Increasing CNN Robustness to Occlusions by Reducing Filter Support

Elad Osherov
Technion, Israel
eladosherov@campus.technion.ac.il

Michael Lindenbaum
Technion, Israel
mic@cs.technion.ac.il

Abstract

Convolutional neural networks (CNNs) provide the current state of the art in visual object classification, but they are far less accurate when classifying partially occluded objects. A straightforward way to improve classification under occlusion conditions is to train the classifier using partially occluded object examples. However, training the network on many combinations of object instances and occlusions may be computationally expensive. This work proposes an alternative approach to increasing the robustness of CNNs to occlusion.

We start by studying the effect of partial occlusions on the trained CNN and show, empirically, that training on partially occluded examples reduces the spatial support of the filters. Building upon this finding, we argue that smaller filter support is beneficial for occlusion robustness. We propose a training process that uses a special regularization term that acts to shrink the spatial support of the filters. We consider three possible regularization terms that are based on second central moments, group sparsity, and mutually reweighted $\mathcal{L}_1$, respectively. When trained on normal (unoccluded) examples, the resulting classifier is highly robust to occlusions. For large training sets and limited training time, the proposed classifier is even more accurate than standard classifiers trained on occluded object examples.

1. Introduction

Deep Convolutional Neural Networks [14] have recently exhibited remarkable performance in the task of image classification [12]. The availability of large amounts of annotated data [3], parallel computational resources such as GPUs, and regularization techniques [20, 9, 26] have contributed greatly to CNN performance. Deeper, more sophisticated, network topologies have continuously provided state-of-the-art results [19, 21, 10].

Nevertheless, CNNs, as well as other visual classification algorithms, are far less accurate when classifying partially occluded objects; see Figure (1). The decrease in performance is especially severe when the classifier is trained

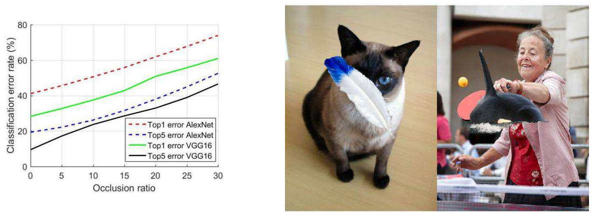

Figure 1: (**Left**) Classification error rates for AlexNet and VGG16 under occlusions of various sizes. Note that even the slightest partial occlusion may significantly reduce classification accuracy. (**Right**) Examples of partially occluded examples used for validation.

as usual, on images of mostly unoccluded objects. This problem could be regarded as a special case of domain adaptation, where the training data and the test data are drawn from different distributions.

Thus, one possible approach to improving classification accuracy under partial occlusion is to train the classifier on partially occluded objects as well [17]. In the first part of this work we experimented with this approach and indeed found that the classification accuracy improved. We also examined the resulting network and observed that (a) its filters (in all layers) tend to be of smaller spatial support, and (b) every filter uses more features from the previous layer. Intuition tells us that smaller spatial support makes the filter more robust to occlusion because the probability that a random occluder will intersect with the filter support and adversely affect its response is smaller. But does it really explain the classifier's greater robustness to occlusions?

To validate the hypothesis that smaller spatial support is indeed important for occlusion robustness, we propose, in the next part of this work, to reduce the spatial support in a different way, by specialized regularization. We propose three possible regularization terms that act to shrink the spatial support of the filters. These regularization terms are based on second central moments, group sparsity, and mutually re-weighted $\mathcal{L}_1$, respectively. When trained on normal (unoccluded) examples, the resulting classifier is usually (for large training sets) more occlusion robust than standard classifiers trained on occluded object examples. This indeed shows that smaller spatial support is beneficial for occluded object classification.

Note that the first approach requires a large number of occluded object examples in order to represent the distribution of all object instances from the class under all typical occluders. To synthesize many occluded object examples from regular images of unoccluded objects, we need to know the object location within the image, which requires a large labeling effort. Natural, unsynthesized occlusion examples are rare, requiring labeling as well. In both cases the number of examples and the computational effort are large. The proposed classifiers with the specialized regularization, on the other hand, are trainable on the usual datasets.

Our contribution in this paper is twofold: first we analyze the effect of training with occlusions on CNN visual classifiers, and in particular show the reduction of the filters' spatial support, accompanied by an increase in its effective depth. Then we show that similarly reduced spatial support of the filters may be obtained by training on unoccluded examples by special regularization. We introduce 3 different types of such regularization and show that it improves the classifier's robustness to occlusions.

The rest of this paper is organized as follows. We start by describing some related work in Section 2. Section 3 analyzes the CNN trained on occluded object examples. Section 4 introduces the occlusion-robust CNN. Finally, we describe our experiments and conclude our work in Sections 5 and 6. Derivation details and more experimental results (with AlexNet and VGG16) are described in the Appendix.

2. Related Work

Partial occlusion is one of the major challenges in visual object classification. Fukushima [5] suggests that human vision tends to struggle when the occlusion pattern is unnoticeable. He proposes a neural network algorithm that first detects the occlusion and then nullifies the corresponding activation map locations. BoW methods [2, 13, 23] generate local pieces of evidence and therefore should be, in principle, relatively robust to occlusions. Nonetheless, they often fail when the object is partially occluded. DPM algorithms [4] still respond positively when one of the parts is undetected, but their performance deteriorates. Other approaches, which relies on HOG features, model the occlusion as a binary coarse grid and explicitly infers it, using a computationally expensive algorithm [25, 6].

CNNs provide the best visual classification results but their accuracy decreases when the network that was trained on unoccluded training images is fed with partially occluded test images [17]. They found that training several sub-networks for each occlusion ratio is sub-optimal, and suggest that architectural changes may be required to improve occlusion robustness. The authors of [8] quantify the invariance of different features of the network to several transformations. They suggest that weight sparsity may contribute to occlusion invariance. A recent contribution focuses on face recognition and achieves occlusion robustness by a specialized loss that relies on classifiers that use localized information [16].

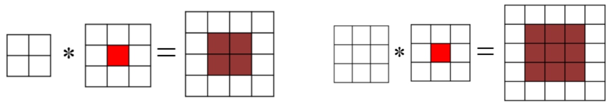

Figure 2: An illustration of how filter support amplifies partial occlusion effect. A 3×3 image with a partial occlusion (red) is filtered with a 2×2 filter. 25% of the feature map elements are corrupted. When filtered with a 3×3 filter, 36% of the feature map elements are corrupted.

