# Local Context Normalization: Revisiting Local Normalization

Anthony Ortiz *[1, 4], Caleb Robinson[3, 4], Dan Morris[2], Olac Fuentes[1], Christopher Kiekintveld[1], Md Mahmudulla Hassan[1], and Nebojsa Jojic †[2]

[1]The University of Texas at El Paso
[2]Microsoft Research
[3]Georgia Institute of Technology
[4]Microsoft AI for Good Research Lab

## Abstract

*Normalization layers have been shown to improve convergence in deep neural networks, and even add useful inductive biases. In many vision applications the local spatial context of the features is important, but most common normalization schemes including Group Normalization (GN), Instance Normalization (IN), and Layer Normalization (LN) normalize over the entire spatial dimension of a feature. This can wash out important signals and degrade performance. For example, in applications that use satellite imagery, input images can be arbitrarily large; consequently, it is nonsensical to normalize over the entire area. Positional Normalization (PN), on the other hand, only normalizes over a single spatial position at a time. A natural compromise is to normalize features by local context, while also taking into account group level information. In this paper, we propose* Local Context Normalization (LCN)*: a normalization layer where every feature is normalized based on a window around it and the filters in its group. We propose an algorithmic solution to make LCN efficient for arbitrary window sizes, even if every point in the image has a unique window. LCN outperforms its Batch Normalization (BN), GN, IN, and LN counterparts for object detection, semantic segmentation, and instance segmentation applications in several benchmark datasets, while keeping performance independent of the batch size and facilitating transfer learning.*

## 1. Introduction

A variety of neural network normalization layers have been proposed in the literature to aid in convergence and sometimes even add desirable inductive bias.

Batch Normalization (BN) is a subtractive and divisive feature normalization scheme widely used in deep learning architectures [12]. Recent research has shown that BN facilitates convergence of very deep learning architectures by smoothing the optimization landscape [32]. BN normalizes the features by the mean and variance computed within a mini-batch. Using the batch dimension while calculating the normalization statistics has two main drawbacks:

- Small batch sizes affect model performance because the mean and variance estimates are less accurate.
- Batches might not exist during inference, so the mean and variance are pre-computed from the training set and used during inference. Therefore, changes in the target data distribution lead to issues while performing transfer learning, since the model assumes the statistics of the original training set [27].

To address both of these issues, Group Normalization (GN) was recently proposed by Wu and He [40]. GN divides channels into groups and normalizes the features by using the statistics within each group. GN does not exploit the batch dimension so the computation is independent of batch sizes and model performance does not degrade when the batch size is reduced. GN shows competitive performance with respect to BN when the batch size is small; consequently, GN is being quickly adopted for computer vision tasks like segmentation and video classification, since batch sizes are often restricted for those applications. When the batch size is sufficiently large, BN still outperforms GN.

BN, GN, IN, and LN all perform "global" normalization where spatial information is not exploited, and all features are normalized by a common mean and variance value. We argue that for the aforementioned applications, local context matters. To incorporate this intuition we propose *Lo-*

*Work partially done while author was interning at Microsoft Research
†Correspondence: jojic@microsoft.com, anthonymlortiz@gmail.com

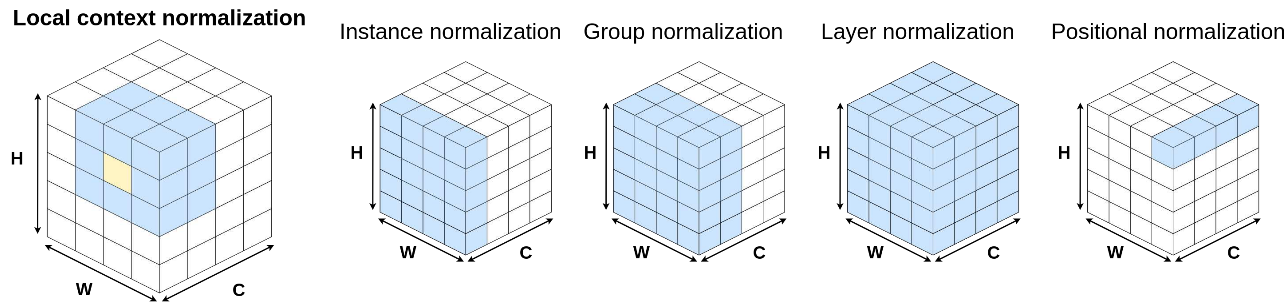


Figure 1: **Proposed *Local Context Normalization* (LCN) layer**. LCN normalizes each value in a channel according to the values in its feature group and spatial neighborhood. The figure shows how our proposed method compares to other normalization layers in terms of which features are used in normalization (shown in blue), where **H**, **W**, and **C** are the height, width, and number of channels in the output volume of a convolutional layer.

