

---

# Accelerating Large-Kernel Convolution Using Summed-Area Tables

---

Linguang Zhang    Maciej Halber    Szymon Rusinkiewicz  
Department of Computer Science  
Princeton University

## Abstract

Expanding the receptive field to capture large-scale context is key to obtaining good performance in dense prediction tasks, such as human pose estimation. While many state-of-the-art fully-convolutional architectures enlarge the receptive field by reducing resolution using strided convolution or pooling layers, the most straightforward strategy is adopting large filters. This, however, is costly because of the quadratic increase in the number of parameters and multiply-add operations. In this work, we explore using learnable *box filters* to allow for convolution with arbitrarily large kernel size, while keeping the number of parameters per filter constant. In addition, we use precomputed *summed-area tables* to make the computational cost of convolution independent of the filter size. We adapt and incorporate the box filter as a differentiable module in a fully-convolutional neural network, and demonstrate its competitive performance on popular benchmarks for the task of human pose estimation.

## 1 Introduction

Fully-convolutional neural networks have seen success in numerous dense prediction tasks since Long et al. [19] adapted architectures that were originally used for image classification. In many cases, achieving high performance requires a large receptive field, and recent network architectures such as the popular ResNet family [10] use a large number of  $3 \times 3$  convolution layers coupled with downsampling to capture large-scale contextual information. While this strategy does increase the receptive field, the use of downsampling prevents the network from generating high-resolution output. Omitting downsampling while still only relying on  $3 \times 3$  convolution is generally infeasible, since the network would need to be significantly deeper to achieve the same receptive field.

Previous work has proposed many ways to deal with this inherent conflict between increasing the receptive field and providing high-resolution output. An immediate solution is to add deconvolution (transposed convolution) layers to upsample the results. Xiao et al. [32] demonstrate that appending deconvolution layers to a ResNet backbone leads to a straightforward solution for restoring the resolution. Similarly, encoder-decoder and U-shaped networks, which are widely used for dense prediction tasks [21, 25], also rely on deconvolution to produce high-resolution predictions.

Because deconvolution adds significant complexity, an alternative is simply to expand the filter size. Naive implementation of this strategy, however, leads to quadratic growth in both the number of operations and the number of parameters, leading to slower computation and greater susceptibility to over-fitting. Dilated convolution [34, 6] uses filters that effectively have a larger size, but utilize zero-padding to get by with fewer operations and parameters. The drawback of dilated convolution, however, is the presence of “gridding” artifacts that can degrade performance in some applications.

In this paper, we explore large-kernel convolution using a classical approach that nevertheless has had limited application in the context of neural networks. Specifically, we exploit *Summed-Area Tables* (SATs), also known as *integral images*, which enable the integral of an arbitrarily-sized rectangular region to be computed in constant time [17]. SATs are therefore ideal for convolving an image

Preprint. Under review.

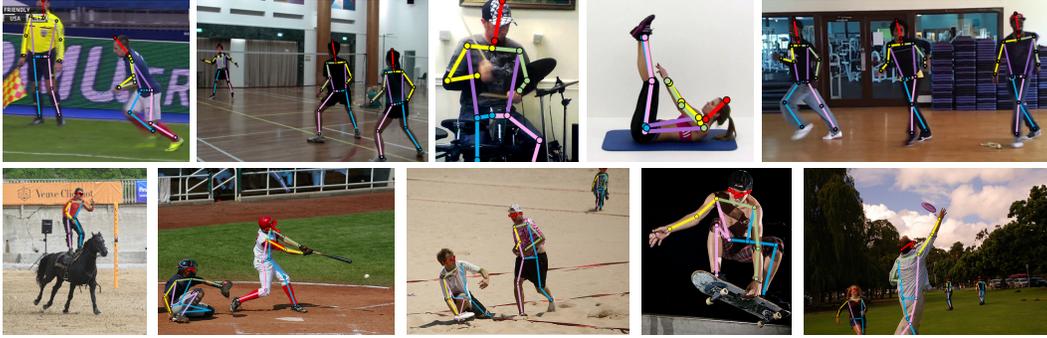


Figure 1: Qualitative results for a dense prediction task — human pose estimation — implemented using the proposed method. **Top:** Results on the test set of MPII Human Pose. **Bottom:** Results on the test-dev2017 set of Microsoft COCO. The proposed method is able to deal with a variety of cases, including multiple people at different scales, self occlusions, and non-standard poses.

with a box filter, since each output pixel requires a constant number of operations. Also, because a box filter can be parameterized using only four variables — the  $(x, y)$  location and size of the box in the kernel — the number of parameters for a box filter is independent of the kernel size. The resulting efficiency, in terms of both computational cost and number of parameters, makes box filters well-suited for tasks that require both a large receptive field and high-resolution output (see Figure 1).

