Global Sum Pooling: A Generalization Trick for Object Counting with Small Datasets of Large Images

Shubhra Aich and Ian Stavness
Department of Computer Science
University of Saskatchewan, Canada
{s.aich,ian.stavness}@usask.ca

Abstract

In this paper, we explore the problem of training one-look regression models for counting objects in datasets comprising a small number of high-resolution, variable-shaped images. We illustrate that conventional global average pooling (GAP) based models are unreliable due to the patchwise cancellation of true overestimates and underestimates for patchwise inference. To overcome this limitation and reduce overfitting caused by the training on full-resolution images, we propose to employ global sum pooling (GSP) instead of GAP or fully connected (FC) layers at the backend of a convolutional network. Although computationally equivalent to GAP, we show through comprehensive experimentation that GSP allows convolutional networks to learn the counting task as a simple linear mapping problem generalized over the input shape and the number of objects present. This generalization capability allows GSP to avoid both patchwise cancellation and overfitting by training on small patches and inference on full-resolution images as a whole. We evaluate our approach on four different aerial image datasets – two car counting datasets (CARPK and COWC), one crowd counting dataset (ShanghaiTech; parts A and B) and one new challenging dataset for wheat spike counting. Our GSP models improve upon the state-of-the-art approaches on all four datasets with a simple architecture. Also, GSP architectures trained with smaller-sized image patches exhibit better localization property due to their focus on learning from smaller regions while training.

1. Introduction

Increasingly complex and large deep learning architectures are being devised to tackle challenging computer vision problems, such as object detection and instance segmentation with hundreds of object classes [18, 21, 7]. However, it is becoming common to deploy highly complex state-of-the-art architectures to solve substantially simpler tasks. Object counting is one such task: counting cars on a freeway or in a parking lot, counting people in a crowd, and counting plants or trees from aerial images. While it is possible to apply very powerful instance segmentation [12] or object detection [29] approaches to counting problems, these architectures require detailed (and time-consuming and tedious to collect) annotations, such as instance segmentation masks or bounding boxes. However, object counting is amenable to weaker labels, such as dot annotations (one dot per instance) or a scalar count per image. Devising simpler deep learning models for less complex computer vision tasks has the benefit of less costly ground-truth labeling, smaller sized networks, more efficient training, and faster inference.

One-look regression models are a class of deep neural network that are well matched to the comparatively simpler problem of object counting. These models use a convolutional front-end combined with fully-connected (FC) or global average pooling (GAP) layers that end in a single unit to generate a scalar count of the number of object instances present in the image [3, 38, 2, 9]. Other variants of this counting network use a final classification layer, where the number of the output units are slightly more than the maximum number of possible object instances in the input [24]. This requires that the maximum number of object instances are known *a priori*, which may be difficult when the number of objects varies with the size of the input. Therefore, in this paper, we focus only on the single output unit models for object counting.

Counting datasets have two common characteristics that complicate the training of one-look models. First, the training set typically consists of a few very high-resolution images. Despite the computational complexity, it might be possible to train on full-sized images as a whole, but there is a high probability of overfitting by blindly memorizing the scalar counts because of the small number of training samples available. Second, images with variable resolution in a single dataset are prevalent because they are often stitched or cropped to a particular region of interest. Many architectures require a pre-defined size for training/test images, and warping images to that pre-defined resolution could make small object instances almost disappear or large object instances become unrealistically large. In this way, downsampling and/or warping of training images can make the counting problem harder by increasing the ratio of resolution of larger to smaller instances present in the image dataset.

A common solution to overcome the challenge of high-resolution, variable-sized images is to use smaller sized, randomly cropped “patches” from the high-resolution raw training images to train the network. Dot annotations can be counted to create an object count per patch, but because the full extent of the object is not annotated, it is not possible to generate patches without partially cutting the objects at the edge of the patch. This type of label noise may be acceptable during training, but at test time, when a total count is required for the high-resolution image, tiled patches would need to be applied to the network with global average pooling (GAP) layers in the backend and the counts per tile summed. We demonstrate later in the experiments section that using GAP with fixed-resolution, smaller patches incorporates both per-patch underestimates and overestimates. Aggregating the counts of all the patches in a single image randomly cancels-out a large number of such overestimates and underestimates; thus giving an apparent impression of a reasonably accurate measure for each image, but obscuring substantial per-patch inference errors. The extent of patchwise cancellation of positive errors by negative errors is random and depends on the pattern of the collocation of the object instances in the images. Also, for rectangular patches, per-patch overestimates and underestimates increase for images with objects not oriented in a spatially vertical or horizontal fashion. We hypothesize that all these shortcomings make the use of patchwise inference with GAP unreliable for counting objects with small datasets of high-resolution images. Previous work has attempted to resolve these patch-wise inference errors empirically, by optimizing the stride of tiled patches based on validation set performance [24]. However, this does not address the fundamental limitation of partial object instances; resulting in unavoidable per-patch counting errors, which are then propagated to the estimate of the full image count.