*cal Context Normalization* (LCN) as a normalization layer which takes advantage of the context of the data distribution by normalizing each feature based on the statistics of its local neighborhood and corresponding feature group. LCN is in fact inspired by computational neuroscience, specifically the *contrast normalization* approach leveraged by the human vision system [21], as well as early generative modeling approaches to co-segmentation [14, 15, 39], where the reasoning about pixel labels is based on shared self-similarity patterns within an image or image window, rather than on shared features across images. LCN provides a performance boost over all previously-proposed normalization techniques, while keeping the advantages of being computationally agnostic to the batch size and suitable for transfer learning. We empirically demonstrate the performance benefit of LCN for object detection as well as semantic and instance segmentation.

Another issue with GN is that because it performs normalization using the entire spatial dimension of the features, when it is used for inference in applications where input images need to be processed in patches, just shifting the input patch for a few pixels produces different predictions. This is a common scenario in geospatial analytics and remote sensing applications where the input tends to cover an immense area [28, 23]. Interactive fine-tuning applications like [29] become infeasible using GN, since a user will not be able to recognize whether changes in the predictions are happening because of fine-tuning or simply because of changes in the image input statistics. With LCN, predictions depend only on the statistics within the feature neighborhood; inference does not change when the input is shifted.

## 2. Related Work

**Normalization in Neural Networks.** Since the early days of neural networks, it has been understood that input normalization usually improves convergence [18, 17]. LeCun et al. showed that convergence in neural networks is faster if the average of each input variable to any layer is close to zero and their covariances are about the same [17]. Many normalization schemes have been proposed in the literature since then [21, 13, 16, 12, 19, 37, 40]. A Local Contrast Normalization Layer was introduced by [13], later referred to as Local Response Normalization (LRN). A modification of this original version of LRN was used by the original AlexNet paper which won the Imagenet [7] challenge in 2012 [16], as well as the 2013 winning entry [42]. Most popular deep learning architectures until 2015 including Overfeat and GoogLeNet [34, 36] also used LRN, which normalizes based on the statistics in a very small window (at most $9 \times 9$) around each feature.

After Ioffe et al. proposed BN in 2015, the community moved towards global normalization schemes where the statistics are computed along entire spatial dimensions [12]. BN normalizes the feature maps of a given mini-batch along the batch dimension. For convolutional layers the mean and variance are computed over both the batch and spatial dimensions, meaning that each location in the feature map is normalized in the same way. Mean and variance are pre-computed on the training set and used at inference time, so when presented with any distribution shift in the input data, BN produces inconsistency at the time of transfer or inference [27]. Reducing the batch size also affects BN performance as the estimated statistics are less accurate.

Other normalization methods [37, 40, 19] have been proposed to avoid exploiting the batch dimension. LN [19] performs normalization along the channel dimension, IN [37] performs normalization for each sample, and GN uses the mean and variance from the entire spatial dimension and a group of feature channels. See Figure 1 for a visual representation of different normalization schemes. Instead of operating on features, Weight Normalization (WN) normalizes the filter weights [31]. These strategies do not suffer from the issues caused by normalizing along the batch dimension, but they have not been able to approach BN performance in most visual recognition applications. Wu and He recently proposed GN, which is able to match BN performance on some computer vision tasks when the batch size is small [40]. All of these approaches perform global normalization, which might wipe out local context. Our proposed LCN takes advantages of both local context around the features and improved convergence from global normalization methods.

**Contrast Enhancement.** In general, contrast varies widely across a typical image. Contrast enhancement is used to boost contrast in the regions where it is low or moderate, while leaving it unchanged where it is high. This requires that the contrast enhancement be adapted to the local image content. Contrast normalization is inspired by computational neuroscience models [13, 21] and reflects certain aspects of human visual perception. This inspired early normalization schemes for neural networks, but contrast enhancement has not been incorporated into recent normalization methods. Perin et al. showed evidence for synaptic clustering, where small groups of neurons (a few dozen) form small-world networks without hubs [25]. For example, in each group, there is an increased probability of connection to other members of the group, not just to a small number of central neurons, facilitating inhibition or excitation within a whole group. Furthermore, these cell assemblies are interlaced so that together they form overlapping groups. Such groups could in fact implement LCN. These groups could also implement more extreme color and feature invariance as in probabilistic index map (PIM) models [14, 39, 15], which assume that the spatial clustering pattern of features (segmentation) is shared across images but the palette (feature intensities in each cluster) can vary freely. PIMs are naturally suited to co-segmentation applications. LCN also emphasizes local similarities among pixel features, but preserves some intensity information, as well.

Local contrast enhancement has been applied in computer vision to pre-process input images [26, 33] ensuring that contrast is normalized across a very small window ($7 \times 7$ or $9 \times 9$ traditionally). Local contrast normalization was essential for the performance of the popular Histogram of Oriented Gradients (HOG) feature descriptors [6]. In this work, we propose applying a similar normalization not only at the input layer, but in all layers of a neural network, to groups of neurons.

