Revisiting Self-Supervised Visual Representation Learning
(2019)

Alexander Kolesnikov, Xiaohua Zhai, Lucas Beyer
Notes

Contributions

The authors uncover numerous insights into self supervised learning. First of all, the standard architecture design recipes do not necessarily translate from the fully-supervised to the self-supervised setting. Architecture choices which negligibly affect performance in the fully labeled setting, may significantly affect performance in the self-supervised setting. Having more filters in a CNN model also tend to produce better learned visual representations. And the evaluation the learned representations using a linear classifier, being sensitive to learning rate scheduling, may take many epochs to converge.

In the paper, the authors use the following models: ResNet (v1 and v2), RevNet and VGG. And the following pretext tasks: Rotation, Exemplar, Jigsaw and Relative Patch Location.

Study setup

One of the main focus points of this paper, is whether the effectiveness of the self-supervised methods changes with different convnet architecture, for this, the authors choose to use Resnet and VGG, but also based on the observation that the quality of the representation does not degrade towards the end of the network as a results of skip connection making residual units invertible, they also use RevNet:

- **ResNet** The first 7x7 conv block outputs $k \times 16$ channels, followed by 4 blocks containing different convolutions and one skip connection, for a ResNet 50, the four blocks contain 3, 4, 6, and 3 such units. The output is a vector of size $512 \times k$, with $k \in \{4, 8, 12, 16\}$. We end up with many variants, residual unit for ResnetV1: `(conv, bn, relu, conv, bn)`, ResnetV2: `(bn, relu, conv, bn, relu, conv)`, and also the possibility of using ReLu before the final average pooling (denoted by (-)).
- **RevNet** slightly modifies the design of the residual unit such that it becomes analytically invertible, the input x is split channel-wise into two equal parts x_1 and x_2, and the output is the concatenation of: $y_2 := x_2$ and $y_1 := x_1 + \mathcal{F}(x_2)$.
- **VGG** consists of a series of 3×3 convolutions followed by ReLu arranged into blocks followed by non-linearities. The VGG19 variant has 5 such blocks of 2, 2, 4, 4, and 4 convolutions respectively. Similar to ResNet, the initial convolution produces $8 \times k$ channels and the fully-connected layers have $512 \times k$ channels.

The self-supervised techniques the authors investigate are:

- **Rotation** Producing 4 copies of an image with 4 rotations (0, 90, 180, 270) and train a model to predict the correct rotation.
- **Exemplar** Every individual image corresponds to its own class, and multiple examples of it are generated by heavy random data augmentation, to model in then trained using a triplet loss between two images of the same class and a negative example.

- **Jigsaw** The task is to recover relative spatial position of 9 randomly sampled image patches after a random permutation of these patches was performed. The patches are sampled with gaps between them and each patch is then independently converted to grayscale with probability 2/3 and normalized to zero mean and unit standard deviation.
- **Relative Patch Location** consists of predicting the relative location of two given patches of an image, the final image representations by averaging representations of 9 cropped patches.

Evaluation: the learned visual representations are evaluated by using them for training a linear logistic regression model to solve multi-class image classification tasks requiring high-level scene understanding by training a linear regression on top of these representation (either by using SGD or L-BFGS)

Results

Table 1. Evaluation of representations from self-supervised techniques based on various CNN architectures. The scores are accuracies (in %) of a linear logistic regression model trained on top of these representations using *ImageNet* training split. Our validation split is used for computing accuracies. The architectures marked by a "(-)" are slight variations described in Section 3.1. Sub-columns such as 4× correspond to widening factors. Top-performing architectures in a column are bold; the best pretext task for each model is underlined.

