

Self-Supervised Feature Learning by Learning to Spot Artifacts

Simon Jenni Paolo Favaro
 University of Bern, Switzerland
 {jenni,favaro}@inf.unibe.ch

Abstract

We introduce a novel self-supervised learning method based on adversarial training. Our objective is to train a discriminator network to distinguish real images from images with synthetic artifacts, and then to extract features from its intermediate layers that can be transferred to other data domains and tasks. To generate images with artifacts, we pre-train a high-capacity autoencoder and then we use a damage and repair strategy: First, we freeze the autoencoder and damage the output of the encoder by randomly dropping its entries. Second, we augment the decoder with a repair network, and train it in an adversarial manner against the discriminator. The repair network helps generate more realistic images by inpainting the dropped feature entries. To make the discriminator focus on the artifacts, we also make it predict what entries in the feature were dropped. We demonstrate experimentally that features learned by creating and spotting artifacts achieve state of the art performance in several benchmarks.

1. Introduction

Recent developments in deep learning have demonstrated impressive capabilities in learning useful features from images [23], which could then be transferred to several other tasks [12, 13, 35, 38]. These systems rely on large annotated datasets, which require expensive and time-consuming human labor. To address these issues *self-supervised learning* methods have been proposed [7, 28, 32, 42, 44]. These methods learn features from images without annotated data. Some introduce a *pretext task* through the following strategy: one withholds some information about the input data and then trains a network to recover it. For example, some methods withhold image regions [32], color [44] or both grayscale and color values [45]; others have withheld the location of patches [7, 28], or additional external information such as egomotion [19]. In self-supervised learning the main challenge is to define a pretext task that relates the most to the final applications of the learned features.



Figure 1. A mixture of real images (green border) and images with synthetic artifacts (red border). Is a good object representation necessary to tell them apart?

Towards this goal, we propose to learn features by classifying images as real or with artifacts (see Figure 1). We aim at creating image artifacts, such that a model capable of spotting them would require an accurate representation of objects and thus build features that could transfer well to tasks such as object classification, detection and segmentation. A first approach to create artifacts is to use inpainting algorithms [2, 6]. Besides being computationally inefficient on large inpainting regions, these methods are unsuitable for training because they may introduce low-level statistics that a neural network could easily learn to detect. This could limit what the network learns about objects. Therefore, instead of editing images at the pixel level, we tamper with their feature representation and create corrupt images so that the texture artifacts are locally unnoticeable, but globally incorrect, as illustrated in Figure 1.

To generate artifacts we first train an autoencoder to reproduce images. Then, we randomly drop entries from the encoded feature (at the bottleneck) so that some information about the input image is lost. We then add a *repair* neural network to the decoder to help it render a realistic image. The repair network inpaints the feature representations at every layer of the decoder, but its limited capacity does not allow it to fully recover the missing information. In this way we obtain an image with artifacts that cannot be detected through local analysis. We then train a discrim-

inator to distinguish real from corrupt images. Moreover, we also make the discriminator output a mask to indicate what feature entries were dropped. This implicitly helps the discriminator focus on the important details in the image. We also use the true mask to restrict the scope of the repair network to the features corresponding to the dropped entries. This limits the ability of the repair network to replace missing information in a globally consistent manner. The repair network and the discriminator are then trained in an adversarial fashion. However, in contrast to other adversarial schemes, notice that our repair network is designed not to completely confuse the discriminator. Finally, we transfer features from the discriminator, since this is the model that learns an approximation of the distribution of images.

Our contributions can be summarized as follows: 1) a novel feature learning framework based on detecting images with artifacts, which does not require human annotation; 2) a method to create images with non-trivial artifacts; 3) our features achieve state of the art performance on several transfer learning evaluations (ILSVRC2012 [5], Pascal VOC [11] and STL-10 [4]).

2. Prior Work

This work relates to several topics in machine learning: adversarial networks, autoencoders and self-supervised learning, which we review briefly here below.

Adversarial Training. The use of adversarial networks has been popularized by the introduction of the generative adversarial network (GAN) model by Goodfellow *et al.* [14]. Radford *et al.* [34] introduced a convolutional version of the GAN along with architectural and training guidelines in their DCGAN model. Since GANs are notoriously difficult to train much work has focused on heuristics and techniques to stabilize training [36, 34]. Donahue *et al.* [8] extend the GAN model with an encoder network that learns the inverse mapping of the generator. Their encoder network learns features that are comparable to contemporary unsupervised methods. They show that transferring the discriminator features in the standard GAN setting leads to significantly worse performance. In contrast, our model demonstrates that by limiting the capabilities of the generator network to local alterations the discriminator network can learn better visual representations.

Autoencoders. The autoencoder model is a common choice for unsupervised representation learning [16], with two notable extensions: the denoising autoencoder [41] and the variational autoencoder (VAE) [21]. Several combinations of autoencoders and adversarial networks have recently been introduced. Makhzani *et al.* [26] introduce adversarial autoencoders, which incorporate concepts of VAEs and GANs by training the latent hidden space of an autoencoder to match (via an adversarial loss) some pre-defined prior distribution. In our model the autoencoder network is sep-

arated from the generator (*i.e.*, repair network) and we do not aim at learning semantically meaningful representations with the encoder.

