Representation Learning by Learning to Count

Mehdi Noroozi[1] Hamed Pirsiavash[2] Paolo Favaro[1]
University of Bern[1] University of Maryland, Baltimore County[2]
{noroozi,favaro}@inf.unibe.ch {hpirsiav@umbc.edu}

Abstract

We introduce a novel method for representation learning that uses an artificial supervision signal based on counting visual primitives. This supervision signal is obtained from an equivariance relation, which does not require any manual annotation. We relate transformations of images to transformations of the representations. More specifically, we look for the representation that satisfies such relation rather than the transformations that match a given representation. In this paper, we use two image transformations in the context of counting: scaling and tiling. The first transformation exploits the fact that the number of visual primitives should be invariant to scale. The second transformation allows us to equate the total number of visual primitives in each tile to that in the whole image. These two transformations are combined in one constraint and used to train a neural network with a contrastive loss. The proposed task produces representations that perform on par or exceed the state of the art in transfer learning benchmarks.

1. Introduction

We are interested in learning representations (features) that are discriminative for semantic image understanding tasks such as classification, detection, and segmentation. A common approach to obtain such features is to use supervised learning. However, this requires manual annotation of images, which is costly, time-consuming, and prone to errors. In contrast, unsupervised or self-supervised feature learning methods exploiting unlabeled data can be much more scalable and flexible.

Some recent feature learning methods, in the so-called *self-supervised learning* paradigm, have managed to avoid annotation by defining a task which provides a *supervision signal*. For example, some methods recover color from gray scale images and vice versa [43, 21, 44, 22], recover a whole patch from the surrounding pixels [33], or recover the relative location of patches [9, 29]. These methods use information already present in the data as supervision signal so that supervised learning tools can be used. A rationale behind self-supervised learning is that pretext tasks that relate the most to the final problems (*e.g.*, classification and detection) will be more likely to build relevant representations.

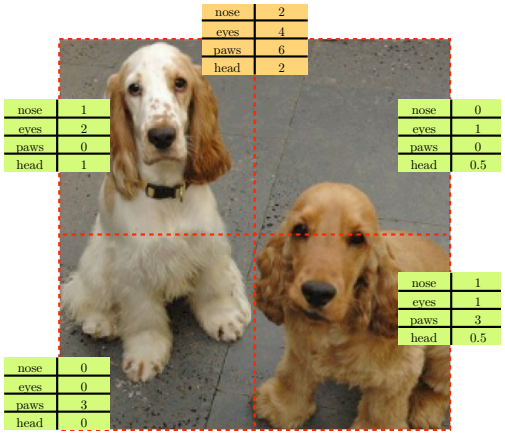


Figure 1: The number of visual primitives in the whole image should match the sum of the number of visual primitives in each tile (dashed red boxes).

As a novel candidate pretext task, we propose *counting visual primitives*. It requires discriminative features, which can be useful to classification, and it can be formulated via detection. To obtain a supervision signal useful to learn to count, we exploit the following property: If we partition an image into non-overlapping regions, the number of visual primitives in each region should sum up to the number of primitives in the original image (see the example in Fig. 1). We make the hypothesis that the model needs to disentangle the image into high-level factors of variation, such that the complex relation between the original image and its regions is translated to a simple arithmetic operation [3, 35]. Our experimental results validate this hypothesis both qualitatively and quantitatively.

While in this work we focus on a specific combination of transformations, one can consider more general relationships (*i.e.*, beyond counting, scaling, and tiling) as super-

vision signals. The same procedure that we introduce is therefore applicable to a broader range of tasks as long as it is possible to express the transformation in feature space caused by a transformation in image space [24].

Our contributions are: 1) We introduce a novel method to learn representations from data without manual annotation; 2) We propose exploiting counting as a pretext task and demonstrate its relation to counting visual primitives; 3) We show that the proposed methodology learns representations that perform on par or exceed the state of the art in standard transfer learning benchmarks.

2. Prior Work

In this work we propose to learn a representation without relying on annotation, a problem that is typically addressed via unsupervised learning. An example of this approach is the autoencoder [14, 40], which reconstructs data by mapping it to a low-dimensional feature vector. A recent alternative approach is *self-supervised learning*, which is a technique that substitutes the labels for a task with *artificial* or *surrogate* ones. In our work such artificial labels are provided by a counting constraint. In many instances, this technique can be seen as recasting the unsupervised learning problem of finding $p(\mathbf{x}) = p(\mathbf{x}_1, \mathbf{x}_2)$, where $\mathbf{x}^\top = [\mathbf{x}_1^\top \; \mathbf{x}_2^\top]$ is a random variable, as a partly supervised one of finding $p(\mathbf{x}_2|\mathbf{x}_1)$, so that we can write $p(\mathbf{x}_1, \mathbf{x}_2) = p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_1)$ (cf. eq. (5.1) in [12]). In our context, the data sample $\mathbf{x}$ collects all available information, which can be just an image, but might also include egomotion measurements, sound, and so on. In the literature, self-supervised methods do not recover a model for the probability function $p(\mathbf{x}_1)$, since $p(\mathbf{x}_2|\mathbf{x}_1)$ is sufficient to obtain a representation of $\mathbf{x}$. Most methods are then organized based on the choice of $\mathbf{x}_1$ and $\mathbf{x}_2$, where $\mathbf{x}_2$ defines the surrogate labels. Below we briefly summarize methods based on their choice for $\mathbf{x}_2$, which leads to a regression or classification problem.