3. The Effect of Occlusions on CNNs

Our first goal is to characterize the networks that are more successful with respect to recognition under partial occlusion. One example is a network trained on partially occluded training examples, which is therefore more robust to occlusions. Since CNNs are composed of linear filters, they are greatly affected by occlusions. A linear filter that sums values from all the objects' regions responds very differently when some of these values change significantly, as is the case under occlusion. Therefore an occlusion-robust classifier should use features that do not rely on the spatial support of the entire object but only on part of it. We are interested in the spatial support of the filters' receptive fields. Instead of using a strict definition of spatial support, namely the set of locations associated with nonzero filter weights, we prefer softer measures that take into consideration the magnitude and location of the filter weights and are referred to as effective spatial support; see Section 3.1.

To examine the change in effective spatial support, we trained two CNNs. One was trained in the usual way, with (unoccluded) images taken from the ImageNet data set [18]. The other was trained on both unoccluded and partially occluded images. We are interested in comparing the corresponding filters across the two networks, and more specifically the effective spatial support of their receptive fields. To establish a correspondence between the filters from both CNNs, we conducted a somewhat more complex experiment; see Section 5.1 for details. As expected, we found that networks trained with occluded examples are better able to recognize partially occluded objects. We also found that these networks are composed of filters with smaller spatial support relative to filters in networks trained as usual. Smaller spatial support means that more neurons would respond independently of the occlusion; see Figure 2. We argue that the smaller spatial support is indeed a major reason for the advantage of the occlusion trained networks and shall provide further evidence for this claim in Section 5.

In the rest of this section we elaborate on this finding, suggest several measures for spatial support, and show empirically that training with occluded examples indeed results in filters of lower spatial support.

3.1. Measures of Spatial Support

We start by briefly describing the CNN structure we use, and the associated notations. A CNN is typically composed of several convolution layers, and then several fully connected layers topped by a classifier. A convolution layer is composed of K linear filters followed by nonlinearities and optional pooling stages. Every filter is specified by a 3D matrix of size D $\times$ M $\times$ N. We will use the tensor notation [7], with a 4D $\mathbf{W}^l$ tensor, where l represents the layer number. A single element (i.e. a weight) of this tensor is denoted $\mathbf{W}^l_{kdmn}$, where k is the filter index, d is the filter depth (or input channel), and m, n are the spatial indexes.

We shall be interested in the spatial distribution of the weight values. Therefore, we focus on sub-filters of size M $\times$ N, corresponding to common k, d values. These 2D sub-filters are called kernels. We consider each kernel separately, and characterize the distribution of its weight values as described below. Then we get more concise characterizations by summing these measures over the different kernels and the different filters in each layer.

The measures of spatial distribution should not depend on the magnitude of the weights or on their signs. Therefore, we use their normalized values. Formally, let $\mathbf{W}^l_{kd::}$ be the 2D matrix of weight associated with the (k, d)-th kernel. Then, the tensor $\mathbf{V}$ of the absolute value of the filter weights and the tensor $\hat{\mathbf{V}}$ of the normalized filter weights are specified by:

$$\mathbf{V}^l_{kd::} = |\mathbf{W}^l_{kd::}|, \;\; \hat{\mathbf{V}}^l_{kd::} = \frac{|\mathbf{W}^l_{kd::}|}{||\mathbf{W}^l_{kd::}||_1}. \tag{1}$$

The resulting normalized weight kernels may be interpreted as a (2D) probability distribution.

We shall use two alternative measures for characterizing the spatial support of the filter weights.

Spatial entropy - Entropy, which is a measure of uncertainty for random variables, may be applied to positive vectors of unit $\mathcal{L}_1$ length as a diversity measure. We shall use it here as a measure for the scattering of filter weights, and refer to it as spatial entropy. Thus, the spatial entropy H_S of the (k, d)-th 2D kernel $\mathbf{W}^l_{kd::}$ is:

$$(H_S)^l_{k,d} = -\sum_{m,n} \hat{\mathbf{V}}^l_{kdmn} log(\hat{\mathbf{V}}^l_{kdmn}). \tag{2}$$

The entropy of the filter is the average of its kernel entropies, weighted by the relative $\mathcal{L}_1$ energy:

$$(H_S)^l_k = \sum_{d=1}^{D} H^l_{k,d} \cdot \frac{\|\mathbf{W}^l_{kd::}\|_1}{\|\mathbf{W}^l_{k:::}\|_1}. \tag{3}$$

(Other types of weighting could be used as well.) Note that the entropy is indifferent to permutations of the weights, and therefore it is only indicative of the spatial support. If the entropy value is large, then the distribution is closer to uniform and the support is large as well. When it is small, however, it could correspond to two significant weights in nearby locations (small spatial support) or to two significant weights in far locations (large spatial support). We call this spatial entropy because it refers to the scattering of weights along spatial coordinates in the different kernels and not to their scattering for different filters or depths.

Second central moment - Another measure that can be used to estimate the scattering of the filter weights is the second moment. Here, again, we begin by normalizing the weights so that their sum is one. Then, following rigid body mechanics, we calculate the center of mass and the second moment relative to it. A filter with substantial weights far from its center will have a large second central moment.

Consider a 2D discrete grid of masses $\{w_{mn}\}$. The ij moment is $M_{ij} = \sum_{m,n} w_{mn} m^i n^j$. Specifically, M_{00} is the zeroth moment, which is simply the sum of the weights, and M_{10}, M_{01} are the first moments. The center of mass is defined as $(\mu_x = M_{10}/M_{00}, \mu_y = M_{01}/M_{00})$. The second central moments are $C_{ij} = \sum_{m,n} w_{mn}(m-\mu_x)^i(n-\mu_y)^j$. The latter moments (known as moments of inertia in mechanics) are common measures for the concentration of mass.

We shall use the absolute and the normalized weights specified in eq.(1), and treat these filter weights as point masses. The moments and the associated centers of mass are calculated from the absolute and normalized weights separately for each kernel, and denoted $(M_{ij})^l_{kd}, (\mu_x)^l_{kd}, (\mu_y)^l_{kd}, (C_{ij})^l_{kd}$ and $\widehat{(M_{ij})}^l_{kd}, \widehat{(\mu_x)}^l_{kd}, \widehat{(\mu_y)}^l_{kd}, \widehat{(C_{ij})}^l_{kd}$ respectively.

