

# LIP: Local Importance-based Pooling

Ziteng Gao Limin Wang\* Gangshan Wu

State Key Laboratory for Novel Software Technology, Nanjing University, China

## Abstract

Spatial downsampling layers are favored in convolutional neural networks (CNNs) to downscale feature maps for larger receptive fields and less memory consumption. However, for discriminative tasks, there is a possibility that these layers lose the discriminative details due to improper pooling strategies, which could hinder the learning process and eventually result in suboptimal models. In this paper, we present a unified framework over the existing downsampling layers (e.g., average pooling, max pooling, and strided convolution) from a local importance view. In this framework, we analyze the issues of these widely-used pooling layers and figure out the criteria for designing an effective downsampling layer. According to this analysis, we propose a conceptually simple, general, and effective pooling layer based on local importance modeling, termed as Local Importance-based Pooling (LIP). LIP can automatically enhance discriminative features during the downsampling procedure by learning adaptive importance weights based on inputs. Experiment results show that LIP consistently yields notable gains with different depths and different architectures on ImageNet classification. In the challenging MS COCO dataset, detectors with our LIP-ResNets as backbones obtain a consistent improvement ( $\geq 1.4\%$ ) over the vanilla ResNets, and especially achieve the current state-of-the-art performance in detecting small objects under the single-scale testing scheme.<sup>1</sup>

## 1. Introduction

For discriminative tasks like image classification [8] and object detection [27], the modern architectures of convolutional neural networks (CNNs) mostly utilize spatial downsampling (pooling) layers to reduce the spatial size of feature maps in the hidden layers. Such pooling layers are for larger receptive fields and less memory consumption, especially in extremely deep networks [34, 15]. The widely-used max pooling, average pooling, and strided convolution use a sliding window whose stride is larger than 1 and pool

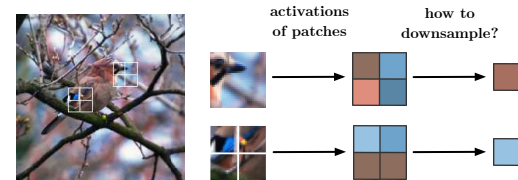


Figure 1: Illustration of our motivation. (Left to right) the original image, some nearby patches, corresponding illustrative activations in the feature map to downsample, and last, the output activations that we need. Here, red-tone activations are caused by the foreground bird. Blue-tone activations are caused by the background clutter in the top patch or the representative blue feather in the bottom patch. We want to preserve the red-tone activations in the top patch window and the blue-tone activations in the bottom. The downsampling method should recognize discriminative features adaptively across sliding windows.

features by different strategies in each local window. But these layers might prevent discriminative details from being well preserved, which are crucial for recognition and detection task. This is especially undesirable for discriminative features of tiny objects, as such details might be diluted with clutter activations or even not be sampled by improper downsampling strategies.

In this paper, we aim to address these issues raised by the existing downsampling layers. To analyze their drawbacks, we present a unified framework from a local importance view. Under this new perspective, the existing pooling procedure could be seen as aggregating features with their local importance in each sliding window. To our best knowledge, we are the first to present a framework from the importance view for downsampling layers, which allows us to analyze and improve the pooling methods in a more principled way. As a result, we show that average and max pooling are both suboptimal due to the strong assumption or the invalid prior knowledge. Strided convolution adopts the improper interval sampling and also fails to model importance adaptively. To overcome their limitations, we present a new pooling method to learn importance weights automatically, coined as *Local Importance-based Pooling* (LIP).

Basically, we argue that not all nearby pixels contribute

\*Corresponding author.

<sup>1</sup>Code is available at <https://github.com/sebgao/LIP>.

equally and some features are more discriminative than the others within a neighborhood in the downsampling procedure, as illustrated in Figure 1. Therefore, it is expected to explicitly model the local importance and build a metric measure over pixels within local neighborhoods. From this analysis, we propose the LIP to meet the requirement of an ideal pooling operation. Specifically, LIP proposes to learn the metric of importance by a subnetwork based on the input features automatically. In this sense, LIP is able to adaptively determine which features are more important to be kept through downsampling. For instance, LIP enables the network to preserve features of tiny targets while discarding false activations of the background clutter when recognizing or detecting small objects. Moreover, LIP is a more generic pooling method than the existing methods, in sense that it is capable of mimicking the behavior of average pooling, max pooling and detail-preserving pooling [33].

Experiments show LIP outperforms baseline methods by a large margin on ImageNet [8] with different architectures. We also evaluate our LIP backbones on the challenging COCO detection task [27], where localizing small objects play an important role. The both one- and two-stage detectors with our LIP-ResNets as backbones obtain a consistent improvement over the vanilla ResNets, and in particular achieve a new state-of-the-art performance in detecting small objects under the single-scale testing scheme.

## 2. Related Work

Downsampling layers as basic layers in CNNs were proposed with LeNet-5 [20] as a way to reduce spatial resolution by summing out values in a sliding window. Spatial downsampling procedures also exist in traditional methods. For example, HOG and SIFT [7, 29] aggregated the gradient descriptors within each spatial neighborhood. Bag of words (BoW) based models also used intensive pooling in object recognition as to obtain more robust representations against translation and scale variance [37, 19].