Considering the complications of datasets with a small number of high-resolution variable-sized images, an ideal solution would be a particular kind of model that can be trained with small-sized random patches (to reduce the risk of overfitting or memorization) and then generalize its performance over arbitrarily large resolution test samples. In this paper, we devise such a model using a set of traditional convolutional and pooling layers in the front-end and replacing the fully connected (FC) layers or global average pooling (GAP) layer with the new global sum pooling (GSP) operation. We show that the use of this GSP

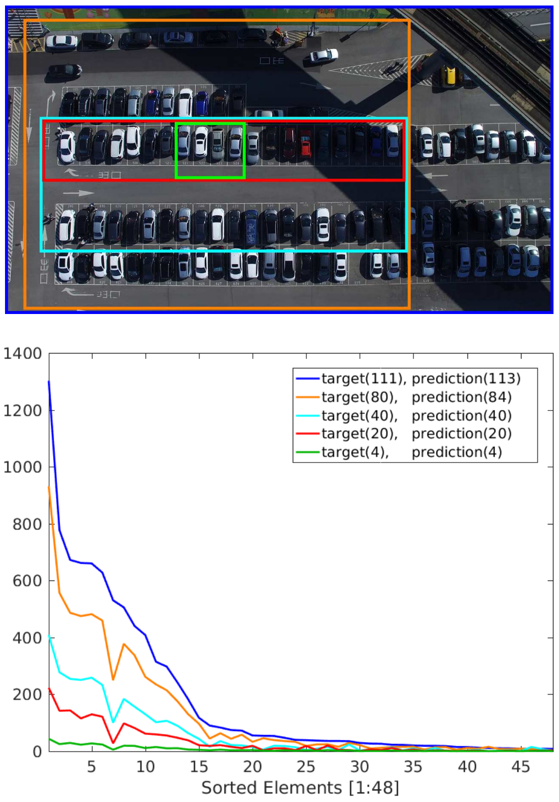

Figure 1. (Left) Sample image with multiple cropping shown using bounding boxes with different colors. (Right) Activations of the first 48 elements sorted in descending order incurred by these cropped samples after GSP operation shown using the corresponding colors of the bounding boxes in the left. For consistency, sorting indices of the full-resolution input are used to sort others. The plot of the values demonstrates the fact of learning a linear mapping of the object counts by our GSP-CNN model regardless of input shape.

layer allows the network to train on image patches and infer accurate object counts on full sized images. Although from a computational perspective, the summation operation in GSP is very similar to the averaging operation in GAP, GSP exhibits the non-trivial property of generalization for counting objects over variable input shapes, which GAP does not. To the best of our knowledge, this is the first work introducing the GSP operation as a replacement of GAP or FC layers. We evaluated GSP models on four different datasets — two for counting cars, one for crowd counting, and one for counting wheat spikes. Our experimental results demonstrate that GSP helps to generate more localized activations on object regions (Figure 2) and achieve better generalization performance which is consistent with our hypothesis.

Our paper makes the following contributions:

- We describe the limitations of existing architectural designs for object counting on datasets with fewer, high-resolution training samples. With extensive experimentation, we demonstrate the patchwise cancellation effect on per-patch overestimates and underestimates of using global average pooling (GAP) with tiled patches. We argue that such randomization makes GAP a fragile candidate for counting architectures.

- We propose global sum pooling (GSP) as an alternative to GAP or FC layers as a remedy to the problems regarding variable resolution images, and overfitting on small datasets of large images. We demonstrate that GSP models can be trained on smaller random patches and used for inference on full resolution images. This is because the GSP models learn a mapping linear to the number of objects regardless of input resolution, which is impossible with GAP or FC layer models.
- We benchmark GSP on four heterogeneous (two car parks, one crowd counting, and one wheat spike counting) datasets. Our GSP model beats state-of-the-art approaches with much better saliency mapping on all these datasets. Results were obtained with a simple convolutional front-end which demonstrates the simplicity and efficacy of GSP for object counting.

2. Related Work

Much of the recent literature on object counting is based on estimating different kinds of activation maps because these approaches are applicable to datasets with high-resolution images. Lempitsky and Zisserman [19] incorporate the idea of per-pixel density map estimation followed by regression for object counting. This regression approach is further enhanced by [5] by adding an interactive user interface. Fiaschi et al. [10] employ random forest to regress the density map and object count. Fully convolutional network [41] is also used for contextual density map estimation irrespective of the input shape. Proximity map, which is the proximity to the nearest cell center, is also estimated in [42] as an alternative to traditional density map approximation. Another variant of density map is proposed in the Count-ception paper [8], where the authors use fully convolutional network [22] to regress the count map followed by scalar count retrieval adjusting the redundant coverage proportional to the kernel size. Wheat spike images have been previously investigated for controlled imaging environments using density maps [26].

There also exists an extensive body of work on crowd counting [1]. Here, we review some of the recent CNN based approaches. Wang et al. [40] employed a one-look CNN model first on dense crowd counting. Zhang et al. [43] developed the *cross-scene* crowd counting approach. They use alternative optimization criteria for counting and density map estimation. Also, instead of single Gaussian kernels to generate a ground truth density map, they use multiple kernels along with the idea of perspective normalization. Cross-scene adaptation is done by finetuning the network with training samples similar to test scenes. Similar to the gradient boosting machines [11], Walach and Wolf [39] iteratively add additional computational blocks in their architecture to train on the residual error of the previous block, which they call layered boosting. Shang et al. [33] use an LSTM [14] decoder on GoogLeNet [36] features to extract a patchwise local count and generate a global count from them using FC layers. CrowdNet [6] uses the combination of shallow and deep networks to acquire multi-scale information in density map approximation for crowd counting. Another approach [44] for multi-scale context aggregation for density map estimation use multi-column networks with different kernel sizes. Hydra CNN [25] employ three convolutional heads to process image pyramids and combine their outputs with additional FC layers to approximate the density map at a lower resolution. Switching CNN architecture [32] proposes a switching module to decide among different sub-networks to process images with different properties.

