bilateral filter [31, 2, 51, 53], which is a technique for creating adaptive smoothing filters that preserve object boundaries. Noting that CRFs make heavy use of bilateral filters to sharpen posterior estimates, we next describe how to simplify and improve CRFs using our segmentation-aware masks (Sec. 3.3). Finally, in Sec. 3.4 we introduce *segmentation-aware convolution*, where we merge segmentation-aware masks with intermediate convolution operations, giving rise to segmentation-aware networks.

3.1. Learning segmentation cues

The first goal of our work is to obtain segmentation cues. In particular, we desire features that can be used to infer – for each pixel – what other pixels belong to the same object (or scene segment).

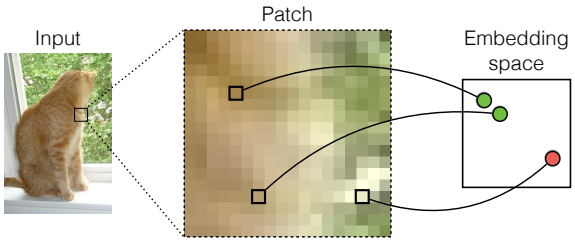

Figure 2: Visualization of the goal for pixel embeddings. For any two pixels sampled from the same object, the embeddings should have a small relative distance. For any two pixels sampled from different objects, the embeddings should have a large distance. The embeddings are illustrated in 2D; in principle, they can have any dimensionality.

Given an RGB image, $\mathbf{I}$, made up of pixels, $\mathbf{p} \in \mathbb{R}^3$ (*i.e.*, 3D vectors encoding color), we learn an embedding function that maps (*i.e.*, embeds) the pixels into a feature space where semantic similarity between pixels can be measured as a distance [11]. Choosing the dimensionality of that feature space to be $D = 64$, we can write the embedding function as $f : \mathbb{R}^3 \mapsto \mathbb{R}^D$, or more specifically, $f(\mathbf{p}) = \mathbf{e}$, where $\mathbf{e}$ is the embedding for pixel $\mathbf{p}$.

Pixel pairs that lie on the same object should produce similar embeddings (*i.e.*, a short distance in feature-space), and pairs from different objects should produce dissimilar embeddings (*i.e.*, a large distance in feature-space). Figure 2 illustrates this goal with 2D embeddings. Given semantic category labels for the pixels as training data, we can represent the embedding goal as a loss function over pixel pairs. For any two pixel indices i and j, and corresponding embeddings $\mathbf{e}_i, \mathbf{e}_j$ and object class labels $\mathbf{l}_i, \mathbf{l}_j$, we can optimize the same-label pairs to have "near" embeddings, and the different-label pairs to have "far" embeddings. Using α and β to denote the "near" and "far" thresholds, respectively, we can define the pairwise loss as

$$\ell_{i,j} = \begin{cases} \max\left(\|\mathbf{e}_i - \mathbf{e}_j\| - \alpha, 0\right) & \text{if } \mathbf{l}_i = \mathbf{l}_j \\ \max\left(\beta - \|\mathbf{e}_i - \mathbf{e}_j\|, 0\right) & \text{if } \mathbf{l}_i \neq \mathbf{l}_j \end{cases}, \quad (1)$$

where $\|\cdot\|$ denotes a vector norm. We find that embeddings learned from L^1 and L^2 norms are similar, but L^1-based embeddings are less vulnerable to exploding gradients. For thresholds, we use $\alpha = 0.5$, and $\beta = 2$. In practice, the specific values of α and β are unimportant, so long as $\alpha \leq \beta$ and the remainder of the network can learn to compensate for the scale of the resulting embeddings, *e.g.*, through λ in upcoming Eq. 3.

To quantify the overall quality of the embedding function, we simply sum the pairwise losses (Eq. 1) across the image. Although for an image with N pixels there are N^2 pairs to evaluate, we find it is effective to simply sample

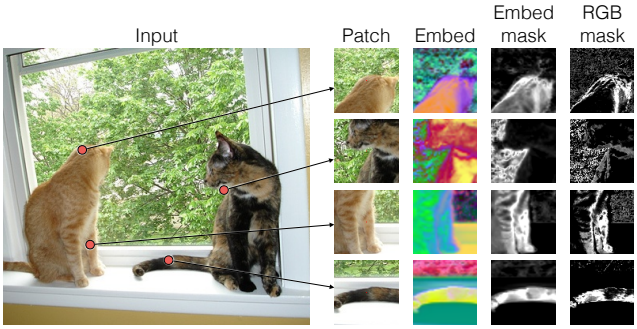

Figure 3: Embeddings and local masks are computed densely for input images. For four locations in the image shown on the left, the figure shows (left-to-right) the extracted patch, the embeddings (compressed to three dimensions by PCA for visualization), the embedding-based mask, and the mask generated by color distance.

pairs from a neighborhood around each pixel, as in

$$\mathcal{L} = \sum_{i \in N} \sum_{j \in \mathcal{N}_i} \ell_{i,j}, \tag{2}$$

where $j \in \mathcal{N}_i$ iterates over the spatial neighbors of index i. In practice, we use three overlapping 3×3 neighborhoods, with atrous factors [9] of 1, 2, and 5. We train a fully-convolutional CNN to minimize this loss through stochastic gradient descent. The network design is detailed in Sec. 4.