Regression. In recent work Pathak *et al*. [33] choose as surrogate label $\mathbf{x}_2$ a region of pixels in an image (*e.g*., the central patch) and use the remaining pixels in the image as $\mathbf{x}_1$. The model used for $p(\mathbf{x}_2|\mathbf{x}_1)$ is based on generative adversarial networks [13, 35]. Other related work [43, 21] maps images to the Lab (luminance and opponent colors) space, and then uses the opponent colors as labels $\mathbf{x}_2$ and the luminance as $\mathbf{x}_1$. Zhang *et al*. [44] combine this choice to the opposite task of predicting the grayscale image from the opponent colors and outperform prior work.

Classification. Doersch *et al*. and Noroozi & Favaro [9, 29] define a categorical problem where the surrogate labels are the relative positions of patches. Other recent works use as surrogate labels ego-motion [1, 17], temporal ordering in video [27, 32], sound [31], and physical interaction [34].

In contrast to these works, here we introduce a different formulation to arrive at a supervision signal. We define the *counting* relationship *"having the same number of visual primitives"* between two images. We use the fact that this relationship is satisfied by two identical images undergoing certain transformations, but not by two different images (although they might, with very low probability). Thus, we are able to assign a binary label (same or different number of visual primitives) to pairs of images. We are not aware of any other self-supervised method that uses this method to obtain the surrogate labels. In contrast, Wang and Gupta [41] impose relationships between triplets of different images obtained through tracking. Notice that also Reed *et al*. [36] exploit an explicit relationship between features. However, they rely on labeling that would reveal the relationship between different input images. Instead, we only exploit the structure of images and relate different parts of the same image to each other. Due to the above counting relationship our work relates also to *object counting*, which we revise here below.

Object counting. In comparison to other semantic tasks, counting has received little attention in the computer vision community. Most effort has been devoted to counting just one category and only recently it was applied to multiple categories in a scene. Counting is usually addressed as a supervised task, where a model is trained on annotated images. The counting prediction can be provided as an object density map [15, 39, 23] or simply as the number of counted objects [5, 6]. There are methods to count humans in crowds [4, 42, 8], cars [28], and penguins [2]. Some recent works count common objects in the scene without relying on object localization [6, 37].

In this work, we are not interested in the task of counting *per se*. As mentioned earlier on, counting is used as a pretext task to learn a representation. Moreover, we do not use labels about the number of objects during training.

3. Transforming Images to Transform Features

One way to characterize a feature of interest is to describe how it should vary as a function of changes in the input data. For example, a feature that counts visual primitives should not be affected by scale, 2D translation, and 2D rotation changes of the input image. Other relationships might indicate instead that a feature should increase its values as a result of some transformation of the input. For example, the magnitude of the feature for counting visual primitives applied to half of an image should be smaller than when applied to the whole image. In general, we propose to learn a deep representation by using the known relationship between input and output transformations as a supervisory signal. To formalize these concepts, we first need to introduce some notation.

Let us denote a color image with $\mathbf{x} \in \mathbb{R}^{m \times n \times 3}$, where $m \times n$ is the size in pixels and there are 3 color channels (RGB). We define a family of image transformations

$\mathcal{G} \triangleq \{G_1, \ldots, G_J\}$, where $G_j : \mathbb{R}^{m \times n \times 3} \mapsto \mathbb{R}^{p \times q \times 3}$, with $j = 1, \ldots, J$, that take images $\mathbf{x}$ and map them to images of $p \times q$ pixels. Let us also define a feature $\phi : \mathbb{R}^{p \times q \times 3} \mapsto \mathbb{R}^k$ mapping the transformed image to some k-dimensional vector. Finally, we define a feature transformation $g : \mathbb{R}^k \times \cdots \times \mathbb{R}^k \mapsto \mathbb{R}^k$ that takes J features and maps them to another feature. Given the image transformation family $\mathcal{G}$ and g, we learn the feature ϕ by using the following relationship as an *artificial supervisory signal*

$$g\left(\phi(G_1 \circ \mathbf{x}), \ldots, \phi(G_J \circ \mathbf{x})\right) = \mathbf{0} \qquad \forall \mathbf{x}. \quad (1)$$

In this work, the transformation family consists of the *downsampling* operator D, with a downsampling factor of 2, and the *tiling* operator T_j, where $j = 1, \ldots, 4$, which extracts the $j-$th tile from a 2×2 grid of tiles. Notice that these two transformations produce images of the same size. Thus, we can set $\mathcal{G} \equiv \{D, T_1, \ldots, T_4\}$. We also define our desired relation between counting features on the transformed images as $g(\mathbf{d}, \mathbf{t_1}, \ldots, \mathbf{t_4}) = \mathbf{d} - \sum_{j=1}^{4} \mathbf{t_j}$. This can be written explicitly as

$$\phi(D \circ \mathbf{x}) = \sum_{j=1}^{4} \phi(T_j \circ \mathbf{x}). \quad (2)$$

We use eq. (2) as our main building block to learn features ϕ that can count visual primitives.

