Content-Adaptive Downsampling in Convolutional Neural Networks

Robin Hesse[1]   Simone Schaub-Meyer[1,2]   Stefan Roth[1,2]

[1]Department of Computer Science, TU Darmstadt   [2]hessian.AI
{robin.hesse, simone.schaub, stefan.roth}@visinf.tu-darmstadt.de

Abstract

Many convolutional neural networks (CNNs) rely on progressive downsampling of their feature maps to increase the network's receptive field and decrease computational cost. However, this comes at the price of losing granularity in the feature maps, limiting the ability to correctly understand images or recover fine detail in dense prediction tasks. To address this, common practice is to replace the last few downsampling operations in a CNN with dilated convolutions, allowing to retain the feature map resolution without reducing the receptive field, albeit increasing the computational cost. This allows to trade off predictive performance against cost, depending on the output feature resolution. By either regularly downsampling or not downsampling the entire feature map, existing work implicitly treats all regions of the input image and subsequent feature maps as equally important, which generally does not hold. We propose an adaptive downsampling scheme that generalizes the above idea by allowing to process informative regions at a higher resolution than less informative ones. In a variety of experiments, we demonstrate the versatility of our adaptive downsampling strategy and empirically show that it improves the cost-accuracy trade-off of various established CNNs.
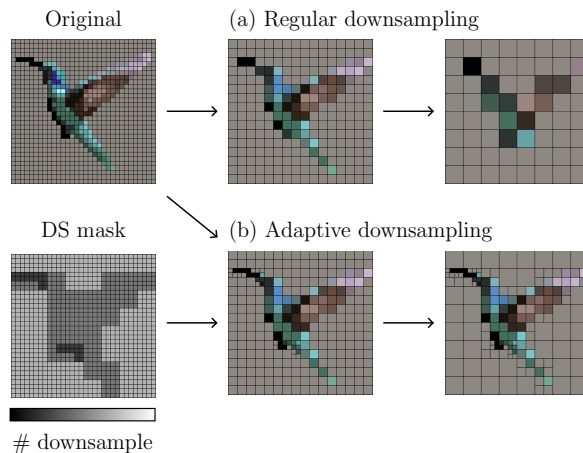
1. Introduction

When humans are exposed to a complex task, they focus on relevant aspects to make optimal use of the brain's limited capacities [2]. For instance, when categorizing the bird's species in Fig. 1, our brain allocates significantly more computational resources for the bird than for the background. While naturally occurring in humans, this adaptive allocation of resources is not utilized in most of today's deep learning architectures. In this work, we propose an adaptive downsampling scheme for Convolutional Neural Networks (CNNs) [23] that mimics above adaptive allocation of resources by *simultaneously* processing different re-

Code available at https://github.com/visinf/cad/.

Figure 1. *Illustration of our content-adaptive method compared to regular downsampling. (a) In regular downsampling, every second pixel is sampled (repeated twice here). Note how the result is almost incognizable and important detail like the beak, claws, and tail of the bird is lost. (b) In our adaptive downsampling, a precomputed downsampling mask defines the number of downsampling operations applied to each pixel. The resulting representation (zoom in to avoid aliasing) contains information of locally varying resolution. This allows to semi-continuously adjust the representation size to keep more detail where it matters.*

gions of an image or feature map *at different resolutions*.

Many CNNs used as backbones in computer vision rely on a progressive application of pooling or strided convolution [17, 22, 37] to increase the network's receptive field and decrease computational cost [46]. However, this progressive, regular downsampling comes at the price of losing fine detail in the feature maps, limiting the network's ability to correctly understand images [46] or recover fine detail in dense prediction tasks [4], such as at object boundaries. To approach this problem, Yu *et al*. [46] replace the last few downsampling operations in a CNN with dilated convolutions [45]. Although increasing the computational cost, this allows to keep the resolution of feature maps without reducing the network's receptive field. However, by regu-

larly downsampling, respectively *not* downsampling [46], *entire* feature maps, most existing CNNs implicitly assume that all regions of the input image and subsequent feature maps are equally important, which generally does not hold [21, 29, 31] as, *e.g*., also shown by locally salient attribution maps [18, 36, 39].

In contrast, our *locally adaptive* downsampling scheme makes a more realistic assumption and can be considered as a generalization of both regular downsampling in CNNs and dilated convolutions [46]. By processing task-relevant regions at a higher resolution than unimportant ones, we can maintain the fine granularity of important regions while efficiently processing unimportant regions at a lower resolution, ultimately leading to an improved trade-off between the accuracy and computational expense of various popular CNNs. Contrary to existing adaptive downsampling methods [21, 29] that adaptively downsample the input image *before* passing it through a CNN, our approach allows for adaptive downsampling of feature maps *within* a CNN.

Specifically, we make the following contributions: *(1)* We present a novel adaptive downsampling scheme for CNNs, allowing to process different feature map regions at different resolutions. *(2)* As different amounts of computational resources are being allocated for different regions, our method can be considered as a novel realization of "focus" in CNNs. This is fundamentally different from existing attention mechanisms, *e.g*., in transformers [8], where attention is implemented via adaptive weighting of input features. As a consequence, an uninformative black image and an informative natural image would still require the same inference time in a transformer, while the black image would be processed faster in our proposed adaptive downsampling. *(3)* We provide an accompanying modified instantiation of submanifold sparse convolution [14] that allows to efficiently perform standard convolution on multi-resolution grids that occur in our adaptive downsampling. *(4)* Thanks to carefully designing the proposed method to satisfy certain guarantees, it can be used in a plug-and-play fashion within existing backbones even without retraining, making it exceptionally versatile and practical. *(5)* We further empirically show with two computer vision tasks that our method is an effective and application-agnostic tool to improve the cost-accuracy trade-off of different established CNNs that build on regular downsampling.

2. Related Work

Adaptive (down)sampling. The goal of adaptive sampling approaches is to sample an image or feature map spatially to obtain optimal performance or properties of a CNN. An early instance are Spatial Transformer Networks [19], which aim to increase invariance to different geometric transformations by learning to spatially transform feature maps. Deformable convolutions [7] use two-dimensional

offsets to the regular sampling locations of standard convolutions to allow for more flexible handling of geometric variations and different receptive field sizes within one layer. CF-ViT [5] entails a multi-stage approach where in each stage the input patch resolution of the most important patches is increased to refine the result of a vision transformer. Talebi and Milanfar [41] show that resizing images with learned resizers, instead of linear ones, can improve the accuracy of recognition models. Recasens *et al*. [31] use an auxiliary saliency network to estimate the most important image regions that are then used for non-uniformly downsampling the input image, leading to an amplification of salient regions. Marin *et al*. [29] train an auxiliary network to predict a non-uniform sampling grid that is denser near semantic boundaries and use it to non-uniformly downsample input images to retain fine detail for segmentation tasks. Jin *et al*. [21] propose a similar idea as [29], particularly for ultra-high-resolution images, additionally including the segmentation accuracy in the training objective of the sampling grid estimator. Our proposed approach is fundamentally different from the above methods that adaptively downsample the image *before* passing it through a CNN [21, 29, 31], instead of *within* the CNN. As a result, they severely sacrifice accuracy, making them primarily appropriate for very high-resolution images.