3.2. Segmentation-aware bilateral filtering

The distance between the embedding at one index, $\mathbf{e}_i$, and any other embedding, $\mathbf{e}_j$, provides a magnitude indicating whether or not i and j fall on the same object. We can convert these magnitudes into (unnormalized) probabilities, using the exponential distribution:

$$\mathbf{m}_{i,j} = \exp(-\lambda \|\mathbf{e}_i - \mathbf{e}_j\|), \tag{3}$$

where λ is a learnable parameter specifying the hardness of this decision, and the notation $\mathbf{m}_{i,j}$ denotes that i is the reference pixel, and j is the neighbor being considered. In other words, considering all indices $j \in \mathcal{N}_i$, $\mathbf{m}_i$ represents a foreground-background segmentation mask, where the central pixel i is defined as the foreground, *i.e.*, $\mathbf{m}_{i,i} = 1$. Figure 3 shows examples of the learned segmentation masks (and the intermediate embeddings), and compares them with masks computed from color distances. In general, the learned semantic embeddings successfully generate accurate foreground-background masks, whereas the color-based embeddings are not as reliable.

A first application of these masks is to perform a segmentation-aware smoothing (of pixels, features, or predictions). Given an input feature $\mathbf{x}_i$, we can compute a segmentation-aware smoothed result, $\mathbf{y}_i$, as follows:

$$\mathbf{y}_i = \frac{\sum_k \mathbf{x}_{i-k} \mathbf{m}_{i,i-k}}{\sum_k \mathbf{m}_{i,i-k}}, \tag{4}$$

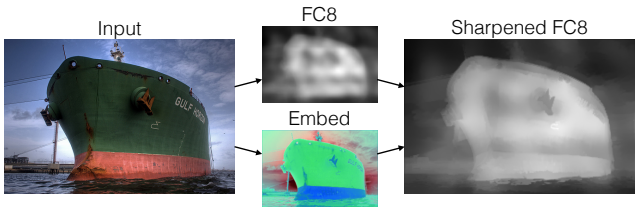

Figure 4: Segmentation-aware bilateral filtering. Given an input image (left), a CNN typically produces a smooth prediction map (middle top). Using learned per-pixel embeddings (middle bottom), we adaptively smooth the FC8 feature map with our segmentation-aware bilateral filter (right).

where k is a spatial displacement from index i. Equation 4 has some interesting special cases, which depend on the underlying indexed embeddings $\mathbf{e}_j$:

- if $\mathbf{e}_j = 0$, the equation yields the average filter;
- if $\mathbf{e}_j = i$, the equation yields Gaussian smoothing;
- if $\mathbf{e}_j = (i, \mathbf{p}_i)$, where $\mathbf{p}_i$ denotes the color vector at i, the equation yields bilateral filtering [31, 2, 51, 53].

Since the embeddings are learned in a CNN, Eq. 4 represents a generalization of all these cases. For comparison, Jampani *et al.* [25] propose to learn the kernel used in the bilateral filter, but keep the arguments to the similarity measure (*i.e.*, $\mathbf{e}_i$) fixed. In our work, by training the network to provide convolutional embeddings, we additionally learn the arguments of the bilateral distance function.

When the embeddings are integrated into a larger network that uses them for filtering, the embedding loss function (Eq. 2) is no longer necessary. Since all of the terms in the filter function (Eq. 4) are differentiable, the global objective (*e.g.*, classification accuracy) can be used to tune not only the input terms, $\mathbf{x}_i$, but also the mask terms, $\mathbf{m}_{i,j}$, and their arguments, $\mathbf{e}_j$. Therefore, the embeddings can be learned end-to-end in the network when used to create masks. In our work, we first train the embeddings with a dedicated loss, then fine-tune them in the larger pipeline in which they are used for masks.

Figure 4 shows an example of how segmentation-aware bilateral filtering sharpens FC8 predictions in practice.

3.3. Segmentation-aware CRFs

Segmentation-aware bilateral filtering can be used to improve CRFs. As discussed earlier, dense CRFs [28] are effective at sharpening the prediction maps produced by CNNs [9, 61].

