Learning ABCs: Approximate Bijective Correspondence for isolating factors of variation with weak supervision
Supplemental Material

Contents

S1. Mutual information calculation details and supporting measurements

A sample of 256 Shapes3D image representations, found by ABC, are shown in Fig. S1a,b by their first two principal components. The information about certain generative factors is qualitatively apparent, by the organization of the points by color. Below we explain in detail how the information content was quantitatively assessed for the results of the main text.

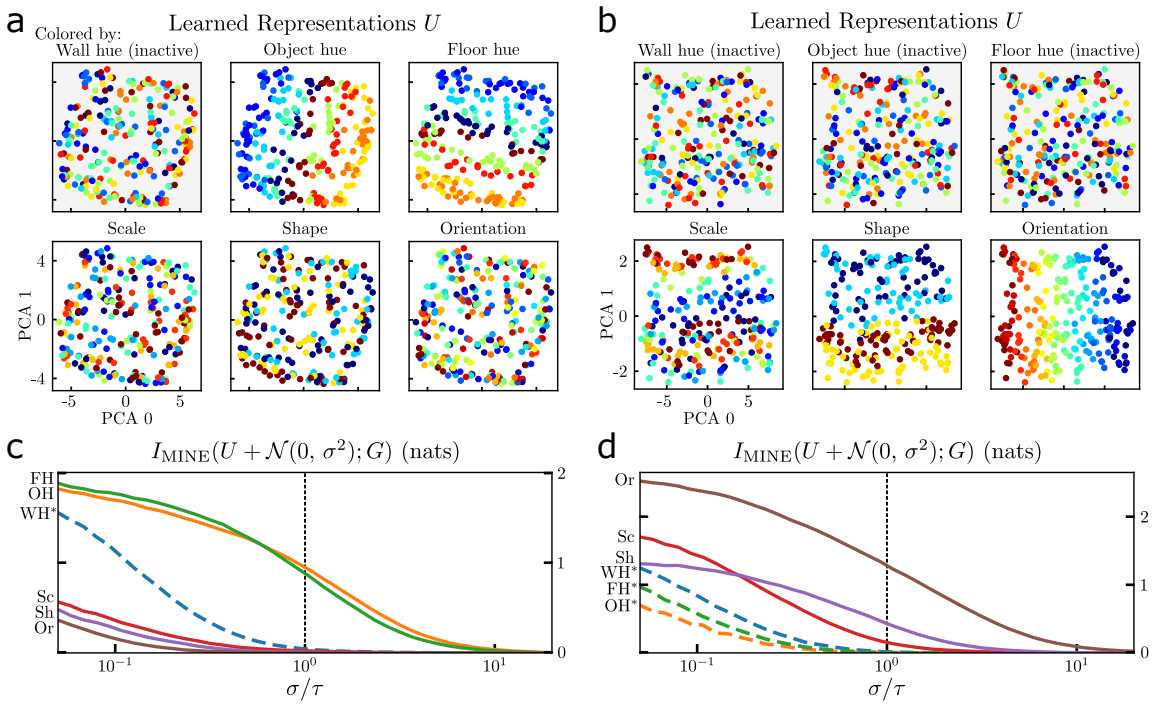
Calculation of mutual information. To estimate the mutual information $I(U;G)$ for the Shapes3D experiments using MINE [1], we train a statistics network T. We use a simple fully connected network whose input is the concatenation of the 64-dimensional embedding U and the 1-dimensional value for the particular generative factor G. It contains three layers of 128 units each with ReLU activations, with a final one-dimensional (scalar) output. The loss is the negated neural information measure of [1],

$$\mathcal{L} = \log(\mathbb{E}_{u\sim P(U),g\sim P(G)}[\exp(T(u,g))]) - \mathbb{E}_{u,g\sim P(U,G)}[T(u,g)] \tag{1}$$

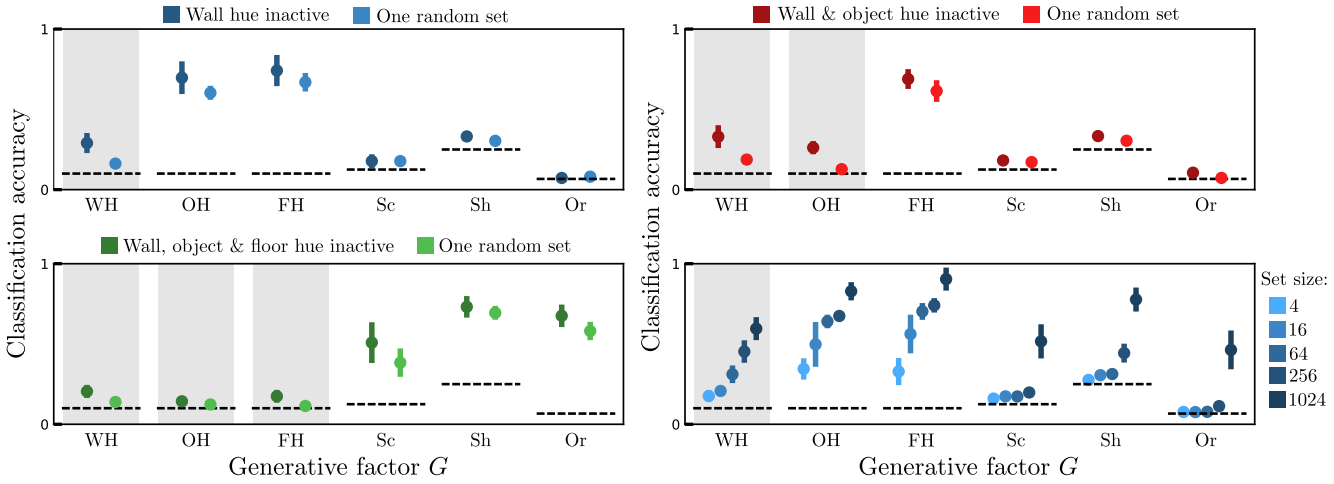
At a high level, the network exploits the difference between the joint distribution $P(U,G)$, where the embedding is properly matched with its correct generative factor, and the product of marginals $P(U)P(G)$, which is simulated by shuffling the labels for the first term in the loss. This difference between the joint and the marginals is the mutual information of the two variables. We train with a learning rate of 3×10^{-4} and a batch size of 256 for 20,000 steps, which we found to be sufficient for convergence. The estimate of the mutual information we report is the average value of the neural information measure over 256,000 samples from the dataset. A new statistics network is trained for each of the six generative factors.