This relationship has a bearing also on *equivariance* [24]. Equivariance, however, is typically defined as the property of a given feature. In this work we invert this logic by fixing the transformations and by finding instead a representation satisfying those transformations. Moreover, equivariance has restrictions on the type of transformations applied to the inputs and the features.

Notice that we have no simple way to control the scale at which our counting features work. It could count object parts, whole objects, object groups, or any combination thereof. This choice might depend on the number of elements of the counting vector ϕ, on the loss function used for training, and on the type of data used for training.

4. Learning to Count

We use convolutional neural networks to obtain our representation. In principle, our network could be trained with color images $\mathbf{x}$ from a large database (*e.g.*, ImageNet [38] or COCO [25]) using an l_2 loss based on eq. (2), for example,

$$\ell(\mathbf{x}) = \left| \phi(D \circ \mathbf{x}) - \textstyle\sum_{j=1}^{4} \phi(T_j \circ \mathbf{x}) \right|^2. \quad (3)$$

However, this loss has $\phi(\mathbf{z}) = \mathbf{0}$, $\forall \mathbf{z}$, as its trivial solution. To avoid such a scenario, we use a contrastive loss [7], where we also enforce that the counting feature should be

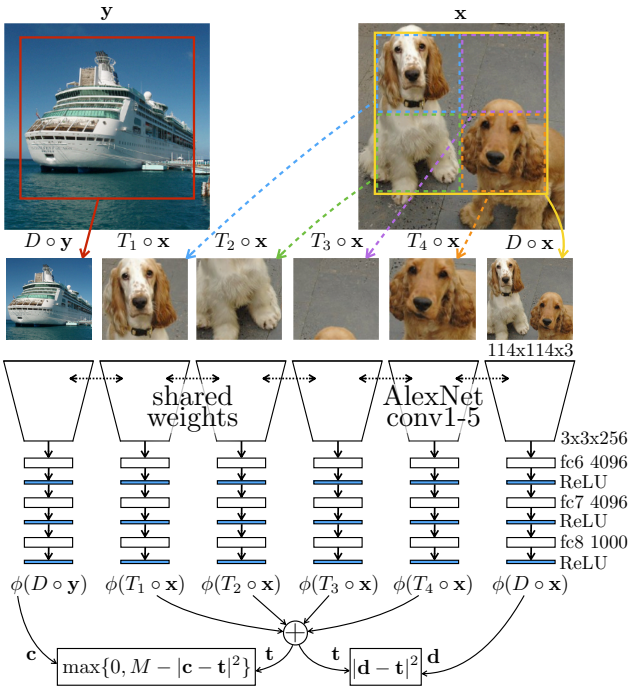


Figure 2: **Training AlexNet to learn to count.** The proposed architecture uses a siamese arrangement so that we simultaneously produce features for 4 tiles and a downsampled image. We also compute the feature from a randomly chosen downsampled image ($D \circ \mathbf{y}$) as a contrastive term.

different between two randomly chosen different images. Therefore, for any $\mathbf{x} \neq \mathbf{y}$, we would like to minimize

$$\ell_{\text{con}}(\mathbf{x}, \mathbf{y}) = \left| \phi(D \circ \mathbf{x}) - \textstyle\sum_{j=1}^{4} \phi(T_j \circ \mathbf{x}) \right|^2 + \max\left\{0, M - \left| \phi(D \circ \mathbf{y}) - \textstyle\sum_{j=1}^{4} \phi(T_j \circ \mathbf{x}) \right|^2\right\} \quad (4)$$

where in our experiments the constant scalar $M = 10$.

Least effort bias. A bias of the system is that it can easily satisfy the constraint (3) by learning to count as few visual primitives as possible. Thus, many entries of the feature mapping may collapse to zero. This effect is observed in the final trained network. In Fig. 3, we show the average of features computed over the ImageNet validation set. There are only 30 and 44 non zero entries out of 1000 after training on ImageNet and on COCO respectively. Despite the sparsity of the features, our transfer learning experiments show that the features in the hidden layers (`conv1-conv5`) perform very well on several benchmarks. In our formulation (4), the contrastive term limits the effects of the least effort bias. Indeed, features that count very few visual primitives cannot differentiate much the content across different images. Therefore, the contrastive term will introduce a tradeoff that

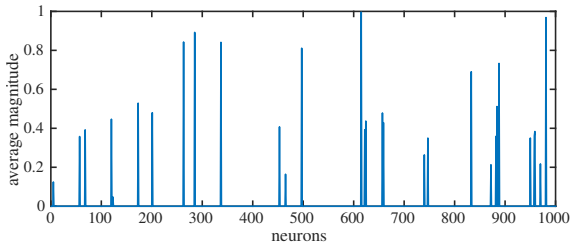


Figure 3: **Average response of our trained network on the ImageNet validation set.** Despite its sparsity (30 non zero entries), the hidden representation in the trained network performs well when transferred to the classification, detection and segmentation tasks.

will push features towards counting as many primitives as is needed to differentiate images from each other.