Detail-preserving pooling. While above adaptive downsampling methods are concerned with the question of *how many* features to sample per region, detail-preserving pooling is concerned with the question of *which* features or feature combinations to sample to better retain important detail. Mixed Pooling [44] randomly selects max or average pooling. Lee *et al*. [24] learn a weighted combination of max and average pooling that is dependent on the pooling region. In L_p pooling [15], orders p_j are learned to interpolate between different pooling operators with $p_j = 1$ corresponding to average pooling and $p_j = \infty$ to max pooling. Detail-preserving pooling [35] is a learnable adaptive pooling method that magnifies important detail, making use of inverse bilateral filters. Local importance-based pooling [11] preserves discriminative detail by training an auxiliary network to predict adaptive importance maps that are used to aggregate features for downsampling. SoftPool [38] minimizes information loss using a softmax-weighted sum of feature activations. Note that adaptive downsampling methods [21, 31], including our approach, also incorporate regular pooling layers, and thus, advanced pooling operations such as the above can be used complementarily.

Layer aggregation and architecture. Besides the above, one can also use early high-resolution feature maps of CNNs and more complex network architectures to improve the granularity of feature maps or the output. Hypercolumns [16] build a feature vector for any pixel by concatenating activations from all feature maps above that pixel.

Fully convolutional networks [26] refine semantic segmentations by combining earlier layers of higher resolution with later layers of lower resolution. The U-Net [34] architecture extends the previous idea by introducing skip connections between all layers of corresponding resolutions. Pinheiro *et al*. [30] propose to gradually refine the output segmentation by adding information from earlier layers in a top-down fashion. Feature pyramid networks [25] produce feature pyramids by combining high-level and low-level features of a CNN. Instead of simple one-step skip-connections, Yu *et al*. [47] incorporate more depth and sharing in their layer aggregation. HRNet [43] goes even further and processes high-resolution and low-resolution streams in parallel while repeatedly exchanging information between the streams. Yu *et al*. [46] improve the granularity of feature maps by substituting downsampling operations with dilated convolutions [45], allowing to retain the feature map resolution while keeping the original receptive field of the CNN. Generally, high-level feature maps yield semantically stronger features [17,25], and thus, using auxiliary information from low-level layers might not be optimal. Further, combining layers of different resolutions or keeping higher resolutions can potentially introduce new parameters, increase computational cost, and/or raise memory usage.

Sparse convolution. While sparse convolution is not directly related to adaptive downsampling, it plays an essential role in our method. Initial works on sparse convolution [10, 12, 13, 33] improve the computational cost of standard convolution by processing only active elements in sparse inputs, such as point clouds [10], handwritten digits [12], or 3D grids [33]. Submanifold sparse convolution [14] avoids a growing number of active elements caused by standard convolutions that dilate the sparse data with each layer. Contrary to sparse convolution, which is only used to completely ignore inactive elements, we adapt sparse convolution to work with our proposed multi-resolution feature maps.

3. Content-adaptive Downsampling in CNNs

In this work, we postulate that not all feature map regions of a CNN are equally important and that, therefore, different regions should be processed at *different resolutions*. By processing only a subset of the most important feature map regions at higher resolution while downsampling less important ones, we can combine the smaller representation size of regular downsampling with the higher feature map granularity of dilated convolutions [46]. As a consequence, we can gain relatively large improvements in predictive performance while only moderately increasing the computational cost compared to regular downsampling.

The high-level idea behind our approach is illustrated in Fig. 1(b). Contrary to regular downsampling (Fig. 1(a)), where the image or feature map is downsampled uniformly by sampling every second pixel, we utilize an adaptive downsampling scheme (Fig. 1(b)) that retains higher resolution in areas with fine detail while downsampling less informative ones. In the following, we will outline the details of our method with a focus on two-dimensional feature maps with channels, *e.g*., in CNNs for vision tasks. However, the extension to other dimensionalities, such as 1D signals and 3D temporal or volumetric data, works analogously.

Regular downsampling. Let $f \in \mathbb{R}^{H \times W \times C}$ be a feature map (or image) of height H, width W, and depth C. We denote regular downsampling by a factor of $d \in \mathbb{N}_{>1}$ as the process of reducing the spatial dimension of f by outputting one element $e \in \mathbb{R}^C$ from each non-overlapping $d \times d$ patch, such that the resulting feature map f_d is of shape $\frac{H}{d} \times \frac{W}{d} \times C$. Generally, regular downsampling $\psi \colon \mathbb{R}^{H \times W \times C} \mapsto \mathbb{R}^{H/d \times W/d \times C}$ in CNNs is realized by applying a patch-wise downsampling function $\psi' \colon \mathbb{R}^{d \times d \times C} \mapsto \mathbb{R}^C$ to all HW/d^2 non-overlapping $d \times d$ patches $f'_{i,j}$ in f:

$$f_d = \psi(f) = \left(\psi'(f'_{1,1}), \ldots, \psi'(f'_{H/d,W/d}) \right). \quad (1)$$

The above definition of regular downsampling generalizes to many commonly used downsampling methods used in CNNs. For example, ψ' can take the form of various pooling operations, *e.g*., [15, 35, 44], or it can select an element based on its spatial location to implement uniform downsampling, *i.e*., sampling every d-th element. When following a convolution of stride 1, this corresponds conceptually to a strided convolution with a stride of d.

Adaptive downsampling. In this work, we generalize the above regular downsampling scheme by allowing some patches $f'_{i,j}$ to *not be downsampled*, and therefore, retaining their *original resolution* of $d \times d$. This is accomplished by providing an additional downsampling mask $m \in \{0,1\}^{H/d \times W/d}$ as input, specifying which patches $f'_{i,j}$ should be downsampled ($m_{i,j} = 1$) and which not ($m_{i,j} = 0$). For now, we assume that this downsampling mask is given, and go later into more detail about how to compute it in a content-adaptive way. Formally, our novel adaptive downsampling ψ_a is defined as

$$\psi_a(f, m) = \left(\psi'_a(f'_{1,1}), \ldots, \psi'_a(f'_{H/d,W/d}) \right),$$

$$\text{with } \psi'_a(f'_{i,j}) = \begin{cases} f'_{i,j} & \text{if } m_{i,j} = 0 \\ \psi'(f'_{i,j}) & \text{if } m_{i,j} = 1, \end{cases} \quad (2)$$

denoting a patch-wise adaptive downsampling as indicated by the downsampling mask m.