Self-supervised Learning. A recently introduced paradigm in unsupervised feature learning uses either naturally occurring or artificially introduced supervision as a means to learn visual representations. The work of Pathak *et al.* [32] combines autoencoders with an adversarial loss for the task of inpainting (*i.e.*, to generate the content in an image region given its context) with state-of-the-art results in semantic inpainting. Similarly to our method, Denton *et al.* [6] use the pretext task of inpainting and transfer the GAN discriminator features for classification. However, we remove and inpaint information at a more abstract level (the internal representation), rather than at the raw data level. This makes the generator produce more realistic artifacts that are then more valuable for feature learning. Wang and Gupta [42] use videos and introduce supervision via visual tracking of image-patches. Pathak *et al.* [31] also make use of videos through motion-based segmentation. The resulting segmentation is then used as the supervisory signal. Derscher *et al.* [7] use spatial context as a supervisory signal by training their model to predict the relative position of two random image patches. Noroozi and Favaro [28] build on this idea by solving jigsaw puzzles. Zhang *et al.* [44] and Larsson *et al.* [24] demonstrate good transfer performance with models trained on the task of image colorization, *i.e.*, predicting color channels from luminance. The split-brain autoencoder [45] extends this idea by also predicting the inverse mapping. The recent work of Noroozi *et al.* [29] introduces counting of visual primitives as a pretext task for representation learning. Other sources of natural supervision include ambient sound [30], temporal ordering [27] and physical interaction [33]. We compare our method to the above approaches and show that it learns features that, when transferred to other tasks, yield a higher performance on several benchmarks.

3. Architecture Overview

We briefly summarize the components of the architecture that will be introduced in the next sections. Let \mathbf{x} be a training image from a dataset and $\hat{\mathbf{x}}$ be its version with artifacts. As model, we use a neural network consisting of the following components (see also Figure 2):

1. Two *autoencoder* networks $\{E, D_1, D_2, D_3, D_4, D_5\}$, where E is the encoder and $D = \{D_1, D_2, D_3, D_4, D_5\}$ is the decoder, pre-trained to reproduce high-fidelity real images \mathbf{x} ; $\phi(\mathbf{x})$ is the output of the encoder E on \mathbf{x} ;
2. A spatial mask Ω to be applied to the feature output of E ; the resulting masked feature is denoted $\hat{\phi}(\mathbf{x}) = \Omega \odot \phi(\mathbf{x}) + (1 - \Omega) \odot (u * \phi(\mathbf{x}))$, where u is some uniform spatial kernel so that $u * \phi(\mathbf{x})$ is a feature average, and \odot denotes the element-wise product in the

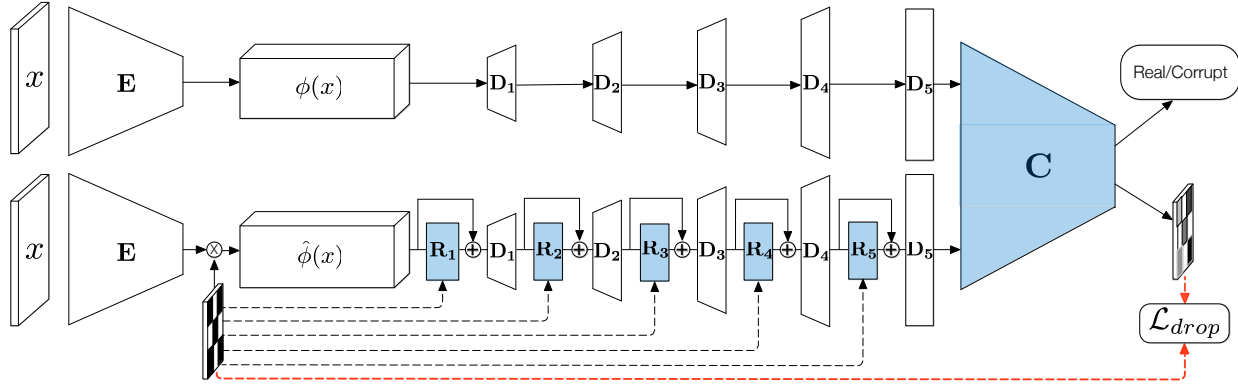


Figure 2. The proposed architecture. Two autoencoders $\{E, D_1, D_2, D_3, D_4, D_5\}$ output either real images (top row) or images with artifacts (bottom row). A discriminator C is trained to distinguish them. The corrupted images are generated by masking the encoded feature $\phi(x)$ and then by using a repair network $\{R_1, R_2, R_3, R_4, R_5\}$ distributed across the layers of the decoder. The mask is also used by the repair network to change only the dropped entries of the feature (see Figure 5 for more details). The discriminator and the repair network (both shaded in blue) are trained in an adversarial fashion on the real/corrupt classification loss. The discriminator is also trained to output the mask used to drop feature entries, so that it learns to localize all artifacts.

spatial domain (the mask is replicated along the channels);
3. A *discriminator* network C to classify x as real images and \hat{x} as fake; we also train the discriminator to output the mask Ω , so that it learns to localize all artifacts;
4. A *repair* network $\{R_1, R_2, R_3, R_4, R_5\}$ added to the layers of one of the two decoder networks; the output of a layer R_i is masked by Ω so that it affects only masked features.

The repair network and the discriminator are trained in an adversarial fashion on the real/corrupt classification loss.