Network architecture. In principle, the choice of the architecture is arbitrary. For ease of comparison with state-of-the-art methods when transferring to classification and detection tasks, we adopt the AlexNet architecture [20] as commonly done in other self-supervised learning methods. We use the first 5 convolutional layers from AlexNet followed by three fully connected layers ($(3\times3\times256)\times4096$, 4096×4096, and 4096×1000), and ReLU units. Note that 1000 is the number of elements that we want to count. We use ReLU in the end since we want the counting vector to be all positive. Our input is 114×114 pixels to handle smaller tiles. Because all the features are the same, training with the loss function in eq. 4 is equivalent to training a 6-way siamese network, as shown in Fig. 2.

5. Experiments

We first present the evaluations of our learned representation in the standard transfer learning benchmarks. Then, we perform ablation studies on our proposed method to show quantitatively the impact of our techniques to prevent poor representations. Finally, we analyze the learned representation through some quantitative and qualitative experiments to get a better insight into what has been learned. We call the activation of the last layer of our network, on which the loss (4) is defined, the *counting vector*. We evaluate whether each unit in the counting vector is counting some visual primitive or not. Our model is based on AlexNet [20] in all experiments. In our tables we use boldface for the top performer and underline the second top performer.

Implementation Details. We use caffe [18] with the default weight regularization settings to train our network. The learning rate is set to be quite low to avoid divergence. We begin with a learning rate of 10^{-4} and drop it by a factor of 0.9 every $10K$ iterations. An important step is to normalize the input by subtracting the mean intensity value and dividing the zero-mean images by their standard deviation.

Method	**Ref**	**Class.**	**Det.**	**Segm.**
Supervised [20]	[43]	79.9	56.8	48.0
Random	[33]	53.3	43.4	19.8
Context [9]	[19]	55.3	46.6	-
Context [9]*	[19]	65.3	51.1	-
Jigsaw [30]	[30]	<u>67.6</u>	**53.2**	<u>37.6</u>
ego-motion [1]	[1]	52.9	41.8	-
ego-motion [1]*	[1]	54.2	43.9	-
Adversarial [10]*	[10]	58.6	46.2	34.9
ContextEncoder [33]	[33]	56.5	44.5	29.7
Sound [31]	[44]	54.4	44.0	-
Sound [31]*	[44]	61.3	-	-
Video [41]	[19]	62.8	47.4	-
Video [41]*	[19]	63.1	47.2	-
Colorization [43]*	[43]	65.9	46.9	35.6
Split-Brain [44]*	[44]	67.1	46.7	36.0
ColorProxy [22]	[22]	65.9	-	**38.0**
WatchingObjectsMove [32]	[32]	61.0	<u>52.2</u>	-
Counting		**67.7**	51.4	36.6

Table 1: **Evaluation of transfer learning on PASCAL.** Classification and detection are evaluated on PASCAL VOC 2007 in the frameworks introduced in [19] and [11] respectively. Both tasks are evaluated using mean average precision (mAP) as a performance measure. Segmentation is evaluated on PASCAL VOC 2012 in the framework of [26], which reports mean intersection over union (mIoU). (*) denotes the use of the data initialization method [19].

5.1. Transfer Learning Evaluation

We evaluate our learned representation on the *detection*, *classification*, and *segmentation* tasks on the **PASCAL** dataset as well as the *classification* task on the **ImageNet** dataset. We train our counting network on the 1.3M images from the training set of ImageNet. We use images of 114×114 pixels as input. Since we transfer only the convolutional layers, it has no effect on the transferred models and evaluation. A new version of [29] has been released [30], where the standard AlexNet is used for transfer learning. All the numbers in our comparisons are from that version.

5.1.1 Fine-tuning on PASCAL

In this set of experiments, we fine-tune our network on the PASCAL VOC 2007 and VOC 2012 datasets, which are standard benchmarks for representation learning. Fine-tuning is based on established frameworks for object classification [19], detection [11] and segmentation [26] tasks. The classification task is a multi-class classification problem, which predicts the presence or absence of 20 object classes. The detection task involves locating objects by specifying a bounding box around them. Segmentation assigns the label of an object class to each pixel in the image. As shown in Table 1, we either outperform previous methods or achieve the second best performance. Notice

Method	Ref	conv1	conv2	conv3	conv4	conv5
Supervised [20]	[44]	19.3	36.3	44.2	48.3	50.5
Random	[44]	11.6	17.1	16.9	16.3	14.1
Context [9]	[44]	16.2	23.3	30.2	31.7	29.6
Jigsaw [30]	[30]	**18.2**	28.8	34.0	33.9	27.1
ContextEncoder [33]	[44]	14.1	20.7	21.0	19.8	15.5
Adversarial [10]	[44]	17.7	24.5	31.0	29.9	28.0
Colorization [43]	[44]	12.5	24.5	30.4	31.5	30.3
Split-Brain [44]	[44]	17.7	29.3	**35.4**	**35.2**	**32.8**
Counting		18.0	**30.6**	34.3	32.5	25.7

Table 2: **ImageNet classification with a linear classifier.** We use the publicly available code and configuration of [43]. Every column shows the top-1 accuracy of AlexNet on the classification task. The learned weights from `conv1` up to the displayed layer are frozen. The features of each layer are spatially resized until there are fewer than 9K dimensions left. A fully connected layer followed by softmax is trained on a 1000-way object classification task.