One possible drawback of box filters is the difficulty of detecting complex spatial patterns. In the context of neural networks, this issue is alleviated, as the intermediate feature maps often contain many channels. Almost any complex kernel can be well approximated by combining a large and diverse collection of box filters with appropriate weights. This observation implies that one can first convolve the image with a collection of box filters, then linearly combine the filtered images with different weights, which approximates convolving the image with various complex kernels. We show how to achieve this with depth-wise convolution followed by  $1 \times 1$  convolution.

Our main contribution is a lightweight, fully-convolutional network that uses SATs and box filters to perform large-kernel convolution, efficiently combining high-resolution output with wide receptive fields for pixel-level prediction tasks. To enable this, we show how to implement box convolution in an end-to-end differentiable setting, in which the gradient of a box filter with respect to its parameters (position and size) must be continuous. The resulting formulation uses sub-pixel positioning for all four corners of the box, and is computationally equivalent to conventional  $4 \times 4$  convolution.

The specific task on which we demonstrate our results is human pose estimation, which can be posed as a high-level keypoint detection problem [18]. This choice is motivated by the effectiveness of SATs in tasks such as local feature detection [2] and face detection [31]. We show that the proposed architecture can achieve competitive results on popular benchmarks [1, 18] with only 1.85M parameters (an order of magnitude less than previous methods) and at a lower computational cost. We also demonstrate that our network generalizes better when the training data is limited. To summarize, our contribution is three-fold:

- Deriving a generalizable solution to obtaining gradients for simple kernels that leverage summed-area tables for acceleration.
- Designing a lightweight fully-convolutional network that produces pixel-level prediction with a large receptive field.
- Demonstrating competitive performance on human pose estimation, with improved generalization when trained on limited data.

## 2 Related Work

**Strategies to Enlarge Receptive Field** For visual recognition systems based on convolutional neural networks, having a sufficiently large receptive field to aggregate spatially distant information is critical. For example, image classification networks [8, 16, 10] typically rely on downsampling (e.g., pooling and strided convolution) interleaved with stacks of small kernels, exploiting the fact that downsampling essentially “amplifies” the receptive field of *every* following kernel.

This downsampling, while generally acceptable for tasks such as classification that produce sparse or “global” output, is inappropriate for dense prediction tasks such as semantic segmentation or heatmap-based human pose estimation. To produce pixel-level predictions, a straightforward strategy is to couple downsampling with upsampling (e.g., deconvolution). U-net and stacked-hourglass architectures [25, 21] adopt such a downsampling-upsampling strategy, together with skip connections to retrieve high-frequency details from early stages. Cascaded Pyramid Networks [7] obtain pixel-level predictions by fusing feature maps of different resolutions, which are produced by a progressive downsampling backbone network (e.g., ResNet [10]).

Avoiding downsampling altogether, while maintaining a large receptive field and computational efficiency, can be accomplished using an approach such as dilated convolution [34, 6]. This technique inserts zeros between kernel elements, which produces the benefits of large-kernel convolution while unfortunately also resulting in gridding artifacts. The differentiable box filter used in this work is another instance of large-kernel convolution. Compared to dilated convolution, which only sparsely utilizes pixels within the kernel, a box filter can leverage more pixels and produce a smoother result, depending on the size of the learned box.

**Summed-Area Tables in Computer Vision** Accelerating convolution with simple kernels using summed-area tables has a long history in computer vision. A classic application that popularized SATs is the efficient face detection method by Viola and Jones [31]. Blob detection can also leverage summed-area tables for efficient image filtering when the Laplacian of Gaussian (LoG) is approximated with box filters [2]. Trzcinski and Lepetit [30] showed that complex, large kernels can be approximated with a few box filters, speeding up linear projection.

These ideas naturally lead to the concept of SAT-accelerated box filters as first-class elements in deep convolutional networks, and this was first explored in the work of Burkov and Lempitsky [3]. The authors show that box filters can replace dilated convolution, leading to improved performance in the context of two existing lightweight networks for semantic segmentation [22, 24]. However, that work has several drawbacks that are not shared by our (independently-developed) implementation. First and most importantly, the implementation of Burkov and Lempitsky leads to discontinuities with respect to filter size and position, caused by a failure to perform sub-pixel sampling from the SAT at box corners. In contrast, we perform correct sub-pixel sampling in both the forward and back-propagation passes, and demonstrate that this strategy easily generalizes to kernels other than single boxes. Second, the size of each box in their implementation can be larger than the image itself, and regularization is used to encourage the boxes to shrink. We avoid regularization of box size, which can bias the training process, and instead impose a maximum-size constraint that may be considered analogous to the kernel size in standard convolution. Third, they observe that the learned boxes are unintuitively symmetric with respect to the vertical axis, and that this is not a consequence of data augmentation. We have not observed the same phenomenon, suggesting that our correct sub-pixel sampling in both the forward pass and gradient estimation introduces less bias.