We use the sum of the two normalized second central moments,

$$\tau^l_{kd} = \widehat{(C_{20})}^l_{kd} + \widehat{(C_{02})}^l_{kd}, \tag{4}$$

to characterize the effective spatial support of the kernel. To characterize a filter, we shall take again a weighted sum of this measure over all the kernels in the filter,

$$\tau^l_k = \sum_{d=1}^{D} \tau^l_{kd} \cdot \frac{\|\mathbf{W}_{kd::}\|_1}{\|\mathbf{W}_{k:::}\|_1}. \tag{5}$$

In principle, we could normalize the filter weights, the corresponding moments, and the weights of the kernels in eq.(5) differently, using, say, Euclidean norms. The results are similar. We kept the $\mathcal{L}_1$ norm so that the same normalization works for the spatial entropy. Interestingly, the effective spatial support τ_k^l may be written in a simplified way, using the unnormalized moments:

$$\tau_k^l = \frac{1}{\|\mathbf{W}_{k:::}\|_1} \sum_{d=1}^{D} [(C_{20})_{kd}^l + (C_{02})_{kd}^l]. \tag{6}$$

See eq.(24) in the Appendix for details; This simplification is used later in Section 4.

Compared to the entropy based measure, this measure, based on central moments, is a more direct measure of the concentration of the filter weights. It explicitly considers the spatial location associated with the weights and calculates their distance from the center of mass. A large second moment means that some non-negligible weights are far from the center of mass, implying larger effective spatial support. Therefore, we prefer this measure over the entropy based measure, but shall use both of them below to strengthen the evidence of the shrinking receptive fields.

3.2. The Effect of Occlusions on Spatial Support

We are interested in the change of effective spatial support caused by training the CNN on images of partially occluded objects. Training two CNN networks separately, one on occluded objects and one on unoccluded objects, would result in two sets of unrelated filters. To compare the spatial support of specific filters, we need to maintain a correspondence between pairs of filters, one from each network. To maintain this correspondence, we conducted a more careful experimental procedure, described in Section 5.1. This procedure produces two CNN networks with correspondence between the filters. We calculated the spatial entropy (3) and the second central moment (6) of every filter, as well as the fraction of filters for which the measures were larger in the network trained only with unoccluded examples; see Table 1 and Table6 in the appendix. The results clearly show that the spatial support tends to be smaller when training with occluded examples. Note that the differences in spatial support are actually very small. This is a result of the procedure we use and the price paid for maintaining the correspondence; see Section 5.1 for more details and for related results without filter correspondence.

As discussed above, filters with smaller support are less likely to be influenced by partial occlusion than filters with larger support. This seems to be the source of the preference for smaller support filters observed when training on occluded objects. Note, however, that filters with larger support are more sensitive, in principle, to additional object parts and details, and are therefore potentially more discriminative. Still, the results indicate that the lower sensitivity to occlusion is more significant than the decrease in discriminative power.

conv layer	1	4	7	10	12
Fraction of filters in net A with larger spatial entropy than corresponding filters in net B.	0.551	0.667	0.672	0.684	0.652
Fraction of filters in net A with larger 2nd central moments than corresponding filters in net B.	0.557	0.633	0.649	0.761	0.612

Table 1: The fraction of filters in network A, (**VGG16** trained only with unoccluded examples) that have a larger effective support than the corresponding filters in network B (trained also with occluded examples). The spatial support of the filters tends to be lower if the training images are partially occluded; see appendix A for additional results.

3.3. The Effect of Occlusions on the Effective Depth

The filter spatial support is smaller when training on occluded object images. The influence of the different kernels on the overall filter response is different as well. To measure the contribution of the various kernels to each filter, we consider the energy of the kernel weights, $||\mathbf{W}_{kd::}^l||_2^2$. The normalized energy:

$$S_{kd}^l = \frac{||\mathbf{W}_{kd::}^l||_2^2}{\sum_{i=1}^{D} ||\mathbf{W}_{ki::}^l||_2^2}, \tag{7}$$

serves as a measure of the contribution of every kernel to the filter response. Note that the normalized energy values sum to 1. The distribution of these contributions within a filter may be estimated by the depth entropy H_D, where the subscript D stands for filter depth.

$$(H_D)_k^l = \sum_{j=1}^{D} -S_{kj}^l \cdot log(S_{kj}^l). \tag{8}$$

This entropy can serve as an indication of the number of kernels that significantly influence the filter response, and we refer to it as effective depth. A filter that depends, for example, on a single kernel, would correspond to zero entropy and minimal effective depth, while a filter depending on all kernels equally would correspond to $logD$ entropy and maximal effective depth.

In the same way that we calculated the change in the spatial support, we now calculate the fraction of filters where the depth entropy (8) was larger in the network that was trained only with unoccluded examples, relative to the network that was trained also with occluded examples. The results (Table 2) indicate that the kernel energy in the network trained also on occluded examples is distributed more evenly, i.e., more kernels contribute to the response. This larger effective depth may have two related but different explanations. First, the network trained on partially occluded

conv layer	1	4	7	10	12
Fraction of filters in net A with larger depth entropy than corresponding filters in net B.	0.432	0.271	0.368	0.244	0.268

Table 2: The fraction of filters in network A, (**VGG16** trained only with unoccluded examples) that have a larger depth entropy than the corresponding filters in network B (trained also with occluded examples). When training on partially occluded examples, more kernels play a meaningful role in the feature extraction process; see appendix A for additional results.

examples uses a smaller part of the object to derive the intermediate pieces of evidence and therefore requires more of them to be discriminative. In addition, because different parts of the object may be occluded in different test images, the network should use alternative configurations of features, which again increases the diversity of the kernels.

Interestingly, this effect is more significant for the deeper layers; see Table 2. Neurons in these layers detect coarser level features or full object parts. With training under occlusion, they may be collecting alternative sets of evidence, such as different subsets of the object's parts.

4. Enforcing Small Spatial Support via Regularization

As discussed in the introduction, and empirically verified (Section 3), a straightforward way to improve classification under occlusion conditions is to train the classifier using partially occluded examples. We propose here an alternative approach: training the classifier on normal, unoccluded examples, with bias to smaller spatial support filters. This bias, which relies on the observation from the previous section, is implemented by several special regularization terms. Our motivation is two-fold:

1. Training on many combinations of object instances and occlusions is computationally expensive and/or requires detailed localization annotation; see Figures 5,3. Therefore, an algorithm trainable only on unoccluded object instances, but which still provides robustness to occlusion, is desirable.