These models optimize a Gibbs energy given by

$$E(\mathbf{x}) = \sum_i \psi_u(\mathbf{x}_i) + \sum_i \sum_{j \leq i} \psi_p(\mathbf{x}_i, \mathbf{x}_j), \tag{5}$$

where i ranges over all pixel indices in the image. In semantic segmentation, the unary term ψ_u is typically chosen to be the negative log probability provided by a CNN trained for per-pixel classification. The pairwise potentials take the form $\psi_p(\mathbf{x}_i, \mathbf{x}_j) = \mu(\mathbf{x}_i, \mathbf{x}_j)k(\mathbf{f}_i, \mathbf{f}_j)$, where μ is a label compatibility function (*e.g.*, the Potts model), and $k(\mathbf{f}_i, \mathbf{f}_j)$ is a feature compatibility function. The feature compatibility is composed of an appearance term (a bilateral filter), and a smoothness term (an averaging filter), in the form

$$k(\mathbf{f}_i, \mathbf{f}_j) = w^1 \exp\left(-\frac{\|i-j\|^2}{2\theta_\alpha^2} - \frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\theta_\beta^2}\right) + w^2 \exp\left(-\frac{\|i-j\|^2}{2\theta_\gamma^2}\right), \tag{6}$$

where the w^k are weights on the two terms. Combined with the label compatibility function, the appearance term adds a penalty if a pair of pixels are assigned the same label but have dissimilar colors. To be effective, these filtering operations are carried out with extremely wide filters (*e.g.*, the size of the image), which necessitates using a data structure called a permutohedral lattice [1].

Motivated by our earlier observation that learned embeddings are a stronger semantic similarity signal than color (see Fig. 3), we replace the color vector $\mathbf{p}_i$ in Eq. 6 with the learned embedding vector $\mathbf{e}_i$. The permutohedral lattice would be inefficient for such a high-dimensional filter, but we find that the signal provided by the embeddings is rich enough that we can use small filters (*e.g.*, 13×13), and achieve the same (or better) performance. This allows us to implement the entire CRF with standard convolution operators, reduce computation time by half, and backpropagate through the CRF into the embeddings.

3.4. Segmentation-aware convolution

The bilateral filter in Eq. 4 is similar in form to convolution, but with a non-linear sharpening mask instead of a learned task-specific filter. In this case, we can have the benefits of both, by inserting the learned convolution filter, $\mathbf{t}$, into the equation:

$$\mathbf{y}_i = \frac{\sum_k \mathbf{x}_{i-k}\mathbf{m}_{i,i-k}\mathbf{t}_k}{\sum_k \mathbf{m}_{i,i-k}}. \tag{7}$$

This is a non-linear convolution: the input signal is multiplied pointwise by the normalized local mask before forming the inner product with the learned filter. If the learned filter $\mathbf{t}_i$ is all ones, we have the same bilateral filter as in Eq. 4; if the embedding-based segmentation mask $\mathbf{m}_i$ is all ones, we have standard convolution. Since the masks in this context encode segmentation cues, we refer to Eq. 7 as segmentation-aware convolution.

The mask acts as an applicability function for the filter, which makes segmentation-aware convolution a special case of normalized convolution [27]. The idea of normalized convolution is to "focus" the convolution operator on the part of the input that truly describes the input signal, avoiding the interpolation of noise or missing information. In this case, "noise" corresponds to information coming from regions other than the one to which index i belongs.

Any convolution filter can be made segmentation-aware. The advantage of segmentation awareness depends on the filter. For instance, a center-surround filter might be rendered useless by the effect of the mask (since it would block the input from the "surround"), whereas a filter selective to a particular shape might benefit from invariance to context. The basic intuition is that the information masked out needs to be distracting rather than helping; realizing this in practice requires learning the masking functions. In our work, we use backpropagation to learn both the arguments and the softness of each layer's masking operation, *i.e.*, both $\mathbf{e}_i$ and λ in Eq. 3. Note that the network can always fall back to a standard CNN by simply learning a setting of $\lambda = 0$.

4. Implementation details

This section first describes how the basic ideas of the technical approach are integrated in a CNN architecture, and then provides details on how the individual components are implemented efficiently as convolution-like layers.

4.1. Network architecture

Any convolutional network can be made segmentation-aware. In our work, the technique for achieving this modification involves generating embeddings with a dedicated "embedding network", then using masks computed from those embeddings to modify the convolutions of a given task-specific network. This implementation strategy is illustrated in Figure 5.

The embedding network has the following architecture. The first seven layers share the design of the earliest convolution layers in VGG-16 [7], and are initialized with that network's (object recognition-trained) weights. There is a subsampling layer after the second convolution layer and also after the fourth convolution layer, so the network captures information at three different scales. The final output from each scale is sent to a pairwise distance computation (detailed in Sec. 4.2) followed by a loss (as in Eq. 1), so that each scale develops embedding-like representations. The outputs from the intermediate embedding layers are then upsampled to a common resolution, concatenated, and sent to a convolution layer with 1×1 filters. This layer learns a weighted average of the intermediate embeddings, and creates the final embedding for each pixel.