## 3 Approach

### 3.1 Fast Convolution with Summed-Area Tables

The summed-area table can be used to accelerate image convolution with kernels that only involve rectangular summations. For simplicity, we begin by considering how to efficiently convolve an image (a single-channel feature map) with the simplest box filter. We imagine a filter kernel of maximum size  $k$  that is zero everywhere except for a rectangular sub-region (i.e., box) filled with ones. The extent of the box is specified by four integers  $x_l, x_h, y_l, y_h \in [0, k)$ , and we define the filter  $g$  as:

$$g_{i,j} = \begin{cases} 1 & x_l \leq i \leq x_h \text{ and } y_l \leq j \leq y_h \\ 0 & \text{otherwise.} \end{cases}$$

The above filter can be used to perform general convolution. Denoting the input image as  $\mathcal{I}$  and the output image as  $\mathcal{O}$ , each pixel in the output image is computed as:

$$\mathcal{O}_{x,y} = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \mathcal{I}_{x+i,y+j} g_{i,j} = \sum_{i=y_l}^{y_h} \sum_{j=x_l}^{x_h} \mathcal{I}_{x+i,y+j}. \quad (1)$$

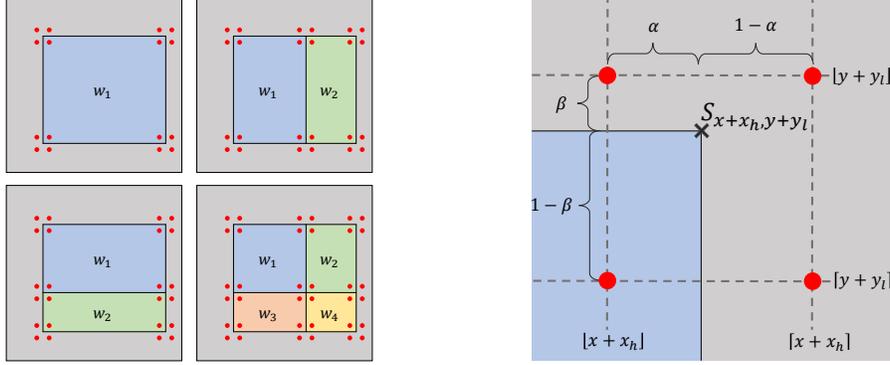


Figure 2: **Left:** a simple box filter, together with variants obtained through kernel splitting. Red dots indicate locations at which the SAT is sampled. **Right:** bilinear interpolation is performed at each corner, with the weights  $\alpha$  and  $\beta$  remaining constant over the course of a single convolution.

The above equation costs at most  $k^2$  multadd (multiply-add) operations to compute each output pixel. However, since every output pixel is the sum of a rectangular region, a precomputed summed-area table can be used to achieve constant-time summation at each output location. The value at location  $(x, y)$  in the summed-area table  $\mathcal{S}$  equals the sum of all pixels above and to the left of  $(x, y)$  in the input image (including  $\mathcal{S}_{x,y}$  itself). Therefore the sum of all pixels enclosed in the box  $[x + x_l, x + x_h] \times [y + y_l, y + y_h]$  can be efficiently computed by sampling at the four corners of the box in the summed-area table:

$$\mathcal{O}_{x,y} = \mathcal{S}_{x+x_h+1, y+y_h+1} + \mathcal{S}_{x+x_l, y+y_l} - \mathcal{S}_{x+x_l, y+y_h+1} - \mathcal{S}_{x+x_h+1, y+y_l}. \quad (2)$$

The summed-area table can be efficiently computed in a single pass over the input image on the CPU, or using row/column parallelization on a GPU. Note that the precomputation cost, relative to the cost of convolution, is quickly amortized when the box becomes larger.

We have so far assumed that the box is aligned with the input, which is a discrete lattice. In a neural network, we would like to make  $x_l, x_h, y_l, y_h$  learnable parameters instead of manually chosen integers, and the resulting continuous optimization naturally leads to non-integer coordinates. One could simply round the sampling points to the nearest integer-valued coordinates, but the rounding operator is unfortunately not differentiable. Burkov and Lempitsky [3] update parameters using approximate gradients derived through other means, such as normalizing the sum by the area of the box [3], but they do not directly address the discontinuity of sampling. We instead *interpolate* among the four nearest values in the SAT to accommodate non-integer coordinates. While any differentiable interpolation function could be used, we adopt bilinear interpolation in our implementation.