To handle the determinism of the embedding network, we add Gaussian distributed noise $\eta\sim\mathcal{N}(0,\sigma^2)$ directly to the embeddings. We show sweeps over the noise scale in Figure S1c,d, where we repeat the calculation for 40 logarithmically spaced values of σ to show the effect of this added noise on the mutual information values. Predictably, the mutual information with respect to all factors is removed with increasing noise. We probe the information content relevant to the InfoNCE loss when the noise and the temperature parameter τ are equal, at which point all information about inactive factors is negligible.

Mutual information versus classification accuracy. To corroborate the Shapes3D mutual information measurements of Section 4.1, we use the common approach of training a simple classifier which takes the learned representations as input and tries to predict the generative factors (Figure S2). We train a different classifier for each generative factor, and use an architecture of 3 fully connected layers with 32 units each, ReLU activation. As with the measurements of mutual information,



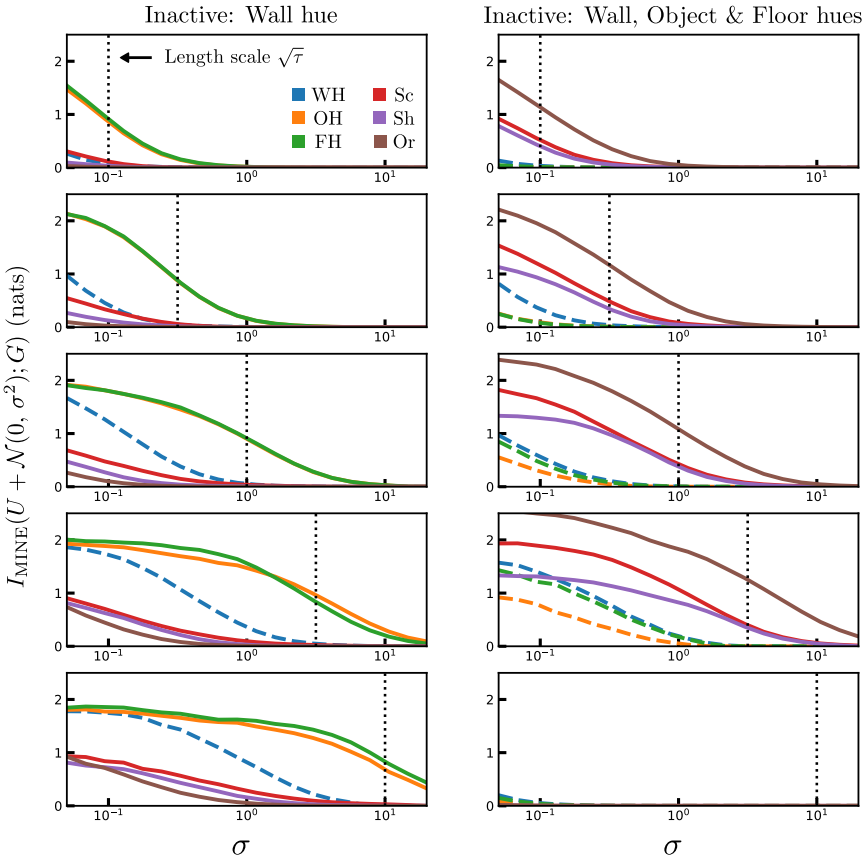
Supplemental Material, Figure S1. ***Information content of ABC-learned representations shows active factor isolation, Shapes3D. (a)*** *Trained with wall hue as the only inactive factor, information about object and floor hue is visually apparent in the first two principal components (> 0.98 of total variance) of the $\mathbb{R}^{64}$ embeddings. Each scatter plot displays the same 256 embeddings, colored according to each generative factor.* ***(b)*** *With all hue factors inactive, the representations become informative about the geometric factors.* ***(c,d)*** *For the networks in* ***(a,b)****, respectively, we estimate the mutual information $I(U; G)$ between the representations and each of the generative factors using MINE [1]. We add Gaussian noise to the representations to probe information content over different length scales in representation space. When σ equals the length scale of the loss (vertical dotted line), there is no information about inactive factors (dashed).*