4. Learning to Spot Artifacts

Our main objective is to train a classifying network (the discriminator) so that it learns an accurate distribution of real images. Prior work [34] showed that a discriminator trained to distinguish real from fake images develops features with interesting abstraction capabilities. In our work we build on this observation and exploit a way to control the level of corruption of the fake images (see Sec. 4.1). Thus, we train a classifier to discriminate between real and corrupt images (see Sec. 4.3). As illustrated earlier on, by solving this task we hope that the classifier learns features suitable for other tasks such as object classification, detection and segmentation. In the next sections we describe our model more in detail, and present the design choices aimed at avoiding learning trivial features.

4.1. The Damage & Repair Network

In our approach we would like to be able to create corrupt images that are not too unrealistic, otherwise, a classifier

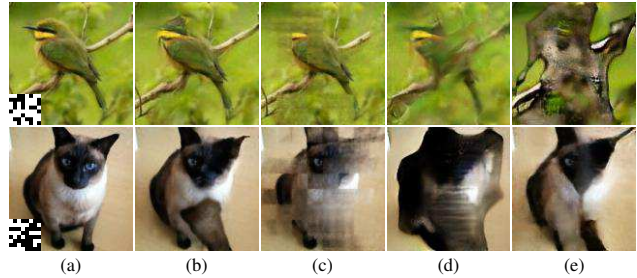


Figure 3. Two examples of corrupt images obtained from our *damage & repair* network. (a) shows two original images from the ImageNet dataset. At the bottom-left corner of those images we show the masks applied to the encoded feature $\phi(x)$. These masks drop on average about 50% of the encoded feature. (b) shows the output corrupt images. The repair network assists the decoder in inpainting texture that is only locally unnoticeable. However, at the global scale the objects are no longer recognizable as valid instances. (c) shows the output of the decoder when the repair network is not active. In this case the artifacts are very visible and easy to detect by exploiting low-level statistics. (d) shows the output of the decoder when the repair network is not masked. The repair network is then able to change the image more globally. This has a negative effect on the discriminator as it fails to predict the mask. (e) shows an example where the images are fed through the *damage & repair* network twice. This results in even more artifacts than in (b).

could distinguish them from real images by detecting only low-level statistics (*e.g.*, unusual local texture patterns). At the same time, we would like to have as much variability as possible, so that a classifier can build a robust model of real images.

To address the latter concern we randomly corrupt real images of an existing dataset. To address the first concern instead of editing images at the pixel-level, we cor-

rupt their feature representation and then partly repair the corruption by fixing only the low-level details. We encode an image \mathbf{x} in a feature $\phi(\mathbf{x}) \in \mathbf{R}^{M \times N \times L}$, where $\phi(\mathbf{x}) = E(\mathbf{x})$, and then at each spatial coordinate in the $M \times N$ domain we randomly drop all the L channels with a given probability $\theta \in (0, 1)$. This defines a binary mask matrix $\Omega \in \{0, 1\}^{M \times N}$ of what feature entries are dropped ($\Omega_{ij} = 0$) and which ones are preserved ($\Omega_{ij} = 1$). The dropped feature channels are replaced by the corresponding entries of an averaged feature computed by convolving $\phi(\mathbf{x})$ with a large uniform kernel u . As an alternative, we also replace the dropped feature channels with random noise. Experimentally, we find no significant difference in performance between these two methods. The mask Ω is generated online during training and thus the same image \mathbf{x} is subject to a different mask at every epoch. If we fed the corrupt feature directly to the decoder D the output would be extremely unrealistic (see Figure 3 (c)). Thus, we introduce a *repair* network that partially compensates for the loss of information due to the mask. The repair network introduces repair layers R_i between layers of the decoder D_i . These layers receive as input the corrupt feature or the outputs of the decoder layers and are allowed to fix only entries that were dropped. More precisely, we define the input to the first decoder layer D_1 as

$$\hat{\phi}(\mathbf{x}) + (1 - \Omega) \odot R_1(\hat{\phi}(\mathbf{x})), \quad (1)$$

where $\hat{\phi}(\mathbf{x}) = \Omega \odot \phi(\mathbf{x}) + (1 - \Omega) \odot (u * \phi(\mathbf{x}))$ and u is a large uniform filter. At the later layers D_2 , D_3 , and D_4 we upsample the mask Ω with the nearest neighbor method and match the spatial dimension of the corresponding intermediate output, *i.e.*, we provide the following input to each layer D_i with $i = 2, 3, 4$

$$D_{i-1} + (1 - U_{i-1}(\Omega)) \odot R_i(D_{i-1}), \quad (2)$$

where U_{i-1} denotes the nearest neighbor upsampling to the spatial dimensions of the output of D_{i-1} . Finally, we also design our encoder E so that it encodes features that are spatially localized and with limited overlap with one another. To do so, we define the encoder $E = \{E_1, E_2, E_3, E_4, E_5\}$ with five layers where E_1 uses 3×3 convolutional filters with stride 1 and the remaining four layers E_2, E_3, E_4 , and E_5 use 2×2 convolutional filters with stride 2. As shown in Figure 4, this design limits the receptive field of each encoded feature entry to a 18×18 pixels patch in the input image with a 2 pixels overlap with neighboring patches. Thus, dropping one of these feature entries is essentially equivalent to dropping one of the 18×18 pixels patches in the input image.