Modern CNNs utilize pooling layers to downscale feature maps mainly for larger receptive field and less memory consumption. VGG [34], Inception [38, 18, 39] and DenseNet [17] used average and max pooling as downsampling layers. ResNet [15] adopted convolutions whose stride is not 1 to extract features at regular non-consecutive locations as downsampling layers.

Some pooling methods, including global average pooling [24], ROI pooling [9], and ROI align [14], aim to downscale feature maps of arbitrary size to a fixed size and therefore enable the network to cooperate with inputs of different sizes. We do not discuss these methods as they are designed to specific architectures. Here, we only focus on pooling layers inside networks, that is, the ones that gradually downscale feature maps by a fixed ratio.

There are some analysis on pooling methods before the

widespread application of CNNs. Boureau *et al.* [2] analyzed average and max pooling in traditional methods, and proved that max pooling can preserve more discriminative features than average pooling in terms of probability. The work [43, 41] showed that pooling can be without specific forms and learning to pool features is beneficial. Our work mainly follows this research line and our results further support these conclusions.

Recent work about pooling has focused on how to better downscale feature maps in CNNs through new pooling layers. Fractional pooling [11] and S3pool [46] tried to improve the way to perform spatial transformation of pooling, which is not the focus of our paper. Mixed and hybrid pooling [45, 21] used various combinations of max and average pooling to perform downscaling.  $L_p$  pooling [13] aggregated activations in the  $L_p$  norm way, which can be viewed as a continuum between max and average pooling controlled by the learned  $p$ . These methods can unify max and average pooling and further improve the performance of networks. However, they can simply learn better pooling method based on average pooling and max pooling, or the combination of them, but fails to provide more insights about general downsampling methods. Saeedan *et al.* [33] argued that details should be preserved and redundant features can be discarded by proposed detail-preserving pooling (DPP). The detail criterion of DPP is relatively hand-crafted by calculating the deviation from statistics of pixels in sliding windows, which is heuristic and may be not optimal.

In this paper, we analyze widely-used pooling layers based on a local importance view, which has not been investigated in previous work. Our proposed LIP, naturally arisen from this concept, outperforms hand-crafted pooling layers by a large margin.

Attention-based methods are recently popular in computer vision community [42, 49]. Our LIP can be also seen as a local attention approach designed for pooling, of which attention weights are in the softmax form. LIP mainly differs from other attention methods in two important aspects for the better compatibility with downsampling procedure: (1) attention weights are produced by local convolutions in logit modules and then normalized locally; (2) LIPs do not adopt the key-query schemes in attention modeling for achieving better shift invariance.

## 3. Local Importance Modeling

In this section, we first present the framework for downsampling layers from local importance modeling view. We discuss some widely-used pooling layers in this framework. Next, we describe our proposed local importance-based pooling (LIP), which naturally arises from this analysis. Finally, we show how to equip popular architectures with LIP layers and then obtain LIP-ResNet and LIP-DenseNet.

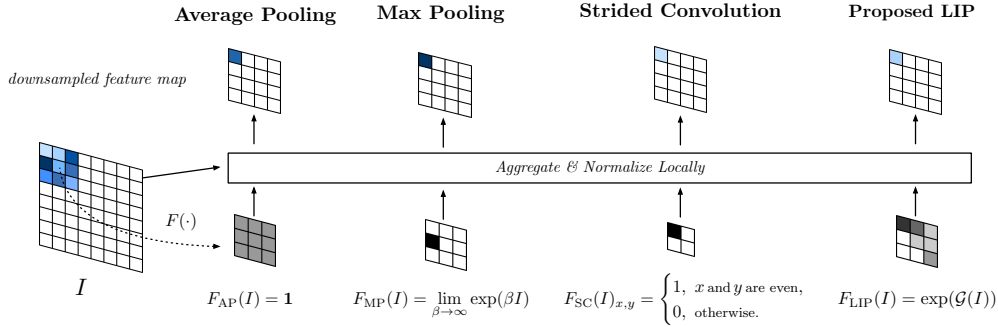


Figure 2: Different downsampling methods viewed in the LAN framework. Activations in the input feature map are blue colored, darker meaning larger. Only activations and corresponding importance within current sliding window are shown. For strided convolution, the window size is equivalent to the stride, which is 2 here.

### 3.1. Framework and Analysis

To analyze the existing downsampling methods and well motivate our LIP, we present a unified framework for downsampling layers from the view of local importance, named *Local Aggregation and Normalization* (LAN). Specifically, given the input feature map  $I$ , the kernel indice set  $\Omega$  consisting of relative sampling locations  $(\Delta x, \Delta y)$  in a sliding window, and the left-top location  $(x, y)$  corresponding to the sliding window in the input feature map with regard to the output location  $(x', y')$ , the LAN framework is formulated as:

$$O_{x',y'} = \frac{\sum_{(\Delta x, \Delta y) \in \Omega} F(I)_{x+\Delta x, y+\Delta y} I_{x+\Delta x, y+\Delta y}}{\sum_{(\Delta x, \Delta y) \in \Omega} F(I)_{x+\Delta x, y+\Delta y}}, \quad (1)$$

where  $F(I)$  is the importance map whose size is the same with  $I$  and  $F(I) \geq 0$  over space. The division  $(x/x', y/y')$  stands for the stride factor, e.g.,  $x = 2x', y = 2y'$  for  $2 \times 2$  stride. We simply denote a stride  $2 \times 2$  as 2 in this paper. As the name of the framework implies, pooling in this view can be seen two steps: **aggregate** features with the importance  $F(I)$  and **normalize** them by importance within local sliding windows. This framework can be extended naturally to the multi-channel situation.