The idea of using a loss at intermediate layers is inspired by Xie and Tu [57], who used this strategy to learn boundary cues in a CNN. The motivation behind this strategy is to provide early layers a stronger signal of the network's end

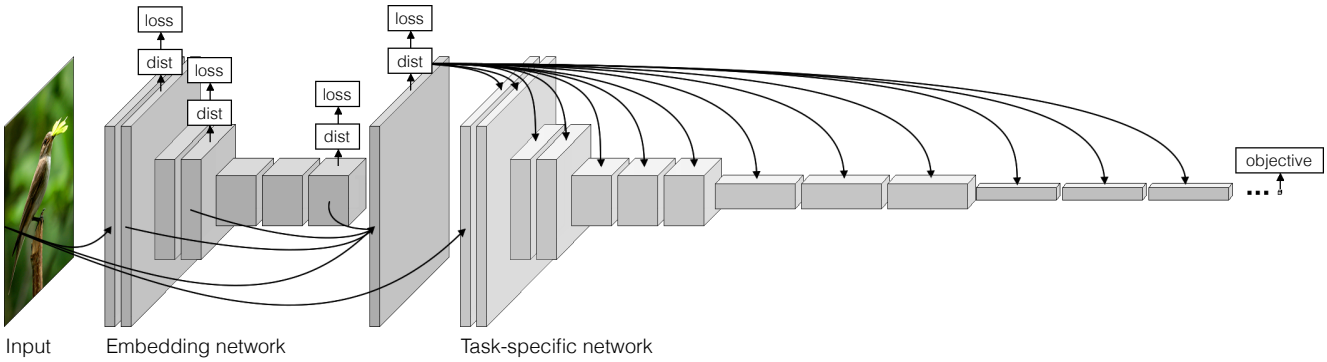

Figure 5: General schematic for our segmentation-aware CNN. The first part is an embedding network, which is guided to compute embedding-like representations at multiple scales, and constructs a final embedding as a weighted sum of the intermediate embeddings. The loss on these layers operates on pairwise distances computed from the embeddings. These same distances are then used to construct local attention masks, that intercept the convolutions in a task-specific network. The final objective backpropagates through both networks, fine-tuning the embeddings for the task.

goal, reducing the burden on backpropagation to carry the signal through multiple layers [30].

The final embeddings are used to create masks in the task-specific network. The lightest usage of these masks involves performing segmentation-aware bilateral filtering on the network's final layer outputs; this achieves the sharpening effect illustrated in Figure 4. The most intrusive usage of the masks involves converting all convolutions into segmentation-aware convolutions. Even in this case, however, the masks can be inserted with no detrimental effect (*i.e.*, by initializing with $\lambda = 0$ in Eq. 3), allowing the network to learn whether or not (and at what layer) to activate the masks. Additionally, if the target task has discrete output labels, as in the case of semantic segmentation, a segmentation-aware CRF can be attached to the end of the network to sharpen the final output predictions.

4.2. Efficient convolutional implementation details

We reduce all steps of the pipeline to matrix multiplications, making the approach very efficient on GPUs. We achieve this by casting the mask creation (*i.e.*, pairwise embedding distance computation) as a convolution-like operation, and implementing it in exactly the way Caffe [26] realizes convolution: via an image-to-column transformation, followed by matrix multiplication.

More precisely, the distance computation works as follows. For every position i in the feature-map provided by the layer below, a patch of features is extracted from the neighborhood $j \in \mathcal{N}_i$, and distances are computed between the central feature and its neighbors. These distances are arranged into a row vector of length K, where K is the spatial dimensionality of the patch. This process turns an $H \times W$ feature-map into an $H \cdot W \times K$ matrix, where each element in the K dimension holds a distance relating that pixel to the central pixel at that spatial index.

To convert the distances into masks, the $H \cdot W \times K$ matrix is passed through an exponential function with a specified hardness, λ. This operation realizes the mask term (Eq. 3). In our work, the hardness of the exponential is learned as a parameter of the CNN.

To perform the actual masking, the input to be masked is simply processed by an image-to-column transformation (producing another $H \cdot W \times K$ matrix), then multiplied pointwise with the normalized mask matrix. From that product, segmentation-aware bilateral filtering is merely a matter of summing across the K dimension, producing an $H \cdot W \times 1$ matrix that can be reshaped into dimensions $H \times W$. Segmentation-aware convolution (Eq. 7) simply requires multiplying the $H \cdot W \times K$ masked values with a $K \times F$ matrix of weights, where F is the number of convolution filters. The result of this multiplication can be reshaped into F different $H \times W$ feature maps.

5. Evaluation

We evaluate on two different dense prediction tasks: semantic segmentation, and optical flow estimation. The goal of the experiments is to minimally modify strong baseline networks, and examine the effects of instilling various levels of "segmentation awareness".