While this use of interpolation leads to a greater number of accesses to the SAT to compute the convolution, we note that the relative cost could be reduced by using more complex kernels. For example, consider the four variants illustrated in Figure 2, left. The use of “kernel splitting” allows the use of kernels in which the box is divided into, say, 2 or 4 pieces with different weights. Sampling from the SAT (illustrated by red dots) along an edge shared by two split boxes can be performed only once, saving parameters and computation.

Regardless of whether a single box or a split kernel is used, we need to compute the gradient of each convolved pixel with respect to box parameters and sampled pixels. Inspired by the Spatial Transformer [13], we use a differentiable interpolation function to obtain the (sub-)gradients when sampling from the SAT. The value sampled from the SAT at (possibly) non-integer-valued coordinates  $(\hat{x}, \hat{y})$  is computed as (see Figure 2, right):

$$\mathcal{S}_{\hat{x}, \hat{y}} = (1 - \alpha)(1 - \beta) \cdot \mathcal{S}_{\lfloor \hat{x} \rfloor, \lfloor \hat{y} \rfloor} + \alpha(1 - \beta) \cdot \mathcal{S}_{\lceil \hat{x} \rceil, \lfloor \hat{y} \rfloor} + (1 - \alpha)\beta \cdot \mathcal{S}_{\lfloor \hat{x} \rfloor, \lceil \hat{y} \rceil} + \alpha\beta \cdot \mathcal{S}_{\lceil \hat{x} \rceil, \lceil \hat{y} \rceil}, \quad (3)$$

where  $\lfloor \cdot \rfloor$  and  $\lceil \cdot \rceil$  are the floor and ceiling operators,  $\alpha = \hat{x} - \lfloor \hat{x} \rfloor$ , and  $\beta = \hat{y} - \lfloor \hat{y} \rfloor$ . The above equation is continuous and differentiable in the interpolation neighborhood, and the partial derivative of  $\mathcal{S}_{\hat{x}, \hat{y}}$  with respect to  $\hat{x}$  can be written as:

$$\frac{\partial \mathcal{S}_{\hat{x}, \hat{y}}}{\partial \hat{x}} = -(1 - \beta) \cdot \mathcal{S}_{\lfloor \hat{x} \rfloor, \lfloor \hat{y} \rfloor} + (1 - \beta) \cdot \mathcal{S}_{\lceil \hat{x} \rceil, \lfloor \hat{y} \rfloor} - \beta \cdot \mathcal{S}_{\lfloor \hat{x} \rfloor, \lceil \hat{y} \rceil} + \beta \cdot \mathcal{S}_{\lceil \hat{x} \rceil, \lceil \hat{y} \rceil}. \quad (4)$$

The partial derivative with respect to  $\hat{y}$  can be computed similarly. The partial derivative of  $\mathcal{S}_{\hat{x}, \hat{y}}$  with respect to each sampled pixel, for example,  $\frac{\partial \mathcal{S}_{\hat{x}, \hat{y}}}{\partial \mathcal{S}_{\lfloor \hat{x} \rfloor, \lfloor \hat{y} \rfloor}}$ , can be trivially computed as  $(1 - \alpha)(1 - \beta)$ .

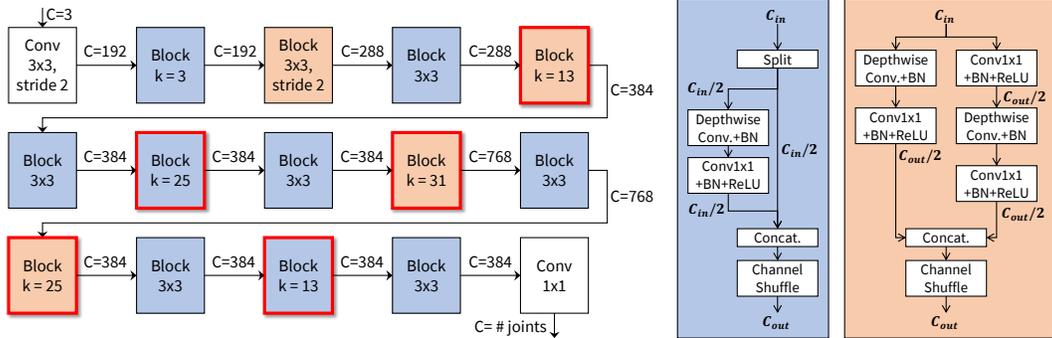


Figure 3: Our dense prediction network. We use blocks with box filters interleaved with blocks with regular  $3 \times 3$  kernels. The two types of blocks (colored differently) are shown on the right [20].