## 3. Local Context Normalization

### 3.1. Formulation

**Local Normalization** In the LRN scheme proposed by [13], every feature $x_{i,h,w}$ – where i refers to channel i and h,w refer to spatial position of the feature – is normalized by equation 1, where $W_{pq}$ is a Gaussian weighting window of size $9 \times 9$, $\sum_{pq} W_{pq} = 1$, $c$ is set to be $mean(\sigma_{hw})$, and $\sigma_{hw}$ is the weighted standard deviation of all features over a small spatial neighborhood. $h$ and $w$ are spatial coordinates, and $i$ is the feature index.

$$\widehat{x}_{ihw} = \frac{x_{ihw} - \sum_{pq} W_{pq} x_{i,h+p,w+q}}{max(c, \sigma_{hw})} \tag{1}$$

**Global Normalization** Most recent normalization techniques, including BN, LN, IN, and GN, apply global normalization. In these techniques, features are normalized following equation 2.

$$\widehat{x}_i = \frac{x_i - \mu_i}{\sigma_i} \tag{2}$$

For a 2D image, $i = (i_B, i_C, i_H, i_W)$ is a 4D vector indexing the features in $(B, C, H, W)$ order, where $B$ is the batch axis, $C$ is the channel axis, and $H$ and $W$ are the spatial height and width axes. $\mu$ and $\sigma$ are computed as:

$$\begin{aligned} \mu_i &= \frac{1}{m} \sum_{k \in S_i} x_k \\ \sigma_i &= \sqrt{\frac{1}{m} \sum_{k \in S_i} (x_k - \mu_i)^2 + \epsilon} \end{aligned} \tag{3}$$

with $\epsilon$ as a small constant. $S_i$ is the set of pixels in which the mean and standard deviation are computed, and $m$ is the size of this set. As shown by [40], most recent types of feature normalization methods mainly differ in how the set $S_i$ is defined. Figure 1 shows graphically the corresponding set $S_i$ for different normalization layers.

For BN, statistics are computed along $(B, H, W)$:

$$\text{BN} \implies S_i = \{k | k_C = i_C\} \tag{4}$$

For LN, normalization is performed per-sample, within each layer. $\mu$ and $\sigma$ are computed along $(C, H, W)$:

$$\text{LN} \implies \mathcal{S}_i = \{k | k_B = i_B\}, \tag{5}$$

For IN, normalization is performed per-sample, per-channel. $\mu$ and $\sigma$ are computed along (H,W):

$$\text{IN} \implies \mathcal{S}_i = \{k | k_B = i_B, k_C = i_C\}, \tag{6}$$

For GN, normalization is performed per-sample, within groups of size $G$ along the channel axis:

$$\text{GN} \implies \mathcal{S}_i = \{k | k_B = i_B, \lfloor \frac{k_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor\}, \tag{7}$$

All global normalization schemes (GN, BN, LN, IN) learn a per-channel linear transformation to compensate for the change in feature amplitude:

$$y_i = \gamma \widehat{x}_i + \beta \tag{8}$$

where $\gamma$ and $\beta$ are learned during training.

**Local Context Normalization** In LCN, the normalization statistics $\mu$ and $\gamma$ are computed following equation 2 using the set $S_i$ defined by 9. We propose performing the normalization per-sample, within a window of size $p \times q$, for groups of filters of size predefined by the number of channels per group ($c\_group$) along the channel axis, as shown in equation 9. instead of number of groups $G$ like commonly done for GN, we use ($c\_group$) as hyper-parameter. We consider windows much bigger than the ones used in LRN and can compute $\mu$ and $\gamma$ in a computationally efficient manner. The size $p$ and $q$ should be adjusted according to the input size and resolution and can be different for different layers of the network.

$$\text{LCN} \implies \mathcal{S}_i = \{k | k_B = i_B, \lfloor \frac{k_C}{c\_group} \rfloor = \lfloor \frac{i_C}{c\_group} \rfloor, \lfloor \frac{k_H}{p} \rfloor = \lfloor \frac{i_H}{p} \rfloor, \lfloor \frac{k_W}{q} \rfloor = \lfloor \frac{i_W}{q} \rfloor \}, \quad (9)$$

**Relation to Previous Normalization Schemes** LCN allows an efficient generalization of most previously proposed mini-batch-independent normalization layers. Like GN, we perform per-group normalization. If the chosen $p$ is greater than or equal to $H$ and the chosen $q$ is greater than or equal to $W$, LCN behaves exactly as GN, but keeping the number of channels per group fixed throughout the network instead of the number or groups. If in that scenario the number of channels per group (c_group) is chosen as the total number of channels (c_group = C), LCN becomes LN. If the number of channels per group (c_group) is chosen as 1 (c_group = 1), LCN becomes IN.

## 3.2. Implementation

LCN can be implemented easily in any framework with support for automatic differentiation like PyTorch [24] and TensorFlow [2]. For an efficient calculation of mean and variance, we used the summed area table algorithm, also known in computer vision as the integral image trick [38], along with dilated convolutions [41, 3]. Algorithm 1 shows the pseudo-code for the implementation of LCN. We first create two integral images using the input features and the square of the input features. Then, we apply dilated convolution to both integral images with proper dilation (dilation depends on c_group, p, and q), kernel and stride of one. This provides us the sum and sum of squares tensors for each feature $x_{ihw}$ within the corresponding window and group. From the sums and sum of square tensors we obtain mean and variance tensors needed to normalize the input features. Note that the running time is constant with respect to the window size making LCN efficient for arbitrarily large windows.

