

Supplemental Material for Fast and Flexible Indoor Scene Synthesis via Deep Convolutional Generative Models

Daniel Ritchie * Kai Wang * Yu-an Lin
Brown University

{daniel_ritchie, kai_wang, yu-an.lin@brown.edu }

1. Model Architecture Details

Here we give specific details about the neural network architectures used for each of our system’s modules. For reference, we also reproduce the pipeline overview figure from the main paper (Figure 1).

1.1. Next Category

The module uses a Resnet18 [1] to encode the scene image. It also extract the counts of all categories of objects in the scene (i.e. a “bag of categories” representation), as in prior work [2], and encodes this with a fully-connected network. Finally, the model concatenates these two encodings and feeds them through another fully-connected network to output a probability distribution over categories. At test time, the module samples from the predicted distribution to select the next category. Figure 2 shows the architecture diagram for this network.

1.2. Location

Figure 3 shows the architecture diagram for this module. It uses a Resnet34 [1] to encode the scene image. It is followed by five “up-convolutional” (i.e. transpose convolution) blocks (*UpConvBlock*). Up-convolution is done by first nearest-neighbor upsampling the input with scale factor of 2, and then applying a 3x3 convolution. Finally, we apply a 1x1 convolution to generate a $(C + 1) \times 64 \times 64$ distribution over categories and location, where C is the number of categories for the room type.

Since the target output during the training process (exact location of the object centroids for the room) is different from the outcome we prefer (a smooth distribution over all possible locations), the module has a high potential to overfit. To alleviate this, we apply dropout before and after the Resnet34 encoder, and also before the final 1x1 convolution. We also apply L2 regularization in the training process. We found this combination of techniques effective at preventing overfitting, though we have not quantitatively evaluated the behavior of each individual component.

1.3. Orientation

Given a translated top-down scene image and object category, the orientation module predicts what direction an object of that category should face if placed at the center of the image. Figure 4 shows the architecture diagram for this module. We assume each category has a canonical front-facing direction. Rather than predict the angle of rotation θ , which is circular, we instead predict the front direction vector, i.e. $[\cos \theta, \sin \theta]$. This must be a normalized vector, i.e. the magnitude of $\sin \theta$ must be $\sqrt{1 - \cos^2 \theta}$. Thus, our module predicts $\cos \theta$ along with a Boolean value giving the sign of $\sin \theta$ (more precisely, it predicts the probability that $\sin \theta$ is positive). Here, we found using separate network weights per category to be most effective.

The set of possible orientations has the potential to be multimodal: for instance, a bed in the corner of a room may be backed up against either wall of the corner. To allow our module to model this behavior, we implement it with a conditional variational autoencoder (CVAE) [3]. Specifically, we use a CNN to encode the input scene (the *Conditional Prior*), which we then concatenate with a latent code z sampled from a multivariate unit normal distribution, and then feed to a fully-connected *Decoder* to produce $\cos \theta$ and the sign of $\sin \theta$. At training time, we use the standard CVAE loss formulation to learn an approximate posterior distribution over latent codes).

Since interior scenes are frequently enclosed by rectilinear architecture, objects in them are often precisely aligned to cardinal directions. A CVAE, however, being a probabilistic model, samples noisy directions. To allow our module to produce precise alignments when appropriate, this module includes a second CNN (the *Snap Predictor*) which takes the input scene and predicts whether the object to be inserted should have its predicted orientation “snapped” to the nearest of the four cardinal directions.