Most of the approaches described above attempt to approximate a final activation map under different names, i.e. density map, count map, and proximity map, which is then post-processed to obtain the count information; thus resulting in a multi-stage pipeline. In this regard, one-look models are simpler and faster than these map estimation approaches. The main idea behind using one-look regression models [3, 2, 40, 38, 9, 24] for object counting is to utilize weaker ground truth information like dot annotations, in contrast to more sophisticated models for object detection [29, 27] or instance-level segmentation [12, 30, 28] that require stronger and more tedious to collect ground truth labels. The domain knowledge of spatial collocation of cars in the car parks is exploited in the layout proposal network [15] to detect and count cars. The COWC dataset paper [24] uses multiple variants of the hybrid of residual [13] and Inception [37] architectures, called ResCeption, as the one-look model for counting cars patchwise. During inference, the authors determine the stride based on the validation set. This kind of hybrid models are also used in [2] to estimate plant characteristics from images. However, recent work on heatmap regulation (HR) [4] describes the philosophical limitation of using one-look models and tries to improve its performance by regulating the final activation map with a Gaussian approximation of the ground-truth activation map. In this paper, using GSP and training with smaller samples, we obtain similar final activation maps to HR without using any extra supervision channel in our model.

3. Our Approach

To overcome the generalization challenges for object counting from a small number of high-resolution, variable

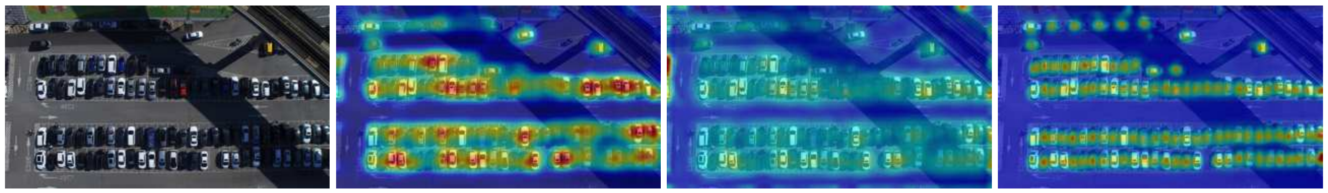

Figure 2. Sample image for car counting [15] along with superimposed activation heatmaps for different one-look regression models, from left-to-right: original image, the baseline GAP model, our GSP model trained with full-resolution images, and GSP trained with 224×224 randomly cropped patches.

sized images, and to avoid the problem of partial object counting while cropping random patches, we propose an architecture that can learn to count from images regardless of their shape. Architectures with final FC layers pose strict requirements about the input images' shape, whereas architectures that combine CNN with additional nonlinear and normalization layers are more flexible. We take inspiration from recent image classification architectures [20, 13, 37, 16] that replace FC layers with a simple GAP layer. Using GAP greatly reduces the number of parameters (to help reduce overfitting), emphasizes the convolutional front-end of the models, permits training and testing on variable size input images, and provides intuitive visualizations of the activation maps [45]. For an object counting task, however, the averaging operation of GAP lacks the ability to generalize over variable resolution input images.

The difference between GAP and GSP for object counting can be illustrated by a hypothetical example. For simplicity of illustration, here we consider an ideal environment where the resolution of object instances falls within a fixed range over a dataset, but this is not a requirement for the GSP approach. Also, without loss of generality, we assume that objects are uniformly distributed, which means that the number of objects within an image is expected to scale with the image resolution. For example, if a $W \times W$ region contains $\mathcal{C}$ objects, then a $mW \times mW$ region would be expected to contain m^2W objects (m is a scalar). If we train a network containing a stack of convolution layers followed by a GAP layer on $W \times W$ samples, our models will learn to generate the expected count of $\mathcal{C}$ with an equivalent vector representation as the output of GAP. During inference, with a $mW \times mW$ image, the last convolution layer will generate m^2 adjacent, spatial feature responses, each representing the expected count of $\mathcal{C}$. This convolutional representation is appropriate to predict an expected count of $m^2\mathcal{C}$. However, the GAP layer will average over all the m^2 spatial sub-regions and obtain an equivalent representation of $\mathcal{C}$. Hence, the averaging operation is not suitable for modeling the proportional scaling of the number of objects with the size of input.

Another option might be to divide the variable resolution input images into fixed size, adjacent, and non-overlapping patches during inference and then sum up the count over all of the filled patches to retrieve the final count. Computationally, such tiling preserves the efficiency of inference in a single pass. Inference on overlapping patches with density estimation is also possible [8], but quadratically increases the computational complexity. Any patch-wise inference scheme, however, has a significant limitation that is unavoidable from the modeling perspective. During inference, the network produces both overestimates and underestimates over all the patches extracted from a single image. For example, for a single image, the amount of overestimates and underestimates are $\mathcal{E}_{\mathcal{O}}$ and $\mathcal{E}_{\mathcal{U}}$, respectively. Although the actual difference between ground truth count and prediction is $\mathcal{E}_{\mathcal{O}} + \mathcal{E}_{\mathcal{U}}$, by summing up the patch counts, we get an apparent error of $|\mathcal{E}_{\mathcal{O}} - \mathcal{E}_{\mathcal{U}}|$. Thus, the measured difference on the whole image, in this case, depends on the difference between overestimate $\mathcal{E}_{\mathcal{O}}$ and underestimate $\mathcal{E}_{\mathcal{U}}$, not on their absolute value. For example, we will get a very low error even if $\mathcal{E}_{\mathcal{O}}$ and $\mathcal{E}_{\mathcal{U}}$ are quite high but are almost equal. Thus, when aggregating the patch count, overestimate and underestimate get nullified by each other randomly on which the model has no control. We call this effect "patchwise cancellation". The amount of such patchwise cancellation depends on various properties of the dataset, such as the density and types of the objects, their collocation patterns, their comparative resolution in the image, and so on. In the experiments section, we show that although for GAP models used on adjacent patches for inference, the difference between ground truth and summed up prediction is sometimes reasonably small, the actual overestimate and underestimate are pretty high, validating our explanation of patchwise cancellation. Therefore, GAP with adjacent patches is not a reliable solution for inference on variable-sized, high-resolution images.