Supplemental Material, Figure S2. ***Corroborating I_{MINE} with classification task.*** *As a proxy for the mutual information, we use the test set classification accuracy of networks trained to predict the six generative factors, one network per factor. As in Figure 3 of the main text, the shaded columns indicate which of the generative factors were inactive while training ABC. Gaussian-distributed random noise with spread σ corresponding to the length scale set by τ in the ABC loss was added to the embeddings to remove information on length scales less than the characteristic length scale of the ABC loss. The dashed lines show the classification accuracy that would result from random guessing.*

there is the issue of evaluating a deterministic network which in general preserves all information [2]. By adding Gaussian noise with magnitude $\sigma = \sqrt{\tau}$, the classification task reproduces the qualitative behavior of Figure 3. Namely, when one or two hue factors are inactive, information about the remaining hue factor(s) is enhanced and information about the inactive factor(s) is suppressed. When all three hue factors are inactive, then and only then is information about the three geometric factors enhanced. There is no substantial difference in the semi-supervised setting, where one set of each mini-batch has no inactive factors.

S2. The role of length scales in isolating factors of variation



Supplemental Material, Figure S3. ***Temperature sets the length scale of the cutoff between active and inactive factors.*** *We train with negative squared Euclidean distance between embeddings as the similarity measure, which makes $\sqrt{\tau}$ a natural length scale for embedding space. By varying the temperature used during training (varying vertically across the five rows), we mark the length scale $\sqrt{\tau}$ with a dotted vertical line in each subplot. Predictably, the magnitude of the noise σ at which information about inactive factors is removed scales with $\sqrt{\tau}$. Had negative Euclidean distance been used instead, we would expect the scaling to follow τ. The bottom right subplot shows one of the limits of varying the temperature of the ABC loss: when it is too large compared to the spread of the initialized embeddings, training is often unsuccessful.*

The ABC loss operates over a characteristic scale in embedding space, set by the temperature parameter τ which plays a role in both the soft nearest neighbor calculation and the InfoNCE loss. When using a similarity measure derived from Euclidean distance, this characteristic scale may be interpreted as a length scale. Two embeddings which are separated by less than this length scale effectively have a separation of zero in the eyes of the loss, and there is no incentive to further collapse them. To be specific, when using negative L2 (Euclidean) distance as the similarity metric, the temperature τ is the characteristic length scale. When using L2 squared distance, as in the MNIST and Shapes3D experiments, the square root of the temperature is the characteristic length scale. With cosine similarity, as in the pose estimation experiments of Section 4.3 of the main text, temperature sets a characteristic angular difference between embeddings.

For downstream tasks, including lookup using the embeddings, this length scale is generally irrelevant. However, measuring

the mutual information requires the addition of noise with a particular scale, and the freedom in choosing this parameter begs the question of a relevant scale in embedding space. As a fortunate consequence, it allows a precise definition of the factor isolation that results from ABC. We show in Figure S3 several Shapes3D experiments where the temperature τ during training took different values. The mutual information is measured as in Figure S1c,d with a sweep over the magnitude of the added noise.

The vertical dashed line in each run shows the characteristic length scale, $\sqrt{\tau}$, and it is clear to see information about the inactive factor(s) (indicated by dashed lines) decaying to zero below the length scale. The predicted behavior, of object and floor hue being isolated when wall hue is inactive, and of the geometric factors being isolated when all three hue factors are inactive, happens in nearly all the runs. The length scales of everything, as measured by the magnitude σ of the noise where the information decays, expand with increased temperature.

There is a limit to this behavior, however, which is shown in the bottom right subplot. When the temperature is too large compared to the initial separations of the embeddings, there is too little gradient information for even the Adam optimizer to leverage, and training is unsuccessful.

Summary. ABC's isolation of factors has a precise meaning in representation space: Information about inactive factors is confined to scales less than the characteristic scale set by the temperature during training, and the isolated active factors inform the structure of embedding space over larger scales. We demonstrate this by removing information over different scales in representation space through additive noise and mutual information measurements.

S3. Synthetic-to-real pose transfer: extended results

We include in-domain pose estimation results for the car category in Table S1. While no baseline method for the pose transfer task was particularly successful (Table 1 of the main text), all are performant when tested in the synthetic domain. The difficulty arises from the large domain gap between synthetic and real images, and not from the ability to extract pose information from the images of the training set.

	Med (°) ↓	Acc. @30° ↑
CCVAE [6]	12.7	0.67
ML-VAE [3]	9.3	0.84
LORD [4]	9.9	0.69
ABC	5.8	0.83

Table S1. ***Pose estimation in synthetic domain: cars.*** *All baselines perform well when testing on unseen instances from the synthetic domain, highlighting the difficulty of surmounting the domain gap for the pose transfer results of Table 1 of the main text.*