An important observation is that shifting  $\hat{x}$  or  $\hat{y}$  by an integer number of pixels does not change the value of  $\alpha$  or  $\beta$ . For instance,  $\alpha = \hat{x} - \lfloor \hat{x} \rfloor = \hat{x} + 1 - \lfloor \hat{x} + 1 \rfloor$ . This implies that we can precompute the weights for bilinear interpolation, which involve only  $\alpha$  and  $\beta$ , if we assume the most common scenario that the convolution is performed with an integer-valued stride (e.g., stride = 1). With precomputed interpolation weights, sampling a pixel from the SAT costs 4 multadds. Therefore, the total number of multadds per pixel spent on computing the region sum becomes 16 when adopting bilinear interpolation. In other words, since we sample 16 pixels and multiply with 16 precomputed weights, our differentiable box filter is *computationally equivalent* to a  $4 \times 4$  conventional convolution. The weights can also be precomputed when kernel splitting is applied: the weight of each split box can be combined with its interpolation weights.

### 3.2 Implementation Details

While a single box filter is simple, multiple box filters can be linearly combined to approximate more complex kernels [30]. To leverage this property, we incorporate the differentiable box filter into a depth-wise convolution layer in which each box filter is convolved with one channel of the input feature map. In many deep learning frameworks (e.g., Caffe [14] or PyTorch [23]), standard (non depth-wise) convolution is usually converted to matrix multiplication, which allows the use of well-optimized General Matrix Multiply (`gemm`) implementations. However, we have observed that implementing depth-wise convolution in this manner is inefficient, because laying out the patches as a matrix and invoking `gemm` introduces memory and computational overhead. We instead parallelize the computation of each output pixel directly, which is significantly faster in practice. Our CUDA implementation, wrapped as a standalone PyTorch layer, is included in the supplemental material.

We re-parameterize the four box coordinates  $x_l, x_h, y_l, y_h$  into the  $[-1, 1]$  range, relative to a maximum box size, and convert them back during forward propagation. We clip the parameters if they grow beyond  $[-1, 1]$ , and ensure that  $x_l \leq x_h$  and  $y_l \leq y_h$  after each iteration. The parameters are initialized uniformly in the range  $[-0.5, 0.5]$  to prevent frequent clipping at the beginning of training.

### 3.3 Dense Prediction Network

We incorporate box convolution into a network architecture (Figure 3) with two “building blocks” inspired by ShuffleNetV2 [20]. The blue “blocks” only convolve with half of the input channels and “shuffle” them with the other, unmodified half. This diversifies the effective receptive field across channels. The orange “blocks” are responsible for changing the number of feature channels. Within the architecture, we interleave regular  $3 \times 3$  convolutions with box convolutions of various maximum kernel sizes  $k$ . We employ more channels and larger kernels in the middle of the network, reducing the kernel size towards the end to produce a sharp prediction. The resolutions of the intermediate feature maps in our network are the same as the output — typically a quarter of the input resolution.

## 4 Experiments

We test our method on the task of human pose estimation (more specifically, estimating the coordinates of each joint). While it is possible to directly regress the joint locations [5, 29], most recent methods

Table 1: Comparisons on the MPII Human Pose dataset. “Pretrain” indicates that the backbone network is pretrained on the ImageNet classification task.

Method	Pretain	#Params	FLOPs	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	PCKh
8-stage hourglass [21]	N	25.1M	19.1G	98.2	96.3	91.2	87.1	90.1	87.4	83.6	90.9
SimpleBaseline [32]	N	34.0M	12.0G	98.0	95.3	89.1	83.9	88.3	83.7	79.1	88.7
SimpleBaseline [32]	Y	34.0M	12.0G	98.1	96.0	90.3	85.6	89.6	86.1	81.9	90.1
Dilated Convolution	N	1.88M	7.7G	97.7	94.6	88.8	83.5	87.4	82.3	77.7	88.0
3 × 3 Convolution	N	1.87M	7.6G	95.6	92.2	82.0	75.6	76.3	69.3	63.8	80.2
Ours	N	1.85M	7.7G	98.1	95.7	90.6	85.7	89.2	84.4	79.6	89.6

estimate a heatmap for each joint and pick the highest value as the keypoint [21, 27, 32, 12, 33, 26]. We train and evaluate our method on two datasets: MPII Human Pose [1] and Microsoft COCO [18]. To provide supervision for training, the target heatmap of each joint is generated by centering a Gaussian kernel with standard deviation equal to 2.0 at the ground-truth location.

#### 4.1 MPII Human Pose Dataset

The MPII Human Pose dataset [1] consists of around 25k images extracted from online videos. Each image contains one or more people, with over 40k people annotated in total. Among the 40k samples, ~28k samples are for training and the remainder are for testing. We use the training/validation split provided by Tompson et al. [28]; test annotations are not publicly available.