5.1. Semantic segmentation

Semantic segmentation is evaluated on the PASCAL VOC 2012 challenge [16], augmented with additional images from Hariharan *et al.* [21]. Experiments are carried out with two different baseline networks, "DeepLab" [9] and "DeepLabV2" [10]. DeepLab is a fully-convolutional version of VGG-16 [7], using atrous convolution in some layers to reduce downsampling. DeepLabV2 is a fully-convolutional version of a 101-layer residual network (ResNet) [24], modified with atrous spatial pyramid pooling and multi-scale input processing. Both networks are initial-

Table 1: PASCAL VOC 2012 validation results for the various considered approaches, compared against the baseline. All methods use DeepLab as the base network; "BF" means bilateral filter; "SegAware" means segmentation-aware.

Method	IOU (%)
DeepLab	66.33
...+ CRF	67.60
...+ 9×9 SegAware BF	66.98
...+ 9×9 SegAware BF $\times 2$	67.36
...+ 9×9 SegAware BF $\times 4$	67.68
...with FC6 SegAware	67.40
...with all layers SegAware	67.94
...with all layers SegAware + 9×9 BF	68.00
...with all layers SegAware + 7×7 BF $\times 2$	68.57
...with all layers SegAware + 5×5 BF $\times 4$	68.52
...with all layers and CRF SegAware	**69.01**

ized with weights learned on ImageNet [46], then trained on the Microsoft COCO training and validation sets [33], and finally fine-tuned on the PASCAL images [16, 21].

To replace the densely connected CRF used in the original works [9, 10], we attach a very sparse segmentation-aware CRF. We select the hyperparameters of the segmentation-aware CRF via cross validation on a small subset of the validation set, arriving at a 13×13 bilateral filter with an atrous factor of 9, a 5×5 spatial filter, and 2 meanfield iterations for both training and testing.

We carry out the main set of experiments with DeepLab on the VOC validation set, investigating the piecewise addition of various segmentation-aware components. A summary of the results is presented in Table 1. The first result is that using learned embeddings to mask the output of DeepLab approximately provides a 0.6% improvement in mean intersection-over-union (IOU) accuracy. This is achieved with a single application of a 9×9 bilateral-like filter on the FC8 outputs produced by DeepLab.

Once the embeddings and masks are computed, it is straightforward to run the masking process repeatedly. Applying the process multiple times improves performance by strengthening the contribution from similar neighbors in the radius, and also by allowing information from a wider radius to contribute to each prediction. Applying the bilateral filter four times increases the gain in IOU accuracy to 1.3%. This is at the cost of approximately 500 ms of additional computation time. A dense CRF yields slightly worse performance, at approximately half the speed (1 second).

Segmentation-aware convolution provides similar improvements, at less computational cost. Simply making the FC6 layer segmentation-aware produces an improvement of approximately 1% to IOU accuracy, at a cost of +100 ms,

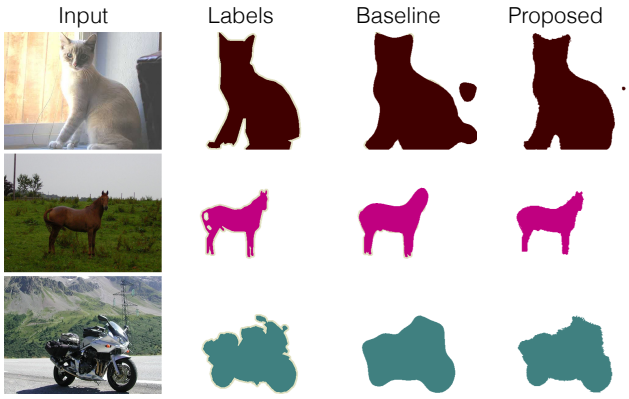

Figure 6: Visualizations of semantic segmentations produced by DeepLab and its segmentation-aware variant on the PASCAL VOC 2012 validation set.

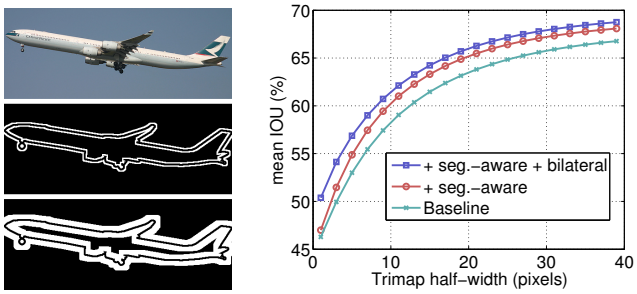

Figure 7: Performance near object boundaries ("trimaps"). Example trimaps are visualized (in white) for the image in the top left; the trimap of half-width three is shown in the middle left, and the trimap of half-width ten is shown on the bottom left. Mean IOU performance of the baseline and two segmentation-aware variants are plotted (right) for trimap half-widths 1 to 40.

while making all layers segmentation-aware improves accuracy by 1.6%, at a cost of just +200 ms.