In Figure 3 we show two examples of real images and corresponding corrupt images obtained under different conditions. Figure 3 (a) shows the original images and the mask

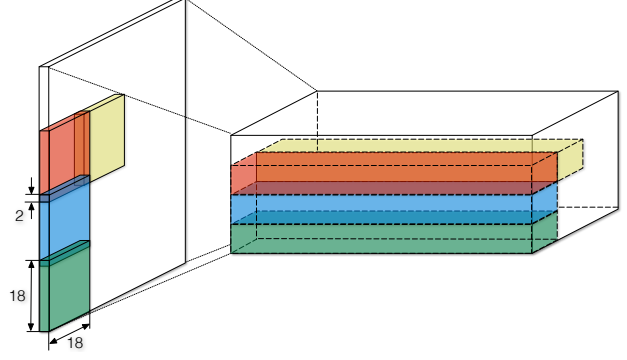


Figure 4. Encoder design. On the left we show the input image and on the right we show the corresponding feature encoding $\phi(\mathbf{x})$. We use 5 layers, where the first one uses 3×3 convolutional filters with stride 1 and the remaining 4 use 2×2 convolutional filters with stride 2. As can be observed, this results in almost separate receptive fields for each feature entry of $\phi(\mathbf{x})$. Each entry corresponds to a 18×18 pixels patch in the original input and overlaps 2 pixels with the neighboring patches (1 pixel each side). This encoding ensures a strong spatial locality to each entry of the encoded features.

Ω , where $M, N = 8$, as inset on the bottom-left corner. The dropping probability is $\theta = 0.5$. Figure 3 (b) shows the resulting corrupt images obtained by our complete architecture. Notice that locally it is impossible to determine whether the image is real or corrupt. Only by looking at a larger region, and by exploiting prior knowledge of what real objects look like, it is possible to determine that the image is corrupt. For comparison purposes we show in Figure 3 (c) and (d) the corresponding corrupt images obtained by disabling the repair network and by using the repair network without the mask restriction, respectively. Finally, in Figure 3 (e) we apply the repair network twice and observe that more easy-to-detect low-level artifacts have been introduced.

4.2. Replication of Real Images

Given the corrupt images generated as described in the previous section, we should be ready to train the discriminator C . The real images could indeed be just the original images from the same dataset that we used to create the corrupt ones. One potential issue with this scheme is that the discriminator may learn to distinguish real from corrupt images based on image processing patterns introduced by the decoder network. These patterns may be unnoticeable to the naked eye, but neural networks seem to be quite good at spotting them. For example, networks have learnt to detect chromatic aberration [7] or downsampling patterns [29].

To avoid this issue we use the same autoencoder $\{E, D\}$ used to generate corrupt images, also to replicate real images. Since the same last layer D_5 of the decoder is used to generate the real and corrupt images, we expect both images

to share the same processing patterns. This should discourage the discriminator from focusing on such patterns.

We therefore pre-train this autoencoder and make sure that it has a high capacity, so that images are replicated with high accuracy. The training of E and D is a standard optimization with the following least-squares objective

$$\mathcal{L}_{\text{auto}} = \sum_{\mathbf{x} \sim p(\mathbf{x})} |D(E(\mathbf{x})) - \mathbf{x}|^2. \quad (3)$$

4.3. Training the Discriminator

As just discussed in Sec. 4.2, to replicate real images we use an autoencoder $\{E, D\}$ and, as described in Sec. 4.1, to create corrupt images we use a *damage & repair* autoencoder $\{\Omega \odot E, \hat{D}\}$, where $\hat{D} = \{R_1, D_1, R_2, D_2, R_3, D_3, R_4, D_4, R_5, D_5\}$. Then, we train the discriminator and the repair subnetwork $R = \{R_1, R_2, R_3, R_4, R_5\}$ via adversarial training [14]. Our discriminator C has two outputs, a binary probability $C^{\text{class}} \in [0, 1]$ for predicting real vs corrupt images and a prediction mask $C^{\text{mask}} \in [0, 1]^{M \times N}$ to localize artifacts in the image. Given an image \mathbf{x} , training the discriminator C and the repair network R involves solving

$$\begin{aligned} \mathcal{L}_{\text{class}} = \min_R \max_C \sum_{\mathbf{x} \sim p(\mathbf{x})} & \log C^{\text{class}}(D(\phi(\mathbf{x}))) \\ & + \log(1 - C^{\text{class}}(\hat{D}(\hat{\phi}(\mathbf{x}))))). \end{aligned} \quad (4)$$

We also train the discriminator to predict the mask Ω by minimizing

$$\begin{aligned} \mathcal{L}_{\text{mask}} = \min_C \sum_{\hat{\mathbf{x}}} \sum_{ij} \Omega_{ij} \log \sigma(C_{ij}^{\text{mask}}(\hat{\mathbf{x}})) \\ + (1 - \Omega_{ij}) \log(1 - \sigma(C_{ij}^{\text{mask}}(\hat{\mathbf{x}}))) \end{aligned} \quad (5)$$

where $\hat{\mathbf{x}} = \hat{D}(\hat{\phi}(\mathbf{x}))$ and $\sigma(z) = 1/(1+e^{-z})$ is the sigmoid function.

4.4. Implementation

Let (64)3c2 denote a convolutional layer with 64 filters of size 3×3 with a stride of 2. The architecture of the encoder E is then defined by (32)3c1-(64)2c2-(128)2c2-(256)2c2-(512)2c2. The decoder network D is given by (256)3rc2-(128)3rc2-(64)3rc2-(32)3rc2-(3)3c1 where *rc* denotes resize-convolutions (i.e., bilinear resizing followed by standard convolution). Batch normalization [18] is applied at all layers of E and D except the last convolutional layer of D . All convolutional layers in E and D are followed by the leaky-ReLU activation $f(x) = \max(x/10, x)$. The filtering of $\phi(x)$ with u is realized using 2D average pooling with a kernel of size 3×3 .