S4. Ablative studies on the pose estimation tasks

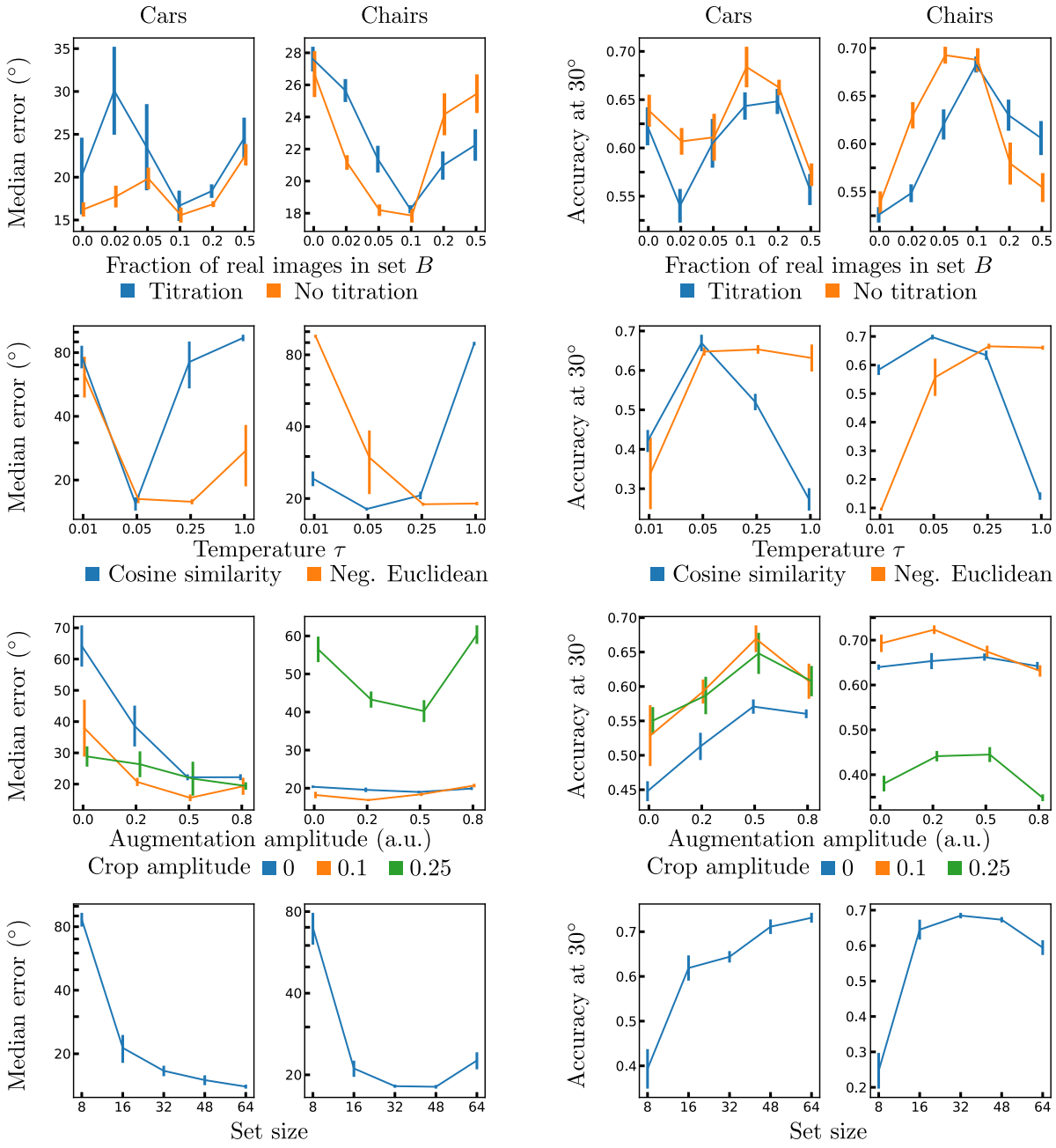
In Figures S4 and S5 we show ablative studies on the pose estimation experiments of Section 4.3 of the main text, for training with the ABC loss and no pose annotations (Table 1) and the experiment where the ABC loss combined with the spherical regression method of [7], utilizing pose annotations on the synthetic images (Table 2).

On both tasks, there is an optimal proportion of real images, though it is much lower for regression. Gradual titration of real images into the unconstrained set $\mathcal{B}$ was neutral or negative for the lookup task (Figure S4, top row) and generally positive for the regression task (Figure S5, top row). Cosine similarity outperforms negative Euclidean distance, and we show the dependence on temperature τ in the second row of Figure S4.

The car and chair categories present different challenges for pose estimation – e.g. an approximate front-back symmetry for cars, greater class diversity for chairs, outdoor versus indoor settings for cars versus chairs, etc. Several of the ablated factors cause differing effects on the performance for the two categories.