**Algorithm 1** LCN pseudo-code

**Input:** $x$: input features of shape [B, C, H, W],
$c\_group$: number of channels per group ,
$window\_size$: spatial window size as a tuple (p, q),
$\gamma$, $\beta$: scale and shifting parameters to be learned
**Output:** $\{y = LCN_{\gamma,\beta}(x)\}$

$S \leftarrow dilated\_conv(I(x), d, k)$ `/* I(x) is integral image of x, dilation d is (c_group,p,q), kernel k is a tensor with -1 and 1 to substract or add dimension */`

$S_{sq} \leftarrow dilated\_conv(I(x_{sq}), d, k)$ `// I(`$x_{sq}$`) is integral image of` $x_{sq}$

$\mu \leftarrow \frac{S}{n}$ `// Compute Mean` $n = c\_group * p * q$

$\sigma^2 \leftarrow \frac{1}{n}(S_{sq} - \frac{S \odot S}{n})$ `// compute Variance`

$\widehat{x} \leftarrow \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}}$ `// Normalize activation`

$y \leftarrow \gamma \widehat{x} + \beta$ `// Apply affine transform`

# 4. Experimental Results

In this section we evaluate our proposed normalization layer for the tasks of object detection, semantic segmentation, and instance segmentation in several benchmark datasets, and we compare its performance to the best previously known normalization schemes.

## 4.1. Semantic Segmentation on Cityscapes

Semantic segmentation consists of assigning a class label to every pixel in an image. Each pixel is typically labeled with the class of an enclosing object or region. We test for semantic segmentation on the Cityscapes dataset [5] which contains 5,000 finely-annotated images. The images are divided into 2,975 training, 500 validation, and 1,525 testing images. There are 30 classes, 19 of which are used for evaluation.

**Implementation Details.** We train state-of-the-art HRNetV2 [35] and HRNetV2-W18-Small-v1 networks as baselines [1]. We follow the same training protocol as [35]. The data is augmented by random cropping (from 1024 $\times$ 2048 to 512 $\times$ 1024), random scaling in the range of [0.5, 2], and random horizontal flipping. We use the Stochastic Gradient Descent (SGD) optimizer with a base learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005. The poly learning rate policy with the power of 0.9 is used for reducing the learning rate as done in [35]. All the models are trained for 484 epochs. We train HRNetV2 using four GPUs and a batch size of two per GPU. We then substitute sync-batch normalization layers by BN, GN, LCN

[1] We used the official implementation code from: https://github.com/leoxiaobin/deep-high-resolution-net.pytorch

Table 1: Cityscapes Semantic Segmentation Performance

| Method | Normalization | mIoU Class (%) | Pixel Acc. (%) | Mean Acc. (%) |
|---|---|---|---|---|
| HRNetV2 W48 | BN | 76.22 | **96.39** | 83.73 |
| HRNetV2 W48 | GN | 75.08 | 95.84 | 82.70 |
| HRNetV2 W48 | LCN (ours) | **77.49** | 96.14 | **84.60** |
| HRNetV2 W18 Small v1 | BN | 71.27 | **95.36** | 79.49 |
| HRNetV2 W18 Small v1 | IN | 69.74 | 94.92 | 77.77 |
| HRNetV2 W18 Small v1 | LN | 66.81 | 94.51 | 75.46 |
| HRNetV2 W18 Small v1 | GN | 70.31 | 95.03 | 78.99 |
| HRNetV2 W18 Small v1 | LCN (ours) | **71.77** | 95.26 | **79.72** |
| $\Delta$ GN | | 1.46 | 0.23 | 0.73 |

and compare results. We do exhaustive comparisons using HRNetV2-W18-Small-v1, which is a smaller version of HRNetV2; all training details are kept the same except for the batch size, which is increased to four images per GPU for faster training.

**Quantitative Results.** Table 1 shows the performance of the different normalization layers on the Cityscapes validation set. In addition to the mean of class-wise intersection over union (mIoU), we also report pixel-wise accuracy (Pixel Acc.) and mean of class-wise pixel accuracy (Mean Acc.).

We observe that our proposed normalization layer outperforms all other normalization techniques including BN. LCN is almost 1.5% better than the best GN configuration in terms of mIoU. For LCN, c_group was chosen as 2, with a window size of 227 $\times$ 227 ($p$ = $q$ = 227) for HRNetV2 W18 Small v1 and 255 $\times$ 255 for HRNetV2 W48. For GN, we tested different numbers of groups as shown in Table 2, and we report the best (using 16 groups) for comparison with other approaches in Table 1. Table 2 shows that GN is somewhat sensitive to the number of groups, ranging from 67% to 70.3% mIoU. Table 2 also shows results for IN and LN, both of which perform worse than the best GN performance. These results were obtained using HRNetV2-W18-Small-v1 network architecture. It is important to mention that we used the same learning rate values to train all models, which implies that LCN still benefits from the same fast convergence as other global normalization techniques; this is not true for local normalization schemes such as LRN, which tend to require lower learning rates for convergence.