The discriminator network C is based on the standard AlexNet architecture [23] to allow for a fair comparison

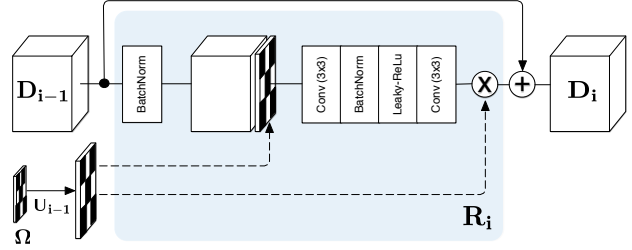


Figure 5. The repair block. We use a residual block design. We also concatenate the drop mask to the output of the batch normalization layer as an additional channel before feeding it to the first convolutional layer. We also gate the output of the last convolutional layer with the same drop mask. This ensures that only corrupted regions can be altered by the repair block.

with other methods. The network is identical to the original up to conv5. We drop pool5 and use a single 3×3 convolutional layer for the mask prediction. For the classification we remove the second fully-connected layer. Batch normalization is only applied after conv5 during unsupervised training and removed in transfer experiments. The standard ReLU activation is used throughout C . The repair layers follow a similar design as the residual blocks found in ResNet [15]. Their design is illustrated in Figure 5.

Adam [20] with an initial learning rate of $3 \cdot 10^{-4}$ and the momentum parameter $\beta_1 = 0.5$ is used for the optimization. We keep all other hyper-parameters at their default values. During training we linearly decay the learning rate to $3 \cdot 10^{-6}$. The autoencoder is pre-trained for 80 epochs and the *damage & repair* network is trained for 150 epochs on random 128×128 crops of the 1.3M ImageNet [5] training images.

5. Experiments

5.1. Ablation Analysis

We validate the main design choices in our model by performing transfer learning for classification on STL-10 [4]. All results were obtained with models trained for 400 epochs on the unlabelled training set and supervised transfer learning for 200 epochs on the labelled training set. During transfer the standard AlexNet architecture is used with the exception that we drop the pool5 layer in order to handle the smaller image size. The weights in the convolutional layers are transferred while the fully-connected layers are randomly initialized. We perform the following set of experiments:

- (a) **Input image as real:** We show that it is better to use autoencoded images as real examples for discriminator training. The rationale here is that the discriminator could exploit the pixel patterns of the decoder network to decide between real and corrupted images. We also observed common GAN artifacts in this setting (see

Figure 6 (a)). In our model, the inputs to the discriminator pass through the same convolutional layer.

(b) Distributed vs. local repair network: Here we illustrate the importance of distributing the repair network throughout the decoder. In the local case we apply the five repair layers consecutively before the first decoder layer. We observe more low-level artifacts in this setup (see Figure 6 (b)).

(c)-(f) Dropping rate: We show how different values of θ influence the performance on classification. The following values are considered: 0.1, 0.3, 0.5 (the baseline), 0.7 and 0.9. The dropping rate has a strong influence on the performance of the learnt features. Values around 0.7 give the best results and low dropping rates significantly reduce performance. With low dropping rates it is unlikely that object parts are corrupted. This makes the examples less valuable for learning semantic content.

(g) Without mask prediction: This experiment demonstrates that the additional self-supervision signal provided through the drop mask improves performance.

(h) 3×3 encoder convolutions: In this experiment we let the encoder features overlap. We replace the 2×2 convolutions with 3×3 convolutions. This increases the receptive field from 18×18 to 33×33 and results in an overlap of 15 pixels. We observe a small decrease in transfer performance.

(i) No gating in repair layers: We demonstrate the influence of training without gating the repair network output with the drop mask. Without gating, the repair network can potentially affect all image regions.

(j) No history of corrupted examples: Following [39] we keep a buffer of corrupted examples and build minibatches where we replace half of the corrupted examples with samples from the buffer. Removing the history has a small negative effect on performance.

(k) No repair network: In this ablation we remove the repair network completely. The poor performance in this case illustrates the importance of adversarial training.

(l) GAN instead of damage & repair: We test the option of training a standard GAN to create the fake examples instead of our damage-repair approach. We observe much poorer performance and unstable adversarial training in this case.

The resulting transfer learning performance of the discriminator on STL-10 is shown in Table 1. In Figure 6 we show renderings for some of the generative models.

5.2. Transfer Learning Experiments

We perform transfer learning experiments for classification, detection and semantic segmentation on standard datasets with the AlexNet discriminator pre-trained using a dropping rate of $\theta = 0.7$.

Table 1. Influence of different architectural choices on the classification accuracy on STL-10 [4]. Convolutional layers were pre-trained on the proposed self-supervised task and kept fixed during transfer for classification.