1.4. Dimensions

Given a scene image transformed into the local coordinate frame of a particular object category, the dimensions

*Equal contribution

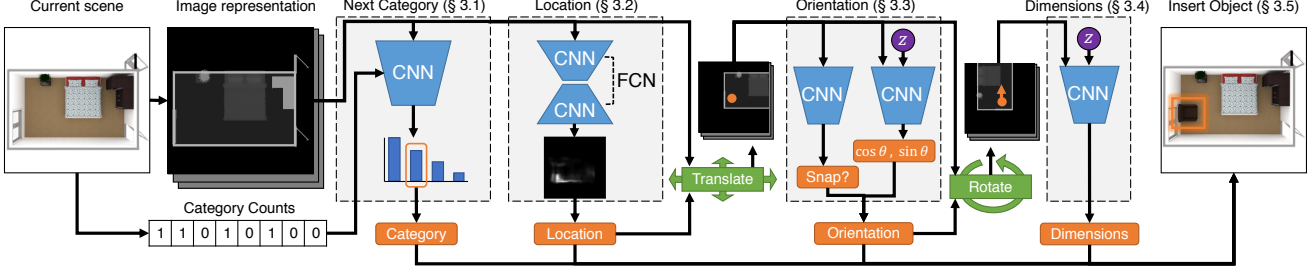


Figure 1. Overview of our automatic object-insertion pipeline. We extract a top-down-image-based representation of the scene, which is fed to four decision modules: which category of object to add (if any), the location, orientation, and dimensions of the object.

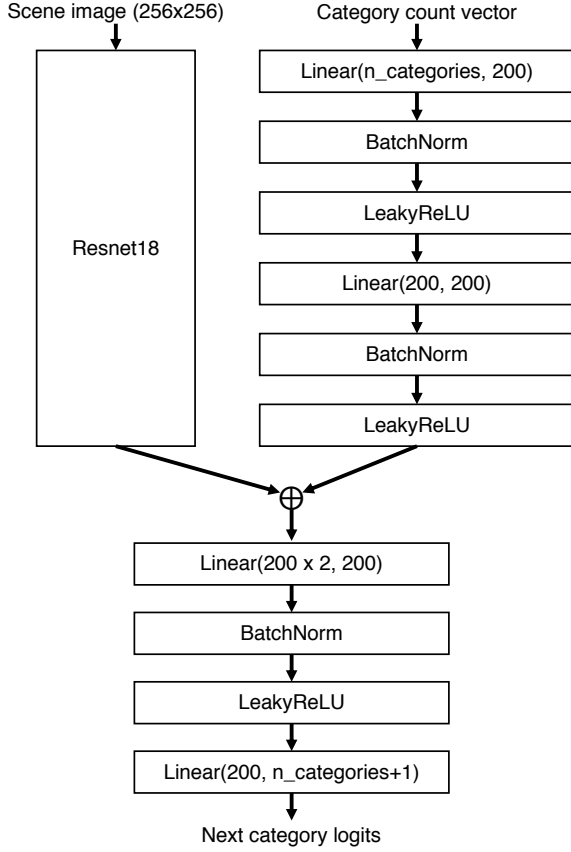


Figure 2. Architecture diagram for the next category prediction module.

module predicts the spatial extent of the object. That is, it predicts an object-space bounding box for the object to be inserted. This is also a multimodal problem, even more so than orientation (e.g. many wardrobes of varying lengths can fit against the same wall). Again, we use a CVAE for this: a CNN encodes the scene, concatenates it with z , and then uses a fully-connected decoder to produce the $[x, y]$ dimensions of the bounding box. Figure 5 shows the architecture diagram for this module.

The human eye is very sensitive to errors in size, e.g. a too-large object that penetrates the wall next to it. To

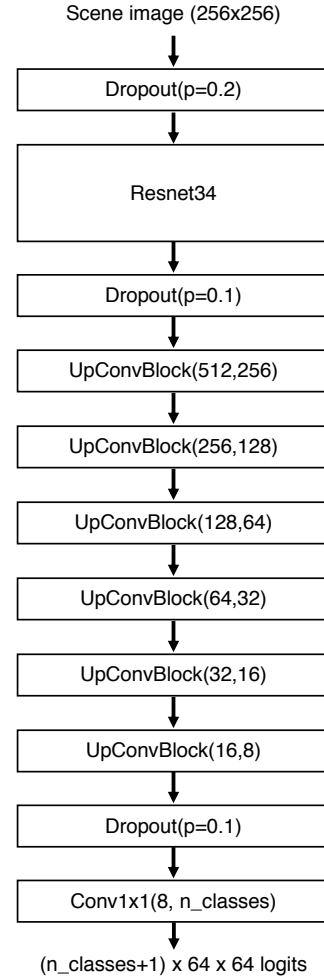


Figure 3. Architecture diagram for the location prediction module. An *UpConvBlock* is a 3x3 transpose convolution with stride 2 followed by a Batch Normalization layer and a ReLU layer.