**Sensitivity to Number of Channels per Group.** We tested the sensitivity of LCN to the number of channels per group (c_group) parameter by training models for different values of c_group while keeping the window size fixed to 227 $\times$ 227 ($p$ = $q$ = 227). Table 3 shows the performance of LCN for the different number of channels per group, which is fairly stable among all configurations.

**Sensitivity to Window Size.** We also tested how LCN performance varies with respect to changes in window size while keeping the number of channels per group fixed. The results are shown in Table 4. The bigger the window size is the closer LCN gets to GN. When the window size (p, q) is equal to the entire spatial dimensions LCN becomes GN. From Table 4 we see how performance decreases as the window size gets closer to the GN equivalent.

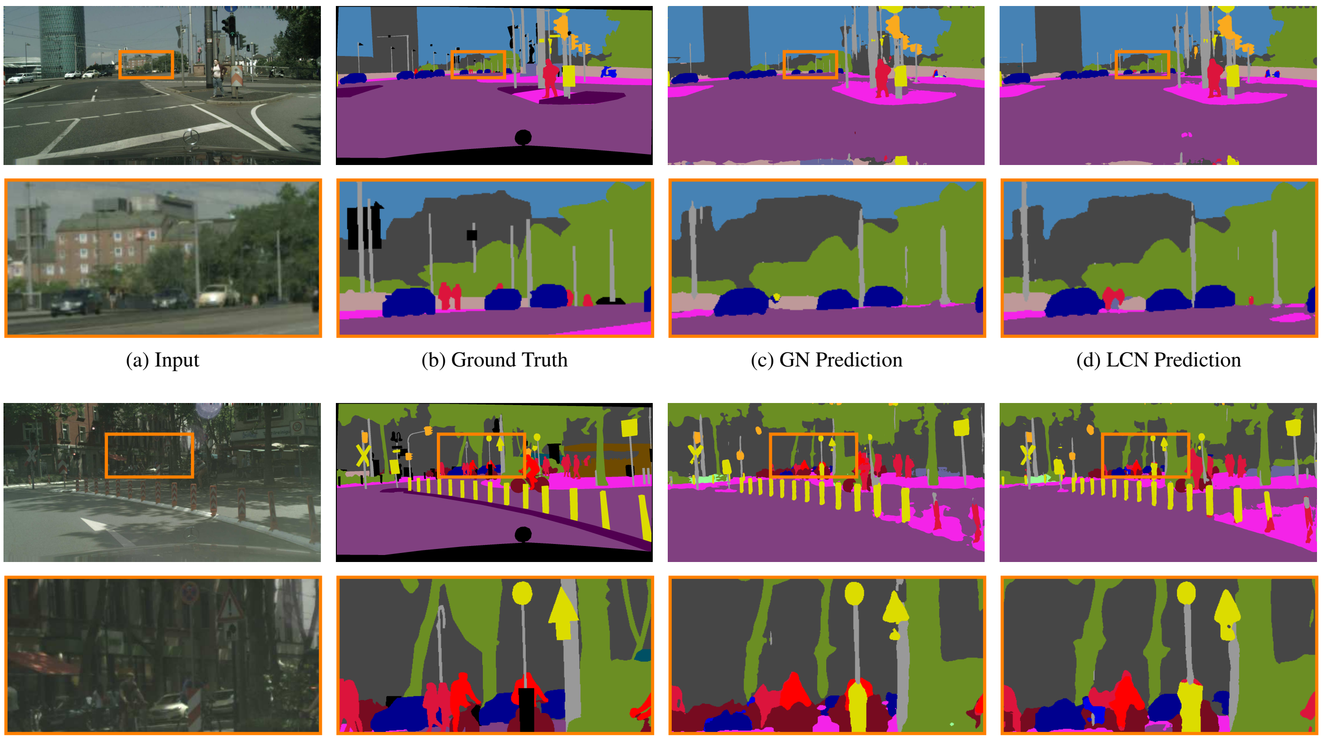
**Qualitative Results** Figure 2 shows two randomly selected examples of the semantic segmentation results obtained from HRNetV2-W18-Small-v1 using GN (last column) and LCN (second-to-last column) as the normalization layers. The second and fourth rows are obtained by maximizing the orange area from the images above them. By zooming in and looking at the details in the segmentation results, we see that LCN allows sharper and more accurate predictions. Carefully looking at the second row, we can observe how using GN HRNet misses pedestrians, which are recognized when using LCN. From the last row, we can see that using LCN results in sharper and less discontinuous predictions. LCN allows HRNet to distinguish between the bike and the legs of the cyclist while GN cannot. LCN also provides more precise boundaries for the cars in the background than GN.

## 4.2. Object Detection and Instance Segmentation on Microsoft COCO Dataset

We evaluate our LCN against previously-proposed normalization schemes for object detection and instance segmentation. Object detection involves detecting instances of objects from a particular class in an image. Instance segmentation involves detecting and segmenting each object in an image. The Microsoft COCO dataset [20] is a high-quality dataset which provides labels appropriate for both detection and instance segmentation and is the standard dataset for both tasks. The annotations include both pixel-level segmentation masks and bounding boxes for objects belonging to 80 categories.