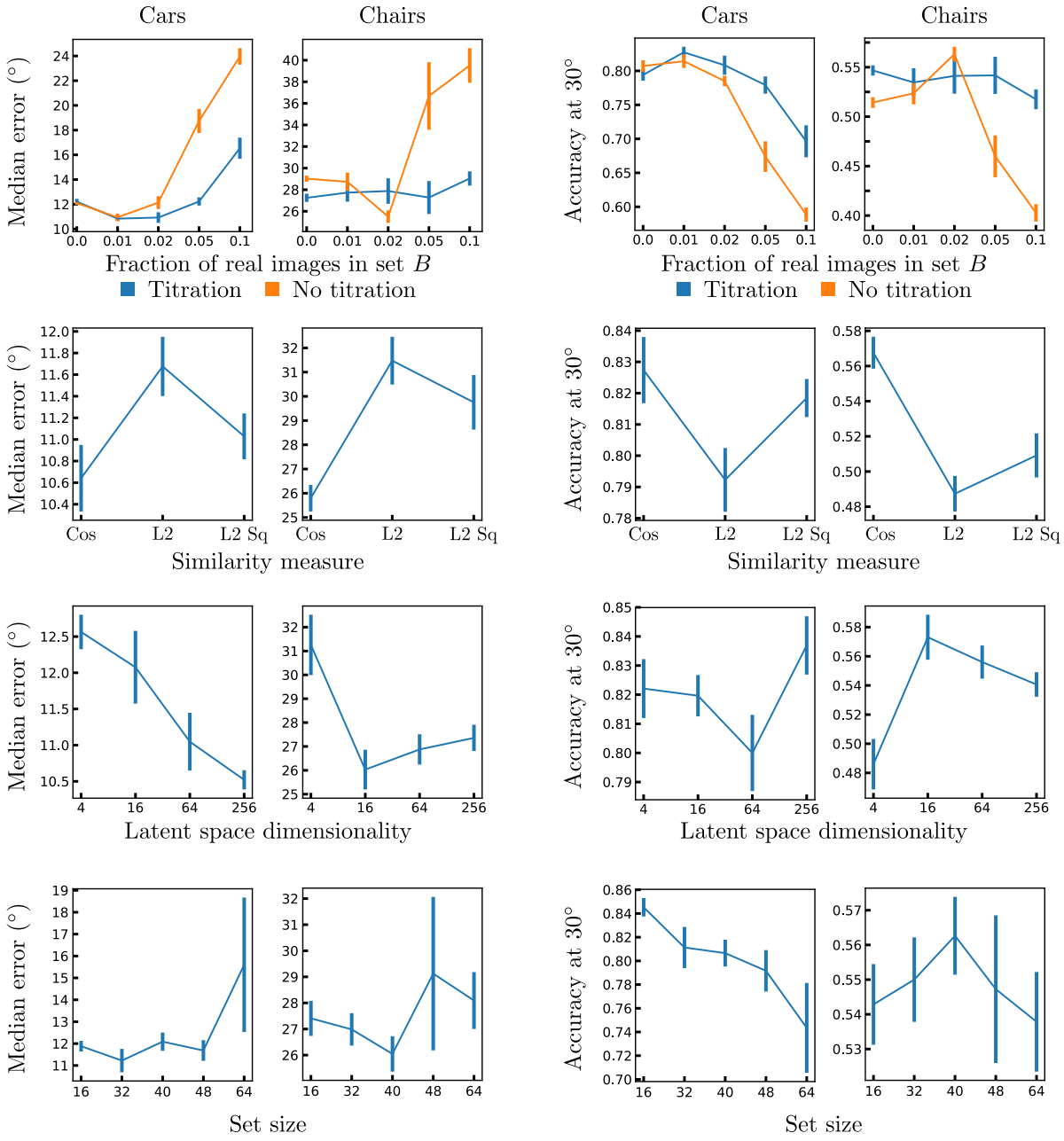
For instance, there is an apparent difference between the two categories in the dependence on the augmentation scheme, shown in the third row of Figure S4. Randomly translating the bounding box by 0.1 of its height and width helps both categories, but more than that and the chair performance greatly suffers.

Another difference between the categories is seen in the final row of Figure S4, where increasing the set size during training only helps pose estimation on cars. For the largest set size, however, chair pose estimation begins to suffer. We presume the pressure to isolate more active factors of variation from increased set size can actually be harmful to the pose estimation task if



Supplemental Material, Figure S4. ***Ablative studies on Pascal3D+ pose lookup with ABC embeddings.*** *Error bars are the standard error of the mean over 8 random seeds for each configuration. We show results on the Pascal3D+ test split for the car and chair categories. For each row, the training configuration is the same as described in Section S8 with only the listed aspect of training being changed. In the first row, no titration means to the fraction of real images in set B are present from the beginning of training. The augmentation amplitude in the third row controls the coloring changes discussed in Section S5. The crop amplitude is another form of augmentation, though we separate it for clarity. It controls the random translation of the bounding box, as a fraction of the dimensions of the bounding box.*

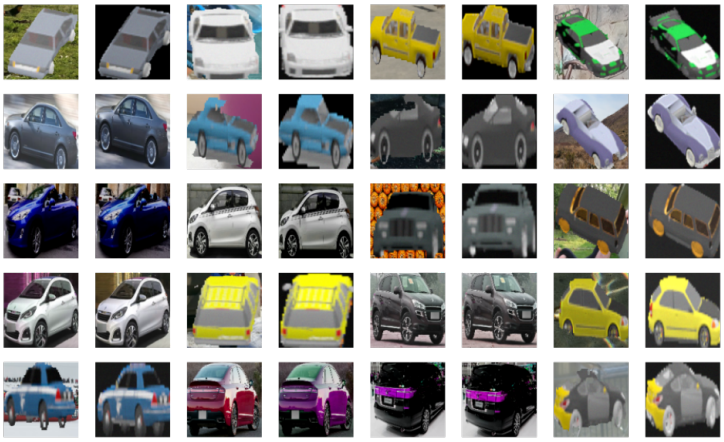
unrelated factors confound the pose estimation during lookup. Set size similarly shows mixed effects for the regression task, shown in the final row of Figure S5.