fine-tune the prediction results, we include an adversarial loss term in the CVAE training. This loss uses a convolutional *Discriminator* which takes the input scene concatenated channel-wise with the signed distance field (SDF) of the predicted bounding box. As with orientation, this module also uses separate network weights per category.

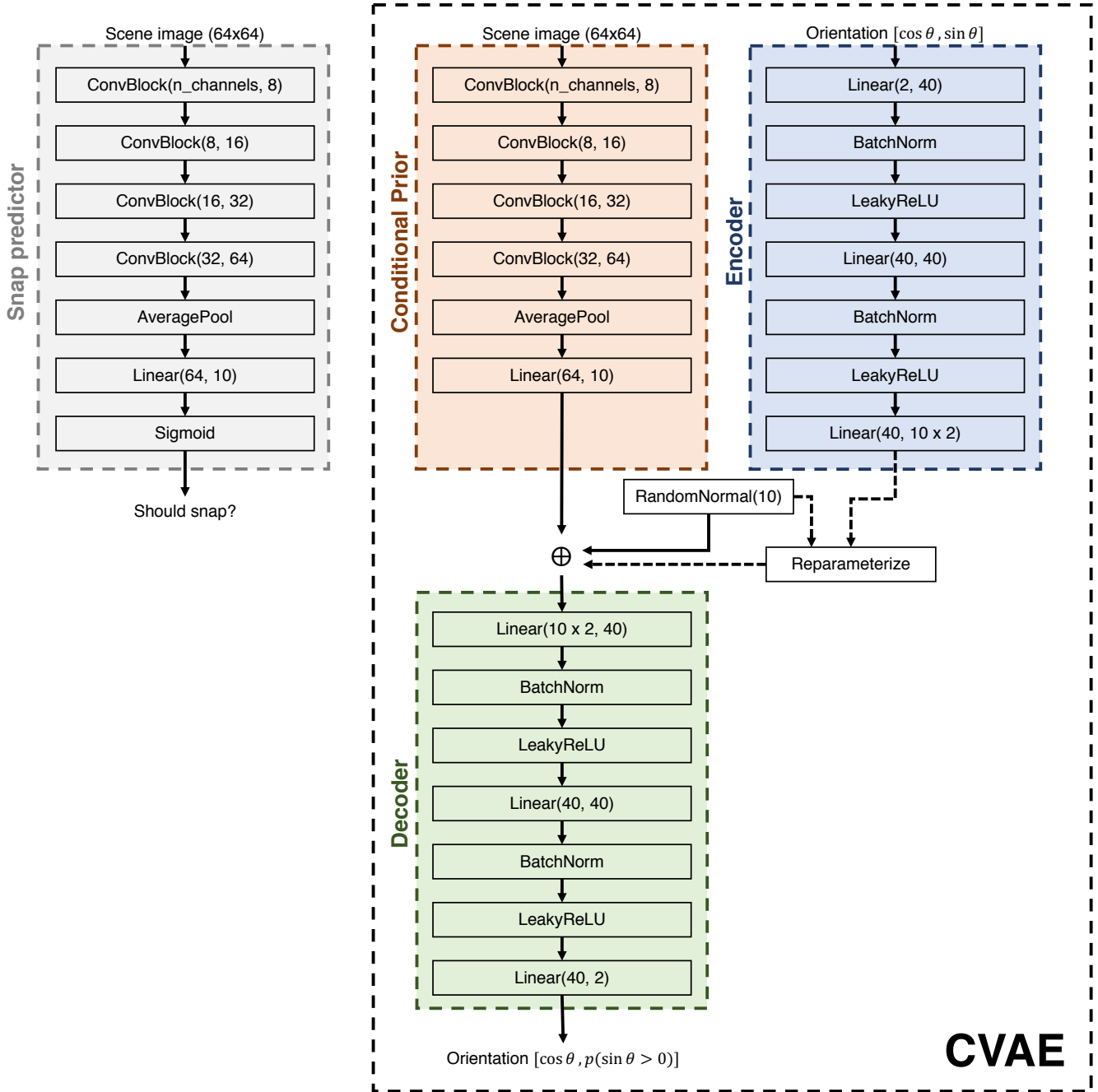


Figure 4. Architecture diagram for the orientation prediction module. A *ConvBlock* is a 3x3 convolution with stride 2 followed by a Batch Normalization layer and a ReLU layer.