2. While we have shown that training on partially occluded examples reduces the filters' spatial support, we did not provide evidence that this reduction contributes significantly to occlusion robustness. It could be that other properties of the network, learned from the occluded examples, provide this robustness and the reduced spatial support is only a side effect. By showing that a network trained only on unoccluded object examples is robust to occlusion, we provide direct evidence that small spatial support provides robustness.

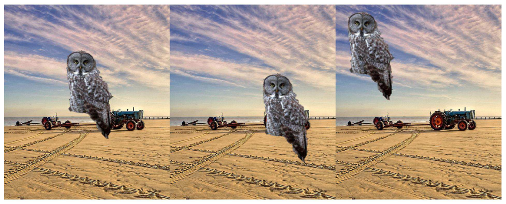

Figure 3: Different occlusion scenarios. When the object's location is not given (as is the case in ImageNet), creating synthetic partial occlusions by placing an occluder in a random location may be problematic. For example, the owl occluder, covering 10% of the full image may partially cover the tractor target, completely cover it, or not cover it as all.

4.1. Convolution layer regularization terms

We shall train CNNs composed of convolution layers and fully connected layers. The training is carried out by minimizing a loss function:

$$L = L_0 + \sum_{l=1}^{L} \lambda^l R^l, \tag{9}$$

where L_0 is the data loss function, λ^l are regularization strength coefficients, which may differ for each layer, and R^l is a regularization term for the l-th layer, which penalizes for large spatial support of the filters in this layer. We shall consider several alternative regularization options.

4.1.1 Regularization by minimizing second central moments

We first propose to use the second central moment of the different convolution filters as a regularization term. Thus, the regularization term R^l becomes:

$$R^l_{CM} = \sum_{k=1}^{K} \tau^l_k. \tag{10}$$

Following eq.(24), this term and its derivative become:

$$R^l_{CM} = \sum_{k=1}^{K} \frac{1}{||\mathbf{w}_{k:::}||_1} \sum_{d=1}^{D} [(C_{20})^l_{kd} + (C_{02})^l_{kd}]. \tag{11}$$

$$\frac{\partial R^l_{CM}}{\partial \mathbf{W}^l_{kdij}} = \left[\frac{sign(\mathbf{W}^l_{kdij})}{||\mathbf{W}_{k:::}||_1} - \frac{\mathbf{W}^l_{kdij}}{||\mathbf{W}_{k:::}||_1^2}\right] \cdot [(i-(\mu_x)^l_{kd})^2 + (j-(\mu_y)^l_{kd})^2]; \tag{12}$$

see appendix A. Note that the term $\frac{\mathbf{W}^l_{kdij}}{||\mathbf{W}_{k:::}||_1^2}$ is negligible in comparison to $\frac{sign(\mathbf{W}^l_{kdij})}{||\mathbf{W}_{k:::}||_1}$, in each case where the filter is not extremely sparse, which is a reasonable assumption. This means that the sign of the derivative is determined only by $sign(\mathbf{W}^l_{kdij})$. The term $\frac{1}{||\mathbf{w}_{k:::}||_1}$ only changes the relative importance of the regularization term within the overall loss function, and therefore may be replaced by changing λ^l. Moreover, $||\mathbf{w}_{k:::}||_1$ changes very slowly in practice. Thus, a simplified expression for the derivative, which we use in the optimization, is:

$$\frac{\partial R^l_{CM}}{\partial \mathbf{W}^l_{kdij}} \approx sign(\mathbf{W}^l_{kdij})[(i-(\mu_x)^l_{kd})^2 + (j-(\mu_y)^l_{kd})^2]. \tag{13}$$

This regularization term shrinks weights that are far from the center of mass of each kernel.

4.1.2 Regularization by enforcing group sparsity

Another way to reduce the effective support of the kernels is to shrink together specific filter weight subsets, chosen to explicitly reduce the support, instead of shrinking each filter weight independently using the usual L2 regularization. We propose to use structured sparsity regularization, and specifically group sparsity regularization (also known as group Lasso [28]), used for sparse signal representation. Sparsity with group regularization prefers not only that the number of non-zero weights will be small, but also that these weights will come from a minimal number of weight subsets (or groups); see Figure 6 in Appendix A for an illustration of the groups used in our experiments. The regularization term R^l in this case is:

$$R^l_{GS} = \sum_{k,d}\sum_{r}\sqrt{\sum_{i,j}(\mathbf{W}^l_{kd::})^2 .* G_r}, \tag{14}$$

where every G_r is an indicator matrix, the same dimension as the kernel, which indicates whether the (i,j)-th weights is a member of the group. The derivative of this term with respect to $\mathbf{W}^l_{kdij}$ is:

$$\frac{\partial R^l_{GS}}{\partial \mathbf{W}^l_{kdij}} = \sum_r \frac{\mathbf{W}^l_{kdij}G_r(i,j)}{\sqrt{\sum_{m,n}(\mathbf{W}^l_{kd::})^2 .* G_r}}. \tag{15}$$

It is possible to shrink all the kernels using several sparsity groups, using the same sparsity group, or using a different group for each one. In our implementation we randomly chose a single group for each kernel.

4.1.3 Regularization by mutually re-weighted L1

Finally, inspired by the results of [8], which showed that promoting sparsity in the network weights increases network invariance to various deformations, we suggest using a variation of weighted $\mathcal{L}_1$ regularization. It is known that $\mathcal{L}_1$ regularization induces sparsity, while retaining the desired

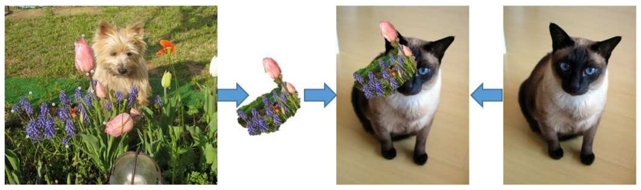

Figure 4: Generating occluded examples. (**Left**) An occluding patch is acquired from a different object category using intelligent scissors and is re-scaled to a certain occlusion ratio, with respect to the target image. (**Right**) The target image is occluded using the previously acquired occluder.

