Acknowledgements

The authors are grateful to Carl Doersch, Raia Hadsell, and Evan Shelhamer for insightful discussions and feedback on the manuscript.

A. Appendix

A.1. Implementation: data augmentation

Self-supervised pretraining. Each image is randomly augmented twice, resulting in two images: $\boldsymbol{x}, \boldsymbol{x}'$. The augmentations are constructed as compositions of the following operations, each applied with a given probability:

1. random cropping: a random patch of the image is selected, whose area is uniformly sampled in $[0.08 \cdot \mathcal{A}, \mathcal{A}]$, where $\mathcal{A}$ is the area of the original image, and whose aspect ratio is logarithmically sampled in $[3/4, 4/3]$. The patch is then resized to 224 ×224 pixels using bicubic interpolation;
2. horizontal flipping;
3. color jittering: the brightness, contrast, saturation and hue are shifted by a uniformly distributed offset;
4. color dropping: the RGB image is replaced by its grey-scale values;
5. gaussian blurring with a 23×23 square kernel and a standard deviation uniformly sampled from $[0.1, 2.0]$;
6. solarization: a point-wise color transformation $x \mapsto x \cdot \mathbb{1}_{x<0.5} + (1-x) \cdot \mathbb{1}_{x \geq 0.5}$ with pixels x in $[0, 1]$.

The augmented images $\boldsymbol{x}, \boldsymbol{x}'$ result from augmentations sampled from distributions $\mathcal{T}$ and $\mathcal{T}'$ respectively. These distributions apply the primitives described above with different probabilities, and different magnitudes. The following table specifies these parameters for the SimCLR [9] and BYOL frameworks [21], which we adopt for DetCon$_S$ and DetCon$_B$ without modification.

	DetCon$_S$		DetCon$_B$	
Parameter	$\mathcal{T}$	$\mathcal{T}'$	$\mathcal{T}$	$\mathcal{T}'$
Random crop probability	1.0			
Flip probability	0.5			
Color jittering probability	0.8			
Color dropping probability	0.2			
Brightness adjustment max	0.8		0.4	
Contrast adjustment max	0.8		0.4	
Saturation adjustment max	0.8		0.2	
Hue adjustment max	0.2		0.1	
Gaussian blurring probability	1.0	0.0	1.0	0.1
Solarization probability	0.0	0.0	0.0	0.2

Transfer to COCO. When fine-tuning, image are randomly flipped and resized to a resolution of $u \cdot 1024$ pixels on the longest side, where u is uniformly sampled in $[0.8, 1.25]$, then cropped or padded to a 1024×1024 image. The aspect ratio is kept the same as the original image. During testing, images are resized to 1024 pixels on the longest side then padded to 1024×1024 pixels.

Transfer to PASCAL. During training, images are randomly flipped and scaled by a factor in $[0.5, 2.0]$. Training and testing are performed with 513×513-resolution images.

Transfer to Cityscapes. During training, images are randomly horizontally flipped and scaled by a factor in $[0.5, 2.0]$, with minimum step size 0.25 within that range. Training is performed on 769×769-resolution images and testing is performed on 1025×2049-resolution images.

Transfer to NYU-Depth v2. The original 640×480 frames are down-sampled by a factor of 2 and center-cropped to 304×228 pixels. For training, images are randomly flipped horizontally and color jittered with the same grayscale, brightness, saturation, and hue settings as [21].

A.2. Implementation: architecture

Our default feature extractor is a ResNet-50 [27]. In Section 4.1 we also investigate deeper architectures (ResNet-101, -152, and -200), and a wider model (ResNet-200 ×2) obtained by scaling all channel dimensions by a factor of 2.

As detailed in Section 3.1, this encoder yields a grid of hidden vectors which we pool within masks to obtain a set of vectors $\boldsymbol{h_m}$ representing each mask. These are then transformed by a projection head g (and optionally a prediction head q) before entering the contrastive loss.

DetCon$_S$. Following SimCLR, the projection head is a two-layer MLP whose hidden and output dimensions are 2048 and 128. The network uses the learned parameters θ for both views.

DetCon$_B$. Following BYOL, the projection head is a two-layer MLP whose hidden and output dimensions are 4096 and 256. The network uses the learned parameters θ for processing one view, and an exponential moving average of these parameters ξ for processing the second. Specifically, ξ is updated using $\xi \leftarrow \lambda \cdot \xi + (1-\lambda) \cdot \theta$, where the decay rate λ is annealed over the course of training from λ_0 to 1 using a cosine schedule [21]. λ_0 is set to 0.996 when training for 1000 epochs and 0.99 when training for 300 epochs. The projection of the first view is further transformed with a prediction head, whose architecture is identical to that of the projection head.