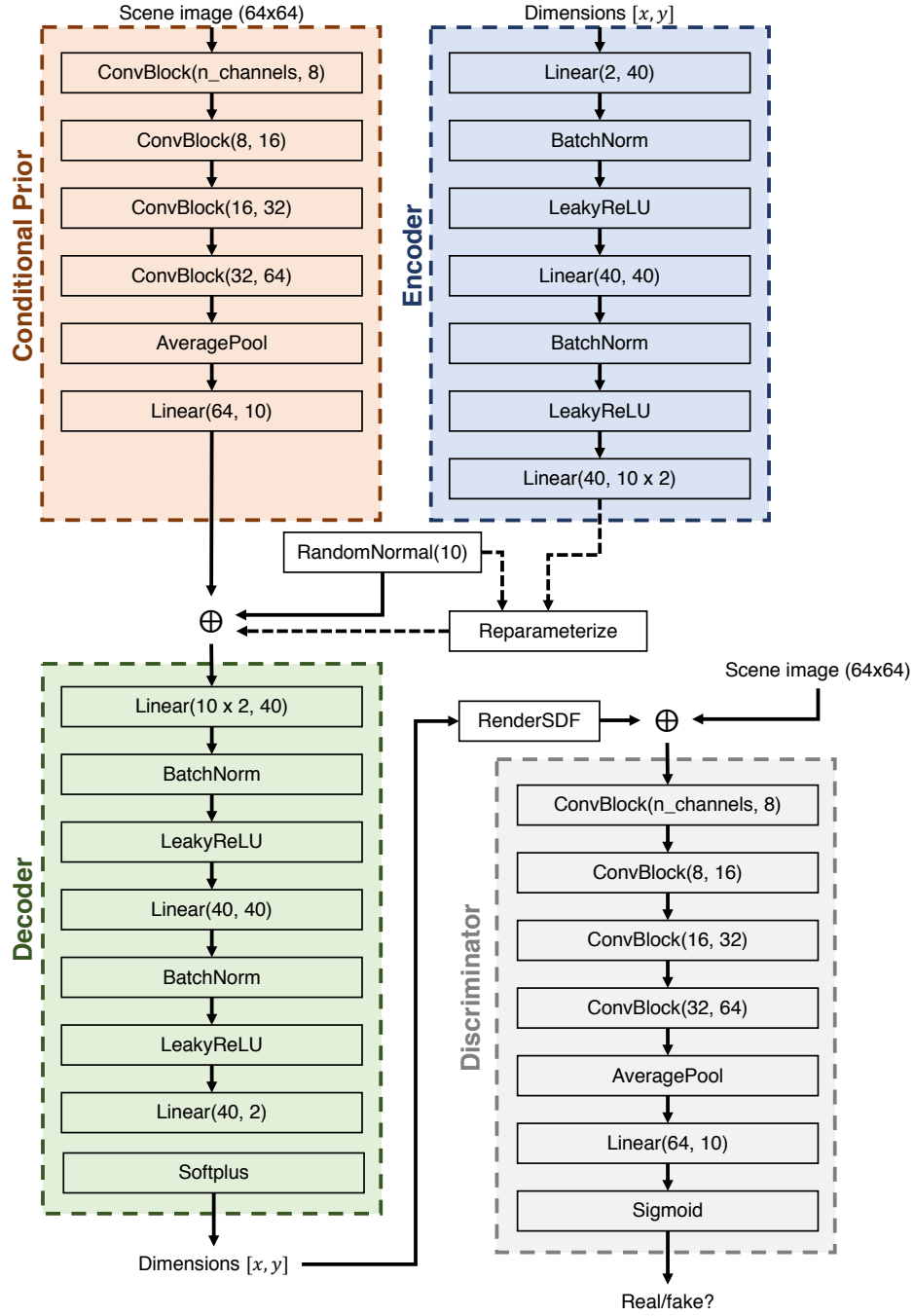


Figure 5. Architecture diagram for the dimensions prediction module. A *ConvBlock* is a 3x3 convolution with stride 2 followed by a Batch Normalization layer and a ReLU layer.

2. Dataset Details

We adopt similar dataset filtering strategies as that of prior work [2], with a few notable differences:

1. We manually selected a list of frequently-occurring objects, which we allow to appear on top of other objects (only on the visible top surface, i.e. no televisions contained in a TV stand). We remove all second tier objects whose parents were filtered out.
2. To facilitate matching objects by bounding box dimensions, we discard rooms containing objects which are scaled by more than 10% along any dimensions. For objects scaled by less than that, we remove the scaling from their transformation matrices.
3. We augment the living room and office dataset with 4 different rotations (0° , 90° , 180° , 270°) of the same room during training, to reduce overfitting, particularly for the location module.

Table 1 shows the counts of all categories appearing in the four types of rooms used in this work, where possible second tier categories are highlighted with bold. The categories are arranged in the order that they are given to the category module.

3. Performance of Each Model Component

Table 2 shows the performance of each of our modules on a held-out test set of scene data. Different metrics are reported for different modules, as appropriate. We have no natural baseline to which to compare these numbers. As an alternative, we report the improvement in performance relative to a randomly-initialized network.

4. Generalization