One can see pooling in this framework as weighted sum over each window where weights are locally normalized importance:

$$\frac{F(I)_{x+\Delta x, y+\Delta y}}{\sum_{(\Delta x, \Delta y) \in \Omega} F(I)_{x+\Delta x, y+\Delta y}}, \quad (2)$$

for  $I_{x+\Delta x, y+\Delta y}$ , which we term simply as local importance. Therefore, local importance stands for weights of features within a sliding window. We can analyze which features in downsampling procedures are more important than others nearby by  $F(I)$ .

Our motivation is that since the feature pooling procedure is intrinsically lossy as it squeezes large input into small output, it is necessary to carefully consider which features to sample and how to aggregate them in a small sliding

window as shown in Figure 1. Sampled features should be discriminative enough for the target tasks. The LAN framework provides a principled way to understand and improve these pooling methods by studying the corresponding importance function  $F$ . Next, we analyze some widely-used downsampling layers in this framework and figure out the requirement of an ideal pooling operation. Figure 2 shows some of these downsampling methods viewing in the framework.

**Average and max pooling.** As discussed in [2], given  $F(I) = \exp(\beta I)$ ,  $\beta = 0$  gives average pooling and  $\beta \rightarrow \infty$  gives max pooling. Average pooling associates features with the same importance to all locations during aggregation in a small window, while max pooling put all attention on the largest activation within a neighborhood. We argue that both of them are suboptimal. Average pooling harms discriminative but small features and cause blurry downsampling features due to the strong assumption of the local equality of features. Max pooling as an improvement over average pooling on feature selection, however, assumes that the most discriminative feature should be of the maximum activation. This assumption mainly has two drawbacks. First, the prior knowledge that the maximum activation stands for the most discriminative detail, may not be always true. Second, the max operator over sliding windows hinders gradient-based optimization since in the back-propagation gradients are assigned only to the local maximums, as discussed in [33]. These sparse gradients would further enhance this inconsistency, in sense that discriminative activations will never become maximums unless current maximums are suppressed.

**Strided convolutions.** Strided convolutions can be seen as dense convolutions whose stride is 1, followed by spatial subsampling [47]. This spatial subsampling can be interpreted as downsampling in our framework with

$$F(I)_{x,y} = \begin{cases} 1, & \text{if } x \text{ and } y \text{ are both multiples of } s, \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

where  $I$  is densely convolved features and  $s$  is both the

stride factor and sliding window size. From this perspective, the downsampling part of strided convolutions fails to model the importance in downsampling procedures adaptively. Moreover, it focuses only on one fixed location within each sliding window and discards the rest. This fixed interval sampling scheme will limit shift invariance, as convolutional patterns are required to appear at specific and non-consecutive locations to activate. In this sense, minor shifts and distortions can lead to great changes in downsampled features and thus disturb the shift invariance of CNNs [47]. For the case of strided  $1 \times 1$  convolutions, it is even worse since the feature maps are not fully utilized [16] and it will incur gradient checkerboard problem [32].

**Detail-preserving pooling.** Recent proposed detail-preserving pooling (DPP) [33] uses the detail criterion as importance function  $F$ , which is measured by the deviations of features from the activation statistics in sliding windows. DPP solves the problem of max pooling by designing more sophisticated importance function and ensuring the continuity for better gradient optimization. However, the assumption in DPP is heuristic and the more detailed feature might be the less discriminative ones. For example, the background clutter could be more detailed than a bird of solid color in foreground. Therefore, DPP might preserve the less discriminative details to outputs. Hand-crafted importance functions in max pooling and DPP incorporate the general prior knowledge into downsampling procedure, which might lead to the inconsistency with the final target of discriminative tasks.

**Requirements of ideal pooling.** From the analysis above, we can figure out the requirement of an ideal pooling layer. First, the downsampling procedure is expected to handle minor shifts and distortions as much as possible, and thus should avoid adopting the fixed interval sampling scheme, i.e.,  $F$  used by strided convolutions. Second, the importance function  $F$  should be selective to the discriminative features rather than manually designed based on prior knowledge, i.e.,  $F$  used in max pooling and DPP. This discriminativeness measure should be adaptive to different tasks and automatically determined by the final objective.

### 3.2. Local Importance-based Pooling

To meet requirements of ideal pooling arisen from local importance view in the LAN framework, we propose local importance-based pooling (LIP). By using a learnable network  $\mathcal{G}$  in  $F$ , the importance function now is not limited in hand-crafted forms and able to learn the criterion for the discriminativeness of features. Also, we restrict the window size of LIP to be not less than stride to fully utilize the feature map and avoid the issue of fixed interval sampling scheme. More specifically, the importance function in LIP is implemented by a tiny fully convolutional network (FCN) [28], which learns to produce the importance map