Ablation experiment	Accuracy
Baseline (dropping rate = 0.5)	79.94%
(a) Input image as real	74.99%
(b) Distributed vs. local repair network	77.51%
(c) Dropping rate = 0.1	70.92%
(d) Dropping rate = 0.3	76.26%
(e) Dropping rate = 0.7	81.06%
(f) Dropping rate = 0.9	79.60%
(g) Without mask prediction	78.44%
(h) 3×3 encoder convolutions	79.84%
(i) No gating in repair layers	79.66%
(j) No history of corrupted examples	79.76%
(k) No repair network	54.74%
(l) GAN instead of damage & repair	56.59%

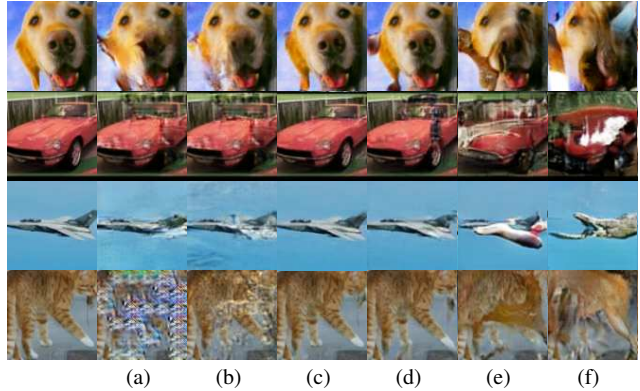


Figure 6. The *damage & repair* network renderings. The left-most column shows the input image. Column (a) shows results when input images are used as real examples. Note that this introduces commonly observed GAN artifacts. Column (b) shows results with the local instead of distributed repair network. Columns (c)-(f) show results with dropping rates of 0.1, 0.3, 0.7 and 0.9.

5.2.1 Classification on STL-10

The STL-10 dataset [4] is designed with unsupervised representation learning in mind and is a common baseline for comparison. The dataset contains 100,000 *unlabeled* training images, 5,000 *labeled* training images evenly distributed across 10 classes for supervised transfer and 8,000 test images. The data consists of 96×96 color images. We randomly resize the images and extract 96×96 crops. Unsupervised training is performed for 400 epochs and supervised transfer for an additional 200 epochs.

We follow the standard evaluation protocol and perform supervised training on the ten pre-defined folds. We compare the resulting average accuracy to state-of-the-art results in Table 2. We can observe an increase in performance

Table 2. Comparison of test-set accuracy on STL-10 with other published results. Following the guidelines in [4] the average accuracy from models trained on the ten pre-defined folds is reported. We train a linear classifier on top of conv5 features for a fair comparison with the other methods.

Model	Accuracy	SD
Dosovitskiy <i>et al.</i> [9]	74.2%	± 0.4
Dundar <i>et al.</i> <i>et al.</i> [10]	74.1%	-
Huang <i>et al.</i> [17]	76.8%	± 0.3
Swersky <i>et al.</i> [40]	70.1%	± 0.6
Zhao <i>et al.</i> [46]	74.3%	-
Denton <i>et al.</i> [6] (finetuned)	77.8%	± 0.8
Ours (conv1-conv5 frozen)	76.9%	± 0.2
Ours (conv1-conv5 finetuned)	80.1%	± 0.3

Table 3. Transfer learning results for classification, detection and segmentation on Pascal VOC2007 and VOC2012 compared to state-of-the-art feature learning methods.

Model	[Ref]	Classification	Detection	Segmentation
		(mAP)	(mAP)	(mIU)
Krizhevsky <i>et al.</i> [23]	[44]	79.9%	56.8%	48.0%
Random	[32]	53.3%	43.4%	19.8%
Agrawal <i>et al.</i> [1]	[8]	54.2%	43.9%	-
Bojanowski <i>et al.</i> [3]	[3]	65.3%	49.4%	-
Doersch <i>et al.</i> [7]	[8]	65.3%	51.1%	-
Donahue <i>et al.</i> [8]	[8]	60.1%	46.9%	35.2%
Jayaraman & Grauman [19]	[19]	-	41.7%	-
Krähenbühl <i>et al.</i> [22]	[22]	56.6%	45.6%	32.6%
Larsson <i>et al.</i> [24]	[24]	65.9%	-	<u>38.0%</u>
Noroozi & Favaro [28]	[28]	67.6%	53.2%	37.6%
Noroozi <i>et al.</i> [29]	[29]	<u>67.7%</u>	51.4%	36.6%
Owens <i>et al.</i> [30]	[30]	61.3%	44.0%	-
Pathak <i>et al.</i> [32]	[32]	56.5%	44.5%	29.7%
Pathak <i>et al.</i> [31]	[31]	61.0%	52.2%	-
Wang & Gupta [42]	[22]	63.1%	47.4%	-
Zhang <i>et al.</i> [44]	[44]	65.9%	46.9%	35.6%
Zhang <i>et al.</i> [45]	[45]	67.1%	46.7%	36.0%
Ours	-	69.8%	<u>52.5%</u>	38.1%

over the other methods. Note that the models compared in Table 2 do not use the same network architecture, hence making it difficult to attribute the difference in performance to a specific factor. It nonetheless showcases the potential of the proposed self-supervised learning task and model.

5.2.2 Multilabel Classification, Detection and Segmentation on PASCAL VOC

The Pascal VOC2007 and VOC2012 datasets consist of images coming from 20 object classes. It is a relatively challenging dataset due to the high variability in size, pose, and position of objects in the images. These datasets are standard benchmarks for representation learning. We only transfer the convolutional layers of our AlexNet based discriminator and randomly initialize the fully-connected layers. The data-dependent rescaling proposed by Krähenbühl

Table 4. Validation set accuracy on ImageNet with linear classifiers trained on the frozen convolutional layers after unsupervised pre-training. Results for the other methods are taken from [29].