Instead of average-pooling the final feature maps, we propose a summation or mere aggregation of the input over the spatial locations only. From the previous example, this aggregation of m^2 similar sub-regions, each with a count of $\mathcal{C}$, would produce the desired expected value of $m^2\mathcal{C}$. Following the nomenclature of GAP, we call this operation global sum pooling (GSP). Although GAP and GSP are computationally similar operations, conceptually GSP provides the ability to use CNN architectures for generalized training and inference on variable shaped inputs in a

Table 1. Statistics of the datasets used for evaluation

Dataset	#Images (Train, Test)	Resolution	Total Count (Train, Test)	Range of Count (Train, Test)
CARPK	(989, 459)	(720 × 1280)	(42274, 47500)	([1, 87], [2, 188])
ShanghaiTech-A	(300, 182)	(200×300) – (1024×992)	(162413, 78862)	([33, 3138], [66, 2256])
ShanghaiTech-B	(400, 316)	(768 × 1024)	(49151, 39121)	([12, 576], [9, 539])
COWC	(32, 20)	∼(18k × 18k) – (2k × 2k)	(37890, 3456)	([45, 13086], [10, 881])
Wheat-Spike	(10, 10)	∼(1000, 3000)	(10112, 9989)	([796, 1287], [749, 1205])

simple and elegant way. Moreover, due to the single pass inference regardless of input resolution, GSP does not suffer from patchwise cancellation.

Linear mapping: Learning to count regardless of the input image shape necessarily means that the convolutional front-end of the network should learn a linear mapping task, where the output vector of GSP will scale proportionally with the number of objects present in the input image. Figure 1 shows a sample 720 × 1280 image from the CARPK [15] aerial car counting dataset. On the right of Figure 1, we plot the largest 48 activations of the 512-vector output of the GSP layer of our model described later, for different-sized sub-regions of the same sample image. Here, the elements are sorted in descending order for the full resolution image, and the same ordering is used for the activations of the sub-regions. The model producing these activations was trained on 224 × 224 randomly cropped samples. From this figure, it is evident that our model is able to learn a linear mapping function from the image space to the high-dimensional feature space, where the final count is a simple linear regression or combination of the extracted feature values.

Weak instance detector and region classifier: An advantage of training on small input sizes is that it guides the network to behave like a weak object instance detector even though we only provide weak labels (a scalar count per image region). Training on sub-regions of a large input image helps the network to better disambiguate the true object regions from the object-like background sub-regions, resulting in improved performance. For example, when training the network with full images, all of which have a non-zero object count, the network never faces a complete background sample from which it can extract background information similar to any binary region classification problem. On the other hand, when we train with small randomly-cropped regions of the input image many background-only samples are fed to the network, instantiating a more rigorous learning paradigm even with weak count labels. Class-activation map (CAM) [45] visualizations illustrate that the GSP model trained with small sub-regions better captures localization information (Figure 2). Training the GAP or GSP models with full-resolution images results in a less uniform distribution of activation among object regions and less localized activations inside object regions as compared to the GSP model trained with smaller patches.

Architecture: We attach a GSP layer after the convolutional front-end of VGG16 [34] model pretrained on ImageNet [31]. GSP produces a 512-dimensional vector, which is converted to a scalar count by a linear layer. We faced no problems with the potential numerical instability caused by large, unnormalized values after spatial summation, even when training the GSP models with full resolution images.

4. Experiments

Datasets: We evaluate object counting with GSP on four datasets: CARPK [15] (overhead view of different car parks), ShanghaiTech [44] (crowd images collected from the web and streets of Shanghai), COWC [24] (overhead view of cars in residential areas and highways), and a wheat spike (WS) dataset [17] (overhead view of mature wheat plants). CARPK and ShanghaiTech-B contain constant resolution images, whereas ShanghaiTech-A, COWC, and WS have large images with variable resolutions. All datasets have comparatively few training and test images. Statistics of these datasets are listed in Table 1.

Metrics: We adopt the evaluation metrics from MCNN [44] and COWC [24] papers along with one additional metric: the percentage of MAE over expected ground truth, which we call the relative MAE (%RMAE) (Equation 1).

$$\begin{cases} \textit{Mean Absolute Error (MAE)} = \frac{\sum_i |\hat{y}_i - y_i|}{N} \\ \textit{\%MAE} = \frac{\sum_i |\hat{y_i} - y_i|}{N y_i} \times 100 \\ \textit{Relative MAE (\%RMAE)} = \frac{MAE \times N}{\sum_i y_i} \times 100 \\ \textit{Root-Mean-Square Error (RMSE)} = \sqrt{\frac{\sum_i (\hat{y}_i - y_i)^2}{N}} \\ \textit{\%RMSE} = \sqrt{\frac{\sum_i (\hat{y}_i - y_i)^2}{N y_i^2}} \times 100 \end{cases} \tag{1}$$