Method	conv1	conv2	conv3	conv4	conv5
Places labels [45]	22.1	35.1	40.2	43.3	44.6
ImageNet labels [20]	22.7	34.8	38.4	39.4	38.7
Random	15.7	20.3	19.8	19.1	17.5
Context [9]	19.7	26.7	31.9	32.7	30.9
Jigsaw [30]	23.0	31.9	35.0	34.2	29.3
Context encoder [33]	18.2	23.2	23.4	21.9	18.4
Sound [31]	19.9	29.3	32.1	28.8	29.8
Adversarial [10]	22.0	28.7	31.8	31.3	29.7
Colorization [43]	16.0	25.7	29.6	30.3	29.7
Split-Brain [44]	21.3	30.7	34.0	34.1	**32.5**
Counting	**23.3**	**33.9**	**36.3**	**34.7**	29.6

Table 3: **Places classification with a linear classifier.** We use the same setting as in Table 2 except that to evaluate generalization across datasets, the model is pretrained on ImageNet (with no labels) and then tested with frozen layers on Places (with labels). The last layer has 205 neurons for scene categories.

that while classification and detection are evaluated on VOC 2007, segmentation is evaluated on VOC 2012. Unfortunately, we did not get any performance boost when using the method of Krähenbühl *et al*. [19].

5.1.2 Linear Classification on Places and ImageNet

As introduced by Zhang *et al*. [43], we train a linear classifier on top of the frozen layers on ImageNet [38] and Places [45] datasets. The results of these experiments are shown in Tables 2 and 3. Our method achieves a performance comparable to the other state-of-the-art methods on the ImageNet dataset and shows a significant improvement on the Places dataset. Training and testing a method on the same dataset type, although with separate sets and no labels, may be affected by *dataset bias*. To have a better assessment of the generalization properties of all the competing methods, we suggest (as shown in Table 3) using the ImageNet dataset for training and the Places benchmark for testing (or vice versa). Our method archives state-of-the-art results with the `conv1-conv4` layers on the Places dataset. Interestingly, the performance of our `conv1` layer is even **higher than the one obtained with supervised learning** when trained either on ImageNet or Places labels.

Interpolation method	Training size	Color space	Counting dimension	Detection performance
Mixed	1.3M	RGB/Gray	20	50.9
Mixed	128K	Gray	1000	44.9
Mixed	512K	Gray	1000	49.1
Mixed	1.3M	RGB	1000	48.2
Mixed	1.3M	Gray	1000	50.4
Linear	1.3M	RGB/Gray	1000	48.4
Cubic	1.3M	RGB/Gray	1000	48.9
Area	1.3M	RGB/Gray	1000	49.2
Lanczos	1.3M	RGB/Gray	1000	47.3
Mixed	1.3M	RGB/Gray	1000	**51.4**

Table 4: **Ablation studies.** We train the counting task under different interpolation methods, training size/color, and feature dimensions, and compare the performance of the learned representations on the detection task on the PASCAL VOC 2007 dataset.

5.2. Ablation Studies

To show the effectiveness of our proposed method, in Table 4 we compare its performance on the detection task on PASCAL VOC 2007 under different training scenarios. The first three rows illustrate some simple comparisons based on feature and dataset size. The first row shows the impact of the counting vector length. As discussed earlier on, the network tends to generate sparse counting features. We train the network on ImageNet with only 20 elements in the counting vector. This leads to a small drop in the performance, thus showing little sensitivity with respect to feature length. We also train the network with a smaller set of training images. The results show that our method is sensitive to the size of the training set. This shows that the counting task is non-trivial and requires a large training dataset.

The remaining rows in Table 4 illustrate a more advanced analysis of the counting task. An important part of the design of the learning procedure is the identification of trivial solutions, *i.e*., solutions that would not result in a useful representation and that the neural network could converge to. By identifying such *trivial learning* scenarios, we can provide suitable countermeasures. We now discuss possible *shortcuts* that the network could use to solve the counting task and also the techniques that we use to avoid them.

train\test	Linear	Cubic	Area	Lanczos	Mixed	std
Linear	**0.33**	0.63	0.33	0.65	0.48	0.18
Cubic	0.79	**0.25**	0.78	0.22	0.52	0.32
Area	0.32	0.85	**0.31**	0.95	0.50	0.34
Lanczos	1.023	0.31	1.02	**0.19**	0.58	0.45
Mixed	0.36	**0.29**	0.37	0.30	0.34	**0.04**

Table 5: **Learning the downsampling style.** The first column and row show the downsampling methods used during the training and testing time respectively. The values in the first block show the pairwise error metric in eq. (6) between corresponding downsampling methods. The last column shows the standard deviation of the error metric across different downsampling methods at test time.