To evaluate if our models are merely “memorizing” the training scenes, we measure similar a generated room can be to a room in the training set. To do so, we use the same scene-to-scene similarity function as in prior work [2] and compute the maximal similarity score of a generated room against 5,000 rooms in the training set. We plot the score distribution for 1,000 generated rooms in Figure 6. For comparison, We also compute the same score for 1,000 rooms from the training set (which are disjoint from the 5,000 aforementioned rooms). In general, the behavior for the synthesized rooms is similar to that of scenes from the dataset. Our model definitively does not just memorize the training data, as it is actually less likely for our model to synthesize a room that is very similar to one from the training set. It is also less likely for our model to synthesize something that is very different from all other rooms in the training set. This is coherent with our impression: that our

model suffers from minor mode collapses, and does not capture all possible unique room layouts. Finally, the large spike in extremely-similar rooms for the dataset-to-dataset comparison (the tall orange bar on the far right of the plot) is due to exact duplicate scenes with exist in the training set.

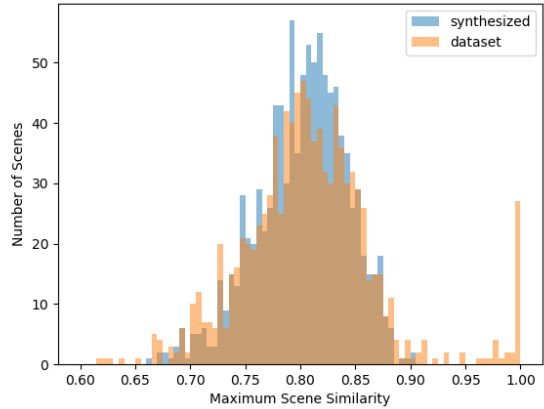


Figure 6. Plotting the maximal similarity score of a bedroom against 5,000 rooms from the training set. We plot the distribution of results for 1,000 synthesized rooms and 1,000 held out rooms in the training set (disjoint from the 5,000)

References

- [1] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR 2016*, 2016. 1
- [2] Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Deep Convolutional Priors for Indoor Scene Synthesis. In *SIGGRAPH 2018*, 2018. 1, 5
- [3] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28*. 2015. 1

