A. Network implementation details

All the models in this paper are based on the modules: Encoder (Enc), Decoder (Dec), latent canonicalizers (Can) and a linear classifier (Cls). Specifically the classifier baseline is a composition of Enc+Cls; the AE is a composition Enc+Dec. For AE+Cls we compose a linear classifier and supervised for both reconstruction and classification loss. For our proposed network, we also have 6 latent canonicalizers, used both individually and in pairs.

Loss weights: We performed a hyper-parameter sweep over the classification loss weight ($\in 10, 20, 50$) and found 50 to be the strongest baseline. We also used 50 as the classification loss for our network.

Network architecture: The Encoder is comprised of 3 modules, each includes 3 layers of 3x3 2D CNN with 64 latent dimensions followed by a batch norm and leaky-ReLU (parameter=0.1). A 2x2 max-pool and 50% dropout is done after each module and finally a max-pool to get a 64 dimensional latent vector, z.

The Decoder is comprised of 3 layers of 3x3 transposed convolutions with dimensionalities: (64, 64, 32) each followed by a batch normalization and ReLU. Finally, a last transposed convolution from 64 to 3 dimensions and a ReLU.

Training hyperparameters: The networks were implemented using the PyTorch framework (35). Pre-training on simulated data and refinement on real data were done using the Adam optimizer (19) with default parameters, with learning rates of $1e-3$ and $1e-4$ respectiverly. Pretraining was done for 400 epochs and refinement was done for roughly 50 epochs. The implementation was done

The classifier and the latent canonicalizers are simply single linear layers with 64 dimensions (corresponding to the dimensionality of the latent, $\boldsymbol{z}$).

B. Simulator Details

To support our training requirements, we built a generator for SVHN-like images. Importantly, we do not aim to mimic the true SVHN dataset statistics. We make only very general simplistic assumptions such as a 32x32 image size, that images contain a centralized digit, etc. We build the simulator based on the *imgaug* library (15) which allows to easy control of the different factors of variation. The specific factors used to generate the images are given in the table together with their set of values. The first row of Table A1 shows the factors used for canonicalization. For each of these (except for noise, blur and crop which are canonicalized together to serve as the bypass), we generate a copy of the image with that factor set to its canonical value. The canonical values are reported in row 3 of the table. They are chosen arbitrarily but are the same for the entire train set. Following the 73257 real SVHN train set samples (not used in this work) we generated a synthetic set of size 75000 total images. Some examples of generated images can be seen in Figure A1. The set together with the simulator code will be made publicly available.

Figure A1: Examples of simulated SVHN images.

Supervised factors	**Rotation**	**Font color**	**Background color**	**Font scale**	**font type**	**shear**	**Noise , Blur , Crop**
Range	$[-15, 15]$	$\{0, .., 255\}^3$	$\{0, ..., 255\}^3$	$[1.0, 1.6]$	6 types	$[-5, 5]$	$\mathcal{N}(0, 0.04)$, $[0, 1.5]$, $[0, 0.1]$
Canonical value	0	$(200, 200, 200)$	$(100, 100, 100)$	1.0	type 1	0	$(0, 0, 0)$
Unupervised factors	**Cval**	**Scale**	**Number of digits**	**Translation**			
Range	$0, ..255$	$[0.95, 1.2]$	$1, ..., 4$	$\pm Poiss(1)$			

Table A1: Simulator factors and their range of values.

Domain gap between synthetic and real images: Having created quite a generic set of digit images with naive assumptions, one may ask whether it is at all useful for the target domain. By taking intermediate checkpoints during training and measuring the classification error without fine-tuning on the real test set, we see in Figure A2 that indeed there is good correlation the train set error and the test set accuracy.

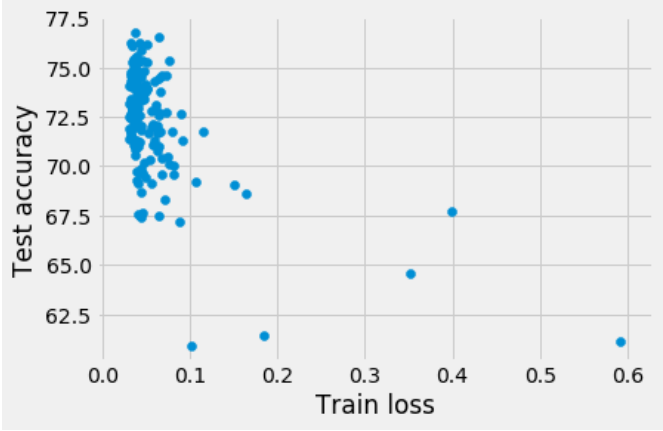

Figure A2: Train error on synthetic images vs. Test accuracy on real SVHN images.

Sampling the space of compositions: Scene complexity grows exponentially with the number of factors. One goal of this work is to show we can get good performance even when sampling a fraction of this space. To get a rough estimate of that percentage we can bin even just the supervised factors to [30, 64, 64, 6, 6, 10] (following the order of table A1) discrete values. This gives a space of $\approx 44M$ possibilities. Noticing the coarse binning of color-space and the fact we haven't included the unsupervised factors, we conclude that our 75000 samples set size is at least 3 order of magnitude smaller than the number of possible factor combinations.

C. Analysis of representations – additional material

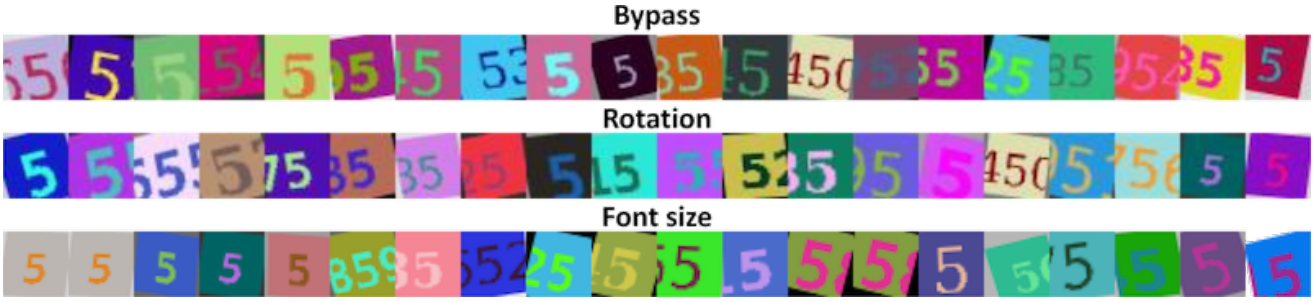

Figure A3: Comparison between the representation learned by the canonicalizers and the bypassed representation. Here we show an extended version of Figure 5 also visualizing the bypassed latent code (top row). The first principal component of $\boldsymbol{z}$ does not show any obvious grouping or pattern.