Models & Training: We train both GSP and GAP models on full-resolution images and on randomly cropped patches of resolutions 224, 128, 96, and 64. For GSP models, inference is done on full resolution images regardless of their shape and input size used at training, which is not possible for GAP models. For GAP models, we provide error metrics in two forms. First, we report errors over the cumulative patch counts that we denote by *GAP-C (GAP-Cumulative)*. However, as described before, such error is a misinterpretation of the actual per patch error of the GAP models. Therefore, another error is estimated per tiled patch and all the per patch errors over the single im-

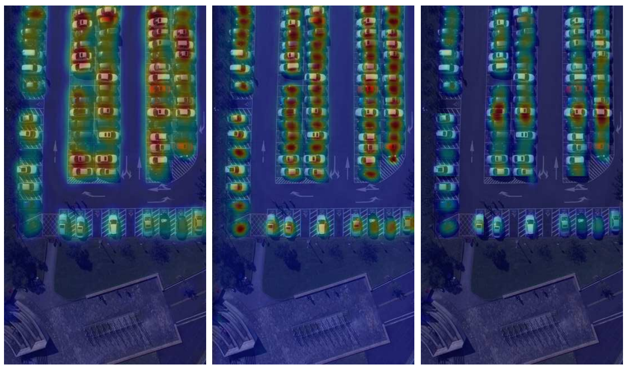
Figure 3. Activation maps for CARPK generated by the GAP-Full (left), GSP-224 (middle), and GAP-224(right) models. Activations are more uniformly distributed and more concentrated inside object regions for the GSP-224 model.

Table 2. GSP-GAP comparison on CARPK dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
Full	GAP	19.61	21.65	23.78	42.87	18.95
	GSP	32.94	36.23	39.46	70.91	31.83
224	GAP-C	7.65	9.59	9.34	15.65	7.39
	GAP-PS	19.20	21.42	19.01	26.38	16.67
	GSP	**5.46**	**8.09**	**12.21**	**44.36**	**5.28**
128	GAP-C	8.66	11.30	11.90	26.64	8.37
	GAP-PS	22.14	25.94	21.43	34.05	17.88
	GSP	6.70	10.21	8.74	19.15	6.48
96	GAP-C	10.72	13.63	17.23	49.35	10.36
	GAP-PS	44.41	48.49	39.23	54.88	32.89
	GSP	10.63	11.37	22.27	62.87	10.27
64	GAP-C	23.20	27.78	42.16	115.61	22.42
	GAP-PS	52.81	57.24	52.51	110.44	34.98
	GSP	32.09	36.02	31.64	34.39	31.01

Table 3. Results on CARPK dataset

Method	MAE	RMSE
YOLO [27, 15]	48.89	57.55
Faster R-CNN [29, 15]	47.45	57.39
One-Look Regression [24, 15]	59.46	66.84
LPN [15]	13.72	21.77
HR [4]	7.88	9.30
Ours (GSP-224)	**5.46**	**8.09**

age are summed up under the tag of *GAP-PS (GAP-Patch-Summed)*.

In order to train on image patches, we compute a count per patch based on the number of central object regions within the patch. The CARPK dataset provides bounding boxes, which we shrink down to 25% along each dimension and to define a central region for each car instance. The shrinking prevents object regions from overlapping and makes it so that we only count objects that are mostly inside the cropped patch. The ShanghaiTech, COWC, and Wheat-Spike datasets provide dot annotations appropriate to train our models.

CARPK dataset: For this car park dataset, we found that GSP models trained with 128 $\times$ 128 and 224 $\times$ 224 samples perform much better than the same model trained with smaller patches, like 64 $\times$ 64 and 96 $\times$ 96 (Table 3). The reason for GAP-C showing apparently lower error is the patchwise cancellation of patchwise overestimate and underestimate of GAP models which is evident from the numerics of GAP-PS.

Figure 3 compares CAM heatmaps superimposed on original images for the baseline GAP-Full model and our best performing GSP-N model and GAP-N (N=224 for both). The activation maps of the GAP model are variable over the object regions, indicating that some of the objects are being highly emphasized than others, whereas the GSP-224 activations are more uniform, showing that all the instances are getting more or less equal attention from the network. Moreover, the GSP-224 activations better localized within object sub-regions than GAP model, which demonstrates that GSP-N models with small N work as a better object detector or binary region classifier than the baseline models.

We believe that the poor performance of GSP-N models for smaller N (64 and 96) is not a characteristic of the model itself. Instead, the poor performance can be attributed to the training procedure that we followed in this paper. As already stated, we shrink the bounding boxes for CARPK dataset to disambiguate the overlapping bounding boxes. However, such shrinking poses restrictions on using arbitrarily small sample sizes in training. If the patch size is close to the object resolution (the average resolution of the bounding boxes in the training set of CARPK dataset is about 40 pixels) and the objects are close together (which cars are in a parking lot), a patch is likely to include one complete object with several other instances partially cut at the edge of the patch. Because we disambiguate object counts by shrinking the boxes, depending on the relative orientation between the object and its encompassing box, and its portion inside the cropped patch, it might be taken into account for counting or not. Therefore, this aspect of our training paradigm is a bit randomized. For comparatively larger sample size, such as 128 and 224, we anticipate that this problem of random consideration of the partial objects in the border is less frequent than the smaller sized patches, such as 64 and 96. In this regard, the optimal sample size depends on the average resolution of the object instances and their relative placement in the images of a particular dataset.

ShanghaiTech dataset: This is the largest crowd counting dataset in terms of the number of counts (Table 1). It comprises two parts – part A images are randomly collected from the web and part B is acquired from the busy streets of Shanghai. Table 4 and 5 enlist the comparative performance of GSP and GAP models on these subsets. Table 6 reports the comparison with state-of-the-art approaches.