Bedroom									
Name	door	window	double bed	wardrobe	single bed	desk	stand	bunker bed	dresser
Count	8335	7104	4230	6690	2032	2586	6508	593	2222
Name	tv stand	plant	sofa	dressing table	office chair	coffee table	sofa chair	ottoman	shelving
Count	1432	1341	393	1730	2077	1048	727	1142	901
Name	floor lamp	armchair	daybed	table lamp	baby bed	piano	shoes cabinet	straight chair	hanger
Count	1387	703	181	2844	266	102	619	595	540
Name	television	bench chair	toy	laptop	loudspeaker	chair	book	console	whiteboard
Count	1252	134	231	1010	438	98	858	368	105
Name	stool	pedestal fan	vase	fishbowl					
Count	178	320	395	59					
Living Room									
Name	door	window	sofa	sofa chair	coffee table	plant	tv stand	floor lamp	fireplace
Count	2286	1789	1661	983	1336	1059	696	651	257
Name	piano	wardrobe	ottoman	shelving	armchair	loudspeaker	dresser	television	toy
Count	85	172	314	309	204	384	64	447	56
Name	stand	vase	console	straight chair	shoes cabinet	bench chair	stool	hanger	laptop
Count	101	282	187	51	31	26	33	34	72
Name	table lamp	pedestal fan	fishbowl	book	cup	fruit bowl	glass	bottle	
Count	68	40	21	69	59	17	16	15	
Office									
Name	door	window	desk	sofa	office chair	plant	shelving	sofa chair	wardrobe
Count	1572	1307	1616	313	1577	557	764	290	319
Name	armchair	piano	tv stand	coffee table	floor lamp	straight chair	dresser	bench chair	ottoman
Count	300	75	137	137	256	274	84	37	112
Name	fireplace	whiteboard	laptop	stand	toy	table lamp	book	loudspeaker	shoes cabinet
Count	38	100	377 102		52	294	356	113	39
Name	hanger	stool	television	vase	pedestal fan	water machine	console	fishbowl	cup
Count	51	56	84	132	49	32	30	5	27
Bathroom									
Name	door	window	bathtub	shower	sink	toilet	shelving	plant	washer
Count	7448	4299	5441	5308	4627	6437	3963	1045	1428
Name	bidet	wardrobe	stand	dresser	floor lamp	ottoman	cabinet	hanger	trash can
Count	1753	459	315	89	231	95	57	111	409
Name	toy	coffee table	straight chair	vase					
Count	126	31	20	31					

Table 1. Counts for all the object categories appearing in the four types of rooms used in this paper, in the order that they are presented to the category prediction module. Bold category name indicates that this can be a second tier object

Room Type	Cat (Top1)	Cat (Top5)	Loc (X-Ent)	Orient (ELBo)	Orient-Snap (Acc.)	Dims (ELBo)
Bedroom	0.5000 (+0.4225)	0.8650 (+0.7600)	0.0030 (-98.61%)	0.0899 (-54.20%)	0.8821 (+0.3821)	0.0018 (-97.74%)
Living	0.5375 (+0.5312)	0.8719 (+0.8001)	0.0035 (-98.56%)	0.1093 (-44.99%)	0.8902 (+0.3902)	0.0018 (-97.79%)
Office	0.5664 (+0.5525)	0.8948 (+0.7629)	0.0038 (-98.25%)	0.0639 (-68.03%)	0.9482 (+0.4482)	0.0015 (-98.14%)
Bathroom	0.6180 (+0.5873)	0.9573 (+0.8597)	0.0020 (-85.00%)	0.0906 (-53.23%)	0.9419 (+0.4419)	0.0019 (-97.64%)

Table 2. Performance of each component of our model on held-out test data. *Acc.* is binary classification accuracy; *Top N* is top-n multiclass classification accuracy; *X-Ent* is cross-entropy; *ELBo* is the standard Evidence Lower Bound objective for variational autoencoders. The numbers in parentheses show the improvement relative to a randomly-initialized network.