Multi-resolution grid convolution. As the resulting feature map $\psi_a(f, m)$ consists of multiple resolutions, we lose the regular grid structure of our data necessary to work with standard convolutional layers. To address this, we project
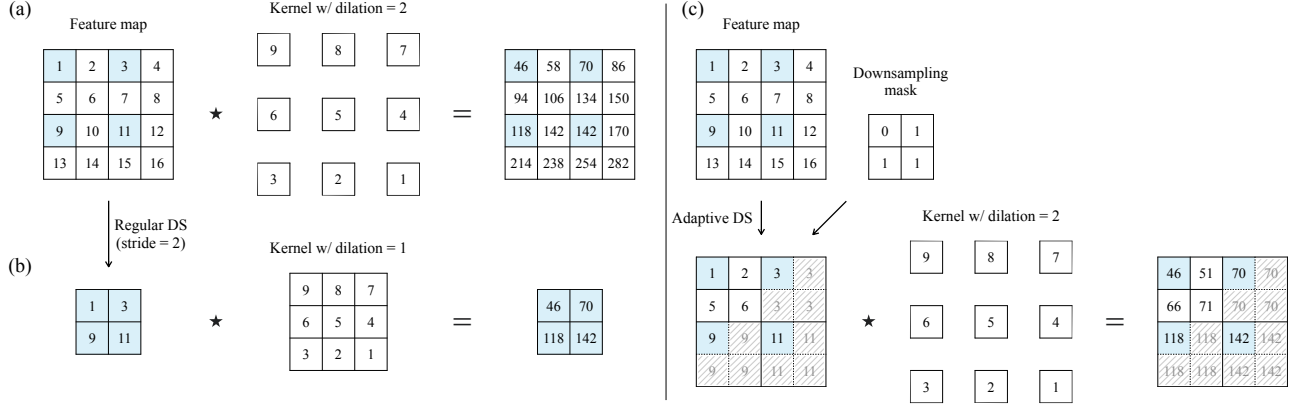
Figure 2. *Illustration of different methods to handle feature map resolution in CNNs.* ⋆ denotes the two-dimensional cross-correlation operator (w/ zero padding). *(a)* Dilated convolution increases the receptive field (equally to downsampling) while retaining the feature resolution [46]. *(b)* Standard strided convolution in CNNs (with stride=2) corresponds to uniformly downsampling the feature map by sampling every second element. This is typically followed by a standard convolution. *(c)* In our adaptive scheme, the downsampling mask is used to downsample a subset of the feature map and low-resolution entries are projected into the high-resolution grid; shaded cells (hatched) denote inactive elements. Afterward, sparse convolution is applied to the active elements (black font) of the multi-resolution feature map to produce an output with the same, multiple resolutions as the input. Note how the (loop) invariant of all algorithms (Guarantee 2) ensures that the elements highlighted in blue, both before and after convolution, are the same across the different methods.

the lower-resolution features onto their corresponding location in the high-resolution grid, as illustrated in Fig. 2(c). Naturally, high-resolution patches $f'_{i,j}$ that are filled with a lower-resolution element will have empty cells (Fig. 2(c) – shaded cells) and, thus, are sparse. We exploit the resulting sparsity by processing our feature representations in subsequent layers using sparse convolution [10, 13, 33]. This allows to only consider *active* elements, and hence, uses less computation in regions of lower effective resolution. To avoid submanifold dilation [14], *i.e.*, that inactive elements become active after convolution, we suggest a modified instance of submanifold sparse convolution [14] (SSC). SSC ensures that input and output contain the same set of active elements and thus enables us to keep the resolution of each respective feature map or image region unchanged. To retain the same receptive field as the convolutional layer would have with regular downsampling, we increase the dilation factor to d. A remaining challenge is that at specific feature map locations (see, *e.g.*, Fig. 2(c) – cell with 6) non-central elements of the convolutional kernel could fall onto inactive elements, causing the result to be corrupted. To avoid this, we modify SSC to assume that all inactive elements take the value of the corresponding active element within their patch (shown in Fig. 2(c) as shaded cells).

Extension to CNNs. Similarly to Yu *et al.* [46], our proposed adaptive downsampling is incorporated into a standard CNN with regular downsampling, *i.e.*, pooling or strided convolution, by substituting the last n downsampling operations with adaptive downsampling. Further, we substitute all convolutional layers that follow the $i \in \{1, \ldots, n\}$ adaptive downsampling operations with our pro-

posed sparse convolution with a dilation factor of d^i to keep the same receptive field as the original network. To perform multiple adaptive downsampling operations in succession, we only consider the elements belonging to the currently lowest available resolution in our multi-resolution feature map and convert them into a dense representation. We then perform adaptive downsampling on these elements according to Eq. (2). As before, the resulting features are projected to their corresponding location in the multi-resolution feature map. Consequently, patches already containing higher-resolution elements are excluded from the downsampling process. This implies that areas, where resolution is retained, cannot be downsampled at a later downsampling step, and thus, for $d = 2$, one obtains a quadtree-like resolution pattern as seen in Fig. 1(b). A more detailed explanation of this process can be found in the supplementary.

Guarantees. The resulting CNN can further be seen as a generalization of both regular CNNs with downsampling as well as of the dilated convolution of Yu *et al.* [46]. When setting all elements of the downsampling mask to one, we obtain a standard CNN with regular downsampling. When setting all elements of the downsampling mask to zero, we obtain exactly dilated convolution [46] for higher-resolution feature maps. By simultaneously processing different feature map regions at *different* resolutions, controlled by the downsampling mask, we establish a combination of the above methods that allows us to "interpolate" between different output strides in a *locally adaptive* fashion. An important desideratum behind our approach is that – like dilated convolution [46] – additional training of the backbone is not mandatory. Therefore, it is exceptionally easy to use

4

Table 1. *Oracle experiment for semantic segmentation.* For each model and output stride (OS), we report the mean IoU (mIoU) on the Cityscapes [6] evaluation split, the theoretical (average) number of multiply-adds (#MA), and the min./max. inference times in seconds. Our adaptive downsampling (OS=8→32) improves the cost-accuracy trade-off of all models, indicated by the on par mIoU scores and the significantly reduced #MA and inference times, compared to OS=8.