Our GSP-224 (part A) and GSP-128 (part B) models outperform state-of-the-performance approaches. Note that,

Table 4. GSP-GAP comparison on ShanghaiTech-A dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
Full	GAP	143.13	199.79	43.21	64.84	33.09
	GSP	153.38	259.04	31.98	38.91	35.46
224	GAP-C	83.50	124.39	23.01	34.89	19.31
	GAP-PS	104.13	140.67	28.03	37.37	24.08
	GSP	**70.69**	**103.58**	**19.66**	**29.37**	**16.34**
128	GAP-C	83.75	124.79	23.28	34.27	19.36
	GAP-PS	113.31	148.58	30.63	38.50	26.20
	GSP	71.19	111.86	18.73	29.28	16.46
96	GAP-C	86.81	124.88	23.31	30.71	20.07
	GAP-PS	127.63	162.94	33.35	37.79	29.51
	GSP	78.25	116.69	19.87	27.21	18.09
64	GAP-C	100.69	140.53	26.87	33.80	23.28
	GAP-PS	160.38	197.79	42.16	45.99	37.08
	GSP	107.81	151.72	29.98	37.97	24.93

Table 5. GSP-GAP comparison on ShanghaiTech-B dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
Full	GAP	12.91	20.19	13.44	21.66	10.45
	GSP	12.26	19.49	12.01	19.56	9.92
224	GAP-C	9.70	16.03	7.69	10.16	7.84
	GAP-PS	18.44	24.04	15.87	17.38	14.91
	GSP	9.96	16.67	8.07	10.71	8.06
128	GAP-C	10.24	16.81	8.43	11.34	8.29
	GAP-PS	24.05	30.31	21.24	22.76	19.45
	GSP	**9.13**	**15.94**	**7.05**	**9.24**	**7.39**
96	GAP-C	11.16	17.60	9.31	11.93	9.03
	GAP-PS	28.73	34.74	25.49	26.76	23.24
	GSP	9.48	15.40	7.70	10.21	7.67
64	GAP-C	15.94	21.57	15.69	18.69	12.90
	GAP-PS	39.06	46.88	35.30	36.75	31.60
	GSP	14.35	22.95	12.44	15.85	11.61

Table 6. Results on ShanghaiTech dataset

Method	Part A		Part B	
	MAE	RMSE	MAE	RMSE
CS-CNN [43]	181.8	277.7	32.0	49.8
MCNN [44]	110.2	173.2	26.4	41.3
FCN [23]	126.5	173.5	23.76	33.12
Cascaded MTL [35]	101.3	152.4	20.0	31.1
Switching-CNN [32]	90.4	135.0	21.6	33.4
Ours (GSP-224 and -128)	**70.7**	**103.6**	**9.1**	**15.9**

Table 7. GSP-GAP comparison on COWC dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
224	GAP-C	17.54	22.98	36.47	50.05	10.15
	GSP	8.85	13.01	10.70	14.99	5.12
128	GAP-C	20.05	43.18	13.89	17.72	11.60
	GSP	8.45	13.09	12.22	17.84	4.89
96	GAP-C	15.45	25.64	10.44	11.99	8.94
	GSP	**8.20**	**12.53**	11.13	16.38	**4.75**
64	GAP-C	24.34	45.16	19.30	24.09	14.09
	GSP	11.15	23.61	**5.72**	**8.43**	6.45

although GAP models apparently provide good accuracy, their per patch error is pretty high indicating a considerable amount of patchwise cancellation. This is also evident from Figure 4. In this figure, both the best performing GSP and GAP models show similar saliency maps validating our claim that patchwise cancellation is heavily responsible for the comparatively poor performance of GAP models.

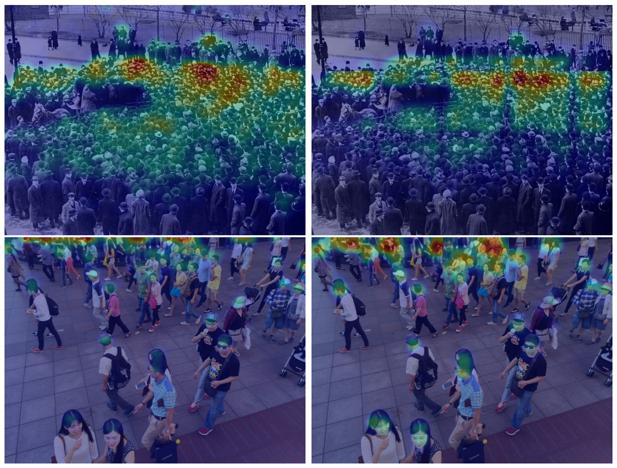
Figure 4. (Left) Saliency maps generated by the best GSP models on ShanghaiTech-A (top) and -B (bottom) datasets. (Right) Same for the best GAP models. GSP and GAP models exhibit similar activations, except GSP models are free from *patchwise cancellation* effect due to single inference on full image.

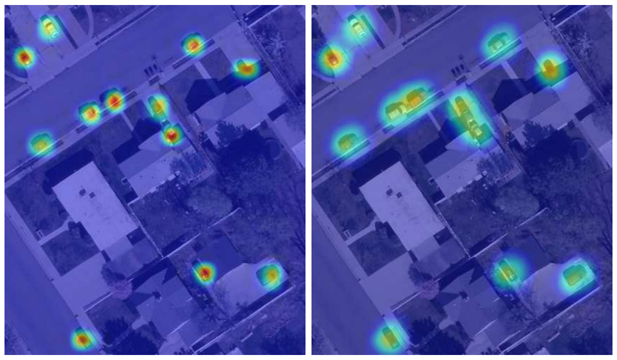
Figure 5. Superimposed activation maps for GSP-64 (left) and GSP-224 (right) on the cropped image of COWC dataset. Activations are better localized the GSP-64 model.