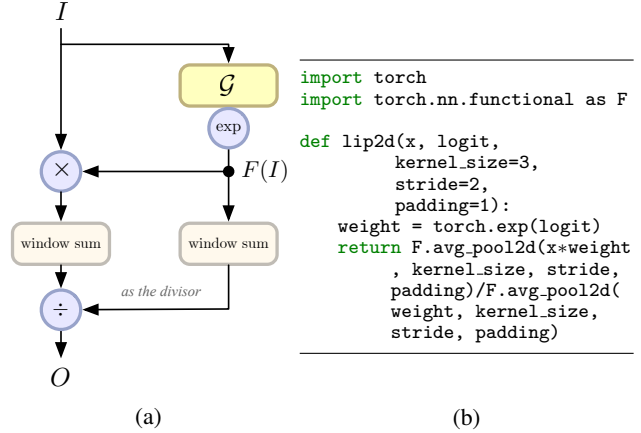


Figure 3: LIP operator and its PyTorch implementation. The logit module  $\mathcal{G}$  is not shown in (b). The ‘window sum’s in (a) mean locally summing within sliding windows.

based on inputs in an end-to-end manner. To make the importance weights non-negative and easy to optimize, we add  $\exp(\cdot)$  operation on top of  $\mathcal{G}$ , that is:

$$F(I) = \exp(\mathcal{G}(I)), \quad (4)$$

where  $\mathcal{G}$  and  $\mathcal{G}(I)$  are named the logit module and the logit respectively as  $\mathcal{G}(I)$  is the logarithm of the importance. In contrast to the hand-crafted form specified by prior knowledge in max pooling or DPP, the logit module  $\mathcal{G}$  is able to learn a better and more compatible importance criterion for both the network and target task. More concretely, according to Equation (1), LIP is then written as:

$$O_{x',y'} = \frac{\sum_{(\Delta x, \Delta y) \in \Omega} I_{x+\Delta x, y+\Delta y} \exp(\mathcal{G}(I))_{x+\Delta x, y+\Delta y}}{\sum_{(\Delta x, \Delta y) \in \Omega} \exp(\mathcal{G}(I))_{x+\Delta x, y+\Delta y}}. \quad (5)$$

With LIP, discriminative features can be automatically emphasized during downsampling procedure by learning a larger value of  $\mathcal{G}(I)$  at the corresponding locations. In the current implementation of LIP, the logit is calculated in a channel wise manner. Figure 3 shows the diagram and PyTorch implementation of LIP.

**Deformable modeling of LIP.** At the macro level, learnable importance function  $F$  of LIP enables the network to model deformation of objects by learning a good effective allocation of features into downsampling with adaptive importance weights. Different from deformable convolutions [6, 50] to sample features by bilinear interpolation with adaptive offsets, LIP explicitly performs spatially dynamic feature selection based on inputs and thus has deformable receptive fields. Empirical evidence of the deformable capacity of LIP is shown and discussed in Section 4.2.

### 3.3. Exemplars: LIP-ResNet and LIP-DenseNet

ResNet [15] and DenseNet [17] are typical architectures among modern CNNs. ResNet mainly uses strided convo-

lutions as downsampling layers except one max pooling in the bottom. DenseNet utilizes average pooling in transition blocks, and in the bottom a strided convolution layer and max pooling like ResNet to downscale feature maps.

**Architectures with LIP.** We adopt the revised ResNet [12] as our plain ResNet baseline, where residual branches employ  $3 \times 3$  kernel for strided convolutions, shown in Figure 4a. To build LIP variants, we replace max pooling in the bottom and strided convolutions in downsampling blocks with LIP. As discussed in Section 3.1, strided convolutions in ResNet could be replaced by a dense convolution and a following LIP. However, this substitution is computational intensive and memory inefficient. We instead first downscale features and then perform convolution. In this sense, we use a LIP and a following convolution to replace strided convolutions in residual and shortcut branches, as shown in Figure 4b. To keep receptive fields the same and avoid the interval sampling problem, we set the window size of LIP to  $3 \times 3$  and the following convolution to  $1 \times 1$ . We leave the global average pooling in the top of ResNet unchanged. Total 7 layers (1 for max pooling,  $3 \times 2$  for strided convolution) are replaced with LIP layers. We name this modified ResNet architecture as **LIP-ResNet**. For DenseNet, we replace  $2 \times 2$  average pooling layers in transition blocks and  $3 \times 3$  max pooling in the bottom by LIP layers of same configurations of window size. The global average pooling also remains unchanged like LIP-ResNet. Total 4 layers (1 max pooling and 3 average pooling) are replaced by LIP layers, and the resulted network is termed as **LIP-DenseNet**.

**Design of logit modules.** In the current implementation, we design two forms of logit modules for LIP layers, called the projection and the bottleneck form, respectively. Structures of logit modules are shown in Figure 4d and 4e. In projection form, the logit module in LIP is simply composed of a  $1 \times 1$  convolution layer. The logit module of bottleneck form is like residual branches in bottleneck blocks [15], which aims to capture spatial information in an efficient way. This form is denoted as Bottleneck- $x$ , where  $x$  is number of channels in the the input and output of  $3 \times 3$  convolution. To further reduce computational complexity of bottleneck logit modules in LIP-ResNet, the first  $1 \times 1$  convolution and  $3 \times 3$  convolution are shared between the residual and shortcut branches in a building block. The input of logit modules here is changed to the feature map fed into the building block, i.e., the top cyan circle in Figure 4b, instead of the feature map to downsample. Bottleneck- $x$  logit module in LIP substitution for replacing max pooling in ResNet and DenseNet is simply a  $3 \times 3$  convolution.