Table 2: GN Performance for Different Numbers of Groups

| Method | Number of Groups | mIoU Class (%) | Pixel Acc. (%) | Mean Acc. (%) |
|---|---|---|---|---|
| HRNetV2 W18 Small v1 | 1 (=LN) | 66.81 | 94.51 | 75.46 |
| HRNetV2 W18 Small v1 | 2 | 69.28 | 94.78 | 77.39 |
| HRNetV2 W18 Small v1 | 4 | 67.00 | 94.50 | 76.13 |
| HRNetV2 W18 Small v1 | 8 | 67.67 | 94.76 | 75.81 |
| HRNetV2 W18 Small v1 | 16 | **70.31** | **95.03** | **78.99** |
| HRNetV2 W18 Small v1 | C (=IN) | 69.74 | 94.92 | 77.77 |



(a) Input (b) Ground Truth (c) GN Prediction (d) LCN Prediction

(e) Input (f) Ground Truth (g) GN Prediction (h) LCN Prediction

Figure 2: **Qualitative results on Cityscapes.** Going from left to right, this figure shows: **Input**, **Ground Truth**, **Group Norm Predictions**, and **Local Context Norm Predictions**. The second and fourth rows are obtained by maximizing the orange area from the images above. We observe how LCN allows the model to detect small objects missed by GN and offers sharper and more accurate predictions.

These computer vision tasks in general benefit from higher-resolution input. We experiment with the Mask R-CNN baselines [9], implemented in the publicly available Detectron codebase. We replace BN and/or GN by LCN during finetuning, using the model pre-trained from ImageNet using GN. We fine-tune with a batch size of one image per GPU and train the model using four GPUs.

The models are trained in the COCO [20] train2017 set and evaluated in the COCO val2017 set (a.k.a. minival). We report the standard COCO metrics of Average Precision (AP), $AP_{50}$, and $AP_{75}$, for both bounding box detection ($AP^{bbox}$) and instance segmentation ($AP^{mask}$).

Table 5 shows the performance of the different normalization techniques[2]. LCN outperforms both GN and BN by a substantial margin in all experiments, even using hyper-parameters tuned for the other schemes.

### 4.3. Image Classification in ImageNet

We also experiment with image classification using the ImageNet dataset [7]. In this experiment, images must be classified into one of 1000 classes. We train on all training images and evaluate on the 50,000 validation images, using the ResNet models [11].

[2]Our results differ slightly from the ones reported in the original paper, but this should not affect the comparison across normalization schemes.

Table 3: LCN sensitivity to number of channels per group for a fixed window size (227, 227)

| Method | Channels per Group | mIoU Class (%) | Pixel Acc. (%) | Mean Acc. (%) |
|---|---|---|---|---|
| HRNetV2 W18 Small v1 | 2 | **71.77** | **95.26** | **79.72** |
| HRNetV2 W18 Small v1 | 4 | 70.26 | 95.07 | 78.49 |
| HRNetV2 W18 Small v1 | 8 | 70.14 | 94.97 | 78.11 |
| HRNetV2 W18 Small v1 | 16 | 70.11 | 94.78 | 79.10 |

Table 4: LCN sensitivity to Window Size

| Method | Window Size | mIoU Class (%) | Pixel Acc. (%) | Mean Acc. (%) |
|---|---|---|---|---|
| HRNetV2 Small v1 | 199 | 71.55 | 95.18 | **79.89** |
| HRNetV2 Small v1 | 227 | 71.77 | **95.26** | 79.72 |
| HRNetV2 Small v1 | 255 | **71.80** | 95.18 | 79.26 |
| HRNetV2 Small v1 | 383 | 70.09 | 95.06 | 77.64 |
| HRNetV2 Small v1 | 511 | 70.03 | 95.09 | 77.94 |
| HRNetV2 Small v1 | all/GN | 70.30 | 95.04 | 78.97 |

Table 5: Detection and Instance Segmentation Performance on the Microsoft Coco Dataset

| Method | $\text{AP}^{bbox}$ (%) | $\text{AP}^{bbox}_{50}$ (%) | $\text{AP}^{bbox}_{75}$ (%) | $\text{AP}^{mask}$ (%) | $\text{AP}^{mask}_{50}$ (%) | $\text{AP}^{mask}_{75}$ |
|---|---|---|---|---|---|---|
| R50 BN | 37.47 | 59.15 | 40.76 | 34.06 | 55.74 | 36.04 |
| R50 GN | 37.34 | 59.65 | 40.34 | 34.33 | 56.53 | 36.31 |
| R50 LCN (Ours) | **37.90** | **59.82** | **41.16** | **34.50** | **56.81** | **36.43** |

Table 6: Image Classification Error on Imagenet

| Network Architecture | Normalization | Top 1 Err. (%) | Top 5 Err. (%) |
|---|---|---|---|
| Resnet 50 | BN | 23.59 | 6.82 |
| Resnet 50 | GN | 24.24 | 7.35 |
| Resnet 50 | LCN | 24.23 | 7.22 |

**Implementation Details.** As in most reported results, we use eight GPUs to train all models, and the batch mean and variance of BN are computed within each GPU. We use He's initialization [10] to initialize convolution weights. We train all models for 100 epochs, and decrease the learning rate by $10\times$ at 30, 60, and 90 epochs.