**Evaluation Metrics** The estimated joint locations are evaluated using the Percentage of Correct Keypoints, normalized by head size (PCKh). Specifically, an estimated joint location is regarded as correct if its distance from the ground-truth location is no larger than a constant threshold, normalized by 60% of the diagonal of the head bounding box. As is common practice, we report PCKh@0.5.

**Training and Testing** The location and scale of each person in the image is annotated, and we use it to crop the person and re-size the image to  $256 \times 256$ , which is a common practice. We augment the data via random rotation ( $\pm 30^\circ$ ), scaling ( $1.0 \pm 0.25$ ) and flipping. We train our network using the Adam optimizer [15], with the base learning rate set to  $10^{-3}$ . The network is trained with a batch size of 64 for 140 epochs. The learning rate is decreased by a factor of 10 at the 90th and 120th epochs. Following previous methods [32, 21, 7], we obtain the final heatmaps by averaging the outputs for the input image and its flipped version. The final joint estimation is computed as the location of the highest response, shifted towards the second-highest response by a quarter of the distance [32].

**Results** Table 1 shows results on the test set, while Figure 1, top row, shows qualitative results. SimpleBaseline refers to the publicly available model optimized using a pretrained ResNet50 [10] backbone (on the ImageNet [8] classification task). For a more realistic comparison, we also present results for a version of SimpleBaseline retrained from scratch. Our network, with SAT-accelerated box filters, offers comparable performance to stacked hourglass networks and SimpleBaseline, while using an order of magnitude fewer parameters and 1.5-2.5 times less computation.

In addition to comparing against recent well-performing methods, we perform two ablation studies. First, we replace all of the box filters in the network with conventional  $3 \times 3$  kernels to evaluate how significantly the receptive field influences performance. As shown in Table 1, second-to-bottom row, the larger receptive field is critical to good performance, even though each  $3 \times 3$  kernel has more learnable parameters compared to the box filter (9 vs. 4).

We also compare to dilated convolution (third-from-bottom row), utilizing parameters that match both the computational cost and receptive field of our box filters. (For example, a box filter of kernel size 13 costs the same and has the same receptive field as  $4 \times 4$  convolution with dilation factor 4.) Despite the significantly greater number of parameters available to dilated convolution (16 vs. 4), our box filters achieve higher performance on every joint, suggesting that SAT-accelerated box convolution may be an attractive replacement for dilated convolution. This is likely because each  $4 \times 4$  dilated convolution kernel essentially only utilizes 16 input pixels to produce each output pixel. In contrast, a box filter could leverage more input pixels if the learned box is large.

Evaluating the implementation by Burkov and Lempitsky [3] on the **test set** is unfortunately infeasible at the time of paper submission. In the supplemental material, we have included the comparisons on the **validation set** — both our method and dilated convolution outperform their implementation.

Table 2: Comparisons on the val2017 set of MS COCO. “Pretrain” — backbone is pretrained on ImageNet [8]; “OHKM” — Online Hard Keypoints Mining; “-” — data is not publicly available.

Method	Backbone	Input size	Pretrain	#Params	FLOPs	AP	AP <sup>50</sup>	AP <sup>75</sup>	AP <sup>M</sup>	AP <sup>L</sup>	AR
8-stage hourglass [21]	-	256×192	N	25.1M	14.3G	66.9	-	-	-	-	-
CPN [7]	ResNet50	256×192	Y	27.0M	6.2G	68.6	-	-	-	-	-
CPN + OHKM [7]	ResNet50	256×192	Y	27.0M	6.2G	69.4	-	-	-	-	-
SimpleBaseline [32]	ResNet50	256×192	N	34.0M	8.9G	69.3	88.3	77.0	66.2	75.8	75.3
SimpleBaseline [32]	ResNet50	256×192	Y	34.0M	8.9G	70.4	88.6	78.3	67.1	77.2	76.3
SimpleBaseline [32]	ResNet152	256×192	Y	68.6M	15.7G	72.0	89.3	79.8	68.7	78.9	77.8
Ours	-	256×192	N	1.85M	5.8G	69.9	88.6	76.7	66.1	76.5	75.1

Table 3: Comparisons on the test-dev2017 set of Microsoft COCO.