For more effective modeling and stable training, we apply affine instance normalization [40] as spatial normalization and sigmoid function with a fixed amplification coefficient on the top of each logit module. Affine instance nor-

Method	Top-1	Top-5	#Params	FLOPs
Strided convolution	76.40	93.15	25.6M	4.12G
Average pooling	76.96	93.35	<b>22.8M</b>	<b>3.82G</b>
DPP ( <i>our baseline structure</i> )	76.87	93.30	<b>22.8M</b>	3.83G
DPP ( <i>original structure in [33]</i> )	77.22	93.64	25.6M	6.59G
LIP w Projection	77.49	93.86	24.7M	4.78G
LIP w Bottleneck-64	77.92	93.97	23.2M	4.65G
LIP w Bottleneck-128	<b>78.19</b>	93.96	23.9M	5.33G
LIP w Bottleneck-256	78.15	<b>94.02</b>	25.8M	7.61G

Table 1: ResNet-50 with different downsampling methods.

malization make activations on each channel of each feature map follow normal distribution and then rescale it by learnable affine parameters. The spatial normalization and rescale operation aim to help learn extreme cases such as max pooling. The sigmoid function is used here to maintain numerical stability and the fixed amplification coefficient provides large enough range for logits, which is set to 12 throughout our experiments.

## 4. Experiments

To validate the effectiveness of our LIP, we carry out experiments on the ImageNet 1K classification task [8] and the MS COCO detection task [27].

### 4.1. ImageNet Classification Experiment Setup

ImageNet 1K classification task [8] requires the methods to cope with high-resolution images to capture discriminative details. We use (LIP-)ResNet and (LIP-)DenseNet for our experiments on the ImageNet classification task. For (LIP-)ResNet training, we use 8 GPUs and a mini-batch of 256 inputs, 32 images per GPU. For (LIP-)DenseNet training, we use 4 GPUs and a mini-batch of 256, 64 images per GPU. Our training procedure is generally following the recipe [10] with two minor modifications. One is that we use SGD optimizer to update parameters with the vanilla momentum rather than Nesterov one. The other is that weight decay of  $10^{-4}$  is applied to all learnable parameters including those of Batch Normalization. All LIP layers are initialized to behave like average pooling by initializing parameters of the last convolution in logit modules to 0. All results are reported on the validation set with single-crop testing.

### 4.2. Results on ImageNet and Analysis

**Study on LIPs and different logit modules.** To compare with other pooling methods, we replace all LIP layers in LIP-ResNet by other pooling layers, i.e., average pooling or DPP, and keep the same configuration of window size and stride for fair comparison. The building blocks of these baselines are shown in Figure 4c. Note that these baselines

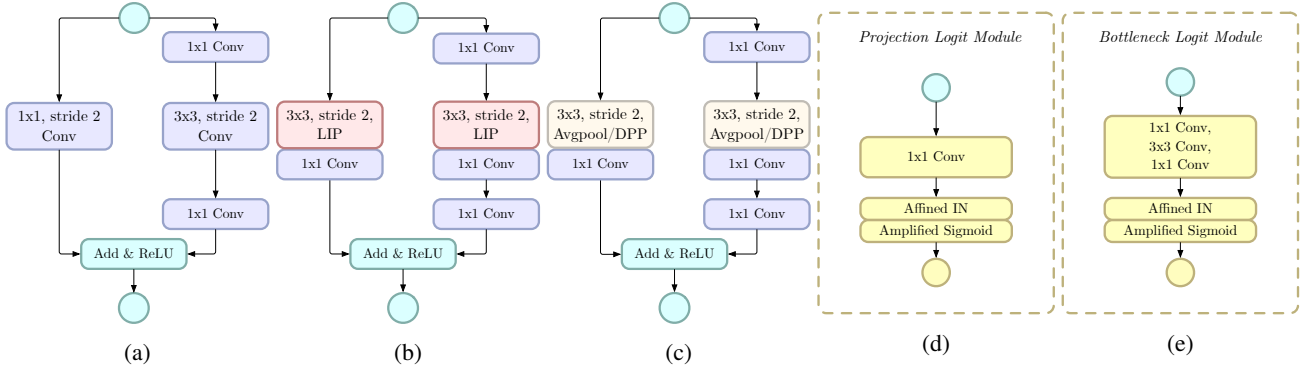


Figure 4: Structures of ResNet building blocks for downsampling and logit modules. There are ResNet building blocks with strided convolutions (a), LIP (b), average pooling or DPP (c). (d) and (e) show the projection and bottleneck logit module. The first two ‘Conv’s in (e) mean convolutions and following affine Instance Normalization and ReLU function.

Layer	Combination of layers in the top			
	A	B	C	D
Affined IN	✓	✓		
Amplified sigmoid	✓		✓	
Top-1	78.19	N/A	77.81	77.89
Top-5	93.96	N/A	93.86	93.86

Table 2: Different top layers on logit modules. Combination D is trained with average pooling within first 2000 iterations and then with LIP to avoid numerical overflow due to  $\exp(\cdot)$  operation along with noisy gradient during early iterations. Combination B falls in training although we tried various ways to avoid numerical problems.

eliminate other factors including receptive fields and non-linearities to to be more consistent with LIP-ResNet. In this study, we resort to the ResNet-50 to perform comparison between different pooling layers.