	Different backbones with DeepLabv3 [3]								
	ResNet-50 [17]			ResNet-101 [17]			ResNet-152 [17]		
OS	16	8	8→32	16	8	8→32	16	8	8→32
mIoU	0.7496	0.7655	0.7658	0.7591	0.7775	0.7748	0.7683	0.7852	0.7848
#MA	$2.6e11$	$8.0e11$	$3.8e11$	$4.2e11$	$1.4e12$	$6.7e11$	$5.7e11$	$1.9e12$	$1.0e12$
Time	0.05	0.11	0.05/0.08	0.07	0.22	0.08/0.15	0.1	0.31	0.11/0.25

	ResNet-101 [17] with different segmentation heads						ResNet-101 [17] with DeepLabv3 [3] and different extensions					
	FCN [26]			DeepLabv3+ [4]			SoftPool [38]			Deformable Convolution [7]		
OS	16	8	8→32	16	8	8→32	16	8	8→32	16	8	8→32
mIoU	0.7154	0.7292	0.7322	0.7745	0.781	0.7805	0.7652	0.7823	0.7819	0.7564	0.7801	0.7797
#MA	$4.2e11$	$1.4e12$	$6.7e11$	$4.2e11$	$1.4e12$	$6.7e11$	$4.2e11$	$1.4e12$	$7.4e11$	$8.0e11$	$1.4e12$	$1.0e12$
Time	0.07	0.22	0.08/0.15	0.07	0.22	0.08/0.15	0.07	0.22	0.08/0.17	0.1	0.21	0.11/0.17

in a plug-and-play fashion. This is possible due to the following two guarantees, which result from our design and hold when substituting regular downsampling, *i.e.*, strided convolution or pooling, with our adaptive downsampling:

Guarantee 1: Due to projecting all elements into the highest-resolution grid and adjusting the dilation factors of subsequent sparse convolutions, we can guarantee that the receptive fields of the CNN and each intermediate layer remain unchanged compared to the corresponding CNN with regular downsampling.

Guarantee 2: All output features of a standard CNN with strided convolution equal the output features at the corresponding high-resolution locations of the same CNN with adaptive downsampling (see Fig. 2(c) – blue cells).

This means that a CNN with adaptive downsampling is not modifying feature map values of a standard CNN with strided convolution, but instead only *adds* information at pixel locations where there was none before, and therefore, refines the feature map. Note that while Guarantee 2 formally only holds for strided convolutions and not for pooling, we show in Sec. 4.3 that applying our adaptive downsampling in a CNN with pooling is still feasible in practice without necessarily requiring retraining.

Content-adaptive downsampling masks. A remaining question is how to obtain the downsampling mask to indicate what regions to process at which resolution in a locally content-adaptive way. In Sec. 4, we show that traditional algorithms, like high-frequency detection or keypoint estimates, can serve as an effective basis to identify important image regions, and thus, to estimate downsampling masks.

Additionally, we demonstrate how to learn an appropriate downsampling mask from data. Specifically, we utilize a shallow CNN that takes as input the feature map of the main model before the adaptive downsampling step, and outputs the downsampling mask. In order to train it end-to-end with the main model, we make the discrete mask estimation differentiable with Gumbel-Softmax [20]. As training our adaptive downsampling end-to-end can result in a downsampling mask of only zeros, *i.e.*, full resolution, we control the amount of invested resources with an additional hyperparameter $\gamma \in [0, 1]$ that defines the desired proportion of active mask elements. Its squared difference to the actual proportion $\hat{m} \in [0, 1]$ of active elements in our downsampling mask is included in the final loss function

$$\mathcal{L} = \alpha\mathcal{L}_1 + \beta(\gamma - \hat{m})^2, \qquad (3)$$

with α and β being weighting factors, and $\mathcal{L}_1$ denoting the loss of the main task, *e.g.*, the segmentation loss.

Limitations. Our approach allows to interpolate between different feature map resolutions, which can drastically improve the cost-accuracy trade-off as shown in Sec. 4. However, some limitations arise that could hinder its effective usage in some applications. First, our method's benefit depends on the given downsampling mask. If the mask does not capture important regions, retaining higher-resolution regions will not help. Second, there may be backbones, especially those with advanced layer aggregation schemes [43, 47], or datasets without fine details where higher-resolution feature maps and, therefore, also our method generally do not yield advantages. However, our analysis and related work [9, 46] show that the highly impactful and widely used CNN backbones, VGG [37] and ResNet [17], benefit from higher-resolution feature representations in various applications.

4. Experiments

To demonstrate the practicality of our proposed adaptive downsampling scheme, we conduct various experiments on two different computer vision tasks confirming the following points: *(1)* As a generalization of [46] and standard CNNs, our method allows to interpolate between different output strides, enabling a more granular control of how many resources should be allocated for the task and input image at hand. *(2)* By selecting task-relevant regions for a specific input, our method can drastically improve the cost-accuracy trade-off of standard CNNs. *(3)* The guarantees satisfied by our method allow to incorporate it in a plug-and-play fashion into pre-trained networks at test time. *(4)* Our method is not limited to a single application and finding appropriate downsampling masks is feasible in practice. *(5)* Our approach generalizes to different backbones, segmentation heads, and extensions.

We start our evaluation with an oracle experiment, demonstrating the potential of our method given a high-quality downsampling mask. Afterward, we present two case studies that show how our method can be used in practice, demonstrating the advantages summarized above.

Experimental setup (segmentation). Experiments for semantic segmentation (Secs. 4.1 and 4.2) have been conducted on the Cityscapes dataset [6]. We report the mean intersection over union (mIoU) on the validation set.

Experimental setup (keypoint description). For the keypoint description experiment (Sec. 4.3), we use the established D2-Net descriptor [9], which takes an image as input and outputs a dense feature map that can be used to locate and describe keypoints. Since D2-Net keypoint localization tends to be imprecise [42], we follow Uzpak *et al.* [42] and instead use SIFT [27] to first detect the 512 most salient keypoints and employ D2-Net only to describe them. Following common practice [28, 32], we report the mean matching accuracy, *i.e.*, the ratio between correct and possible matches, at a three-pixel threshold (MMA@3) on the HPatches dataset [1].

Experimental setup (general). As the focus of this work is the cost-accuracy trade-off of the examined methods, for all experiments we report the respective task-specific metrics, *i.e.*, mIoU and MMA@3, over the required theoretical (average) number of multiply-adds as a proxy of computational cost. As baselines, we consider the original models with varying output strides obtained by using regular downsampling [17, 37] with the standard factor $d = 2$, respectively *not* downsampling and using dilated convolutions [46]. To draw a more conclusive picture, we also report the required inference time in seconds. However, the inference time depends on the used implementation and hardware – here an Nvidia RTX A6000 GPU. For a fair comparison we use the same implementation for our
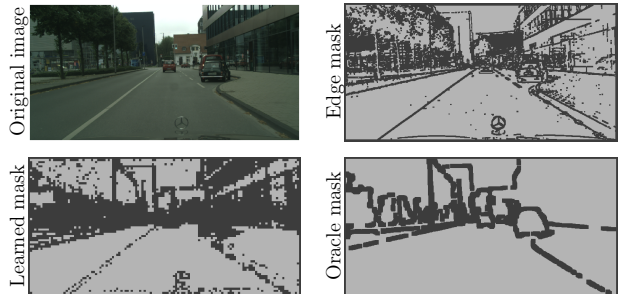

Figure 3. *Example downsampling masks.* A Cityscapes [6] image and the corresponding estimated downsampling masks using our oracle setup, edge detection, and a shallow learned network. Note how low-frequency regions, *e.g.*, the sky and road, belong to the same class, and thus, can be processed at low resolution (bright) to decrease the computational cost.

method and the baselines; please refer to the supplementary for different implementations.