A first potential problem is that the neural network learns trivial features such as low-level texture statistics histograms. For example, a special case is color histograms. This representation is undesirable because it would be semantically agnostic (or very weak) and therefore we would not expect it to transfer well to classification and detection. In general, these histograms would not satisfy eq. (2). However, if the neural network could tell tiles apart from downsampled images, then it could apply a customized scaling factor to the histograms in the two cases and satisfy eq. (2). In other words, the network might learn the following degenerate feature

$$\phi(\mathbf{z}) = \begin{cases} \frac{1}{4}\text{hist}(\mathbf{z}) & \text{if } \mathbf{z} \text{ is a tile} \\ \text{hist}(\mathbf{z}) & \text{if } \mathbf{z} \text{ is downsampled.} \end{cases} \quad (5)$$

Notice that this feature would satisfy the first term in eq. (2). The second (contrastive) term would also be easily satisfied since different images have typically different low-level texture histograms. We discuss below scenarios when this might happen and present our solutions towards reducing the likelihood of trivial learning.

The network recognizes the downsampling style. During training, we randomly crop a 224×224 region from a 256×256 image. Next, we downsample the whole image by a factor of 2. The downsampling style, *e.g.*, bilinear, bicubic, and Lanczos, may leave artifacts in images that the network may learn to recognize. To make the identification of the downsampling method difficult, at each *stochastic gradient descent* iteration, we randomly pick either the bicubic, bilinear, lanczos, or the area method as defined in OpenCV [16]. As shown in Table 4, the randomization of different downsampling methods significantly improves the detection performance by at least 2.2%.

In Table 5, we perform another experiment that clearly shows that network learns the downsampling style. We train our network by using only one downsampling method. Then, we test the network on the pretext task by using only one (possibly different) method. If the network has learned to detect the downsampling method, then it will perform poorly at test time when using a different one. As an error metric, we use the first term in the loss function normalized by the average of the norm of the feature vector. More precisely, the error when the network is trained with the i-th downsampling style and tested on the j-th one is

$$\mathbf{e}_{ij} = \frac{\sum_{\mathbf{x}} \left| \sum_{p=1}^{4} \phi^i \left(T_p \circ \mathbf{x}\right) - \phi^i(D^j \circ \mathbf{x}) \right|^2}{\sum_{\mathbf{x}} \left|\phi^i \left(D^i \circ \mathbf{x}\right)\right|^2} \quad (6)$$

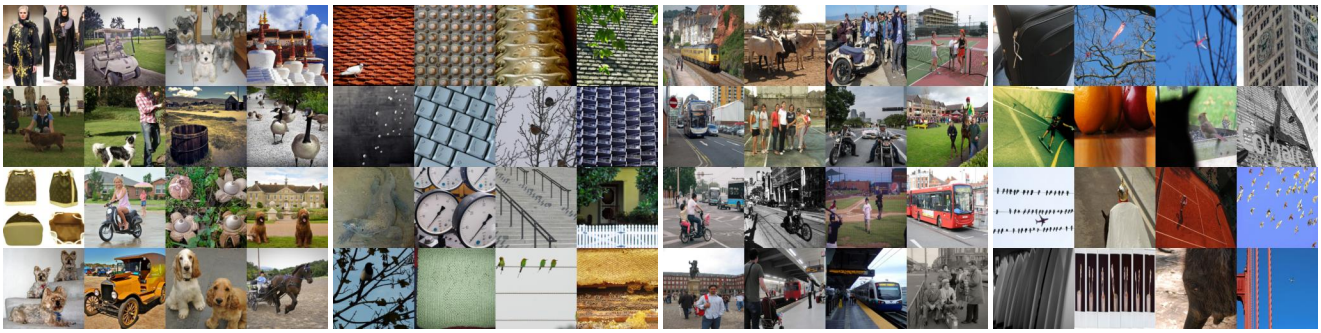
where ϕ^i denotes the counting vector of the network trained with the i-th downsampling method. D^j denotes the downsampling transformation using the j-th method. T_p is the tiling transformation that gives the p-th tile.

Table 5 collects all the computed errors. The element in row i and column j shows the pairwise error metric $\mathbf{e}_{ij}$. The last column shows the standard deviation of this error metric across different downsampling methods. A higher value means that the network is sensitive to the downsampling method. This experiment clearly shows that the network learns the downsampling style. Another observation that can be made based on the similarity of the errors, is that the pairs (linear, area) and (cubic, lanczos) leave similar artifacts in downsampling.

The network recognizes chromatic aberration. The presence of chromatic aberration and its undesirable effects on learning have been pointed out by Doersch *et al*. [9]. Chromatic aberration is a relative shift between the color channels that increases in the outward radial direction. Hence, our network can use this property to tell tiles apart from the dowsampled images. In fact, tiles will have a strongly diagonal chromatic aberration, while the downsampled image will have a radial aberration. We already reduce its effect by choosing the central region in the very first cropping preprocessing. To further reduce its effect, we train the network with both color and grayscale images (obtained by replicating the average color across all 3 channels). In training, we randomly choose color images 33% of the time and grayscale images 67% of the time. This choice is consistent across all the terms in the loss function (*i.e.*, all tiles and downsampled images are either colored or grayscale). While this choice does not completely solve the issue, it does improve the performance of the model. We find that completely eliminating the color from images leads to a loss in performance in transfer learning (see Table 4).