During training, we adopt the data augmentation of Szegedy et al. [36] as used in [40]. We evaluate the top-1 classification error on the center crops of $224 \times 224$ pixels in the validation set. To reduce random variations, we report the median error rate of the final five epochs [8].

As in [40] our baseline is the ResNet trained with BN [11]. To compare with GN and LCN, we replace BN with the specific variant. We use the same hyper-parameters for all models. We set the number of channels per group for LCN as 32, and we used $p = q = 127$ for the window size parameters. Table 6 shows that LCN offers similar performance as GN, but we do not see the same boost in performance observed for object detection and image segmentation. We hypothesize that this happens because image classification is a global task which might not benefit from local context.

### 4.4. Systematic Generalization on INRIA Aerial Imagery Dataset

The INRIA Aerial Image Labeling Dataset was introduced to test generalization of remote-sensing segmentation models [22]. It includes imagery from 10 dissimilar urban areas in North America and Europe. Instead of splitting adjacent portions of the same images into training and test sets, the splitting was done city-wise. All tiles of five cities were included in the training set and the remaining ones are used as the test set. The imagery is orthorectified [22] and has a spatial resolution of 0.3m per pixel. The dataset covers 810 $km^2$ (405 $km^2$ for training and 405 $km^2$ for the test set). Images were labeled for the semantic classes of building and non-building.

**Implementation Details.** We trained different versions of U-Net [30] where just the normalization layer was changed. We trained all models in this set of experiments using $572 \times 572$ randomly sampled patches from all training image tiles. We used the Adam optimizer with a batch size of 12. All networks were trained from scratch with a starting learning

Table 7: Performance in Inria Aerial Image Labeling Dataset. LCN outperforms all the other normalization layers overall.

| Method | Bellingham | | Bloomington | | Innsbruck | | San Francisco | | East Tyrol | | **Overall** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IoU | Acc. | IoU | Acc. | IoU | Acc. | IoU | Acc. | IoU | Acc. | IoU | Acc. |
| U-Net + BN | **65.37** | **96.53** | 55.07 | 95.83 | 67.62 | 96.08 | 72.80 | 91.00 | 67.00 | 96.91 | 67.98 | 95.27 |
| U-Net + GN | 55.48 | 93.38 | 55.47 | 94.41 | 58.93 | 93.77 | 72.12 | 89.56 | 62.27 | 95.73 | 63.71 | 93.45 |
| U-Net + LCN | 63.61 | 96.26 | **60.47** | **96.22** | **68.99** | **96.28** | **75.01** | **91.46** | **68.90** | **97.19** | **69.90** | **95.48** |

rate of 0.001. We keep the same learning rate for the first 60 epochs and decay it to 0.0001 over the next 40 epochs. Every network was trained for 100 epochs. In every epoch 8,000 patches are seen. Binary cross-entropy loss was used as the loss function.

Table 7 summarizes the performance of the different normalization layers in the INRIA aerial image labeling dataset. Our proposed LCN outperforms all the other normalization layers with an overall mIoU almost 2% higher than the next-best normalization scheme, and more than 6% better than GN in terms of overall IoU. LCN provides much better performance than other methods in almost every test city. LCN was trained using a $91 \times 91$ window size and four channels per group.

### 4.5. Land Cover Mapping

Table 8: Landcover Mapping Tested on Maryland 2013 Test

| Method | mIoU (%) | Pixel Acc. (%) |
|---|---|---|
| UNet + BN | 76.69 | 87.15 |
| UNet + GN | 74.15 | 85.18 |
| UNet + LCN | 76.51 | 86.96 |

Finally, we evaluate LCN on a land cover mapping task previously studied in [28, 1]. Land cover mapping is a semantic image segmentation task where each pixel in an aerial or satellite image must be classified as belonging to one of a variety of land cover classes. This process of turning raw remotely sensed imagery into a summarized data product is an important first step in many downstream sustainability related applications. For example, the Chesapeake Bay Conservancy uses land cover data in a variety of settings including determining where to target planting riparian forest buffers [4]. The dataset can be found at [1] and contains 4-channel (red, green, blue, and near-infrared), 1m resolution imagery from the National Agricultural Imagery Program (NAIP) and dense pixel labels from the Chesapeake Conservancy's land cover mapping program over 100,000 square miles intersecting 6 states in the northeastern US. We use the Maryland 2013 subset - training on the 50,000 multi-spectral image patches, each of size $256 \times 256 \times 4$, from the train split. We test over the 20 test tiles[3]. Each pixel must be classified as: water, tree canopy / forest, low vegetation / field, or impervious surfaces.