4.1. Oracle experiment: Feature resolution in semantic segmentation

In our first experiment, we examine the role of feature map resolution in semantic segmentation and whether processing more informative regions at a higher resolution offers advantages and is generally possible. To this end, we first conduct an oracle experiment in which we assume a high-quality downsampling mask to be given. As baselines we use a segmentation model with regular downsampling and two different output strides (OS) of 16 (lower feature resolution) and 8 (higher feature resolution). For our proposed adaptive downsampling with output stride 8 to 32 (OS=8→32), we define the downsampling mask as the pixels that are misclassified by a regular model with OS=32 but correctly classified by a model with OS=8, followed by a dilation with a square kernel size of $K \times K$ (see Fig. 3 and supplementary). This gives us exactly the image regions that benefit from higher resolutions, and thus, can be considered our oracle downsampling mask. To demonstrate the generalizability of our approach, we investigate a variety of different backbones, segmentation heads, and extensions.

Results for our adaptive downsampling and baselines can be seen in Tab. 1. Confirming our assumption and prior work [4, 46], a smaller output stride, respectively higher resolution, increases both the mIoU and computational cost of all the examined baseline models (OS=8 *vs.* OS=16). Moreover, using our adaptive downsampling leads to an mIoU that is on par with the respective regular model with OS=8 while for some images requiring less than 50% of the time and computational cost (*e.g.*, ResNet-101+DeepLabv3). This clearly confirms our hypothesis that only a fraction of the feature map must be processed at a higher resolution to obtain strong predictive performance.
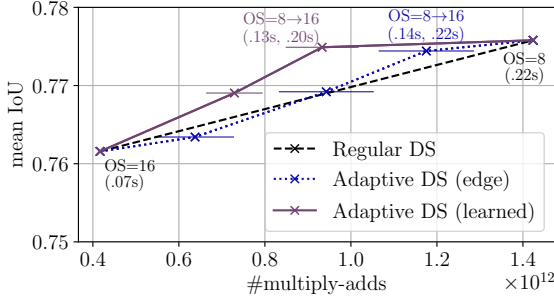
6

Figure 4. *Semantic segmentation results.* mIoU on the Cityscapes [6] evaluation set over the number of multiply-adds for a ResNet-101 [17] backbone with a DeepLabv3 [3] segmentation head, using regular downsampling [46] and our novel adaptive downsampling scheme with a learned mask, respectively an edge detection mask, with varying hyperparameters. Min. and max. inference times for each setup are annotated. To reduce variance, we report the maximum mIoU over three runs. X-confidence intervals show 2 times the standard deviation of the required multiply-adds.

Further, it shows that our proposed method is applicable to various different ResNet backbones (VGG [37] is shown in Sec. 4.3) and segmentation heads. Also, our approach is a novel and orthogonal research direction, which can be used *together* with existing extensions like deformable convolution [7] and advanced pooling strategies [38].

4.2. Case study 1: Semantic segmentation

In our first case study, we again consider semantic segmentation with ResNet101+DeepLabv3 models but now with realistically estimated downsampling masks. We propose two strategies to estimate useful downsampling masks:

Edge mask. From analyzing the problem at hand, we know that segmentation boundaries are likely to align with edges. For this reason, we perform edge detection on the input image to create a downsampling mask. Specifically, we apply a Sobel filter to the original grayscale images and dilate the thresholded result with a square kernel of size 11×11. Computational cost for detecting the edges is neglectable ($3 \times 10^{-3}\%$ of the total multiply-adds).

Learned mask. As described in Sec. 3, we use a shallow CNN to estimate downsampling masks from an intermediate layer of the feature extractor. The mask estimator is trained end-to-end with the segmentation model (see supplementary). The computational cost for the mask estimator is included in the reported multiply-adds and times.

The resulting downsampling masks for a single example image can be seen in Fig. 3. Dark areas denote keeping the higher resolution of an output stride of 8 while bright areas correspond to an output stride of 16 (OS=8→16). In Fig. 4, we report results for our adaptive downsampling with the two proposed strategies. We again observe that by "focusing" on more relevant image regions, *e.g.*, edges, we
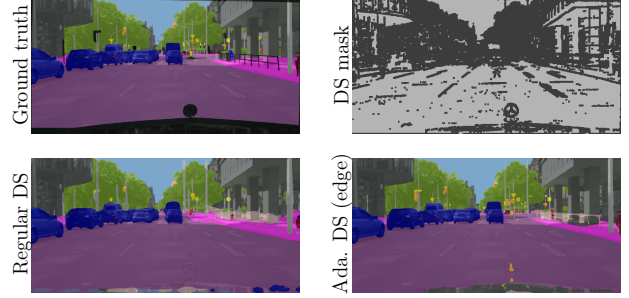

Figure 5. *Qualitative semantic segmentation results.* Estimated segmentation maps using regular downsampling with OS=8 and adaptive downsampling with an output stride of 8 to 16 (OS=8→16) and our edge downsampling (DS) mask.

can improve the cost-accuracy trade-off of an established model. Further, by adjusting the hyperparameters of the mask estimators, we can granularly control the amount of invested resources. Remarkably, adaptive downsampling with a learned mask achieves on par mIoU as regular downsampling with an output stride of 8 (0.775 vs. 0.776) while reducing the required computational cost by approximately 35%. Looking at the annotated minimum and maximum times, we can observe time improvements of up to 40%. However, we observe that for images with a lot of "interesting" content, the worst-case inference time is similar to regular downsampling with OS=8. This case study also demonstrates how naively and easily we can estimate downsampling masks that together with our adaptive downsampling still yield advantages for the task at hand. A visual example of the estimated segmentations is shown in Fig. 5.

Contrary to our work, existing methods that adaptively downsample the input *before* passing it into a CNN [21, 29] lose important image information, resulting in lower mIoU scores of 0.5 or 0.65 on Cityscapes [6]. Hence, they are mainly beneficial for processing ultra-high resolution images and are not considered as competitive baselines here.

4.3. Case study 2: Keypoint description

In our second realistic case study, we consider the problem of keypoint description. To this end, we investigate the established SIFT detector [27] and D2-Net descriptor [9] as described in the experimental setup. D2-Net utilizes a VGG16 [37] backbone that progressively applies max pooling to reduce the resolution of the feature map, and thus, computational load. To demonstrate the simplicity and versatility of our method, we use the original model weights provided by the authors [9] and do *not* perform additional training for any of the following setups. Again confirming prior work [9], in Fig. 6 we see that a higher feature map resolution (*i.e.*, lower OS) increases both the MMA@3 and the computational cost of the baseline model.