To examine where the gains are taking place, we compute each method's accuracy within "trimaps" that extend from the objects' boundaries. A trimap is a narrow band (of a specified half-width) that surrounds a boundary on either side; measuring accuracy exclusively within this band can help separate within-object accuracy from on-boundary accuracy [9]. Figure 7 (left) shows examples of trimaps, and (right) plots accuracies as a function of trimap width. The results show that segmentation-aware convolution offers its main improvement slightly away from the boundaries (*i.e.*, beyond 10 pixels), while bilateral filtering offers its largest improvement very near the boundary (*i.e.*, within 5 pixels).

Combining segmentation-aware convolution with bilateral filtering pushes the gains to 2.2%. Finally, adding a segmentation-aware CRF to the pipeline increases IOU ac-

Table 2: PASCAL VOC 2012 test results.

Method	IOU (%)
DeepLab	67.0
DeepLab+CRF	68.2
SegAware DeepLab	69.0
DeepLabV2	79.0
DeepLabV2+CRF	79.7
SegAware DeepLabV2	**79.8**

curacy by an additional 0.5%, bringing the overall gain to approximately 2.7% over the DeepLab baseline.

We evaluate the "all components" approach on the VOC test server, with both DeepLab and DeepLabV2. Results are summarized in Table 2. The improvement over DeepLab is 2%, which is noticeable in visualizations of the results, as shown in Figure 6. DeepLabV2 performs approximately 10 points higher than DeepLab; we exceed this improvement by approximately 0.8%. The segmentation-aware modifications perform equally well (0.1% superior) to dense CRF post-processing, despite being simpler (using only a sparse CRF, and replacing the permutohedral lattice with basic convolution), and twice as fast (0.5s rather than 1s).

5.2. Optical flow

We evaluate optical flow on the recently introduced FlyingChairs [14] dataset. The baseline network for this experiment is the "FlowNetSimple" model from Dosovitskiy *et al*. [14]. This is a fully-convolutional network, with a contractive part that reduces the resolution of the input by a factor of 64, and an expansionary part (with skip connections) that restores the resolution to quarter-size.

In this context, we find that relatively minor segmentation-aware modifications yield substantial gains in accuracy. Using embeddings pre-trained on PASCAL VOC, we make the final prediction layer segmentation-aware, and add 9×9 bilateral filtering to the end of the network. This reduces the average end-point error (aEPE) from 2.78 to 2.26 (an 18% reduction in error), and reduces average angular error by approximately 6 degrees, from 15.58 to 9.54. We achieve these gains without the aggressive data augmentation techniques pursued by Dosovitskiy *et al*. [14]. Table 3 lists these results in the context of some related work in this domain, demonstrating that the gain is fairly substantial. FlowNetCorr [14] achieves a better error, but it effectively doubles the network size and runtime, whereas our method only adds a shallow set of embedding layers. As shown in Figure 8, a qualitative improvement to the flow fields is easily discernable, especially near object boundaries. Note that the performance of prior FlowNet architectures diminishes with the application of variational refinement [14], likely because this step was not integrated in the training process. The filtering methods of this work, however, are easily integrated into backpropagation.

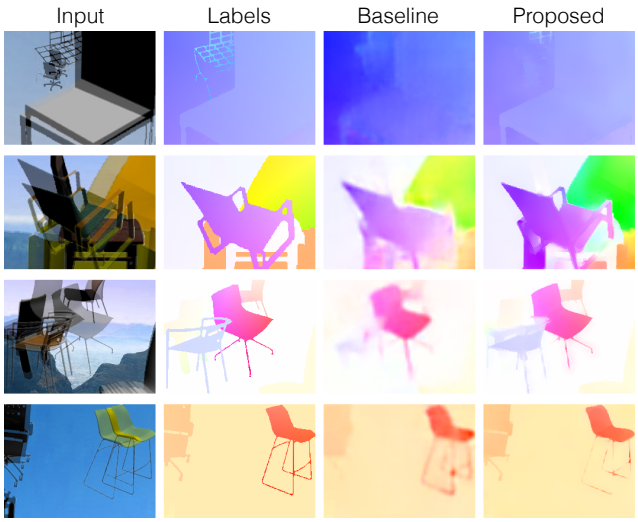

Figure 8: Visualizations of optical flow produced by FlowNet and its segmentation-aware variant on the FlyingChairs test set: segmentation-awareness yields much sharper results than the baseline.

Table 3: FlyingChairs test results.