convexity attribute that the $\mathcal{L}_0$ norm lacks. Sparsity can be achieved effectively using an iterative sequence of minimizations, known as the iterative reweighted $\mathcal{L}_1$ minimization (IRWL1) [1]. Each iteration minimizes a reweighted $\mathcal{L}_1$ norm, where the weights used for the next iteration are computed from the value of the current solution. We propose to use a variation of this algorithm. The original reweighting scheme uses the size of each entry to calculate the weight of this entry (in the next iteration) while we use the other entries of the vector to calculate this weight in the next iteration. In the context of CNN kernels, for a particular filter weight X in some kernel, we use the other filter weights in the same kernel to calculate the weight of X. We call this algorithm Mutually Re-Weighted $\mathcal{L}_1$ (MRWL1) and specify the corresponding regularization term:

$$R^l_{MRWL1} = \sum_{k,d}\sum_{i,j}|\mathbf{W}^l_{kdij}| \cdot \sum_{m\neq i, n\neq j}|\mathbf{W}^l_{kdmn}|. \tag{16}$$

In comparison to $\mathcal{L}_1$ regularization, where the weights shrink at the same rate, MRWL1 constantly changes the shrinking rate for each weight, with respect to its spatial neighbors. MRWL1 tends to shrink smaller weights more rapidly then larger weights, effectively promoting sparsity. The derivative of R^l_{MRWL1} with respect to $\mathbf{W}^l_{kdij}$ is:

$$\frac{\partial R^l_{MRWL1}}{\partial \mathbf{W}^l_{kdij}} = sign(\mathbf{W}^l_{kdij}) \cdot \sum_{m\neq i, n\neq j}|\mathbf{W}^l_{kdmn}|. \tag{17}$$

Note that the weight $\sum_{m\neq i,n\neq j}|\mathbf{W}^l_{kdmn}|$ has a similar effect as the weight $1/(|\mathbf{W}^l_{kdmn}|+\epsilon)$, which would be used if we followed the reweighting proposed in [1], provided that the sum (or norm) of the weights in the kernel is constant.

4.2. Fully connected layer regularization

The first fully connected layer is the deepest layer in the CNN for which the spatial information of the input is explicit. The output of this layer lacks any direct spatial information that can be traced back to the input location of the pixels. Following the discussion in Section 3, we argue that recognition of partially occluded objects would benefit from shrinking the filters' support. The same regularization terms, proposed in Section 4.1 for the convolution filters, can also be used for the optimization of the first fully connected layer weights, while promoting smaller support and sparsity; see Section 5 for experimental results.

4.3. The learning algorithm

The regularization terms presented above are independent with respect to the different kernels, implying that training using the stochastic gradient descent algorithm is efficient. Algorithm 1 describes the SGD training process. The algorithm depends, via a parameter R, on one of the regularization terms,(10),(14) or (16), and uses the associated derivative (13),(15) or (17). The algorithm's other parameters are the learning rate function γ_t, the batch size B, and the regularization strength coefficients $\{\lambda^l\}_{l=1}^{L}$. Additional learning techniques such as momentum, $\mathcal{L}_2$ regularization and batch normalization, may be incorporated.

Algorithm 1 Small support regularized SGD

```
1: Input: training set $D = \{x_{1:N}, y_{1:N}\}$,$R$,B,$\gamma_t$, $\{\lambda^l\}_{l=1}^{L}$
2: $\mathbf{w}_{init} \leftarrow N(0, \sigma)$, $\mathbf{w}_{init} \in \mathbb{R}^{L}_{K\times D\times M\times N}$
3: while ($epoch \leq numberofepochs$) do
4:     for $n = 1...BatchNumber$ do
5:         $\mathbf{w}_n \leftarrow \mathbf{w}_{n-1} - \frac{\gamma_t}{B}\sum_{b=1}^{B}[\frac{\partial L_0(x_b)}{\partial \mathbf{w}} + \lambda^l \frac{\partial R(x_b)}{\partial \mathbf{w}}]$
6:     end for
7:     $epoch \leftarrow epoch + 1$
8: end while
9: return w
```

5. Experiments and Results

We start by describing experiments that demonstrate the influence of training under occlusion on the properties of the network, as discussed in Section 3. We then turn to evaluating the categorization accuracy of algorithms that use the proposed specialized regularization. We use three popular datasets: **CIFAR-10/CIFAR100** [11], and **ImageNet** [18].

Since, to our knowledge, there is no large scale data set that contains occlusion annotations, we generated and used several types of artificial occlusions. The occluders were crops from image of other categories obtained by running the intelligent scissors [15] in random locations. We also experimented with random rectangles and got similar results. The occluders' sizes were characterized by the ratio δ between their sizes and the entire image. Note that $\delta = 0.1$ may correspond to a square occluder whose sides equal 0.32 of the full image—not a small image part. All experiments were done using the MatConvNet toolbox [24].

5.1. Training with partially occluded examples

In the following, we elaborate on the experimental settings used for obtaining the results in Section 3, and show additional results. To evaluate the effect of training with partially occluded examples on the spatial support of the filters, we conducted two types of experiments.

In the first type, we trained 2 CNN networks as follows. Our goal here was to maintain a correspondence between the filters of the two networks. Thus, we first trained both networks identically for 15 epochs on unoccluded examples (from ImageNet [18]). We then continued the training for 5 epochs, with a learning rate that was 100 times smaller than that used in the first 15 epochs. For these last five epochs, the first network was trained on unoccluded examples but the second network was trained on partially occluded examples. In the resulting networks, the corresponding filters are not identical but essentially extracted features with the same functionality. A comparison between the properties of these filters is described in Section 3. We experimented with AlexNet [12], and VGG16 [19], both with batch normalization [10].

In the experiments of the second type, we trained the two CNNs separately, one on unoccluded examples and the other on occluded examples only. Here, correspondence between filters is not known (or does not exist), so comparing specific filters is meaningless. Thus, we only compared the average of the effective spatial support, over all filters in the same layer; see Table 8 in Appendix A. These results further confirm our findings regarding the smaller filter support caused by training on occluded objects.

5.2. Regularization for occlusion robustness

5.2.1 The CIFAR Datasets

The **CIFAR-10/CIFAR100** datasets [11] are both drawn from 80 Million Tiny Images [22]. They both contain 50K training examples and 10K test examples, all of which are 32×32 RGB. **CIFAR10** consists of 10 distinct classes while **CIFAR100** consists of 100. Both training and test data are distributed uniformly in both datasets. We evaluated the recognition accuracy under partial occlusion by training a **LeNet** [14] CNN with different choices of the proposed regularization terms; see Table 3.