[3]Consisting of $\sim 900,000,000$ pixels

**Implementation Details** We trained different versions of U-Net architecture used on [28] for different normalization layers without doing any data augmentation and compared results. We used the Adam optimizer with a batch size of 96. All networks were trained from scratch for 100 epochs with a starting learning rate of 0.001 with decay to 0.0001 after 60 epochs. The multi-class cross-entropy loss was used as criterion. The best GN results are obtained using 8 groups. LCN results are obtained using 4 channels per group and and a $31 \times 31$ window.

Table 8 shows the mean IoU and Pixel Accuracy of the different normalization layers for land cover mapping. LCN outperforms GN for this task with performance slightly lower than BN. We notice that LCN benefits from larger input images. When input images are small like in this setting the performance boost from using LCN becomes smaller.

## 5. Discussion and Conclusion

We proposed *Local Context Normalization* (LCN) as a normalization layer where every feature is normalized based on a window around it and the filters in its group. We empirically showed that LCN outperforms all previously-proposed normalization layers for object detection, semantic segmentation, and instance image segmentation across a variety of datasets. The performance of LCN is invariant to batch size, and it is well-suited for transfer learning and interactive systems.

We note that we used hyper-parameters which were already highly optimized for BN and/or GN without tuning, so it is likely that we could obtain better results with LCN by just searching for better hyper-parameters. In our experiments we also do not consider varying the window size for different layers in the network, but it is a direction worth exploring: adjusting the window size during training via gradient descent may further improve performance for LCN.


## Acknowledgement

The authors thank Lucas Joppa and the Microsoft AI for Earth initiative for their support. A.O. was supported by the Army Research Office under award W911NF-17-1-0370. We thank Nvidia Corporation for the donation of two Titan Xp GPUs used for this research.

## References


[1] Chesapeake land cover. Maryland split. 8

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 4

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *arXiv preprint arXiv:1412.7062*, 2014. 4

[4] Chesapeake Conservancy. Land cover data project 2013/2014. `https://chesapeakeconservancy.org/conservation-innovation-center/high-resolution-data/land-cover-data-project/`, 2016. 8

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 4

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE conference on computer vision and pattern recognition*, 2005. 3

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2, 6

[8] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training Imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 7

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 6

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 7

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Cision and Pattern Recognition*, pages 770–778, 2016. 6, 7

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the International Conference in Machine Learning (ICML)*, 2015. 1, 2

[13] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153. IEEE, 2009. 2, 3

[14] Nebojsa Jojic and Yaron Caspi. Capturing image structure with probabilistic index maps. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004. 2, 3

[15] Nebojsa Jojic, Alessandro Perina, Marco Cristani, Vittorio Murino, and Brendan Frey. Stel component analysis: Modeling spatial correlations in image class structure. In *2009 IEEE conference on computer vision and pattern recognition*, pages 2044–2051. IEEE, 2009. 2, 3

[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. 2

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[18] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 1998. 2

[19] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

[20] 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*, pages 740–755. Springer, 2014. 5, 6

[21] Siwei Lyu and Eero P Simoncelli. Nonlinear image representation using divisive normalization. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 2, 3

[22] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2017. 7

[23] Anthony Ortiz, Alonso Granados, Olac Fuentes, Christopher Kiekintveld, Dalton Rosario, and Zachary Bell. Integrated learning and feature selection for deep neural networks in multispectral images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1196–1205, 2018. 2

[24] 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-W*, 2017. 4

[25] Rodrigo Perin, Thomas K Berger, and Henry Markram. A synaptic organizing principle for cortical neuronal groups. *Proceedings of the National Academy of Sciences*, 108(13):5419–5424, 2011. 3

[26] Nicolas Pinto, David D Cox, and James J DiCarlo. Why is real-world visual object recognition hard? *PLoS computational biology*, 4(1):e27, 2008. 3

[27] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516, 2017. 1, 2

[28] Caleb Robinson, Le Hou, Kolya Malkin, Rachel Soobitsky, Jacob Czawlytko, Bistra Dilkina, and Nebojsa Jojic. Large scale high-resolution land cover mapping with multi-resolution data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12726–12735, 2019. 2, 8

[29] Caleb Robinson, Anthony Ortiz, Kolya Malkin, Blake Elias, Andi Peng, Dan Morris, Bistra Dilkina, and Nebojsa Jojic. Human-machine collaboration for fast land cover mapping. *AAAI Conference on Artificial Intelligence (AAAI 2020)*, 2020. 2

[30] 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*, pages 234–241. Springer, 2015. 7

[31] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016. 2

[32] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018. 1

[33] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *2012 21st International Conference on Pattern Recognition (ICPR 2012)*, pages 3288–3291. IEEE, 2012. 3

[34] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2

[35] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. High-resolution representations for labeling pixels and regions. *arXiv preprint arXiv:1904.04514*, 2019. 4

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 2, 7

[37] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 2

[38] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. 2001. 4

[39] John Winn and Nebojsa Jojic. Locus: Learning object classes with unsupervised segmentation. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 756–763. IEEE, 2005. 2, 3

[40] Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision*, pages 3–19. Springer, 2018. 1, 2, 3, 7

[41] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 4

[42] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 2