Model\Layer	conv1	conv2	conv3	conv4	conv5
Krizhevsky <i>et al.</i> [23]	19.3%	36.3%	44.2%	48.3%	50.5%
Random	11.6%	17.1%	16.9%	16.3%	14.1%
Doersch <i>et al.</i> [7]	16.2%	23.3%	30.2%	31.7%	29.6%
Donahue <i>et al.</i> [8]	17.7%	24.5%	31.0%	29.9%	28.0%
Krähenbühl <i>et al.</i> [22]	17.5%	23.0%	24.5%	23.2%	20.6%
Noroozi & Favaro [28]	<u>18.2%</u>	28.8%	34.0%	33.9%	27.1%
Noroozi <i>et al.</i> [29]	18.0%	30.6%	34.3%	32.5%	25.7%
Pathak <i>et al.</i> [32]	14.1%	20.7%	21.0%	19.8%	15.5%
Zhang <i>et al.</i> [44]	13.1%	24.8%	31.0%	32.6%	31.8%
Zhang <i>et al.</i> [45]	17.7%	29.3%	<u>35.4%</u>	<u>35.2%</u>	<u>32.8%</u>
Ours	19.5%	33.3%	37.9%	38.9%	34.9%

et al. [22] is used in all experiments as is standard practice. The convolutional layers are fine-tuned, *i.e.*, not frozen. This demonstrates the usefulness of the discriminator weights as an initialization for other tasks. A comparison to the state-of-the-art feature learning methods is shown in Table 3.

Classification on VOC2007. For multilabel classification we use the framework provided by Krähenbühl *et al.* [22]. Fine-tuning is performed on random crops of the 'trainval' dataset. The final predictions are computed as the average prediction of 10 random crops per test image. With a mAP of 69.8% we achieve a state-of-the-art performance on this task.

Detection on VOC2007. The Fast-RCNN [12] framework is used for detection. We follow the guidelines in [22] and use multi-scale training and single-scale testing. All other settings are kept at their default values. With a mAP of 52.5% we achieve the second best result.

Semantic Segmentation on VOC2012. We use the standard FCN framework [25] with default settings. We train for 100,000 iterations using a fixed learning rate of 10^{-4} . Our discriminator weights achieve a state-of-the-art result with a mean intersection over union (mIU) of 38.1%.

5.2.3 Layerwise Performance on ImageNet & Places

We evaluate the quality of representations learned at different depths of the network with the evaluation framework introduced in [44]. All convolutional layers are frozen and multinomial logistic regression classifiers are trained on top of them. The outputs of the convolutional layers are resized such that the flattened features are of similar size (~ 9200). A comparison to other models on ImageNet is given in Table 4. Our model outperforms all other approaches in this benchmark. Note also that our conv1 features perform even slightly better than the supervised counterparts. To demonstrate that the learnt representations generalize to other input data, the same experiment was also

Table 5. Validation set accuracy on Places with linear classifiers trained on the frozen convolutional layers after unsupervised pre-training. Results for the other methods are taken from [29].

Model \ Layer	conv1	conv2	conv3	conv4	conv5
Places-labels <i>et al.</i> [23]	22.1%	35.1%	40.2%	43.3%	44.6%
ImageNet-labels <i>et al.</i> [23]	22.7%	34.8%	38.4%	39.4%	38.7%
Random	15.7%	20.3%	19.8%	19.1%	17.5%
Doersch <i>et al.</i> [7]	19.7%	26.7%	31.9%	32.7%	30.9%
Donahue <i>et al.</i> [8]	22.0%	28.7%	31.8%	31.3%	29.7%
Krähenbühl <i>et al.</i> [22]	21.4%	26.2%	27.1%	26.1%	24.0%
Noroozi & Favaro [28]	23.0%	31.9%	35.0%	34.2%	29.3%
Noroozi <i>et al.</i> [29]	23.3%	33.9%	36.3%	34.7%	29.6%
Owens <i>et al.</i> [30]	19.9%	29.3%	32.1%	28.8%	29.8%
Pathak <i>et al.</i> [32]	18.2%	23.2%	23.4%	21.9%	18.4%
Wang & Gupta [42]	20.1%	28.5%	29.9%	29.7%	27.9%
Zhang <i>et al.</i> [44]	16.0%	25.7%	29.6%	30.3%	29.7%
Zhang <i>et al.</i> [45]	21.3%	30.7%	34.0%	34.1%	<u>32.5%</u>
Ours	23.3%	34.3%	36.9%	37.3%	34.4%

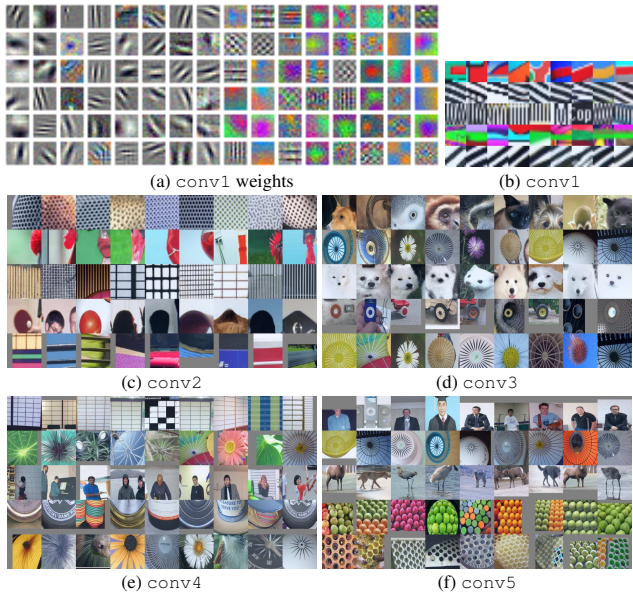


Figure 7. Visualisation of the learnt features at different layers of the AlexNet after unsupervised training. We show conv1 weights and maximally activating image-patches for five neurons at each layer.

performed on the Places [47] dataset. This dataset contains 2.4M images from 205 scene categories. As can be seen in Table 5 we outperform all the other methods for layers conv2–conv5. Note also that we achieve the highest overall accuracy with 37.3%.