The results are reported in Table 1. *First*, the ResNet-50 baseline with average pooling reduces both parameters and FLOPs, but still improves performance over the vanilla ResNet by around 0.5% in top-1 accuracy. This result may be ascribed to the fixed interval sampling issue in strided convolutions, and a similar result was found in [16]. *Second*, for our downsampling method, LIP with the simplest projection logit modules gains a noticeable improvement over these baselines ( $> 0.5\%$  in top-1). This shows that the importance simply learned from the projection logit module is beneficial for downsampling procedure. *Third*, with a more powerful logit module Bottleneck-64, LIP-ResNet further improves accuracy over the projection one with fewer parameters and less computational cost. This demonstrates that spatial information is helpful for designing a better logit module. The performance would saturate when we stretch the bottleneck logit module wider, and the Bottleneck-128 is a good trade-off between computational complexity and recognition performance, improving by 1.79% in top-1 and 0.81% in top-5 over the plain net-

Layer	Combination of LIP substitutions				
	A	B	C	D	E
Max Pooling	✓				
Res <sub>3</sub>	✓	✓			
Res <sub>4</sub>	✓	✓	✓		
Res <sub>5</sub>	✓	✓	✓	✓	
Top-1	78.19	77.87	77.78	76.92	76.40
Top-5	93.96	93.94	93.81	93.37	93.15
#Params	23.9M	23.8M	23.7M	23.9M	25.6M
FLOPs	5.33G	4.87G	4.26G	4.11G	4.12G

Table 3: Different LIP substitution locations. Combination A stands for the ResNet-50 with full 7 LIPs (LIP-ResNet w Bottleneck-128) and E stands for the vanilla ResNet.

work. We adopt LIP with the Bottleneck-128 logit module as our default choice in the remaining experiments. *Finally*, we test the effectiveness of instance normalization and amplified sigmoid function. Results are shown in Table 2. The combination of them improves accuracy by enabling LIP to approximate extreme cases such as max pooling stably.

**LIP layers at various locations.** Table 3 shows the results by placing different numbers of LIPs at different locations. We can find more LIPs generally contributes to better result but LIPs at different locations may not improve performance equally. LIP as the max pooling substitution only improves the top-1 accuracy significantly. We suspect that a single convolution as the logit module at this layer fails to encode enough semantic information to provide powerful logits into LIP. Another possible reason is that high-resolution details may help fine-grained classification but not benefit coarse-grained one. We can also find that the LIP at Res<sub>4</sub> is the most effective one. This might be due to the fact that the feature at this layer contains more semantics and the feature map size is still relatively large for downscaling. For practical applications, we recommend the usage of Combination C in Table 3 due to less parameters and only 3% extra FLOPs compared to the vanilla ResNet.

Architecture	Top-1	Top-5	#Params	FLOPs
ResNet-50	76.40	93.15	25.6M	4.12G
LIP-ResNet-50	78.19	93.96	23.9M	5.33G
ResNet-101	77.98	93.98	44.5M	7.85G
LIP-ResNet-101	79.33	94.60	42.9M	9.06G
ResNet-152*	78.49	94.22	60.2M	11.58G
DenseNet-BC-121	75.62	92.56	8.0M	2.88G
LIP-DenseNet-BC-121	76.64	93.16	8.7M	4.13G

Table 4: ResNets and DenseNets with and without LIP. For ResNet-152, we adopt the result trained by a similar recipe<sup>2</sup>.

But our default choice for the remaining experiments is the full LIP model, i.e., Combination A.

**Different network depth and architectures.** We also evaluate LIP-ResNet and LIP-DenseNet with the deeper network, and the result is summarized in Table 4. We find that LIP-ResNet-50 performs comparably to the vanilla ResNet-101 with only about half parameters and less FLOPs. LIP-ResNet-101 surpasses the vanilla ResNet-152 in both top-1 and top-5 accuracy by a notable margin (0.84% and 0.38%). For DenseNet and LIP-DenseNet, the result is also favorable, demonstrating the effectiveness of our method across different network architectures.

**Visualizations.** As discussed in Section 3.2, LIP enables the network to have capacity of deformable modeling. To show this, we perform some visualizations of LIP layers. We first compute class activation mappings (CAMs) [48] of ResNet-50 models with LIP, average pooling, and strided convolution. Next, we backpropagate activation of specific locations in CAMs to get gradient maps, which are called effective receptive fields [31] of specific locations in the original image context. Results are shown in Figure 5. The CAMs are similar but the gradient maps differ much among three downsampling approaches. The effective receptive field of the model with LIP layers is more compact and mainly focuses on the foreground even when the backpropagated location moves out of the foreground (i.e., Figure 5d). Average pooling and strided convolution ones, however, are interfered more by the background clutter when backpropagating the activation out of the foreground. This comparison shows the deformable modeling capacity of LIP layers. Compared with the average pooling and strided convolution, the clutter and background without discriminative features contribute much less to final recognition results in LIP-ResNet.