Method	Backbone	Input size	Pretrain	#Params	AP	AP <sup>50</sup>	AP <sup>75</sup>	AP <sup>M</sup>	AP <sup>L</sup>	AR
OpenPose [4]	-	-	-	-	61.8	84.9	67.5	57.1	68.2	66.5
Mask-RCNN [11]	ResNet-50-FPN	-	-	-	63.1	87.3	68.7	57.8	71.4	-
Integral Pose [27]	ResNet101	256×256	-	45.0M	67.8	88.2	74.8	63.9	74.0	-
SimpleBaseline [32]	ResNet50	256×192	N	34.0M	68.8	90.3	76.8	65.7	74.5	74.5
SimpleBaseline [32]	ResNet50	256×192	Y	34.0M	70.0	90.9	77.9	66.8	75.8	75.6
CPN [7] (ensembled)	ResNet-Inception	384×288	-	-	73.0	91.7	80.9	69.5	78.1	79.0
SimpleBaseline [32]	ResNet152	384×288	Y	68.6M	73.7	91.9	81.1	70.3	80.0	79.0
Ours	-	256×192	N	1.85M	68.9	90.3	76.0	65.3	74.9	74.3

## 4.2 Microsoft COCO Keypoint Detection

Multi-person pose estimation is one of the tasks on the Microsoft COCO dataset [18], which contains over 200k images and 250k person instances. Each person is annotated with 17 joints. For training, we use the train2017 split, which has ~57k images and ~150k samples. Validation and evaluation are performed on the val2017 (~5k images) and the test-dev2017 (~20k images) splits, respectively.

**Evaluation Metrics** The performance is evaluated using Object Keypoint Similarity (OKS), which is defined as  $\sum_i \exp(-d_i^2/2s^2\kappa_i^2)\delta(v_i>0)/\sum_i \delta(v_i>0)$ , where  $d_i$  is the Euclidean distance from the  $i$ -th predicted joint location to ground truth,  $v_i$  is the visibility flag, and  $\kappa_i$  is a per-keypoint constant that controls falloff. We report the average precision (AP) and recall (AR). Average precision is evaluated for OKS = 0.5 (AP<sup>0.5</sup>), OKS = 0.75 (AP<sup>0.75</sup>), OKS = 0.5:0.05:0.95 (AP), medium objects (AP<sup>M</sup>) and large objects (AP<sup>L</sup>). Average recall is evaluated for OKS = 0.5:0.05:0.95.

**Training** We follow similar training and testing procedures as above, but with slight differences. The input is cropped and re-sized to 256×192. Because the dataset aims for multi-person pose estimation, and the location of each person is not available during testing, we adopt the Faster-RCNN human detector [9], which is also used by SimpleBaseline [32]. During training, we still rely on the ground-truth bounding boxes to determine the location of each sample. The network is trained for 210 epochs, and we decrease the learning rate by a factor of 10 at the 170th and 200th epochs.

**Results** Table 2 shows the results on the val2017 set. Our method outperforms the SimpleBaseline (ResNet50 backbone) trained from scratch, and even performs slightly better than CPN, which uses both online hard keypoint mining and a pretrained backbone. Note that the number of parameters in our network is an order of magnitude lower than previous methods. Table 3 demonstrates that we also achieve competitive results on the test-dev2017 set, when compared to existing methods. Figure 1, bottom row, shows qualitative results on the test-dev2017 set.

## 4.3 Analysis

**Learned Filters** Figure 4 shows a subset of boxes (32 instances) learned in the last layer that uses box filters (13 × 13 kernel) at different stages of training. Burkov and Lempitsky [3] observe that a certain number of boxes learned in their network shrink to the minimal size under the imposed L2-regularization. Such regularization is not necessary in our approach since the size of each box is bounded by a pre-defined kernel. We therefore do not observe any similar effect. They also demonstrate a counter-intuitive phenomenon that their learned boxes tend to be *symmetric w.r.t. the vertical axis*, and “even when horizontal flip augmentations are switched off during training”. The

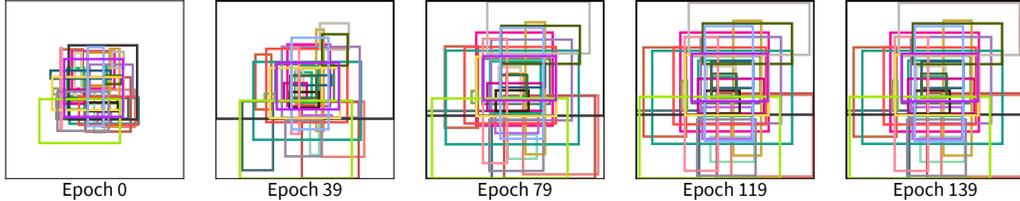


Figure 4: Evolution of learned boxes. As training proceeds, the boxes become more diverse, producing varied features maps. The learning rate is reduced at the 90th and 120th epochs.