Computational cost. The forward pass through a ResNet-50 encoder requires roughly 4B FLOPS. Ignoring the cost of bias terms and point-wise nonlinearities, the projection head in DetCon$_S$ requires 4.4M FLOPS (i.e. 2048×2048 + 2048×128). Since this is calculated 16 times rather than once, it results in an overhead of 67M FLOPS compared to SimCLR. For DetCon$_B$ the combined cost of evaluating the projection and prediction heads results in an additional 173M FLOPS compared to BYOL. Finally, the cost of evaluating the contrastive loss is 134M FLOPS for DetCon$_S$ (i.e. $128 \times 4096 \times 16^2$) and 268M FLOPS for DetCon$_B$. In total DetCon$_S$ requires 201M additional FLOPS and DetCon$_B$ 441M which represent 5.3% and 11.6% of the cost of evaluating the backbone. This overhead is sufficiently small compared to the gain in training iterations required to reach a given transfer performance (e.g. a 500% gain for DetCon$_S$ over SimCLR, and a 333% for DetCon$_B$ over DetCon) for us not further distinguish between gains in computation and training time.

A.3. Implementation: optimization

Self-supervised pretraining. We train using the LARS optimizer [65] with a batch size of 4096 split across 128 Cloud TPU v3 workers. When training on ImageNet we again adopt the optimization details of SimCLR and BYOL for DetCon$_S$ and DetCon$_B$, scaling the learning rate linearly with the batch size and decaying it according to a cosine schedule. For DetCon$_S$ the base learning rate is 0.3 and the weight decay is 10^{-6}. DetCon$_B$ also uses these values when training for 300 epochs; when training for 1000 epochs they are 0.2 and $1.5 \cdot 10^{-6}$.

When pretraining on COCO, we replace the cosine learning rate schedule with a piecewise constant, which has been found to alleviate overfitting [25], dropping the learning rate by a factor of 10 at the 96th and 98th percentiles. For fair comparison we use the same schedules when applying SimCLR to the COCO dataset, which we also find to perform better than the more aggressive cosine schedule.

Transfer to COCO. We fine-tune with stochastic gradient descent, increasing the learning rate linearly for the first 500 iterations and dropping twice by a factor of 10, after $\frac{2}{3}$ and $\frac{8}{9}$ of the total training time, following [62]. We use a base learning rate of 0.3 for ResNet-50 models and 0.2 for larger ones, a momentum of 0.9, a weight decay of $4 \cdot 10^{-5}$, and a batch size of 64 images split across 16 workers.

Transfer to PASCAL. We fine-tune for 45 epochs with stochastic gradient descent, with a batch size of 16 and weight decay of 10^{-4}. The learning rate is 0.02 and dropped by a factor of 10 at the 70th and 90th percentiles.

Transfer to Cityscapes. We fine-tune for 160 epochs with stochastic gradient descent and a Nesterov momentum of 0.9, using a batch size of 2 and weight decay of 10^{-4}. The initial learning rate is 0.005 and dropped by a factor of 10 at the 70th and 90th percentiles.

Transfer to NYU-Depth v2. We fine-tune for 7500 steps with a batch size of 256, weight decay of $5 \cdot 10^{-4}$, and a learning rate of 0.16 scaled linearly with the batch size [21].

	Fine-tune 1×		Fine-tune 2×	
method	AP^{bb}	AP^{mk}	AP^{bb}	AP^{mk}
Supervised	42.0	37.3	43.4	38.4
SimCLR [9]	42.0	37.9	43.8	39.3
InfoMin [55]	42.9	38.6	44.5	39.9
BYOL [21]	43.7	38.8	44.3	39.4
DetCon$_B$	**45.2**	**40.0**	**45.7**	**40.4**

(a) **ResNet-101** feature extractor

	Fine-tune 1×		Fine-tune 2×	
method	AP^{bb}	AP^{mk}	AP^{bb}	AP^{mk}
Supervised	43.4	38.5	43.4	38.5
SimCLR [9]	43.6	39.1	44.9	40.0
BYOL [21]	44.9	40.0	45.7	40.6
DetCon$_B$	**46.0**	**40.6**	**46.4**	**40.7**

(b) **ResNet-152** feature extractor

	Fine-tune 1×		Fine-tune 2×	
method	AP^{bb}	AP^{mk}	AP^{bb}	AP^{mk}
Supervised	43.2	38.3	43.5	38.5
SimCLR [9]	44.3	39.6	45.3	40.3
BYOL [21]	45.6	40.5	45.9	40.5
DetCon$_B$	**47.1**	**41.3**	**47.2**	**41.5**

(c) **ResNet-200** feature extractor

Table A.1. **Comparison to prior art:** all methods are pretrained on ImageNet then fined-tuned on COCO for 12 epochs (1× schedule) or 24 epochs (2× schedule).

A.4. Results: larger models

In Table 2 we compare to prior works on self-supervised learning which transfer to COCO. Here we provide additional comparisons which use larger models (ResNet-101, -152, and -200). We find DetCon to continue to outperform prior work in this higher capacity regime (Table A.1).