### 4.3. MS COCO Detection Experiment Setup

After verifying the effectiveness of LIP on image classification, we now focus on the more challenging detection task. There exists the problem of invisibility of tiny objects

<sup>2</sup><https://github.com/tensorpack/tensorpack/tree/master/examples/ResNet>

Backbone	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>
<i>Faster R-CNN w FPN results</i>						
ResNet-50	37.7	59.3	41.1	21.9	41.5	48.7
LIP-ResNet-50	39.2	61.2	42.5	24.0	43.1	50.3
ResNet-101	39.4	60.7	43.0	22.1	43.6	52.1
LIP-ResNet-101	41.7	63.6	45.6	25.2	45.8	54.0
ResNeXt-101	40.7	62.1	44.5	23.0	44.5	53.6
<i>RetinaNet results</i>						
ResNet-50	36.6	56.6	38.9	19.6	40.3	48.9
LIP-ResNet-50	38.0	58.8	40.5	22.6	41.5	49.9
ResNet-101	38.1	58.1	40.6	20.2	41.8	50.8

Table 5: Faster R-CNN with FPN and RetinaNet with different backbones results on COCO 2017 val set. ResNeXt-101 stands for ResNeXt-64x4d-101 backbone in [44].

in most CNN architectures for detection [23]. This issue is mainly caused by losing discriminative information of small objects during improper downsampling procedure, which is suitable to justify the design of our LIP.

MS COCO detection [27] is a challenging task where the object scale variation is very large and detecting small objects plays a crucial role in final detection performance [35, 36]. We adopt mmdetection codebase [4] for our experiments. Our training strictly follows the default configuration of mmdetection, which includes setting shorter size of the image to 800, using standard horizontal flipping augmentation and ROI Align [14]. In this experiment, we train two detection frameworks: Faster R-CNN with FPN [25] and RetinaNet [26], on the COCO 2017 train set with the pre-trained backbone networks in Section 4.2. We adopt the typical 2× training time scheme for all COCO experiments. The baseline results are reported by evaluating the released detectors in mmdetection model zoo<sup>3</sup>. Detection performance is reported with single-scale testing.

### 4.4. Results on MS COCO and Analysis

The results of different backbones with Faster R-CNN and FPN are shown in Table 5. LIP-ResNet-50 and LIP-ResNet-101 backbones with Faster R-CNN yield 1.5% and 2.3% gain in AP over baselines, showing the effectiveness of our LIP at capturing discriminative features for detection branch. Their improvement gap may be ascribed to the fact that the deeper backbone provides more semantic features to produce better logits for LIP downsampling. For small object detection, the deeper vanilla ResNet only results in 0.2% gain in AP<sub>s</sub>, while LIP-ResNet-101 is better than LIP-ResNet-50 by 1.2% in AP<sub>s</sub>. The improvement of LIP-ResNets in AP<sub>s</sub> over the vanilla ResNets (2.1% and 3.1%) is also notable. These results show that the LIP layers are able to better preserve discriminative features of tiny

<sup>3</sup>Evaluated when this paper was submitted and some baseline results are slightly higher than the officially reported ones in [4].

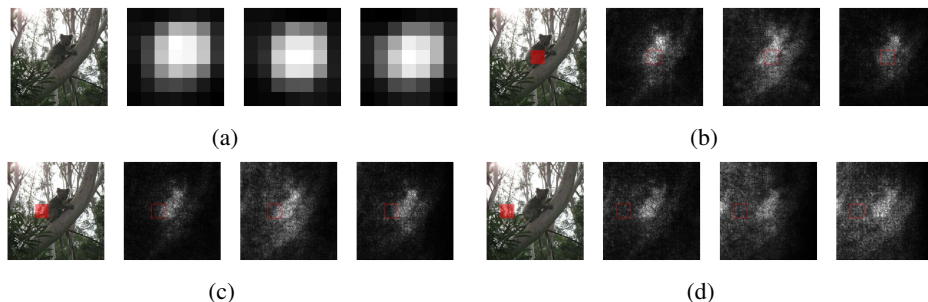


Figure 5: Visualizations of ResNets with different downsampling techniques. For each subfigure, there are an original image and results of ResNet-50 with LIP, average pooling and strided convolution from left to right. (a) denotes the class activation mappings (CAMs) [48] for koala. (b)~(d) denote the effective receptive fields [31] in the image context, namely, backpropagated gradients from specific locations in CAMs (red in original images). Contrast of visualizations is lowered for human vision.

Detection Framework	Backbone	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>t</sub>
Faster R-CNN w FPN [25]	ResNet-101 w FPN	36.2	59.1	39.0	18.2	39.0	48.2
Mask R-CNN [14]	ResNet-101 w FPN	38.2	60.3	41.7	20.1	41.1	50.2
SOD-MTGAN [1]	ResNet-101 w FPN	41.4	63.2	45.4	<u>24.7</u>	44.2	52.6
Grid R-CNN [30]	ResNet-101 w FPN	41.5	60.9	44.5	23.3	44.9	53.1
DCR [5]	ResNet-101-Deformable w FPN	41.7	64.0	45.9	23.7	44.7	53.4
TridentNet [22]	ResNet-101	42.7	63.6	<u>46.5</u>	23.9	<u>46.6</u>	<b>56.6</b>
Cascade R-CNN [3]	ResNet-101 w FPN	<u>42.8</u>	62.1	46.3	23.7	45.5	55.2
Faster R-CNN w FPN & LIP	LIP-ResNet-101 w FPN	42.0	<u>64.3</u>	45.8	<u>24.7</u>	45.2	52.3
Faster R-CNN w FPN & LIP	LIP-ResNet-101-MD w FPN	<b>43.9</b>	<b>65.7</b>	<b>48.1</b>	<b>25.4</b>	<b>46.7</b>	<u>56.3</u>

Table 6: Results on COCO test-dev set. ‘Deformable’ denotes deformable convolutions in [6]. ‘MD’ denotes adding more deformable convolutions. The 1st and 2nd of each criterion are boldface and underlined respectively.

objects. The results with the single-stage RetinaNet also validate the effectiveness of the LIP layer.