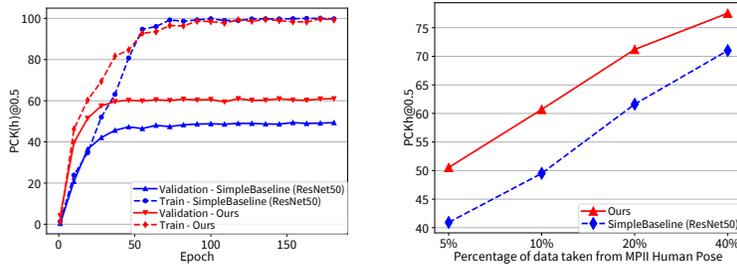


Figure 5: Overfitting analysis on the MPII Human Pose dataset. **Left:** training and validation accuracy of our method and the SimpleBaseline. Training accuracy refers to PCK@0.5 (without head-size normalization since each training sample has been re-sized to  $256 \times 256$  already). **Right:** validation accuracy after training saturates, using different amounts of training data.

learned boxes after convergence in Figure 4 show that we do not encounter this issue. We therefore conclude that four parameters are still necessary to represent each box.

**Overfitting Analysis** Complex neural networks are known to be more likely to overfit to small datasets. A kernel used to convolve with a small feature map would be *poorly utilized*, unless the size of the training data is large. While our network has many fewer parameters than existing architectures, the utilization of each kernel is high since we maintain high-resolution feature maps. This implies that our method may generalize better to new data when the training data is insufficient. To verify this, we take a small subset (10%) of the MPII Human Pose dataset and allow the networks to overfit to it. To make overfitting happen more quickly, we disable random rotation/scaling and keep flip augmentation only. Figure 5, left, shows the training progress of our network and the SimpleBaseline (using a ResNet50 backbone). Surprisingly, the gap in validation accuracy between SimpleBaseline and our method is significant after both networks saturate (i.e., achieving  $\approx 100\%$  training accuracy). While our network converges faster at the beginning, it takes longer to *completely* overfit to the training data, which may explain the higher validation accuracy after convergence. We have tried varying the learning rate for training the SimpleBaseline, but the eventual validation accuracy is nearly unaffected. In Figure 5, right, we show comparisons using different amounts of training data. Our method consistently outperforms the SimpleBaseline.

## 5 Conclusions and Discussion

In this work, we propose a large-kernel convolution layer leveraging summed-area tables to accelerate computation. Using the proposed layer, we design an end-to-end differentiable dense prediction network that produces pixel-level prediction while ensuring a large effective receptive field. We demonstrate through the human pose estimation task that our method leads to competitive performance using many fewer parameters and at lower computational cost.

Our network maintains high-resolution intermediate feature maps only, which consume more GPU memory than previous networks using downsampling. As mentioned previously, reducing the number of channels would hurt the diversity of box filters, thus making performance worse. A solution to this is the kernel splitting strategy we have mentioned in Section 3.1, which trades more `multadds` for a lower memory footprint. Nevertheless, once the learned boxes converge, we can round the sampling points (i.e., box corners) to integers and only fine-tune the rest of the network. Sampling using integer coordinates does not require interpolation, which would use many fewer `multadd` operations.

## References

- [1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3686–3693, 2014.
- [2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417, 2006.
- [3] Egor Burkov and Victor Lempitsky. Deep neural networks with box convolutions. In *Advances in Neural Information Processing Systems*, pages 6214–6224, 2018.
- [4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [5] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4733–4742, 2016.
- [6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(4):834–848, 2018.
- [7] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7103–7112, 2018.
- [8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.
- [9] Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.
- [12] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. DeeperCut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision (ECCV)*, pages 34–50, 2016.
- [13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.
- [14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, pages 675–678, 2014.
- [15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.
- [16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [17] John P Lewis. Fast template matching. In *Vision Interface*, volume 95, pages 15–19, 1995.
- [18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.
- [19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.
- [20] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.
- [21] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*, pages 483–499, 2016.
- [22] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. ENet: A deep neural network architecture for real-time semantic segmentation. *arXiv:1606.02147*, 2016.

- [23] 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 *Advances in Neural Information Processing Systems, Workshop on the Future of Gradient-Based Machine Learning Software and Techniques*, 2017.
- [24] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2017.
- [25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.
- [26] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. *arXiv:1902.09212*, 2019.
- [27] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *European Conference on Computer Vision (ECCV)*, pages 529–545, 2018.
- [28] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656, 2015.
- [29] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1653–1660, 2014.
- [30] Tomasz Trzcinski and Vincent Lepetit. Efficient discriminative projections for compact binary descriptors. In *European Conference on Computer Vision (ECCV)*, pages 228–242, 2012.
- [31] Paul Viola and Michael J Jones. Robust real-time face detection. *International Journal of Computer Vision (IJCV)*, 57(2):137–154, 2004.
- [32] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *European Conference on Computer Vision (ECCV)*, pages 466–481, 2018.
- [33] Wei Yang, Shuang Li, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Learning feature pyramids for human pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1281–1290, 2017.
- [34] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.