Self-Supervised Feature Learning by Learning to Spot Artifacts
(2018)

Simon Jenni Paolo Favaro
Notes

Contributions

The authors propose a new pretext task in a self supervised setting, the objective is to train a discriminator network to distinguish real images from images with synthetic artifacts. To generate these synthetic artifacts, first, a high-capacity auto-encoder is pretrained, then some parts of outputs of the encoders are dropped, and in the decoder, a repair network is added to help generate more realistic entries. And then a discriminator is trained to differentiate between real images from images with synthetic artifacts, the discriminator also outputs a mask to indicate what feature entries were dropped.

The discriminator can then be used as a starting point for downstream tasks, this methods can also generate images with non-trivial artifacts.

Method

The architecture

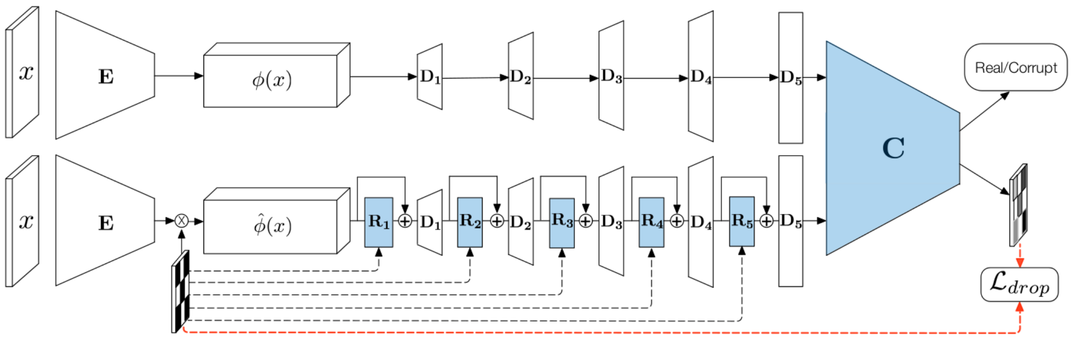
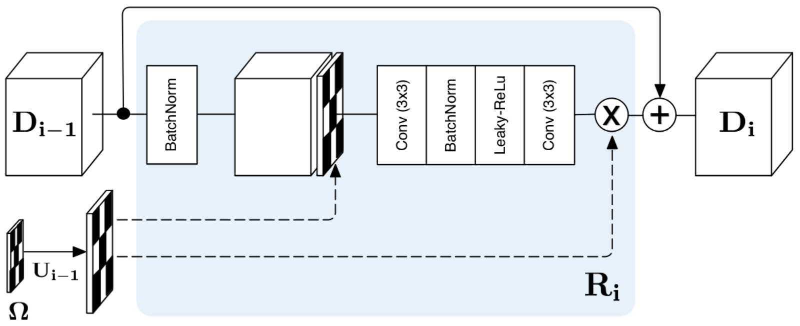


Figure 2. The proposed architecture. Two autoencoders $\{E, D_1, D_2, D_3, D_4, D_5\}$ output either real images (top row) or images with artifacts (bottom row). A discriminator C is trained to distinguish them. The corrupted images are generated by masking the encoded feature $\phi(\mathbf{x})$ and 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.

- Two autoencoder networks $\{E, D_1, D_2, D_3, D_4, D_5\}$, where E is the encoder and $\{D_1, D_2, D_3, D_4, D_5\}$ is the decoder, pre-trained to reproduce high-fidelity real images x; The encoder is {`conv 3x3`, 4 x (`conv 2x2, s=2`)}, and its output is: $\phi(x)$
- A spatial binary mask Ω to be applied to the feature output of E; Each spatial location of the output of E is droped with a probability of $\theta = 0.5$
- A discriminator network C to classify x as real images and $\hat{\mathbf{x}}$ as fake and outputs a mask Ω;

- A repair network $\{R_1, R_2, R_3, R_4, R_5\}$ added to the layers of one of the two decoder networks; at each stage of the decoder, the repair network receives the output of the decoder and an upsampled spatial mask, so that the repair network only changes the parts of dropped by the mask (gating mechanism).



Training

- Pretraining the autoencoder: A simple reconstruction loss: $\mathcal{L}_{\text{auto}} = \sum_{\mathbf{x}\sim p(\mathbf{x})} |D(E(\mathbf{x})) - \mathbf{x}|^2$
- The Discriminator for classification: Given an image x, we train both the discriminator C and the repair network R in a min-max game, which involves solving:

$$\mathcal{L}_{\text{class}} = \min_R \max_C \sum_{\mathbf{x}\sim p(\mathbf{x})} \log C^{\text{class}}\left(D(\phi(\mathbf{x}))\right) + \log\left(1 - C^{\text{class}}\left(\hat{D}(\hat{\phi}(\mathbf{x}))\right)\right)$$

- The Discriminator for mask prediction: the prediction can be viewed as a binary classification over all the spatial position (i, j) of a mask of size $M \times N$:

$$\mathcal{L}_{\text{mask}} = \min_C \sum_{\hat{\mathbf{x}}} \sum_{ij} \Omega_{ij} \log \sigma\left(C^{\text{mask}}_{ij}(\hat{\mathbf{x}})\right)(1 - \Omega_{ij}) \log\left(1 - \sigma\left(C^{\text{mask}}_{ij}(\hat{\mathbf{x}})\right)\right)$$