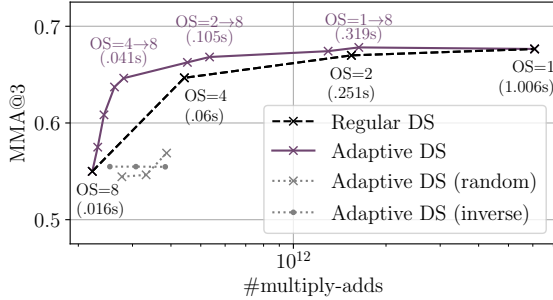Next, we substitute the last 3, 2, or 1 max pooling op-

Figure 6. *Keypoint description results.* MMA@3 over the required number of multiply-adds for a SIFT [27] detector with a pre-trained D2-Net [9] descriptor using regular downsampling [46] with different output strides (OS), and our novel adaptive downsampling scheme using different masks. Given a reasonable downsampling mask, our adaptive downsampling can achieve on par MMA@3 while significantly reducing the computational cost. Actual inference times for a randomly chosen image are annotated.
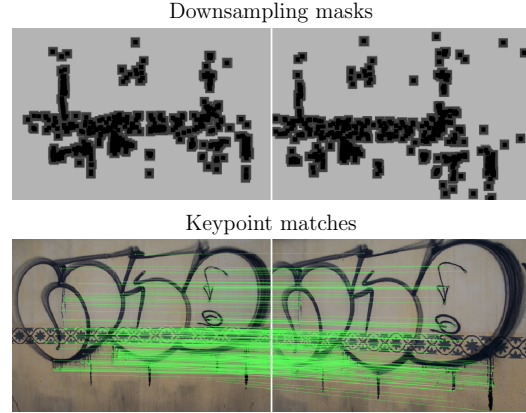

Figure 7. Qualitative results of keypoints matched with our adaptive downsampling scheme and the corresponding downsampling masks. Note how only a fraction of the image is important, and thus, processed at high resolution (dark) to increase the accuracy while saving resources in bright areas of the downsampling mask.

erations with our adaptive downsampling with max pooling (OS=$\{1, 2, 4\} \rightarrow 8$). As only image regions around the keypoints are important for our task at hand, we estimate the downsampling masks by dilating keypoints obtained from SIFT with filters of increasing sizes for each of the up to three downsampling levels (see Fig. 7). We demonstrate a fine-grained control of computational cost by varying the dilation sizes for the downsampling levels. Larger dilation sizes will lead to larger areas being processed at higher resolution, and thus, increase cost as well as MMA@3. The results in Fig. 6 clearly show that, compared to the fixed output strides of the baseline model, our adaptive downsampling yields a significant reduction of the computational cost while achieving the same MMA@3, *e.g.*, we achieve an ∼70% reduction of the computational cost to reach an MMA@3 that is on par with a regular output stride of 1.

Inadequate masks. To evaluate how our method performs with "bad" downsampling masks, in Fig. 6 we additionally report the MMA@3 for adaptive downsampling with an output stride of 4 to 8 using random and unreasonable masks, *i.e.*, the inverse of reasonable masks. Confirming our stated limitations, we see that inadequate downsampling masks do not capture important image regions, and thus, we process unimportant regions at high resolution, leading to an increased computational cost compared to regular downsampling with OS=8 without yielding significant improvements in predictive performance. Note, however, that thanks to our guarantees in Sec. 3, poor downsampling masks still yield comparable predictive performance as regular downsampling with OS=8, showing that these masks do not negatively affect predictive performance and that the model still behaves in an expected way.

5. Conclusion and Discussion

In this work, we propose – to the best of our knowledge – the first downsampling scheme that changes the operating resolution *within* CNNs in a *locally adaptive* fashion. We do so by generalizing standard CNNs [17, 22, 37] and Yu *et al.*'s [46] dilated convolutional networks, allowing us to process feature maps with spatially-varying resolutions. By selecting an appropriate content-adaptive downsampling mask, indicating locally the most important regions that are to be processed at a higher resolution, we can enable CNNs to "focus" more strongly on task-relevant regions. Besides substantially improving the cost-accuracy trade-off in two computer vision tasks, our novel adaptive downsampling enables a more continuous control of the invested computational resources, giving practitioners another degree of freedom to best adapt their model to the available resources. Thanks to carefully designing the proposed method, it satisfies two important guarantees, allowing adaptive downsampling even to be used at test time in a plug-and-play fashion within standard CNNs pre-trained with *regular* downsampling. As our approach improves the cost-accuracy trade-off of various established models, it contributes to saving valuable scarce resources and to the important research direction of more (energy) efficient deep learning [40].

References

[1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, pages 3852–3861, 2017. 6, iii

[2] Merit Bruckmaier, Ilias Tachtsidis, Phong Phan, and Nilli Lavie. Attention and capacity limits in perception: A cellular metabolism account. *Journal of Neuroscience*, 40(35):6801–6811, 2020. 1

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587 [cs.CV]*, 2017. 5, 7, i, ii

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 1, 5, 6, i

[5] Mengzhao Chen, Mingbao Lin, Ke Li, Yongkui Shen, Wu Yongjian, Fei Chao, and Rongrong Ji. CF-ViT: A general coarse-to-fine method for vision transformer. In *AAAI*, 2023. 2

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016. 5, 6, 7, i

[7] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017. 2, 5, 7, i

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[9] Mihai Dusmanu, Ignacio Rocco, Tomás Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A trainable CNN for joint description and detection of local features. In *CVPR*, pages 8092–8101, 2019. 5, 6, 7, 8, ii

[10] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks. In *ICRA*, pages 1355–1361, 2017. 3, 4

[11] Ziteng Gao, Limin Wang, and Gangshan Wu. LIP: Local importance-based pooling. In *ICCV*, pages 3354–3363, 2019. 2

[12] Benjamin Graham. Spatially-sparse convolutional neural networks. *arXiv:1409.6070 [cs.CV]*, 2014. 3

[13] Ben Graham. Sparse 3D convolutional neural networks. In *BMVC*, pages 150.1–150.9, 2015. 3, 4

[14] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, pages 9224–9232, 2018. 2, 3, 4

[15] Caglar Gulcehre, Kyunghyun Cho, Razvan Pascanu, and Yoshua Bengio. Learned-norm pooling for deep feedforward and recurrent neural networks. In *ECML PKDD*, pages 530–546, 2014. 2, 3

[16] Bharath Hariharan, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 3, 5, 6, 7, 8, i, ii

[18] Robin Hesse, Simone Schaub-Meyer, and Stefan Roth. Fast axiomatic attribution for neural networks. In *NeurIPS*2021*, pages 19513–19524. 2

[19] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NIPS*2015*, pages 2017–2025. 2

[20] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017. 5, ii

[21] Chen Jin, Ryutaro Tanno, Thomy Mertzanidou, Eleftheria Panagiotaki, and Daniel C. Alexander. Learning to downsample for segmentation of ultra-high resolution images. In *ICLR*, 2020. 2, 7

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*2012*, pages 1106–1114. 1, 8

[23] Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten ZIP code recognition. *Neural Comput.*, 1(4):541–551, 1989. 1

[24] Chen-Yu Lee, Patrick W. Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, pages 464–472, 2016. 2