When training on unoccluded object examples, without regularization, the error associated with classifying occluded objects (2nd line in Table 3) is much higher than that associated with classifying unoccluded objects (top line). As expected, when training on occluded object examples, the error associated with classifying partially occluded objects decreases (3rd line). The proposed classifiers, when trained on normal (unoccluded examples), reduced the classification error on occluded objects as well, regardless of the choice of regularization term. This improvement came, as

	LeNet-CIFAR10		**LeNet-CIFAR100**	
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.1$	$\delta = 0.2$
ut+uv	18.97		51.55	
ut+ov	27.31	40.03	60.17	72.17
ot+ov	**22.89**	**27.71**	56.43	**60.34**
R_{CM} conv	23.47	34.90	56.78	68.11
R_{CM} conv+fc	23.00	34.30	**56.22**	67.16
R_{GS} conv	24.20	36.70	58.74	70.01
R_{GS} conv+fc	23.72	36.09	**56.21**	69.22
R_{MRWL1} conv	25.05	35.51	58.68	69.21
R_{MRWL1} conv+fc	24.45	35.33	59.78	69.17
R_{L1} conv+fc	27.52	39.93	60.10	72.31
R_{IRWL1}[1] conv+fc	26.45	38.78	58.71	70.11

Table 3: Recognition error rates (%) on the **CIFAR10** and **CIFAR100** datasets. ut: unoccluded train. uv: unoccluded val. ot: occluded train. ov: occluded val. δ: occlusion ratio.

expected, with smaller filter support. Adding a regularization term on the first fully connected (FC) layer further improves classification results, in most cases. For comparison we also experimented with L1 and IRWL1 regularizations.

For **CIFAR10**, training with occluded examples yielded better accuracy than our regularization based approach, although, for moderate occlusion ($\delta = 0.1$), the difference is small (for R_{cm} regularization). For **CIFAR100**, the regularization based approach achieved slightly better results. We associated the difference between these two cases with the smaller number of examples (500 vs 5K) per class in the second case, which is probably not enough for training.

For best performance on occluded objects, λ is set to a relatively high value; see Appendix. When tested on unoccluded objects, with this λ value, performance is slightly reduced. (error rate increases by 0.4%). When setting λ optimally for recognizing unoccluded objects, the results are better than those obtained with $\mathcal{L}_2$ regularization (error rate goes down by 1.7%). Therefore, R_{CM} should be preferred also for general classification. This surprising result is consistent with [27], where similar improvements were obtained by training with occlusions.

5.2.2 The ImageNet Dataset

The picture changes when experimenting with the **ILSVRC2012** classification task [3]. We used **AlexNet** and **VGG16** topologies [12, 19], with batch normalization [10]. We experimented with training several CNNs. The first was trained on unoccluded object examples and tested on both unoccluded and occluded validation sets ($\delta = 0.1$). Next we trained a network with partially occluded object examples, also with a $\delta = 0.1$ occlusion ratio, and tested it on the corresponding occluded validation set. Finally, we trained a network with different regularization terms. For **AlexNet**, R_{CM} regularization was used on the first and second convolution layers as well as on the first fully connected layer and R_{MRWL1} was used on the third, fourth and fifth convolution layers. For **VGG16** we used R_{CM} on the first layer of each cascade. Reducing the support of the first element in the cascade results in support reduction of the entire cascade. The results are given in Table 4. The regularized VGG network achieved a significant advantage over training on occluded examples, and its validation accuracy was close to that obtained without occlusion at all. We associate this advantage with the smaller size of the objects in ImageNet, which implies that the simulated occluder might eliminate the object completely, or not even intersect with it; see Figure 3. This makes training with occluded examples less effective, prolonging training significantly; see Figure5.

ILSVRC12	top-5\top-1 AlexNet	top-5\top-1 VGG16
ut+uv	19.3\41.1	9.5\28.3
ut+ov	26.2\50.6	23.7\37.5
ot+ov	25.2\48.5	16.9\35.9
Support regularized net	**25.0\48.1**	**13.3\33.6**

Table 4: Recognition error rates (%) on the **ILSVRC12** dataset for **AlexNet** and **VGG16**. ut: unoccluded train. uv: unoccluded val. ot: occluded train. ov: occluded val. The last three rows correspond to testing of occluded objects.

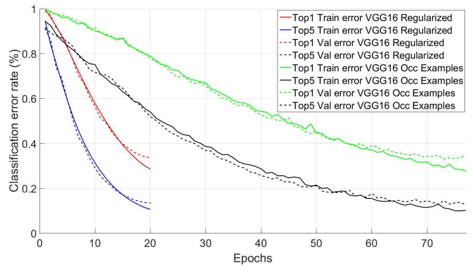

Figure 5: **VGG16** convergence of support-regularized network vs. the common network trained on occluded examples. The regularized network converged 4 times faster.

6. Conclusions

We considered visual classification under occlusion using CNNs. We show that training with partially occluded objects reduces the spatial support of the CNN filters and increase their effective depth. Following these observations, we propose a new learning algorithm that relies on special regularization and trains with regular unoccluded examples, while producing a classifier that is robust to occlusions. In the realistic case of large train sets with weakly annotated images, it trains faster and is more accurate than training with occluded objects. This result is surprising since limiting spatial support does not address all aspects of occlusion.

Acknowledgments

This research was supported by the Israel Science Foundation (grant No.1792\15).

References

[1] E. J. Candes, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier Analysis and Applications*, 2008. 6, 8

[2] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision (ECCV)*, 2004. 2

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 1, 8

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2010. 2

[5] K. Fukushima. Recognition of partly occluded patterns: a neural network model. *Biological Cybernetics*, 2001. 2

[6] T. Gao, B. Packer, and D. Koller. A segmentation-aware object detection model with occlusion handling. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 2

[7] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016. 3

[8] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng. Measuring invariances in deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2009. 2, 6

[9] I. J. Goodfellow, D. Warde-farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. 1

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 1, 7, 8

[11] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 7

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012. 1, 7, 8

[13] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 2

[14] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems (NIPS)*, 1990. 1, 7

[15] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *The 22nd annual conference on Computer Graphics and Interactive Techniques*, 1995. 7

[16] M. Opitz, G. Waltner, G. Poier, H. Possegger, and H. Bischof. Grid loss: Detecting occluded faces. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[17] B. Pepik, R. Benenson, T. Ritschel, and B. Schiele. What is holding back convnets for detection? In *German Conference on Pattern Recognition*, 2015. 1, 2