Supplemental Material, Figure S5. ***Ablative studies on Pascal3D+ with spherical regression + ABC network.*** *Error bars are the standard error of the mean over 10 random seeds for each configuration, with less than 1% of the runs discarded for lack of convergence. We show results on the Pascal3D+ test split for the car and chair categories. For each row, the training configuration is the same as described in Appendix S8 with only the listed aspect of training being changed. In the first row, no titration means to the fraction of real images in set B are present from the beginning of training. The three similarity measures in the second row are cosine similarity, L2 (Euclidean) distance, and squared L2 distance.*

S5. Augmentations used for pose estimation

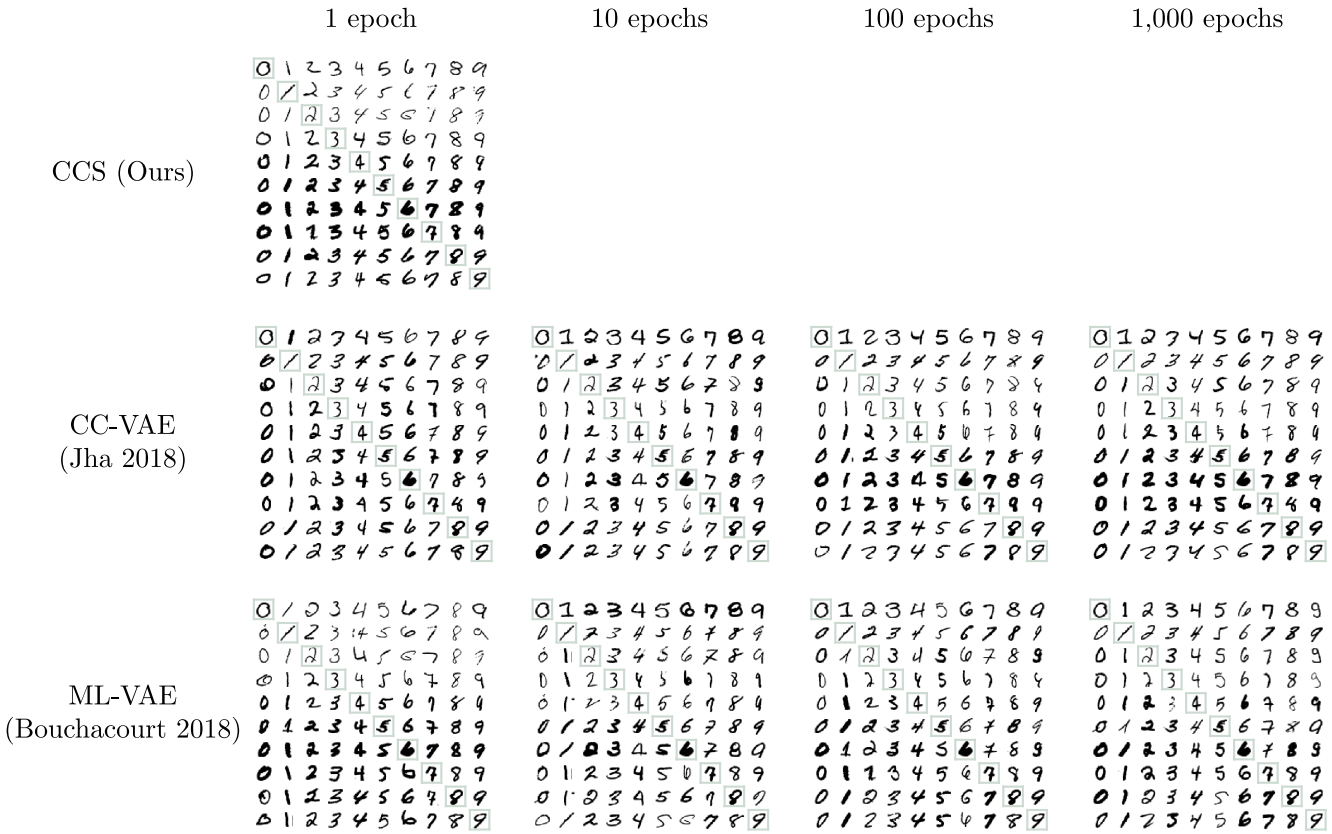
For each real and synthetic image in the pose estimation tasks of Section 4.3 of the main text, we augment twice and train with the double augmentation version of the ABC loss (described in Section 3.1.1), in order to suppress additional nuisance factors from the learned representations. We show in Figure S6 sample augmentation of real and synthetic car images, which include random translations of the bounding box, brightness adjustment, the addition of salt and pepper noise to each pixel, the



Supplemental Material, Figure S6. ***Augmentations used in the pose estimation experiments.*** *We show sample augmentations applied to both real and synthetic cars. These include adjusting brightness and hue, adding normally distributed noise to each pixel, random translations of the crop (bounding box), and replacing the background of synthetic images with random crops from real images.*

addition of a scaled, Sobel-filtered version of the image, and hue adjustment for the real images. We also paint the background of the synthetic images with random crops from ImageNet-A [5].