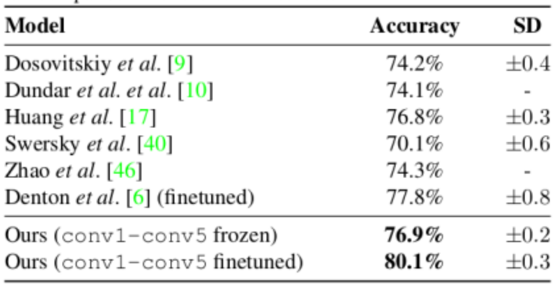
Results

Ablation study Input image as real: Use autoencoded images or real examples for discriminator training. Distributed vs. local repair network: In the local case we apply the five repair layers consecutively before the first decoder layer. Dropping rate: different values of θ, and other tests:

	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%

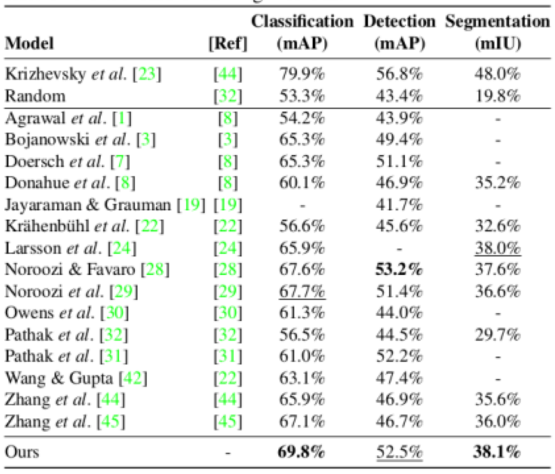
Transfer Learning

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.



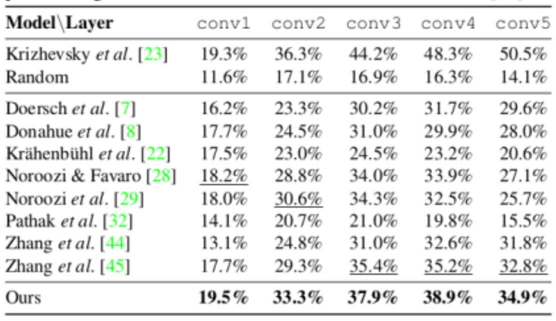
Model	**Accuracy**	**SD**
Dosovitskiy *et al.* [9]	74.2%	±0.4
Dundar *et al.* *et al.* [10]	74.1%	-
Huang *et al.* [17]	76.8%	±0.3
Swersky *et al.* [40]	70.1%	±0.6
Zhao *et al.* [46]	74.3%	-
Denton *et al.* [6] (finetuned)	77.8%	±0.8
Ours (`conv1-conv5` frozen)	**76.9%**	±0.2
Ours (`conv1-conv5` finetuned)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



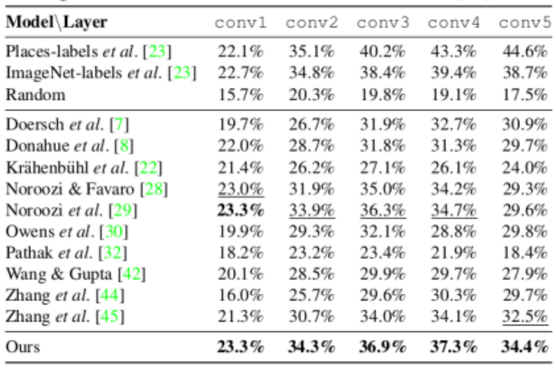
Model	**[Ref]**	**Classification (mAP)**	**Detection (mAP)**	**Segmentation (mIU)**
Krizhevsky *et al.* [23]	[44]	79.9%	56.8%	48.0%
Random	[32]	53.3%	43.4%	19.8%
Agrawal *et al.* [1]	[8]	54.2%	43.9%	-
Bojanowski *et al.* [3]	[3]	65.3%	49.4%	-
Doersch *et al.* [7]	[8]	65.3%	51.1%	-
Donahue *et al.* [8]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

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

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 *et al.* [23]	22.1%	35.1%	40.2%	43.3%	44.6%
ImageNet-labels *et al.* [23]	22.7%	34.8%	38.4%	39.4%	38.7%
Random	15.7%	20.3%	19.8%	19.1%	17.5%
Doersch *et al.* [7]	19.7%	26.7%	31.9%	32.7%	30.9%
Donahue *et al.* [8]	22.0%	28.7%	31.8%	31.3%	29.7%
Krähenbühl *et al.* [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 *et al.* [29]	**23.3%**	33.9%	36.3%	34.7%	29.6%
Owens *et al.* [30]	19.9%	29.3%	32.1%	28.8%	29.8%
Pathak *et al.* [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 *et al.* [44]	16.0%	25.7%	29.6%	30.3%	29.7%
Zhang *et al.* [45]	21.3%	30.7%	34.0%	34.1%	32.5%
Ours	**23.3%**	**34.3%**	**36.9%**	**37.3%**	**34.4%**