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015. 2, 7

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Convention on Learning Representations (ICLR)*, 2015. 1, 7, 8

[20] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 2014. 1

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[22] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2008. 7

[23] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, 2013. 2

[24] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015. 7

[25] A. Vedaldi and A. Zisserman. Structured output regression for detection with partial truncation. In *Advances in neural information processing systems(NIPS)*, 2009. 2

[26] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. 1

[27] X. Wang, A. Shrivastava, and A. Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 8

[28] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2006. 6

A. Supplementary Material

A.1. Second central moment regularization derivative

In the following we provide the full derivation of the second central moment regularization term, R_{CM}, that was introduced in (12). We consider the absolute value of the filter weights, normalized by the $\mathcal{L}_1$ norm of the entire filter (note that it differs from $\hat{V}$ (1)):

$$\tilde{\mathbf{V}}^l_{kd::} = \frac{|\mathbf{W}^l_{kd::}|}{||\mathbf{W}^l_{k:::}||_1}. \tag{18}$$

Let $R^l_{CM} = (R^l_{CM})^x + (R^l_{CM})^y$, where $(R^l_{CM})^x$ and $(R^l_{CM})^y$ correspond to the horizontal second central moment C_{20} and the vertical second central moment C_{02}, respectively. Below we calculate the derivative of $(R^l_{CM})^x$. The derivative of $(R^l_{CM})^y$ is symmetric.

The derivative of $(R^l_{CM})^x$ with respect to a specific filter weight is obtained using the chain rule:

$$\frac{\partial (R^l_{CM})^x}{\partial \mathbf{W}^l_{kdij}} = \frac{\partial (R^l_{CM})^x}{\partial \tilde{\mathbf{V}}^l_{kdij}} \cdot \frac{\partial \tilde{\mathbf{V}}^l_{kdij}}{\partial \mathbf{W}^l_{kdij}}. \tag{19}$$

The explicit term for $(R^l_{CM})^x$ is:

$$\begin{aligned}
&(R^l_{CM})^x = \\
&= \sum_{k,d} \frac{1}{||\mathbf{W}^l_{k:::}||_1} \Big[\sum_{m,n} |\mathbf{W}^l_{kdmn}| \cdot m^2 - \frac{(\sum_{m,n} |\mathbf{W}^l_{kdmn}| \cdot m)^2}{\sum_{m,n} |\mathbf{W}^l_{kdmn}|} \Big] \\
&= \sum_{k,d} \Big[\sum_{m,n} \frac{|\mathbf{W}^l_{kdmn}| \cdot m^2}{||\mathbf{W}^l_{k:::}||_1} - \frac{\Big(\frac{\sum_{m,n} |\mathbf{W}^l_{kdmn}| \cdot m}{||\mathbf{W}^l_{k:::}||_1} \Big)^2}{\frac{\sum_{m,n} |\mathbf{W}^l_{kdmn}|}{||\mathbf{W}^l_{k:::}||_1}} \Big] \\
&= \sum_{k,d} \Big[\sum_{m,n} \tilde{\mathbf{V}}^l_{kdmn} \cdot m^2 - \frac{\Big(\sum_{m,n} \tilde{\mathbf{V}}^l_{kdmn} \cdot m \Big)^2}{\sum_{m,n} \tilde{\mathbf{V}}^l_{kdmn}} \Big].
\end{aligned} \tag{20}$$

This term can now be written using the moments $\tilde{M}_{20}$, $\tilde{M}_{10}$ and $\tilde{M}_{00}$ (which are the moments in the $\tilde{\mathbf{V}}^l_{kdmn}$ values). Calculating the first element of the derivative, with respect to $\tilde{\mathbf{V}}^l_{kdij}$ yields:

$$\begin{aligned}
\frac{\partial (R^l_{CM})^x}{\partial \tilde{\mathbf{V}}^l_{kdij}} &= \frac{\partial}{\partial \tilde{\mathbf{V}}^l_{kdij}} \sum_{u=1,v=1}^{u=K,v=D} \Big[(\tilde{M}_{20})^l_{uv} - \frac{(\tilde{M}^2_{10})^l_{uv}}{(\tilde{M}_{00})^l_{uv}} \Big] \\
&= \Big[i^2 - \frac{2 \cdot (\tilde{M}_{10})^l_{kd} \cdot (\tilde{M}_{00})^l_{kd} \cdot i - (\tilde{M}^2_{10})^l_{kd}}{(\tilde{M}^2_{00})^l_{kd}} \Big] \\
&= \Big[\frac{i^2 \cdot (\tilde{M}^2_{00})^l_{kd} - 2 \cdot (\tilde{M}_{10})^l_{kd} \cdot (\tilde{M}_{00})^l_{kd} \cdot i + (\tilde{M}^2_{10})^l_{kd}}{(\tilde{M}^2_{00})^l_{kd}} \Big] \\
&= \frac{\Big[(\tilde{M}_{00})^l_{kd} \cdot i - (\tilde{M}_{10})^l_{kd} \Big]^2}{(\tilde{M}^2_{00})^l_{kd}} = \Big[i - (\tilde{\mu}_x)^l_{kd} \Big]^2.
\end{aligned} \tag{21}$$

Note that $(\tilde{\mu}_x)^l_{kd} = (\mu_x)^l_{kd}$. Deriving the second term of (19) yields:

$$\begin{aligned}
\frac{\partial \tilde{\mathbf{V}}^l_{kdij}}{\partial \mathbf{W}^l_{kdij}} &= \frac{\partial}{\partial \mathbf{W}^l_{kdij}} \frac{|\mathbf{W}^l_{kdij}|}{||\mathbf{W}^l_{k:::}||_1} = \\
&= \Big[\frac{sign(\mathbf{W}^l_{kdij})}{||\mathbf{W}_{k:::}||_1} - \frac{\mathbf{W}^l_{kdij}}{||\mathbf{W}_{k:::}||_1^2} \Big].
\end{aligned} \tag{22}$$

To sum up, the complete term for the derivative, with respect to a single filter weight $\mathbf{W}^l_{kdij}$ takes the form of:

$$\frac{\partial R^l_{CM}}{\partial \mathbf{W}^l_{kdij}} = \Big[\frac{sign(\mathbf{W}^l_{kdij})}{||\mathbf{W}_{k:::}||_1} - \frac{\mathbf{W}^l_{kdij}}{||\mathbf{W}_{k:::}||_1^2} \Big] \cdot [(i - (\mu_x)^l_{kd})^2 + (j - (\mu_y)^l_{kd})^2]; \tag{23}$$