[25] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 936–944, 2017. 3

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 3, 5, i

[27] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004. 6, 7, 8

[28] Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. ASLFeat: Learning local features of accurate shape and localization. In *CVPR*, pages 6589–6598, 2020. 6

[29] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam S. Tsai, Fei Yang, and Yuri Boykov. Efficient segmentation: Learning downsampling near semantic boundaries. In *ICCV*, pages 2131–2141, 2019. 2, 7

[30] Pedro Oliveira Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *ECCV*, volume 1, pages 75–91, 2016. 3

[31] Adrià Recasens, Petr Kellnhofer, Simon Stent, Wojciech Matusik, and Antonio Torralba. Learning to zoom: A saliency-based sampling layer for neural networks. In *ECCV*, volume 9, pages 52–67, 2018. 2

[32] Jérôme Revaud, César Roberto de Souza, Martin Humenberger, and Philippe Weinzaepfel. R2D2: Reliable and repeatable detector and descriptor. In *NeurIPS*2019*, pages 12405–12415. 6

[33] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *CVPR*, pages 6620–6629, 2017. 3, 4

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015. 3

[35] Faraz Saeedan, Nicolas Weber, Michael Goesele, and Stefan Roth. Detail-preserving pooling in deep networks. In *CVPR*, pages 9108–9116, 2018. 2, 3

[36] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*, 2014. 2

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 5, 6, 7, 8, i

[38] Alexandros Stergiou, Ronald Poppe, and Grigorios Kalliatakis. Refining activation downsampling with SoftPool. In *ICCV*, pages 10337–10346, 2021. 2, 5, 7, i

[39] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *ICML*, pages 3319–3328, 2017. 2

[40] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE*, 105(12):2295–2329, 2017. 8

[41] Hossein Talebi and Peyman Milanfar. Learning to resize images for computer vision tasks. In *ICCV*, pages 487–496, 2021. 2

[42] Ali Uzpak, Abdelaziz Djelouah, and Simone Schaub-Meyer. Style transfer for keypoint matching under adverse conditions. In *3DV*, pages 1089–1097, 2020. 6

[43] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(10):3349–3364, 2021. 3, 5

[44] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. Mixed pooling for convolutional neural networks. In *RSKT*, pages 364–375, 2014. 2, 3

[45] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 1, 3

[46] Fisher Yu, Vladlen Koltun, and Thomas A. Funkhouser. Dilated residual networks. In *CVPR*, pages 636–644, 2017. 1, 2, 3, 4, 5, 6, 7, 8, i

[47] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *CVPR*, pages 2403–2412, 2018. 3, 5
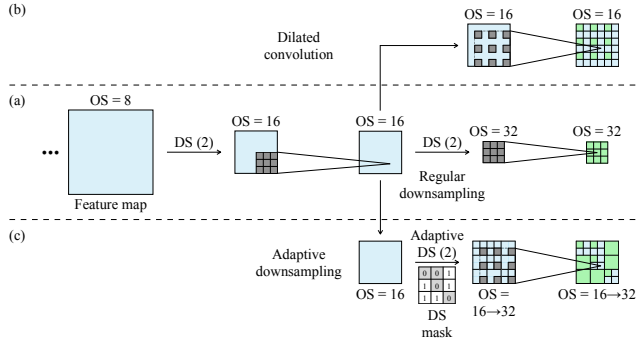
Figure 8. *Illustration of different methods to handle the output strides (OS) in CNNs.* (a) A standard convolutional network progressively downsamples the feature map. The illustrated convolutions, followed by uniform downsampling (DS), correspond conceptionally to strided convolutions with a stride of 2. (b) The last downsampling operation is substituted with a dilated convolution to retain the same receptive field while increasing the feature map resolution [46]. (c) The last downsampling operation is substituted with our novel adaptive downsampling and the dilation is increased to retain the same receptive field. Our approach retains high feature map resolution only where needed.

A. Overview

This appendix provides additional explanations and experimental details for reproducibility purposes, which could not be included in the main text due to space limitations.

B. Addendum to Extension to CNNs

In Sec. 3 of the main paper, we briefly discussed how our method can be applied multiple times in succession to extend it to CNNs. For a more detailed description and illustration of this process, please refer to Fig. 9.

C. Experimental Details

In the following, we provide additional details to ensure reproducibility. All experiments have been conducted using Python, PyTorch 1.11.0 [48], and a single NVIDIA RTX A6000 (48GB) GPU or a single NVIDIA A100-SXM4 (40GB) GPU.

Implementation of different output strides. In all experiments, we examine backbone models with varying output strides. Fig. 8 illustrates how these different output strides (OS) are realized. Fig. 8(a) shows a standard backbone using regular downsampling [17, 37] with the standard downsampling factor $d=2$. Part (b) shows how the backbone can be modified to retain a lower output stride, *i.e.*, higher resolution, by using dilated convolution [46]. Alternatively, it can be modified to contain multiple output strides using our novel adaptive downsampling scheme as shown in (c).

C.1. Oracle experiment: Feature resolution in semantic segmentation

The experiments described in Sec. 4.1 of the main paper have been conducted on the widely used Cityscapes dataset [6] (freely available to academic and non-academic entities for non-commercial purposes), containing 5 000 annotated street scene images. Our code is built on top of the publicly available `DeepLabV3Plus-Pytorch` GitHub repository (MIT License).[1]

We examine different ResNet [17] backbones (50, 101, and 152 layers), three established segmentation models (FCN [26], DeepLabv3 [3], and DeepLabv3+ [4]), and two extensions, *i.e.* Softpool [38] and deformable convolution [7]. Following common practice, we initialize the backbones with weights obtained from pre-training on the ImageNet dataset [49]; they are publicly available in PyTorch. We then fine-tune models with regular output stride on the official Cityscapes [6] training split for 30k iterations using a learning rate of 0.01 for the backbone, a learning rate of 0.1 for the classifier, a batch size of 8, and an SGD optimizer with a weight decay of $1e-4$ and a momentum of 0.9. For the final models with regular and adaptive downsampling, we fine-tune above models for another 10k iterations with learning rates multiplied by 0.1. The downsampling mask dilation sizes K for ResNet-{50, 101, 152} with DeepLabv3 are 33, 33, and 55, respectively. For ResNet-101 with FCN, respectively DeepLabv3+, K is chosen as 33. Finally, for ResNet-101 with DeepLabv3 and the extensions deformable convolution and SoftPool, we use $K = 55$. SoftPool is used as in the original paper [38]. Deformable convolution is only used in the last block of the ResNet backbone and offsets are estimated using a single convolutional layer with a kernel size of 3×3 and without a bias.

