Appendix

A. Dataset Labels

For experiments on the CelebA-HQ dataset, we transform labels manually from 19 classes to 8 or 10 as required for comparison to the baselines. The 19 classes in the original dataset are:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Class	'background'	'skin'	'nose'	'eye_g'	'l_eye'	'r_eye'	'l_brow'	'r_brow'	'l_ear'	'r_ear'	'mouth'	'u_lip'	'l_lip'	'hair'	'hat'	'ear_r'	'neck_l'	'neck'	'cloth'

For experiments with 10 classes and 8 classes we consider the following:

Index	0	1	2	3	4	5	6	7	8	9
Class	'background'	'skin'	'nose'	'eye'	'brow'	'ear'	'mouth'	'hair'	'neck'	'cloth'

Index	0	1	2	3	4	5	6	7
Class	'background'	'skin'	'nose'	'eye'	'brow'	'ear'	'mouth'	'hair'

Note that in the case of transformation from 19 classes to 10, we assign classes 14,15,16 as the 'background'. For the case of transformation from 10 classes to 8, we assign classes 8,9 as the 'background'

B. Network Architectures

For our MLP model we use a 2 layer fully connected network with hidden dimensions 1024 followed by 256. In the case of our modified UNet based model, we first downsample the hypercolumns and then upsample them to the input resolution. For faces and cars, which we process at 512×512 we use Model-A and for cats which we process at 256×256 we use Model-B. The architectures are as given below:

Model-A

Name	Input Resolution	Layer
Conv1	H×W×3840	ConvBNReLU
MP1	H×W×1024	MaxPool
Conv2	H/2×W/2×1024	ConvBNReLU
MP2	H/2×W/2×256	MaxPool
Conv3	H/4×W/4×256	ConvBNReLU
MP3	H/4×W/4×256	MaxPool
Conv4	H/8×W/8×256	ConvBNReLU
MP4	H/8×W/8×256	MaxPool
Conv5	H/16×W/16×256	ConvBNReLU
MP5	H/16×W/16×512	MaxPool
Conv6	H/32×W/32×512	ConvBNReLU
Up1	H/32×W/32×512	Upsample Concat(Conv5)
Conv7	H/16×W/16×1024	ConvBNReLU
Up2	H/16×W/16×256	Upsample Concat(Conv4)
Conv8	H/8×W/8×512	ConvBNReLU
Up3	H/8×W/8×256	Upsample Concat(Conv3)
Conv9	H/4×W/4×512	ConvBNReLU
Up4	H/4×W/4×128	Upsample Concat(Conv2)
Conv10	H/2×W/2×384	ConvBNReLU
Up5	H/2×W/2×256	Upsample Concat(Conv1)
Conv11	H×W×1280	ConvBNReLU
FC	H×W×256	Linear
	H×W×Classes	-

Model-B

Name	Input Resolution	Layer
Conv1	H×W×3840	ConvBNReLU
MP1	H×W×1024	MaxPool
Conv2	H/2×W/2×1024	ConvBNReLU
MP2	H/2×W/2×256	MaxPool
Conv3	H/4×W/4×256	ConvBNReLU
MP3	H/4×W/4×256	MaxPool
Conv4	H/8×W/8×256	ConvBNReLU
MP4	H/8×W/8×256	MaxPool
Conv5	H/16×W/16×256	ConvBNReLU
Up1	H/16×W/16×512	Upsample Concat(Conv4)
Conv6	H/8×W/8×768	ConvBNReLU
Up2	H/8×W/8×256	Upsample Concat(Conv3)
Conv7	H/4×W/4×512	ConvBNReLU
Up3	H/4×W/4×128	Upsample Concat(Conv2)
Conv8	H/2×W/2×384	ConvBNReLU
Up4	H/2×W/2×256	Upsample Concat(Conv1)
Conv9	H×W×1280	ConvBNReLU
FC	H×W×256	Linear
	H×W×Classes	-

C. Re-implementation details

DatasetGAN : We present scores both reported by DatasetGAN in their paper and after using their updated code, where the upsampling technique for hypercolumns has been changed from 'nearest neighbour' to 'bilinear'. The mIoU of the Car category sees a large hike due to this. For the Cat category we filter out 2 images from the Cat16 training set with largely faulty annotations. We train DatasetGAN's model on the remaining images and report scores.

ReGAN : Since ReGAN have not released their code, we re-implemented their method as described in the paper. We use StyleGAN2 trained on FFHQ dataset. We implement their MLP model with 2 hidden layers of 2000 and 200 dimensions each. For distillation we use UNet architecture as specified in their paper. We compare with their 10-shot setting. We use the same 10/190 train/test split as communicated by the authors for both their method and ours.

D. Hyperparameters

In this section we list additional training details which we did not specify in the main text.

MoCoV2 Training : We train MoCoV2 with MLP head with an initial lr of 0.03 and weight decay of 0.0001 for 800 epochs with cosine annealing. We use SGD with momentum of 0.9 and temperature coefficient for the loss as 0.2. For the transforms we use random resized crop, flip, gaussian blur, grayscale and color jitter.

SimSiam Training : We use the official implementation[3] of SimSiam. We train SimSiam with an initial lr of 0.05 and weight decay of 0.0001 for 400 epochs. We notice that training for more epochs (*e.g.* 800) leads to slightly worse performance in the downstream tasks. We use SGD with momentum of 0.9 an same transforms as MoCoV2.

Hypercolumn extraction: We use outputs of ResNet50 blocks from `conv_2x` to `conv_5x`. We use bilinear upsampling to resize to the resolution of input image before concatenation. The number of channels of our concatenated tensor is 3840.

Projector Training : We train for 800 epochs with initial learning rate of 0.001 for the MLP model, while for the UNet based methods we train for 200 epochs with initial lr of 0.0005. For both methods we use weight decay of 0.0005, Adam optimizer and batch size of 2. We also use random resized crop, flip and color jitter for both the methods.

Distillation Training : We use DeepLabV3 with ResNet101 for all experiments. We use initial lr of 0.001 and batch size of 8 with Adam optimizer. We observe that our models converge by the 2nd epoch and we use that model to report mIoUs.

E. Effect of Resolution

Input Resolution for training segmentor	Input Resolution for training MoCoV2		
	96	256	512
96	0.4047	0.4081	0.3984
256	0.4687	0.5022	0.4876
512	0.4389	0.4958	0.5023

We explore the effect of resolution for each training on performance. Here we refer to the trained MoCoV2 + Hypercolumn extraction + Projector as the segmentor. While training the segmentor with input image of resolution R_i we upsample the hypercolumns to and calculate loss on the same resolution R_i. While testing, we upsample the output of network to 512 $\times$ 512 for all experiments to obtain consistent scores. All MoCoV2 models have been trained on the FFHQ dataset, while the segmentor models have been trained on Face34. It can be seen that in this case there is almost no improvement in performance for MoCoV2 trained on 512 $\times$ 512/segmentor trained on 512 $\times$ 512, wrt MoCoV2 trained on 256 $\times$ 256/segmentor trained on 256 $\times$ 256. The performance of MoCo models trained on 96 $\times$ 96 is consistently worse.

[3] https://github.com/facebookresearch/simsiam

F. Some more Qualitative Results

We present more qualitative results of our best method on the Face34 and Car20 testing datasets below. In the case of faces, our model is able to capture all classes well except very fine segments such as some wrinkles. In the case of cars, our model classifies the window of the last example correctly, though it is wrongly annotated.