Table 8. Results on COWC dataset

Method	%MAE	%RMSE
ResCeption [24]	5.78	8.09
ResCeption taller 03 [24]	6.14	7.57
Ours (GSP-64)	**5.72**	8.43

COWC dataset: COWC contains very few training images (32) and the image sizes vary substantially (2220×2220 to 18400×18075), therefore it is an ideal test case for the main features of GSP. Unlike the parking lot datasets, the COWC dataset contains images covering highways and residential areas and therefore cars in these images often appear to be entirely isolated objects in the roads or highways or parked in the residential streets. Each pixel covers 15 cm, resulting in the resolution of the cars ranging from 24 to 48 pixels. Because of the sparsity of the objects in the ultra-high-resolution training images, we extract ~ 8000 samples of resolution 288×288 centered on

Table 9. Results on Wheat-Spike dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
Full	GAP	132.25	153.77	13.82	16.40	13.24
	GSP	161.63	178.11	16.16	17.81	16.18
224	GAP-C	82.19	92.41	8.43	9.45	8.23
	GAP-PS	189.5	195.06	17.94	18.36	17.85
	GSP	108.19	134.01	10.37	12.38	10.83
128	GAP-C	91.00	106.07	9.45	11.07	9.11
	GAP-PS	279.75	284.67	24.93	25.08	24.92
	GSP	85.00	108.87	8.05	9.95	8.51
96	GAP-C	**75.38**	**88.23**	7.83	9.37	7.55
	GAP-PS	351.50	356.98	30.34	30.52	30.26
	GSP	80.00	100.63	7.94	9.93	8.01
64	GAP-C	87.50	99.02	9.19	10.46	8.76
	GAP-PS	514.00	519.63	41.29	41.49	41.07
	GSP	111.38	130.07	11.23	13.01	11.15

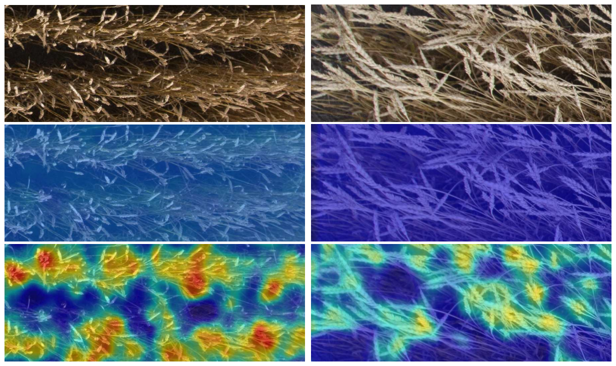
Figure 6. Cropped sample images from Wheat-Spike dataset (top) with superimposed CAM generated by GSP-Full (middle) and GSP-96 (bottom) models.

object sub-regions from the images prior to training. We do this to avoid training on a large number of negative samples that would be the case for random cropping.

For COWC, we could not provide per patch error metrics for GAP models in Table 7 since the test set contains only scalar counts as the ground truth. The smallest patch size (GSP-64) provides comparable performance to previously published results (Table 8). We also see that the activations for GSP-64 are more concentrated on the objects in the image compared to that of GSP-224 (Figure 5). This observation is consistent with our claim that the GSP models trained with smaller sample size tend to localize objects better, particularly when they are relatively isolated from each other.

Wheat-Spike dataset: This dataset [17] is a comparatively challenging one for object counting because of the irregular placement or collocation of wheat spikes. Out of 10 training samples, we use 8 for training and 2 for validation. Like COWC, the Wheat-Spike dataset is an ideal case study for GSP because of the low number of high-resolution training samples. Since the images are high-resolution and sub-regions inside a single image vary quite a bit in terms of brightness, perspective, and variable object shape resulting from natural morphology and wind motion, there are many features inside a single image that any suitable architecture should exploit without memorization or overfitting.

Table 9 reports the comparative performance of GSP and GAP models on this dataset. Although the summed up count for GAP seems to be more accurate than the corresponding GSP models, the surprisingly high aggregate of per-patch error again explains the effect of patchwise cancellation of under/overestimates with GAP models.

Also, the error for the GSP model trained with full-resolution images is quite high – 161.63, about 16% *MAE* compared to the average count of 1000. GSP-96 provides the best performance with MAE of 80.00 (8% of average count). Figure 6 shows cropped samples, their superimposed activation maps from GSP-Full model (middle), and GSP-96 (right). The GSP-96 model is able to identify salient regions in the image well, but for GSP-Full models, it tries to blindly memorize the count from only eight high-resolution images, which is clearly evident from the very uniform heatmap distribution all over the image regardless of foreground and background.