Model	Rotation				Exemplar			RelPatchLoc		Jigsaw	
	4×	8×	12×	16×	4×	8×	12×	4×	8×	4×	8×
RevNet50	**47.3**	**50.4**	**53.1**	**53.7**	**42.4**	45.6	46.4	40.6	45.0	40.1	43.7
ResNet50 v2	43.8	47.5	47.2	47.6	**43.0**	45.7	46.6	42.2	46.7	38.4	41.3
ResNet50 v1	41.7	43.4	43.3	43.2	**42.8**	**46.9**	**47.7**	**46.8**	**50.5**	**42.2**	**45.4**
RevNet50 (-)	45.2	**51.0**	**52.8**	**53.7**	38.0	42.6	44.3	33.8	43.5	36.1	41.5
ResNet50 v2 (-)	38.6	44.5	47.3	48.2	33.7	36.7	38.2	38.6	43.4	32.5	34.4
VGG19-BN	16.8	14.6	16.6	22.7	26.4	28.3	29.0	28.5	29.4	19.8	21.1

Family		ImageNet		Places205	
		Prev.	Ours	Prev.	Ours
A	Rotation[11]	38.7	**55.4**	35.1	**48.0**
R	Exemplar[8]	31.5	46.0	-	42.7
R	Rel. Patch Loc.[8]	36.2	51.4	-	45.3
A	Jigsaw[34, 51]	34.7	44.6	35.5	42.2
V	CC+vgg-Jigsaw++[36]	37.3	-	37.5	-
A	Counting[35]	34.3	-	36.3	-
A	Split-Brain[51]	35.4	-	34.1	-
V	DeepClustering[3]	**41.0**	-	**39.8**	-
R	CPC[37]	48.7†	-	-	-
R	Supervised RevNet50	74.8	74.4	-	58.9
R	Supervised ResNet50 v2	76.0	75.8	-	61.6
V	Supervised VGG19	72.7	75.0	58.9	61.5

† *marks results reported in unpublished manuscripts.*

Table 2. Comparison of the published self-supervised models to our best models. The scores correspond to accuracy of linear logistic regression that is trained on top of representations provided by self-supervised models. Official validation splits of *ImageNet* and *Places205* are used for computing accuracies. The "Family" column shows which basic model architecture was used in the referenced literature: **A**lexNet, **V**GG-style, or **R**esidual.

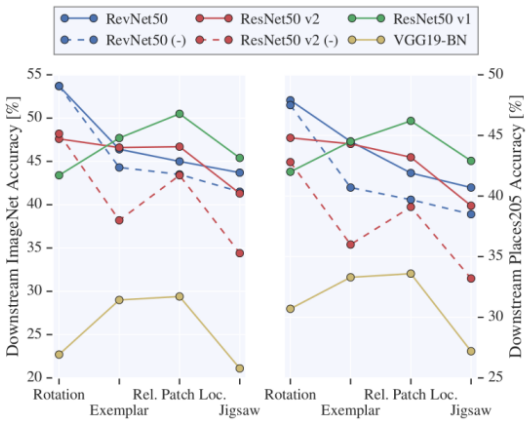

Figure 2. Different network architectures perform significantly differently across self-supervision tasks. This observation generalizes across datasets: *ImageNet* evaluation is shown on the left and *Places205* is shown on the right.

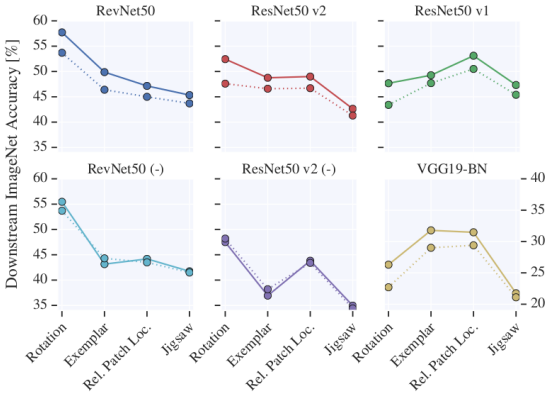

Figure 3. Comparing linear evaluation (·····) of the representations to non-linear (——) evaluation, *i.e.* training a multi-layer perceptron instead of a linear model. Linear evaluation is not limiting: conclusions drawn from it carry over to the non-linear evaluation.