A.2. Simplifying the expression for the effective spatial support of a filter

In the following we provide further details regarding the simplification made in Section 3 eq.(4)-(5) in order to obtain the simplified term in eq.(6), which is used to calculate the second central moment of a filter. This simplification allowed us to calculate $(C_{20})^l_{kd}$ and $(C_{02})^l_{kd}$ using the absolute valued weights, instead of the normalized weights as in $\widehat{(C_{20})}^l_{kd}$ and $\widehat{(C_{02})}^l_{kd}$. Substituting eq.(4) into (5) yield the following:

$$\begin{aligned}
&\sum_{d=1}^{D} \widehat{(C_{20})}^l_{kd} \frac{\|\mathbf{W}_{kd::}\|_1}{\|\mathbf{W}_{k:::}\|_1} = \sum_{d=1}^{D} \Big[\sum_{i,j} \frac{|\mathbf{W}^l_{kdij}|}{||\mathbf{W}^l_{kd::}||_1} \cdot j^2 \\
&- \frac{\big(\sum_{i,j} \frac{|\mathbf{W}^l_{kdij}|}{||\mathbf{W}^l_{kd::}||_1} \cdot j \big)^2}{\sum_{m,n} \frac{|\mathbf{W}^l_{kdmn}|}{||\mathbf{W}^l_{kd::}||_1}} \Big] \cdot \frac{\|\mathbf{W}_{kd::}\|_1}{\|\mathbf{W}_{k:::}\|_1} = \frac{\sum_{d=1}^{D} (C_{20})^l_{kd}}{\|\mathbf{W}_{k:::}\|_1}.
\end{aligned} \tag{24}$$

Hence, using the unnormalized second moments is equivalent up to $\frac{1}{\|\mathbf{W}_{k:::}\|_1}$, which is shown in eq.(6).

A.3. Additional results supporting the observation that training with occluded examples lowers the spatial supports of the filters and increases their effective depths

In the following, we provide further, more detailed results regarding the experiments presented in section 3 and section 5. First we provide the complete results of second central moment and spatial entropy for two VGG16 networks, where the first trained with unoccluded examples, and the second is based on the first, but is trained on occluded examples near the end of the training process; see Sections 3 and 5, for details. The following table completes the results in Table 1, Section 3.2:

	Fraction of filters in net A with larger spatial entropy than corresponding filters in net B	Fraction of filters in net A with larger 2nd central moments than corresponding filters in net B
conv 1	0.551	0.577
conv 2	0.616	0.591
conv 3	0.721	0.611
conv 4	0.667	0.633
conv 5	0.583	0.591
conv 6	0.599	0.701
conv 7	0.672	0.649
conv 8	0.691	0.652
conv 9	0.686	0.599
conv 10	0.684	0.761
conv 11	0.613	0.658
conv 12	0.652	0.612
conv 13	0.634	0.634

Table 5: The fraction of filters in network A, (**VGG16** trained only with unoccluded examples) that has a larger effective support than the corresponding filters in network B (trained also with occluded examples). The spatial support of the filters tends to be lower if the training images are partially occluded.

Next we provide the complete results of the same experiment for networks based on **AlexNet** topology:

conv layer	1	2	3	4	5
Fraction of filters in net A with larger spatial entropy than corresponding filters in net B.	0.641	0.582	0.675	0.597	0.596
Fraction of filters in net A with larger 2nd central moments than corresponding filters in net B.	0.572	0.585	0.675	0.722	0.634

Table 6: The fraction of filters in network A, (**AlexNet** trained only with unoccluded examples) that has a larger effective support than the corresponding filters in network B (trained also with occluded examples). The spatial support of the filters tends to be lower if the training images are partially occluded.

In the following table we present the effective depth for the same experiment for networks based on **AlexNet** topology; see Section 3.3, Table 2:

conv layer	1	2	3	4	5
Fraction of filters in net A with larger depth entropy than corresponding filters in net B.	0.448	0.293	0.361	0.254	0.275

Table 7: The fraction of filters in network A, (**AlexNet** trained only with unoccluded examples) that has a larger depth entropy than the corresponding filters in network B (trained also with occluded examples). When training on partially occluded examples, more kernels play a meaningful role in the feature extraction process; see appendix A for additional results.

As mentioned in Section 3.2, in the experiments presented above, the differences in spatial support are quite small. This is a result of the experimental settings, meant to maintain correspondence between the filters of the two network. We also carried out a different experiment, where we trained two AlexNet networks. The first was trained only on unoccluded examples, while the second was trained only on occluded examples. In this setting, there is no filter correspondence between the two networks. The following table presents the means and the standard deviations of the 2nd central moments in these networks. The results show that the spatial support is indeed lower when training with occluded examples. The difference is larger and statistically significant in the later layers.

conv layer	1	2	3	4	5
(μ,σ) for non occ	5.99,3.12	3.31,0.19	1.13,0.03	1.181,0.032	1.20,0.03
(μ,σ) for occ	5.65,2.71	3.21,0.33	1.01,0.01	1.147,0.001	1.13,0.002

Table 8: A comparison of the mean and standard deviations of the second central moment between two **AlexNet** networks. The first trained on unoccluded examples while the second trained only on partially occluded training examples. The comparison is presented with respect to the different convolution layers. The spatial support is smaller if the training process is conducted on occluded examples.

A.4. The group sparsity masks used in the experiments.

In the following figure, we present an example of the group sparsity masks used with the LeNet network which trained with the CIFAR data sets experiments described in Section 5:

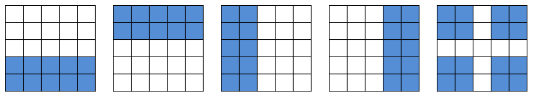

Figure 6: Group sparsity masks, similar to those used in our experiments. White represents unaffected weights, while blue represents the group of weights meant to decay together.

A.5. A note on setting the regularization strength.

Using substantial experimentation (on LeNet5), we came to the conclusion that latter layers in the CNN benefit from larger regularization strength values (λ). The results cited in Table 3 were obtained with $\lambda = 0.0005$ for the first two layers and for larger $\lambda = 0.005$ for the last layers.