To measure the theoretical number of multiply-adds, we use the publicly available code in the `flops-counter.pytorch` GitHub repository (MIT License).[2] To estimate the runtime in seconds, we use 10 warup iterations, followed by another 10 iterations where we measure the average inference time of the backbone. For a fair comparison between equally optimized implementations, we use the same implementation for our proposed method and the baselines in the main paper. However, in practice, over the years there have been major efforts to highly optimize standard convolution in PyTorch. To put our results into context with these optimizations, in Tab. 2 we report inference times for our method, and baseline methods with the default PyTorch implementations. As even the PyTorch version has major impact on the inference time of the baseline models, we report results for two ver-

---

[1] https://github.com/VainF/DeepLabV3Plus-Pytorch
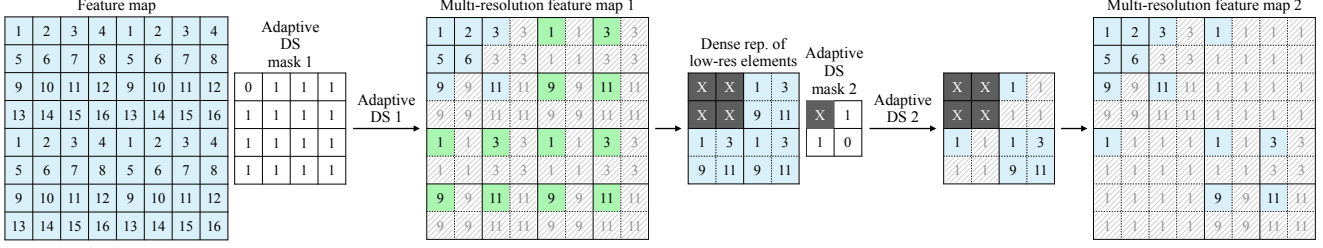[2] https://github.com/sovrasov/flops-counter.pytorch

Figure 9. *Illustration of our adaptive downsampling applied two times in succession with a downsampling factor of $d = 2$.* Starting on the left, adaptive downsampling is applied a first time to a regular *feature map* to generate *multi-resolution feature map 1*. To perform adaptive downsampling a second ($k = 2$) time, we only consider the $d^k \times d^k$ patches of *multi-resolution feature map 1* that have previously been completely downsampled, *i.e.*, that are of lower effective resolution (green cells). The active elements of these patches are projected back to their regular, dense representation (*dense rep. of low-res elements*) to apply adaptive downsampling a second time ('X' denotes areas that cannot be represented at the current resolution as they have not been downsampled). The elements of the resulting feature map are then again projected onto their corresponding location to generate *multi-resolution feature map 2*.

Table 2. *Inference times for different implementations.* We consider the ResNet-101+DeepLabv3 [3, 17] setup from the oracle experiment. We report mIoU, theoretical number of multiply-adds (#MA), and min./max. inference times for our and the default PyTorch implementation with two different PyTorch versions.

OS	Impl.	PyTorch version	#MA	Time(s)	mIoU
16	*Ours*	1.7.1	$4.2e11$	0.08	0.7591
16	*Ours*	1.11.0	$4.2e11$	0.07	0.7591
16	*Default*	1.7.1	$4.2e11$	0.08	0.7591
16	*Default*	1.11.0	$4.2e11$	0.05	0.7591
8	*Ours*	1.7.1	$1.4e12$	0.26	0.7775
8	*Ours*	1.11.0	$1.4e12$	0.22	0.7775
8	*Default*	1.7.1	$1.4e12$	0.24	0.7775
8	*Default*	1.11.0	$1.4e12$	0.14	0.7775
8→32	*Ours*	1.7.1	$6.7e11$	0.09/0.18	0.7748
8→32	*Ours*	1.11.0	$6.7e11$	0.08/0.15	0.7748

sions, *i.e.*, 1.7.1 and 1.11.0.

Given a reasonable downsampling mask, our proposed method (OS=8→32) and implementation can outperform the default PyTorch baseline with an output stride of 8 (similar mIoU and 0.08s *vs*. 0.14s). For PyTorch 1.7.1, our implementation is even similarly efficient as the default implementation for the regular output strides (0.08s for OS=16 and 0.26 *vs*. 0.24 for OS=8). With version 1.11.0, PyTorch became more optimized, and thus, the gap to our implementation increases for the regular output strides (0.07s *vs*. 0.05s for OS=16 and 0.22s *vs*. 0.14s for OS=8). While we are not aware of the nature of the internal optimizations of PyTorch, it is reasonable to assume that similar optimizations could be applied to our implementation.

C.2. Case study 1: Semantic segmentation

The experimental setup in Sec. 4.2 of the main paper follows the same setup as in Appendix C.1. All models are

trained from scratch for 30k iterations. We investigate a ResNet-101 [17] backbone with the DeepLabv3 [3] segmentation model for OS=16 and OS=8. The runtime and theoretical number of multiply-adds are also measured as in Appendix C.1.

The downsampling masks for the edge detection setup are estimated by applying a Sobel filter to the original image, thresholding the result at 0.95, 0.35, respectively 0.15 to control the amount of invested resources, and dilating the result with a square dilation kernel of size 11×11.

To estimate the learned downsampling mask, we use a shallow CNN of 4 layers with a kernel size of 3×3, with 128, 64, 64, and 2 channels, with ReLU activation functions, and max pool with a stride of 2 after the first two layers. The output is fed into a Gumbel-Softmax [20] to obtain the final discretized downsampling mask. We train the mask estimator with a learning rate of 0.001 end-to-end together with the segmentation model. The two setups in Fig. 4 of the main paper are obtained by setting γ, *i.e.*, the fraction of active elements in the downsampling mask, to 0.5 and 0.7.

C.3. Case study 2: Keypoint description

For the keypoint description experiment in Sec. 4.3 of the main paper, we use the D2-Net descriptor [9]. Our code is based on the official, publicly available D2-Net [9] implementation (Clear BSD License).[3] We initialize our models with pre-trained weights that are also available in the official D2-Net [9] repository and do not train or fine-tune for any of the examined setups. We estimate the downsampling masks by dilating keypoints obtained from SIFT with filters of varying sizes for each downsampling level. To sample points with different computational complexity in Fig. 6 of the main paper, we use dilation sizes of – from least to most multiply-adds – (0,0,0), (0,0,5),

---

[3] https://github.com/mihaidusmanu/d2-net

(0,0,11), (0,0,21), (0,0,31), (0,19,35), (0,27,37), (19,41,41), (25,51,51), (∞,∞,∞). Here, the entries in each triplet correspond to the respective scale they operate on, *i.e.*, the first entry is the dilation size for the mask that retains elements at a resolution of OS=1, the second for retaining elements at a resolution of OS=2, and the third for retaining elements at a resolution of OS=4. An entry of 0 means that the entire feature map is downsampled while an entry of ∞ indicates that the resolution of the entire feature map is retained. As such, (0,0,0) corresponds to the regular output stride of 8, while (∞,∞,∞) corresponds to the regular output stride of 1. The used HPatches dataset [1] (MIT License), to the best of our knowledge, contains no personally identifiable information except for one sequence of an image of a prominent person that is already publicly available and not offensive.

References

[48] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. i

[49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(13):211–252, 2015. i