5.3. Analysis

We use visualization and nearest neighbor search to see what visual primitives our trained network counts. Ideally, these visual primitives should capture high-level concepts like objects or object parts rather than low-level concepts like edges and corners. In fact, detecting simple corners will not go a long way in semantic scene understanding. To



(a) (b) (c) (d)

Figure 4: **Examples of activating/ignored images.** (a) and (b) show the top 16 images with the highest and lowest counting feature magnitude from the validation set of ImageNet. (c) and (d) show the top 16 images with the highest and lowest counting feature magnitude from the test set of COCO.

Figure 5: **Image croppings of increasing size.** The number of visual primitives should increase going from left to right.

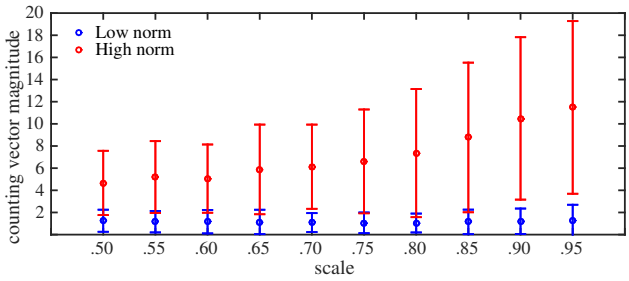


Figure 6: **Counting evaluation on ImageNet.** On the abscissa we report the scale of the cropped region and on the ordinate the corresponding average and standard deviation of the counting vector magnitude.

avoid dataset bias, we train our model on ImageNet (with no labeles) and show the results on COCO dataset.

5.3.1 Quantitative Analysis

We illustrate quantitatively the relation between the magnitude of the counting vector and the number of objects. Rather than counting exactly the number of specific objects, we introduce a simple method to rank images based on how many objects they contain. The method is based on cropping an image with larger and larger regions which are then rescaled to the same size through downsampling (see Fig. 5). We build two sets of 100 images each. We assign images yielding the highest and lowest feature magnitude into two different sets. We randomly crop 10 regions with an area between $50\% - 95\%$ of each image and compute the corresponding counting vector. The mean and the standard deviation of the counting vector magnitude of the cropped images for each set is shown in Fig 6. We observe that our feature does not count low-level texture, and is instead more sensitive to composite images. A better understanding of this observation needs futher investigation.

5.3.2 Qualitative Analysis

Activating/Ignored images. In Fig 4, we show blocks of 16 images ranked based on the magnitude of the counting vector. We observe that images with the lowest feature norms are textures without any high-level visual primitives. In contrast, images with the highest feature response mostly contain multiple object instances or a large object. For this experiment we use the validation or the test set of the dataset that the network has been trained on, so the network has not seen these images during training.

Nearest neighbor search. To qualitatively evaluate our learned representation, for some validation images, we visualize their nearest neighbors in the training set in Fig. 7. Given a query image, the retrieval is obtained as a ranking of the Euclidean distance between the counting vector of the query image and the counting vector of images in the dataset. Smaller values indicate higher affinity. Fig. 7 shows that the retrieved results share a similar scene outline and are semantically related to the query images. Note that we perform retrieval in the counting space, which is the last layer of our network. This is different from the analogous experiment in [19] which performs the retrieval in the intermediate layers. This result can be seen as an evidence that our initial hypothesis, that the counting vectors capture high level visual primitives, was true.

Neuron activations. To visualize what each single counting neuron (*i.e.*, feature element) has learned, we rank images not seen during training based on the magnitude of their neuron responses. We do this experiment on the validation set of ImageNet and the test set of COCO. In Fig. 8,