5.3. Qualitative Analysis of the Features

To better understand what the discriminator has learnt we use different network visualization techniques. We show the learnt conv1 filters as well as maximally activating image-patches [13, 43] for some neurons of each convolutional layer in Figure 7. We observe prominent edge-detectors



Figure 8. Nearest-neighbor retrievals on the ImageNet validation set obtained with conv5 features from the AlexNet discriminator. Nearest-neighbors were computed using cosine-similarity.

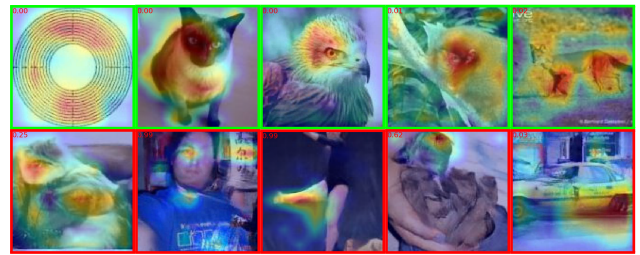


Figure 9. Visualisation of contributing image features for the decision between real images (green border) and images with artifacts (red border). The discriminator appears to focus on object parts such as legs and heads as well as geometric shapes.

in the conv1 filters, much like what can be observed in a supervised AlexNet. Figure 8 shows Nearest-Neighbor retrievals obtained with our conv5 features.

We use Grad-CAM [37] in Figure 9 to illustrate what image regions the discriminator focuses on when making the decision between real and with artifacts. We can observe that the discriminator often looks for missing or existing object parts.

6. Conclusions

We have shown how to learn features by classifying images into real or corrupt. This classification task is designed to use images without human annotation, and thus can exploit large readily-available image datasets. We have tackled this task with a model that combines autoencoders and an assistive network (the repair network) with adversarial networks. The transfer (via fine-tuning) of features learned by the classification network achieves state of the art performance on several benchmarks: ILSVRC2012, Pascal VOC and STL-10.

Acknowledgements. We thank Alexei A. Efros for helpful discussions and feedback. This work was supported by the Swiss National Science Foundation (SNSF) grant number 200021_169622.

References

- [1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015. 7
- [2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000. 1
- [3] P. Bojanowski and A. Joulin. Unsupervised learning by predicting noise. *arXiv preprint arXiv:1704.05310*, 2017. 7
- [4] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011. 2, 5, 6, 7
- [5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 2, 5
- [6] E. Denton, S. Gross, and R. Fergus. Semi-supervised learning with context-conditional generative adversarial networks. *arXiv preprint arXiv:1611.06430*, 2016. 1, 2, 7
- [7] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 1, 2, 4, 7, 8
- [8] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016. 2, 7, 8
- [9] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014. 7
- [10] A. Dundar, J. Jin, and E. Culurciello. Convolutional clustering for unsupervised learning. *arXiv preprint arXiv:1511.06241*, 2015. 7
- [11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2
- [12] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 1, 7
- [13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1, 8
- [14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 2, 5
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 5
- [16] G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, pages 3–3, 1994. 2
- [17] C. Huang, C. Change Loy, and X. Tang. Unsupervised learning of discriminative attributes and visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5175–5184, 2016. 7
- [18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 5
- [19] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1413–1421, 2015. 1, 7
- [20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5
- [21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2
- [22] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015. 7, 8
- [23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 5, 7, 8
- [24] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017. 2, 7
- [25] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 7
- [26] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 2
- [27] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016. 2
- [28] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 1, 2, 7, 8
- [29] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. *arXiv preprint arXiv:1708.06734*, 2017. 2, 4, 7, 8
- [30] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *European Conference on Computer Vision*, pages 801–816. Springer, 2016. 2, 7, 8
- [31] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 7
- [32] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision*

- and *Pattern Recognition*, pages 2536–2544, 2016. 1, 2, 7, 8
- [33] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta. The curious robot: Learning visual representations via physical interactions. In *European Conference on Computer Vision*, pages 3–18. Springer, 2016. 2
 - [34] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2, 3
 - [35] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1
 - [36] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016. 2
 - [37] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *arXiv preprint arXiv:1610.02391*, 2016. 8
 - [38] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014. 1
 - [39] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*, 2016. 6
 - [40] K. Swersky, J. Snoek, and R. P. Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013. 7
 - [41] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010. 2
 - [42] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015. 1, 2, 7, 8
 - [43] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015. 8
 - [44] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016. 1, 2, 7, 8
 - [45] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017. 1, 2, 7, 8
 - [46] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015. 7
 - [47] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 487–495. Curran Associates, Inc., 2014. 8