Method	aEPE	aAE
SPyNet [44]	2.63	-
EpicFlow [45]	2.94	-
DeepFlow [56]	3.53	-
LDOF [6]	3.47	-
FlowNetSimple [14]	2.78	15.58
FlowNetSimple + variational [14]	2.86	-
FlowNetCorr [14]	**2.19**	-
FlowNetCorr + variational [14]	2.61	-
SegAware FlowNetSimple	2.36	**9.54**

6. Conclusion

This work introduces Segmentation-Aware Convolutional Networks, a direct generalization of standard CNNs that allows us to seamlessly accommodate segmentation information throughout a deep architecture. Our approach avoids feature blurring before it happens, rather than fixing it post-hoc. The full architecture can be trained end-to-end. We have shown that this allows us to directly compete with segmentation-specific structured prediction algorithms, while easily extending to continuous prediction tasks, such as optical flow estimation, that currently have no remedy for blurred responses.

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 29(2):753–762, 2010. 5

[2] V. Aurich and J. Weule. Non-linear gaussian filters performing edge preserving diffusion. In *Proceedings of the DAGM Symposium*, pages 538–545, 1995. 3, 4

[3] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk. PN-Net: Conjoined triple deep network for learning local image descriptors. *arXiv:1601.05030*, 2016. 2

[4] G. Bertasius, J. Shi, and L. Torresani. Semantic segmentation with boundary neural fields. In *CVPR*, 2016. 3

[5] J. Bromley, I. Guyon, Y. Lecun, E. Sackinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *NIPS*, 1994. 2

[6] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *PAMI*, 33(3):500–513, 2011. 8

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 5, 6

[8] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille. Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform. In *CVPR*, 2016. 3

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015. 1, 3, 4, 6, 7

[10] L.-C. 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*, 2016. 6, 7

[11] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 2, 3

[12] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015. 2

[13] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. *arXiv:1612.08083*, 2016. 2

[14] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 1, 3, 8

[15] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 1

[16] 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, 2012. 6, 7

[17] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv:1703.10277*, 2017. 2

[18] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007. 2

[19] R. Gadde, V. Jampani, M. Kiefel, and P. V. Gehler. Superpixel convolutional networks using bilateral inceptions. In *ECCV*, 2016. 2

[20] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. Berg. MatchNet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015. 2

[21] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 6, 7

[22] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 3

[23] A. W. Harley, K. G. Derpanis, and I. Kokkinos. Learning dense convolutional embeddings for semantic segmentation. In *ICLR*, 2016. 2

[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[25] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense CRFs and bilateral neural networks. In *CVPR*, 2016. 4

[26] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM-MM*, pages 675–678, 2014. 6

[27] H. Knutsson and C.-F. Westin. Normalized and differential convolution. In *CVPR*, 1993. 5

[28] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *NIPS*, 2011. 3, 4

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[30] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *AISTATS*, 2(3):6, 2015. 6

[31] J.-S. Lee. Digital image smoothing and the sigma filter. *CVGIP*, 24(2):255–269, 1983. 3, 4

[32] M. Leordeanu, R. Sukthankar, and C. Sminchisescu. Efficient closed-form solution to generalized boundary detection. In *ECCV*, pages 516–529, 2012. 2

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, pages 740–755, 2014. 7

[34] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2014. 1, 3

[35] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, pages 289–297, 2016. 2

[36] M. Maire, P. Arbeláez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *CVPR*, 2008. 2

[37] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015. 3

[38] A. Newell and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. *arXiv:1611.05424*, 2016. 2

[39] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 1, 3

[40] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with PixelCNN decoders. *arXiv:1606.05328*, 2016. 2

[41] P. Ott and M. Everingham. Implicit color segmentation features for pedestrian and object detection. In *ICCV*, 2009. 2

[42] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *PAMI*, 1990. 2

[43] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, 2014. 1

[44] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *CVPR*, 2017. 8

[45] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 8

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. 7

[47] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 1

[48] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000. 2

[49] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 2

[50] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[51] S. M. Smith and J. M. Brady. SUSAN – A new approach to low level image processing. *IJCV*, 23(1):45–78, 1997. 3, 4

[52] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015. 2

[53] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998. 2, 3, 4

[54] E. Trulls, I. Kokkinos, A. Sanfeliu, and F. Moreno-Noguer. Dense segmentation-aware descriptors. In *CVPR*, 2013. 2

[55] E. Trulls, S. Tsogkas, I. Kokkinos, A. Sanfeliu, and F. Moreno-Noguer. Segmentation-aware deformable part models. In *CVPR*, 2014. 2

[56] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, pages 1385–1392, 2013. 8

[57] S. Xie and Z. Tu. Holistically-nested edge detection. In *CVPR*, 2015. 5

[58] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 1, 3

[59] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 2

[60] J. Žbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, 2014. 2

[61] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015. 3, 4

Segmentation-Aware Convolutional Networks Using Local Attention Masks Supplementary Material