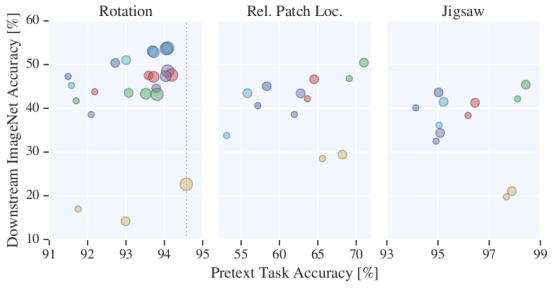

Figure 4. A look at how predictive pretext performance is to eventual downstream performance. Colors correspond to the architectures in Figure 3 and circle size to the widening factor k. Within an architecture, pretext performance is somewhat predictive, but it is not so across architectures. For instance, according to pretext accuracy, the widest *VGG* model is the best one for *Rotation*, but it performs poorly on the downstream task.

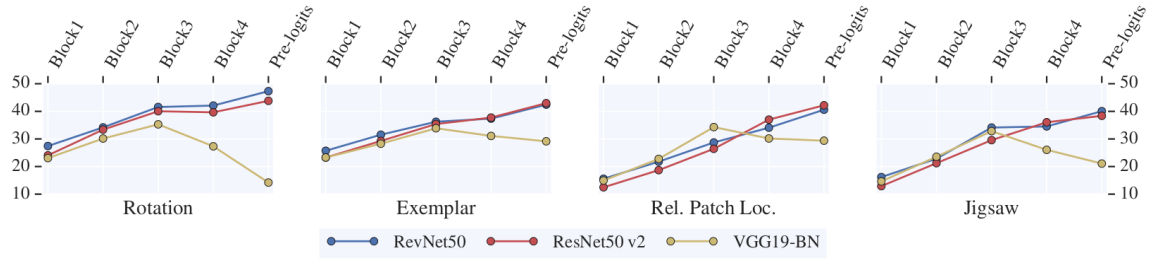

Figure 5. Evaluating the representation from various depths within the network. The vertical axis corresponds to downstream ImageNet performance in percent. For residual architectures, the *pre-logits* are always best.

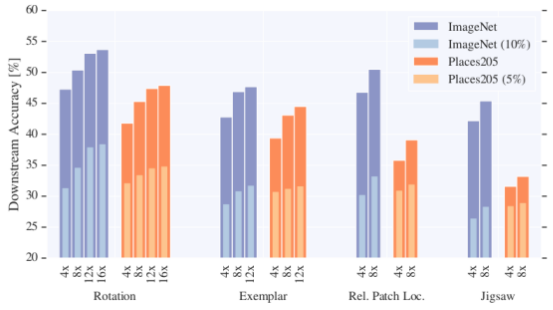

Figure 7. Performance of the best models evaluated using all data as well as a subset of the data. The trend is clear: increased widening factor increases performance across the board.

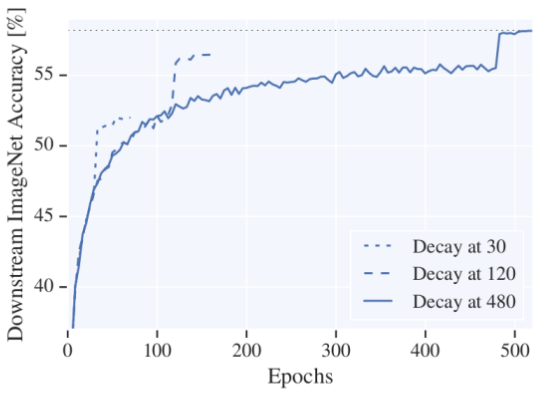

Figure 8. Downstream task accuracy curve of the linear evaluation model trained with SGD on representations from the *Rotation* task. The first learning rate decay starts after 30, 120 and 480 epochs. We observe that accuracy on the downstream task improves even after very large number of epochs.