S6. Digit style isolation: extended results, timing measurements



Supplemental Material, Figure S7. ***Retrieval results over the course of training, comparison.*** *We compare retrieval on the test set of MNIST at various stages of training ABC and the two VAE-based approaches mentioned in the main text. As in Figure 4 of the main text, the query images are the boxed images along the diagonal, and each row is the nearest representative for each class in embedding space. Also as before, in all cases the digit 9 was withheld during training.*

In Figure S7 we compare digit style isolation on MNIST using the output of ABC and the style part of the latent representations yielded by the VAE-based approaches of [6] and [3]. Interestingly, ML-VAE appears to embed the digits with respect to stroke thickness and slant very similarly to ABC at the beginning of training, long before any realistic images are able to be generated, but this clear interpretability of the embeddings fades as training progresses. There are no intermediate results to show for [8], which has no style representations until the second stage of training (the last ten epochs).

S7. Timing calculation on MNIST

	ABC (ours)	Sanchez et al. [8]	CC-VAE [6]
Seconds/epoch	47.8	70.2	150.2

Table S2. **Training timing for style isolation on MNIST (Section 4.2).** These comparisons were run on an NVIDIA Tesla K80.

We compare measurements of training time in Table S2, all run in Tensorflow on an NVIDIA Tesla K80. The discriminative approaches – ABC and [8] – are far faster to train than the generative approach of [6]. ABC is fastest by a wide margin due to its simplicity, requiring only one embedding network and a relatively simple loss calculation, in contrast to the seven networks and involved loss calculations required for [8].

Note that by having the fastest training time per epoch, ABC further widens the gulf to the baselines, which require orders of magnitude more epochs to yield representations which isolate digit style.

S8. Implementation details

Accompanying code may be found on github. For all experiments we use the ADAM optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$). Padding for convolutional layers is always 'valid.'

S8.1. Shapes3D

Layer	Units	Kernel size	Activation	Stride
Conv2D	32	3x3	ReLU	1
Conv2D	32	3x3	ReLU	1
Conv2D	64	3x3	ReLU	2
Conv2D	64	3x3	ReLU	1
Conv2D	128	3x3	ReLU	1
Conv2D	128	3x3	ReLU	2
Flatten	–	–	–	–
Dense	128	–	ReLU	–
Dense	Embedding dimension (64)	–	Linear	–

Table S3. **Architecture used for Shapes3D experiments (Section 4.1).** Input shape is [64, 64, 3].

For the experiments of Figures S1&3 we used the network architecture listed in Table S3, and trained for 2000 steps with a learning rate of 3×10^{-5}. We used a set size of 32 and squared L2 distance as the embedding space metric, with a temperature of 1. To curate a set for training, we randomly sample from among the possible values for the inactive factor(s) and then filter the dataset according to it. This takes longer when there are more inactive factors, as more of the dataset must be sieved out to acquire each stack.

S8.2. MNIST

Layer	Units	Kernel size	Activation	Stride
Conv2D	32	3x3	ReLU	1
Conv2D	32	3x3	ReLU	1
Conv2D	32	3x3	ReLU	2
Conv2D	32	3x3	ReLU	1
Conv2D	32	3x3	ReLU	1
Flatten	–	–	–	–
Dense	128	–	ReLU	–
Dense	Embedding dimension (8)	–	Linear	–

Table S4. **Architecture used for MNIST experiments (Section 4.2).** Input shape is [28, 28, 1].

For the MNIST experiments we used the architecture specified in Table S6. The set size was 64. We used a learning rate of 10^{-4} and trained for 500 steps. We used squared L2 distance as the embedding space metric and a temperature of 1. All instances of the digit 9 are held out at training time, and images of the other digits are formed into stacks before being randomly paired each training batch. This ran in under 30 seconds on an NVIDIA Tesla V100 GPU.

S8.3. Pose estimation

For both the pose estimation lookup (Table 1) and regression (Table 2) tasks, we use the same base network to embed the images, described in Table S5. In contrast to the Shapes3D and MNIST experiments, we train with mini-batches consisting of 4 pairs of image sets, each of size 32. We use cosine similarity and a temperature of 0.1 for lookup and 0.05 for regression. For the lookup task, the network trained for 40k steps with a learning rate that starts at 10^{-4} and decays by a factor of 2 every 10k steps. The beginning of training is purely synthetic images and then ramping up linearly to 10% real images folded into the unconstrained stack, stepping every 4k steps.

Layer	Units	Kernel size	Activation	Stride
ResNet50, up to conv4_block6	–	–	–	–
Conv2D	256	3x3	ReLU	1
Global Average Pooling	–	–	–	–
Flatten	–	–	–	–
Dense	128	–	tanh	–
Dense	Embedding dimension (64)	–	Linear	–

Table S5. **Architecture used for pose estimation experiments (Section 4.3).** Input shape is [128, 128, 3].

