Selfie: Self-supervised Pretraining for Image Embedding

(2019)

Trieu H. Trinh, Minh-Thang Luong, Quoc V. Le

Notes

Contributions

The authors introduce a new self supervised method called *Selfie* (SELF-supervised Image Embedding) that takes advantage of the progress made in language modeling based on attention mechanism, and generalizes BERT to continuous spaces, such as images. Similar to BERT applied to language, some patches of the image are masked out and the objective is to reconstruct the original image, and instead of having a regression target that very sensitive to small perturbations, they propose to use a classification network where the model classifies the right patch to fill in a target masked location.

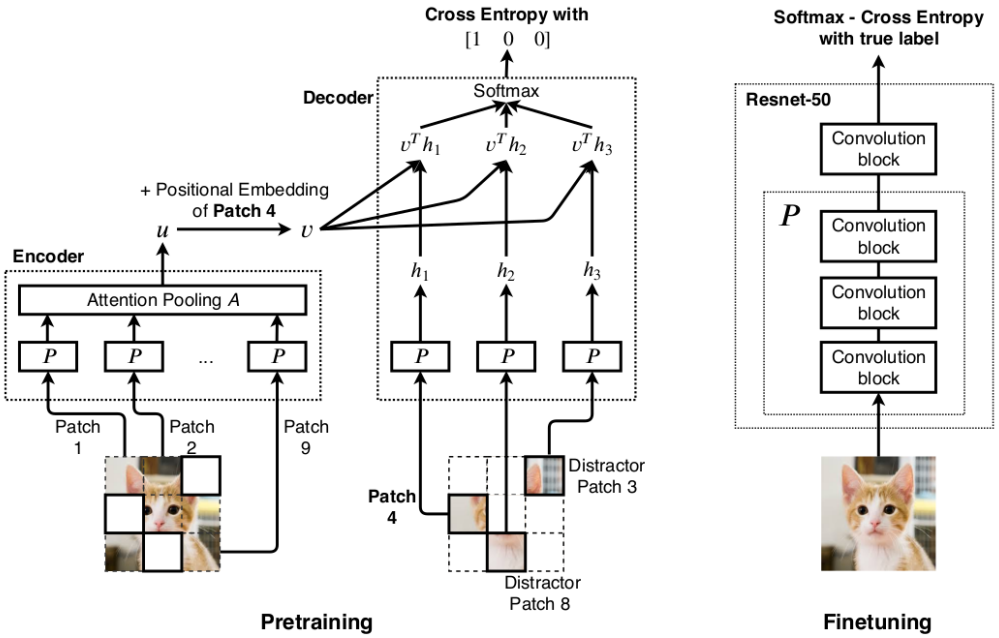

Figure 1: **An overview of** *Selfie*. *(Left) During pretraining*, our method makes use of an encoder-decoder architecture: the encoder takes in a set of square patches from the input image while the decoder takes in a different set. The *encoder* builds a single vector u that represents all of its input patches using a patch processing network P followed by an attention pooling network A. The *decoder* then takes u to predict its own input patches from their positions. Instead of predicting the actual pixel content, the decoder classifies the correct patch from negative examples (distractors) with a cross-entropy loss. In our implementation, we use the first three blocks of ResNet-50 (equivalent to ResNet-36) for P and Transformer layers (Vaswani et al., 2017) for A. Square patches are processed independently by P to produce a feature vector per patch. *(Right) During finetuning*, ResNet-50 is applied *on the full image*. Its first three blocks are initialized from the pretrained P, and the network is finetuned end-to-end.

Method

Self-supervised pretraining

In the pretraining stage, P, a patch processing network (a simple ResNet 50), will be applied to small patches in an image to produce one feature vector per patch for both the encoder and the decoder. In the encoder, the feature vectors are pooled together by an attention pooling network A to produce a single vector u. In the decoder, no pooling takes place; instead the feature vectors are sent directly to the computation loss to form an unsupervised classification task. The representations from the encoder and decoder networks are jointly trained during pretraining to predict what patch is being masked out at a particular location among other distracting patches.

Given N number of patches extracted from a given image (say 9), with a sample fraction $p > 0.5$, a larger subset (say 6 out of 9) of these patches of size $N \times p$ is passed through the encoder and then the attention pooling layer creating a single vector u. The unselected $(1-p) \times N$ patches (the remaining 3) are also passed through the encoder P producing vectors h_i, one of the unselected patches is selected as the target, and then the location embedding corresponding to the selected target patch is added to u, the rest of unselected patches (two) are considered distractor patches. The vectors h_i are used in a dot products with u and passed through a softmax to find the correct patch location, the model (P, softmax layer and attention pooling) are trained using a cross entropy loss. To give correct prediction, the encoder network learns to compress the information in the input image to a vector u such that when seeded by a location of a missing patch, it can recover that patch accurately.

Implementation details

Patch sampling & Processing For small images of size 32×32, the patch size is 8×8. For larger images 224×224 the patch size is 32. Each image is padded with 4 pixels and then randomly cropped to 32×32 / 224×224. The encoder, or the patch processing network used in a ResNet 50 with only the first three blocks, and to get the vectors an average pooling is applied to the outputs.