To compare with the state-of-the-art detectors, we train the deformable backbone (following the placement of more deformable convolutions in [50], but without modulation and feature mimicking) with LIP in Faster R-CNN and FPN framework. The results are shown in Table 6. The detectors with LIP-ResNet-101 are comparable to the state-of-the-art methods by simply using a standard detection pipeline without any specific design. The LIP-ResNet-101-MD backbone can further boost AP to 43.9% and AP<sub>s</sub> to 25.4%, yielding a new state-of-the-art performance in detecting small objects under the single-scale testing scheme.

## 5. Conclusion and Future Work

In this paper, we stress spatial importance modeling in pooling procedures. We have presented the *Local Aggregation and Normalization* (LAN) framework based on local importance to analyze the widely-used pooling layers. Under the framework, we figure out these layers might keep out discriminative features due to using improper downsampling importance maps. Based on this analysis, we have proposed the *Local Importance-based Pooling* (LIP),

a conceptually simple, general, and effective downsampling method, with a goal of learning adaptive and discriminative importance maps to aggregate features for downsampling. Networks with LIPs are able to better preserve the discriminative details, especially those of tiny objects. Experiments on the ImageNet classification task indicate that LIP can capture rich details for holistic image recognition. On the COCO detection task, LIPs enable both one- and two-stage detection frameworks to yield better performance, especially that on small objects. Moreover, detectors with LIP-ResNet backbones reach a new state-of-the-art performance in detecting small objects by simply using a standard detection framework.

In the future, we plan to study more aspects of implementation of LIP, such as logit module design, adaptive pooling size exploration and so on. Meanwhile, we will verify the effectiveness of LIP to more tasks, e.g., pose estimation and image segmentation.

## Acknowledgments

This work is supported by the National Science Foundation of China (No. 61921006, No. 61321491), and Collaborative Innovation Center of Novel Software Technology and Industrialization. The first author would like to thank Nan Wei and Qinshan Zeng for their comments and support.



## References

- [1] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. SOD-MTGAN: small object detection via multi-task generative adversarial network. In *ECCV*, 2018.
- [2] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 2010.
- [3] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In *CVPR*, 2018.
- [4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. In *arXiv*, 2019.
- [5] Bowen Cheng, Yunchao Wei, Honghui Shi, Rogério Schmidt Feris, Jinjun Xiong, and Thomas S. Huang. Revisiting RCNN: on awakening the classification power of faster RCNN. In *ECCV*, 2018.
- [6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.
- [7] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
- [8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- [9] Ross B. Girshick. Fast R-CNN. In *ICCV*, 2015.
- [10] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. In *arXiv*, 2017.
- [11] Benjamin Graham. Fractional max-pooling. In *arXiv*, 2014.
- [12] Sam Gross and Michael Wilber. Training and investigating residual nets. <https://github.com/facebook/fb.resnet.torch>.
- [13] Çağlar Gülçehre, KyungHyun Cho, Razvan Pascanu, and Yoshua Bengio. Learned-norm pooling for deep feedforward and recurrent neural networks. In *ECML PKDD*, pages 530–546, 2014.
- [14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [16] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, 2019.
- [17] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.
- [18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.
- [19] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.
- [20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [21] Chen-Yu Lee, Patrick W. Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016.
- [22] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *arXiv*, 2019.
- [23] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet: A backbone network for object detection. In *ECCV*, 2018.
- [24] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.
- [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*, 2017.
- [26] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.
- [27] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014.
- [28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- [29] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [30] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid R-CNN. In *arXiv*, 2018.
- [31] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard S. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *NIPS*, 2016.
- [32] Sebastian Palacio, Joachim Folz, Jörn Hees, Federico Raue, Damian Borth, and Andreas Dengel. What do deep networks like to see? In *CVPR*, 2018.
- [33] Faraz Saeedan, Nicolas Weber, Michael Goesele, and Stefan Roth. Detail-preserving pooling in deep networks. In *CVPR*, 2018.
- [34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- [35] Bharat Singh and Larry S. Davis. An analysis of scale invariance in object detection SNIP. In *CVPR*, 2018.
- [36] Bharat Singh, Mahyar Najibi, and Larry S. Davis. SNIPER: efficient multi-scale training. In *NeurIPS*, 2018.
- [37] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.
- [38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
- [39] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

- [40] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. In *arXiv*, 2016.
- [41] Lan Wang, Chenqiang Gao, Jiang Liu, and Deyu Meng. A novel learning-based frame pooling method for event detection. *Signal Processing*, 140:45–52, 2017.
- [42] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.
- [43] Guo-Sen Xie, Xu-Yao Zhang, Xiangbo Shu, Shuicheng Yan, and Cheng-Lin Liu. Task-driven feature pooling for image classification. In *ICCV*, 2015.
- [44] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.
- [45] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. Mixed pooling for convolutional neural networks. In *RSKT*, 2014.
- [46] Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogério Schmidt Feris. S3pool: Pooling with stochastic spatial sampling. In *CVPR*, 2017.
- [47] Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, 2019.
- [48] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.
- [49] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *arXiv*, 2019.
- [50] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *arXiv*, 2018.