For regression, the embeddings are then fed, separately for each Euler angle, as input to a 128 unit dense layer with tanh activation, which is then split off into two dense layers with 2 and 4 units and linear activation for the angle magnitude and quadrant, respectively, as in [7]. To maintain consistency between how the embeddings are processed for the ABC loss and how they are fed into the regression sub-network, the embeddings are L2-normalized to lie on the 64-dimensional unit sphere before the regression. The angle magnitudes are passed through a spherical exponential activation function [7], which is the square root of a softmax. The magnitudes are then compared with ground truth $(|\sin\phi_i|, |\cos\phi_i)|$, with i spanning the three Euler angles, through a cosine similarity loss. The quadrant outputs are trained as a classification task with categorical cross entropy against the ground truth angle quadrants, defined as $(\text{sign}(\sin\phi_i), \text{sign}(\cos\phi_i))$. Training proceeds for 60k steps with a learning rate that starts at 10^{-4} and decays by a factor of 2 every 20k steps.

To more closely match the distribution of camera pose in real images, we filter the ShapeNet renderings by elevation: 0.5 radians and 1.3 radians for the max elevation for cars and chairs, respectively (for all of the baselines as well).

S8.4. Factor isolation on real images only

Layer	Units	Kernel size	Activation	Stride
Conv2D	64	3x3	ReLU	1
Conv2D	64	3x3	ReLU	2
Conv2D	128	3x3	ReLU	1
Conv2D	128	3x3	ReLU	2
Conv2D	256	3x3	ReLU	1
Conv2D	256	3x3	ReLU	2
Flatten	–	–	–	–
Dense	128	–	ReLU	–
Dense	Embedding dimension (64)	–	Linear	–

Table S6. **Architecture used for the Extended YaleB Face and EPFL Multi-view Car experiments (Section 4.4).** Input shape is [64, 64, 1] for the face images and [64, 64, 3] for the car images.

S8.4.1 Extended YaleB Face

Images of all 28 people were used for training. There are 9 poses per person and 64 illumination angles. With the illumination as the only active factor, the full set of 64 images was used for each set; with illumination and pose active, 384 images were randomly sampled for each person (out of 576) every time a set was made for training.

Each image was augmented twice with a random shift of contrast, and then a random crop (though the faces were not tight-cropped for this dataset). We used negative L2 distance as the similarity measure, and a learning rate of 3×10^{-4} for 10,000 steps (illumination active) or 50,000 steps (illumination and pose active).

S8.4.2 EPFL Multi-view Car

We used a set size of 81 images randomly sampled from the sequence of images for a particular car. All 20 car sequences were used for training. Each image was augmented twice with a random shift of contrast and saturation, and then a random

crop (though the cars were not tight-cropped for this dataset). We used negative L2 distance as the similarity measure, and a learning rate of 3×10^{-4} for 10,000 steps.

S8.5. Baselines

Imagenet-pretrained ResNet: We use the same ResNet50V2 base as for the ABC embedding network, and compare representations for each image by cosine similarity (which performed better than comparing by L2 distance).

Sanchez et al. [8]: We used the colored-MNIST architecture specifications and hyperparameters described in the Supplemental Material for the MNIST experiments of Section 4.2. As the colored-MNIST factors of variation isolated by [8] are simpler in nature (color of foreground/background from specific digit, versus digit identity from style), we found better results by boosting the dimension of the exclusive representation to 64 (up from the original 8 for the color description).

CCVAE [6] and ML-VAE [3]: We translated the publicly available pytorch code to tensorflow for training MNIST[1],[2]. We were unable to find code for their experiments on larger image sizes, but we followed the encoder and decoder specifications for the 64x64 RGB images in the Supplemental for [6], found here[3], for both methods. We optimized hyperparameters in a grid search around the published numbers, and used a group size for [3] which matched the stack size used for the ABC method. As with [8], we downsized the ShapeNet renderings and Pascal3D+ tight crops to 64x64, after attempts to scale the encoder-decoder architecture up to 128x128 were unsuccessful.

LORD [4]: We ran the publicly available tensorflow code with the parameters for the paper's Cars3D experiments[4]. Because a separate latent vector for each instance must be learned, memory became an issue when using the full training set (as used for all other methods). We used 200 instances each for cars and chairs to closely match the 183 Cars3D instances used in their work, with images downsized and tight-cropped to 64x64 to match this works' architecture.

References

[1] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 531–540. PMLR, 10–15 Jul 2018. S1, S2

[2] Adar Elad, Doron Haviv, Yochai Blau, and Tomer Michaeli. Direct validation of the information bottleneck principle for deep nets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019. S3

[3] Bouchacourt et al. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *AAAI*, 2018. S4, S8, S11

[4] Aviv Gabbay and Yedid Hoshen. Demystifying inter-class disentanglement. In *International Conference on Learning Representations (ICLR)*, 2020. S4, S11

[5] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *arXiv preprint arXiv:1907.07174*, 2019. S7

[6] Ananya Harsh Jha, Saket Anand, Maneesh Singh, and V. S. R. Veeravasarapu. Disentangling factors of variation with cycle-consistent variational auto-encoders, 2018. S4, S8, S11

[7] Shuai Liao, Efstratios Gavves, and Cees G. M. Snoek. Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-spheres. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. S4, S10

[8] Eduardo Hugo Sanchez, Mathieu Serrurier, and Mathias Ortner. Learning disentangled representations via mutual information estimation. In *The European Conference on Computer Vision (ECCV)*, 2020. S8, S11

[1] https://github.com/ananyahjha93/cycle-consistent-vae
[2] https://github.com/DianeBouchacourt/multi-level-vae
[3] https://arxiv.org/pdf/1804.10469.pdf
[4] https://github.com/avivga/lord