Attention Pooling Given the encoded patches in the form of vectors h_i, the attention pooling network aggregate all these vectors into a single vectors u, instead of using a max-pooling or average-pooling (both corresponding to a softmax with different temperature), we use transformer layers (follows the design in BERT: self-attention layer is followed with two fully connected layers, GeLU non linearity and dropout of 0.1). Using a learned vector u_0 that will contain the aggregated information of all the encoded vectors h_i in the output, we pass them (h_i and u_0) through a transformer and only keep the first output. which is the pooling results.

$$u, h_1^{\text{output}}, h_2^{\text{output}}, \ldots, h_n^{\text{output}} = \text{TransformerLayers } (u_o, h_1, h_2, \ldots, h_n)$$

Positional Embedding for small 32×32 images, the patches are of size 8×8, resulting in 16 patches, so a vectors of size 16 is learned for larger images, we have 7×7 number of patches, and instead of learning a vectors of size 16, only $7 + 7 = 14$ positional embeddings are learned to position the patch in the H and V directions.

Results

Table 1: Test accuracy (%) of ResNet-50 with and without pretraining across datasets and sizes.

		Labeled Data Percentage			
		5%	*8%*	*20%*	*100%*
CIFAR-10	Supervised	75.9 ± 0.7	79.3 ± 1.0	88.3 ± 0.3	95.5 ± 0.2
	Selfie Pretrained	75.9 ± 0.4	80.3 ± 0.3	89.1 ± 0.5	95.7 ± 0.1
	Δ	0.0	**+1.0**	**+0.8**	**+0.2**
		5%	*10%*	*20%*	*100%*
ImageNet 32×32	Supervised	13.1 ± 0.8	25.9 ± 0.5	32.7 ± 0.4	55.7 ± 0.6
	Selfie Pretrained	18.3 ± 0.1	30.2 ± 0.5	33.5 ± 0.2	56.4 ± 0.6
	Δ	**+5.2**	**+4.3**	**+0.8**	**+0.7**
ImageNet 224×224	Supervised	35.6 ± 0.7	59.6 ± 0.2	65.7 ± 0.2	76.9 ± 0.2
	Selfie Pretrained	46.7 ± 0.4	61.9 ± 0.2	67.1 ± 0.2	77.0 ± 0.1
	Δ	**+11.1**	**+2.3**	**+1.4**	**+0.1**

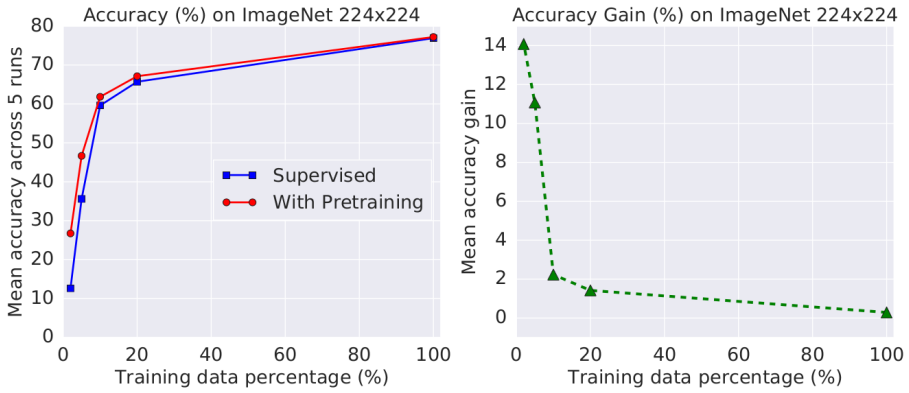

Figure 4: Pretraining with *Selfie* benefits the most when there is much more unlabeled data than labeled data. **Left:** Mean accuracy across five runs on ImageNet 224×224 for purely supervised model versus one with pretraining. **Right:** Mean accuracy gain from pretraining. The improvement quickly diminishes at the 10% mark when there is 10 times more data than the labeled set.

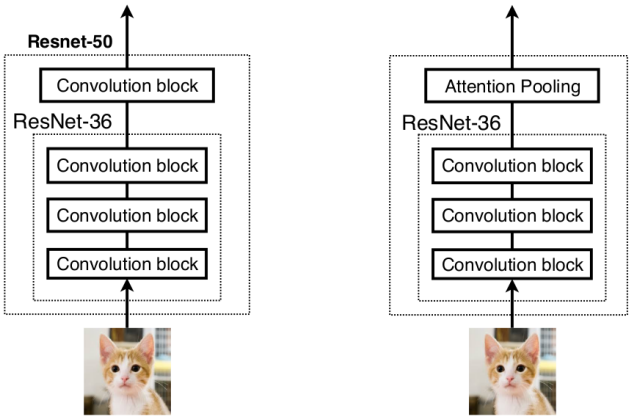

Figure 5: *(Left)* ResNet-50 architecture. *(Right)* ResNet-36 + attention pooling architecture.

Table 2: Accuracy (%) of ResNet-50 and ResNet-36 + attention pooling after finetuning from pretrained weights, found by *Selfie* on limited and full labeled sets. The gain (Δ) indicates how much improvement is made from using attention pooling in place of the last convolution block.

Method	ResNet-50	ResNet-36 + attention pooling	Δ
CIFAR-10 8%	80.3 ± 0.3	81.3 ± 0.1	**+1.0**
ImageNet 10%	61.8 ± 0.2	62.1 ± 0.2	**+0.3**
CIFAR-10 100%	95.7 ± 0.1	95.4 ± 0.2	-0.3
ImageNet 100%	77.0 ± 0.1	77.5 ± 0.1	**+0.5**