Acknowledgments

This research was undertaken thanks in part to funding from the Canada First Research Excellence Fund and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

5. Conclusion and Future work

In this paper, we introduce the global sum pooling operation as a way to train one-look counting models without overfitting on datasets containing few high-resolution images. With detailed experimental results on several datasets, we show that our GSP model, trained with small numbers of input samples, provides more accurate counting results than existing approaches. Also, when the GSP model is trained on small patches, it indirectly receives a weak supervision regarding object position and learns to localize objects better. This is also true for a GAP model, but it lacks the capability to infer counts from full-resolution test images and suffers from high per-patch errors for patchwise inference. This makes GAP unreliable for object counting on variable-sized, high-resolution images. Although we have only addressed object counting in this study, we believe that GSP could be applied to other computer vision tasks, such as classification or object detection. For these tasks, we expect that the scaling property of GSP may be able to utilize the features of image sub-regions over multiple spatial scales better than models that employ GAP or FC layers. In that case, the requirement for fixed-resolution images for many object detection or classification models can be eliminated. We plan to investigate these directions as future work.

References

[1] A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 2017. 3

[2] S. Aich, A. Josuttes, I. Ovsyannikov, K. Strueby, I. Ahmed, H. S. Duddu, C. Pozniak, S. Shirtliffe, and I. Stavness. Deepwheat: Estimating phenotypic traits from crop images with deep learning. In *WACV*, 2018. 1, 3

[3] S. Aich and I. Stavness. Leaf counting with deep convolutional and deconvolutional networks. In *ICCVW*, 2017. 1, 3

[4] Shubhra Aich and Ian Stavness. Improving object counting with heatmap regulation. *CoRR*, abs/1803.05494, 2018. 3, 6

[5] Carlos Arteta, Victor Lempitsky, J. Alison Noble, and Andrew Zisserman. Interactive object counting. In *ECCV*, 2014. 3

[6] Lokesh Boominathan, Srinivas S S Kruthiventi, and R. Venkatesh Babu. Crowdnet: A deep convolutional network for dense crowd counting. In *ACMM*, 2016. 3

[7] Holger Caesar, Jasper R. R. Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. *CoRR*, abs/1612.03716, 2016. 1

[8] J. P. Cohen, G. Boucher, C. A. Glastonbury, H. Z. Lo, and Y. Bengio. Count-ception: Counting by fully convolutional redundant counting. In *ICCVW*, 2017. 3, 4

[9] A. Dobrescu, M. V. Giuffrida, and S. A. Tsaftaris. Leveraging multiple datasets for deep leaf counting. In *ICCVW*, 2017. 1, 3

[10] L. Fiaschi, U. Koethe, R. Nair, and F. A. Hamprecht. Learning to count with regression forest and structured labels. In *ICPR*, 2012. 3

[11] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 2001. 3

[12] K. He, G. Gkioxari, P. Dollr, and R. Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 3

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 4

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 1997. 3

[15] M. R. Hsieh, Y. L. Lin, and W. H. Hsu. Drone-based object counting by spatially regularized regional proposal network. In *ICCV*, 2017. 3, 4, 5, 6

[16] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 4

[17] A. Josuttes, S. Aich, I. Stavness, C. Pozniak, and S. Shirtliffe. Utilizing deep learning to predict the number of spikes in wheat (triticum aestivum). In *Phenome 2018 Posters*, 2018. 5, 8

[18] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017. 1

[19] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *NIPS*. 2010. 3

[20] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013. 4

[21] 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 *ECCV*, 2014. 1

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 3

[23] Mark Marsden, Kevin McGuinness, Suzanne Little, and Noel E. O'Connor. Fully convolutional crowd counting on highly congested scenes. *CoRR*, abs/1612.00220, 2016. 7

[24] T. Nathan Mundhenk, Goran Konjevod, Wesam A. Sakla, and Kofi Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In *ECCV*, 2016. 1, 2, 3, 5, 6, 7

[25] Daniel Oñoro-Rubio and Roberto J. López-Sastre. Towards perspective-free object counting with deep learning. In *ECCV*, 2016. 3

[26] M. P. Pound, J. A. Atkinson, D. M. Wells, T. P. Pridmore, and A. P. French. Deep learning for multi-task plant phenotyping. In *ICCVW*, 2017. 3

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 3, 6

[28] M. Ren and R. S. Zemel. End-to-end instance segmentation with recurrent attention. In *CVPR*, 2017. 3

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *NIPS*. 2015. 1, 3, 6

[30] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. Recurrent instance segmentation. In *ECCV*, 2016. 3

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. 5

[32] D. B. Sam, S. Surya, and R. V. Babu. Switching convolutional neural network for crowd counting. In *CVPR*, 2017. 3, 7

[33] C. Shang, H. Ai, and B. Bai. End-to-end crowd counting via joint learning local and global count. In *ICIP*, 2016. 3

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 5

[35] V. A. Sindagi and V. M. Patel. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *IEEE AVSS*, 2017. 7

[36] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 3

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 3, 4

[38] Jordan R. Ubbens and Ian Stavness. Deep plant phenomics: A deep learning platform for complex plant phenotyping tasks. *Frontiers in Plant Science*, 8:1190, 2017. 1, 3

[39] Elad Walach and Lior Wolf. Learning to count with cnn boosting. In *ECCV*, 2016. 3

[40] Chuan Wang, Hua Zhang, Liang Yang, Si Liu, and Xiaochun Cao. Deep people counting in extremely dense crowds. In *ACMM*, 2015. 3

[41] Weidi Xie, J. Alison Noble, and Andrew Zisserman. Microscopy cell counting and detection with fully convolutional regression networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 2016. 3

[42] Yuanpu Xie, Fuyong Xing, Hai Su, and Lin Yang. Beyond classification: Structured regression for robust cell detection using convolutional neural network. In *MICCAI*, 2015. 3

[43] Cong Zhang, Hongsheng Li, X. Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, 2015. 3, 7

[44] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, 2016. 3, 5, 7

[45] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016. 4, 5