Adam W. Harley
Carnegie Mellon University
aharley@cmu.edu

Konstantinos G. Derpanis
Ryerson University
kosta@ryerson.ca

Iasonas Kokkinos
Facebook AI Research
iasonask@fb.com

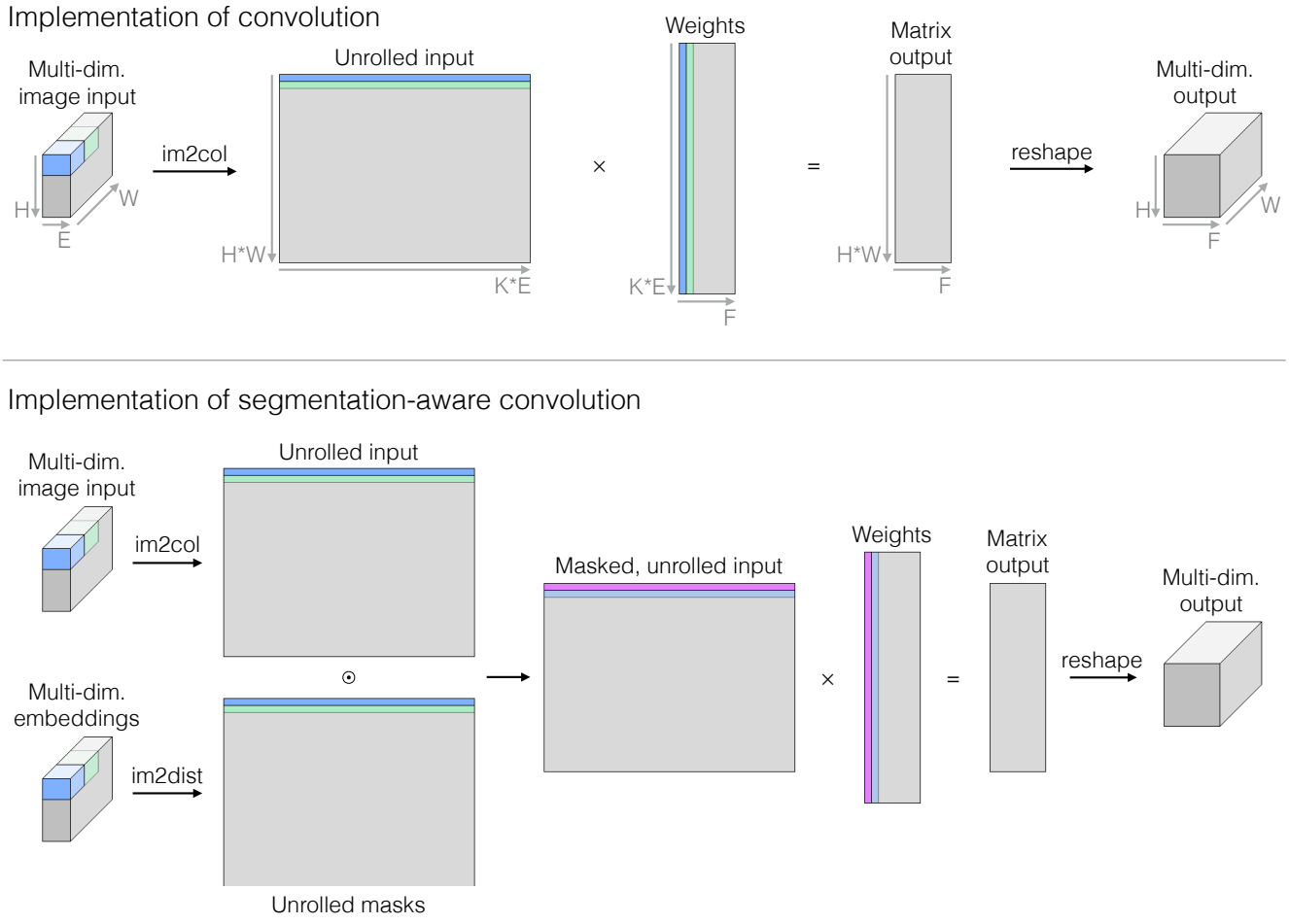

Figure 1: Implementation of convolution in Caffe, compared with the implementation of segmentation-aware convolution. Convolution involves re-organizing the elements of each (potentially overlapping) patch into a column (*i.e.*, *im2col*), followed by a matrix multiplication with weights. Segmentation-aware convolution works similarly, with an image-to-column transformation on the input, an image-to-distance transformation on the embeddings (*i.e.*, *im2dist*), a pointwise multiplication of those two matrices, and then a matrix multiplication with weights. The variables H, W denote the height and width of the input, respectively; E denotes the number of channels in the input; K denotes the dimensionality of a patch (*e.g.*, $K = 9$ in convolution with a 3×3 filter); F denotes the number of filters (and the dimensionality of the output). In both cases, an $H \times W \times E$ input is transformed into an $H \times W \times F$ output.