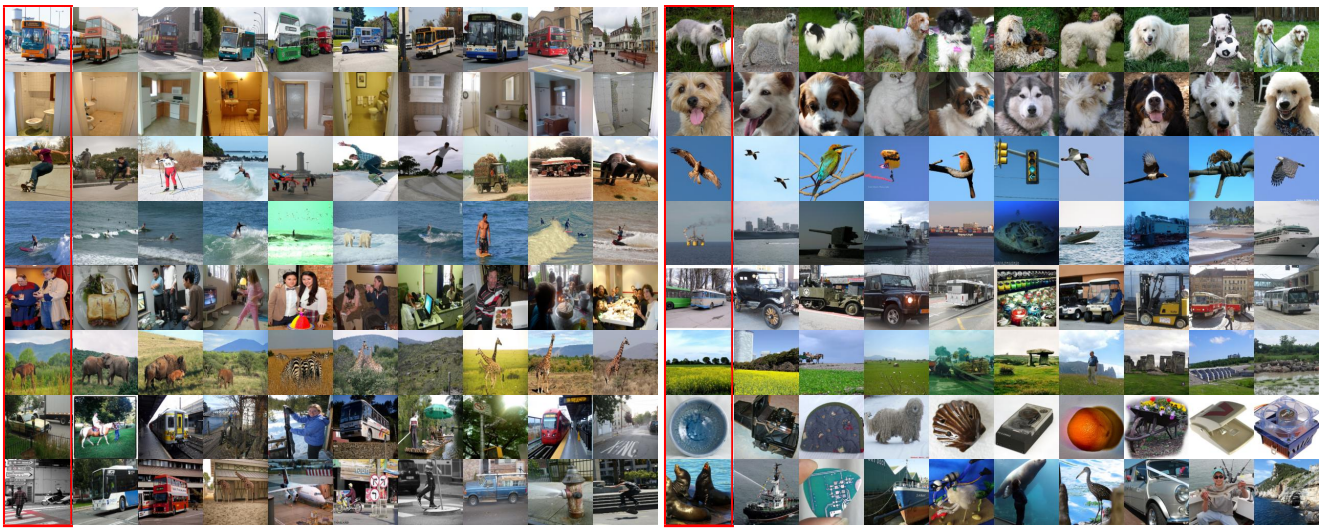


Figure 7: **Nearest neighbor retrievals.** Left: COCO retrievals. Right: ImageNet retrievals. In both datasets, the leftmost column (with a red border) shows the queries and the other columns show the top matching images sorted with increasing Euclidean distance in our counting feature space from left to right. On the bottom 3 rows, we show the failure retrieval cases. Note that the matches share a similar content and scene outline.

Figure 8: Blocks of the 8 most activating images for 4 neurons of our network trained on ImageNet (top row) and COCO (bottom row). The counting neurons are sensitive to semantically similar images. Interestingly, dominant concepts in each dataset, *e.g.*, dogs in ImageNet and persons playing baseball in COCO, emerge in our counting vector.

we show the top 8 most activating images for 4 neurons out of 30 active ones on ImageNet and out of 44 active ones on COCO. We observe that these neurons seem to cluster images that share the same scene layout and general content.

6. Conclusions

We have presented a novel representation learning method that does not rely on annotated data. We used counting as a pretext task, which we formalized as a constraint that relates the "counted" visual primitives in tiles of an image to those counted in its downsampled version. This constraint was used to train a neural network with a contrastive loss. Our experiments show that the learned features count non-trivial semantic content, qualitatively cluster images with similar scene outline, and outperform previous state of the art methods on transfer learning benchmarks. Our framework can be further extended to other tasks and transformations in addition to being combined with partially labeled data in a semi-supervised learning method.

Acknowledgements. We thank Attila Szabó for insightful discussions about unsupervised learning and relations based on equivariance. Paolo Favaro acknowledges support from the Swiss National Science Foundation on project 200021_149227. Hamed Pirsiavash acknowledges support from GE Global Research.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015.

[2] C. Arteta, V. Lempitsky, and A. Zisserman. Counting in the wild. In *ECCV*, 2016.

[3] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. Better mixing via deep representations. In *ICML*, 2013.

[4] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *CVPR*, 2008.

[5] A. B. Chan and N. Vasconcelos. Bayesian poisson regression for crowd counting. In *ICCV*, 2009.

[6] P. Chattopadhyay, R. Vedantam, R. R. Selvaraju, D. Batra, and D. Parikh. Counting everyday objects in everyday scenes. *arXiv preprint arXiv:1604.03505v2*, 2016.

[7] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face. In *CVPR*, 2005.

[8] J. Dai. Generative modeling of convolutional neural networks. In *ICLR*, 2015.

[9] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[10] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2017.

[11] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *NIPS*, 2014.

[14] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science, 313(5786):504507*, 2006.

[15] H. Idrees, K. Soomro, and M. Shah. Detecting humans in dense crowds using locally-consistent scale prior and global occlusion reasoning. *PAMI*, 2015.

[16] Itseez. *The OpenCV Reference Manual*, 2.4.9.0 edition, April 2014.

[17] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM-MM*, 2014.

[19] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. In *ICLR*, 2016.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[21] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016.

[22] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017.

[23] V. Lempitsky and A. Zisserman. Learning to count objects in images. In *NIPS*, 2010.

[24] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, 2015.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

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

[27] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016.

[28] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In *ECCV*, 2016.

[29] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[30] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *arXiv preprint arXiv:1603.09246*, 2016.

[31] A. Owens, J. Wu, J. H. M. annd William T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016.

[32] D. Pathak, R. Girshick, P. Dollr, T. Darrell, and B. Hariharan. Learning features by watching objects move. *arXiv preprint arXiv:1612.06370*, 2016.

[33] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[34] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta. The curious robot: Learning visual representations via physical interactions. In *ECCV*, 2016.

[35] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[36] S. Reed, Y. Zhang, Y. Zhang, and H. Lee. Deep visual analogy-making. In *NIPS*, 2015.

[37] M. Ren and R. S. Zemel. End-to-end instance segmentation with recurrent attention. *arXiv:1605.09410v4*, 2017.

[38] 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. *IJCV*, 2015.

[39] J. Shao, K. Kang, C. C. Loy, and X. Wang. Deeply learned attributes for crowded scene understanding. In *CVPR*, 2015.

[40] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2006.

[41] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

[42] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, 2015.

[43] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016.

[44] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *arXiv preprint arXiv:1611.09842*, 2016.

[45] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. In *NIPS*, 2014.