A Models and training

For our experiments on the standard unsupervised task we used two different VAE architectures. The first one is the same found in Higgins et al. (2017) and uses a 2-layer MLP as an encoder with 1200 units and ReLU non-linearity. The decoder is a 3-layer with the same number of units and the Tanh non-linearity. The second architecture is the one found in Burgess et al. (2018) and consists of a 3-layer CNN with $32{\times}4{\times}2{\times}1$ convolutions and max pooling, followed by a 2-layer MLP with 256 units in each layer. The decoder is defined to be the transpose of this architecture. ReLU non-linearity where applied after each layer of the CNN and the MLP for both the encoder and the decoder. Both models used a Gaussian stochastic layer with 10 units as in the original papers.

We also tested two variants of this last architecture, one found in Mathieu et al. (2019) which changes the shape of the convolution and another with batch normalisation. Neither variant exhibited any improvements to disentanglement or reconstruction on the full dSprite data and so were not included in the rest of the experiments.

For the image composition task we used same as in Burgess et al. (2018) that we described above. The latent transformation layer was parameterized as:

$$h_{transformed} = W_r cat[z_r; action] + W_t cat[z_t; action]$$

where z_r and z_t are the samples from the stochastic layer for reference and transform image, cat is the concatenation operation performed along the column dimension. The output is another 10-dimensional vector with the transformed latent code.

Alternatively we also tried a 3 layer MLP with 100 hidden units, but saw no benefit in performance and decreased disentanglement when trainined on the full dataset.

Training on the unsupervised tasks ran for 100 epochs for dSprites and 65 epochs for Shapes3D, even though models converged before the end. The learning rate was fixed at $1e-4$ and the batch size at 64. β values used were $1, 4, 8, 12, 16$ on the full dSprite dataset. $\beta = 4$ and $\beta = 16$ where not included in the rest of the experiments since the former offered very little disentanglement and the latter very large reconstruction error. For the FactorVAE we used $\gamma = 20, 50, 100$ throughout. In the composition task the models where trained for 100 epochs with $\beta = 1$. Using β higher than 1 interfered with the model's ability to solve the task so they where not used.

For the ground-truth decoders (GT Decoder) we used the same MLP decoder of Higgins et al. (2017) mentioned above. Using deeper decoders with convolutions with/without batch norm after each layer was also tested, but did not provide significan benefits and also decreased the performance on some of the conditions.

All the models where implemented in PyTorch (Paszke et al., 2019) and the experiments where performed using the Ignite and Sacred frameworks (V. Fomin & Tejani, 2020; Klaus Greff et al., 2017).

To measure disentanglement we used the framework proposed by Eastwood & Williams (2018) with a slight modification. The approach consists of predicting each generative factor value, given the latent representations of the training images using a non-linear model. In our case we used the LassoCV regression found in the Sklearn library (Pedregosa et al., 2011) with an α coefficient of 0.01 and 5 cross-validation partitions. Deviating from the original proposal, we do not normalize the inputs to the regression model since we found that this tends to give a lot of weight to dead units (when measured by their KL divergence). This is likely due to the model "killing" these units during training after they start with a high KL value, which might not completely erase the information they carry about a given generative factor.

B Extra plots for Dsprites dataset

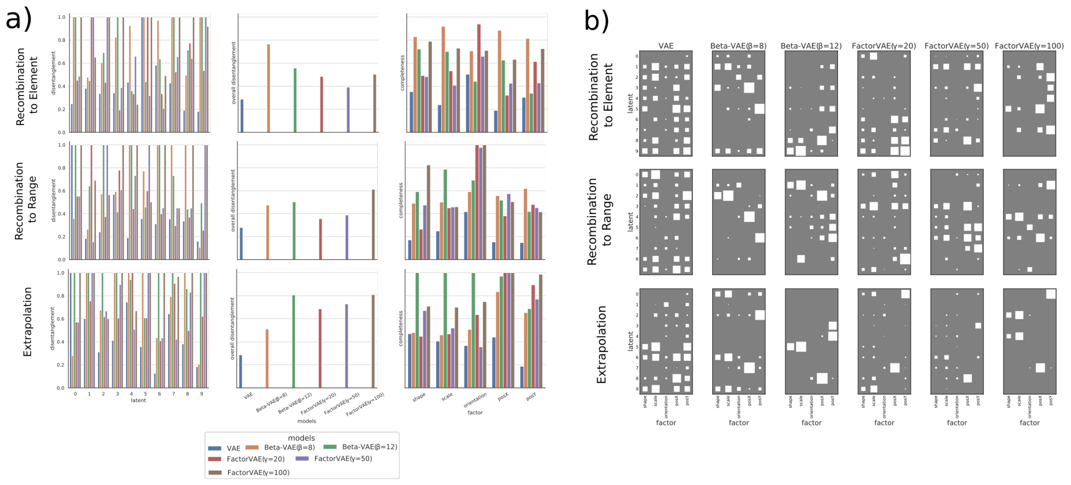

Figure 7: **Disentanglement analysis for dSprites**. The disentanglement analysis results for the dSprites dataset. a) The scores for each of the metrics evaluated by the DCI framework: disentanglement (left), overall disentanglement (middle) and completeness (right) for each of the conditions. b) The matrices of coefficients computed by the framework plotted as Hinton Diagrams. These are used to obtain the quantitative scores in the panel above. They offer a qualitative view of how the model is disentangling. On the left is how perfect disentanglement looks in this framework.

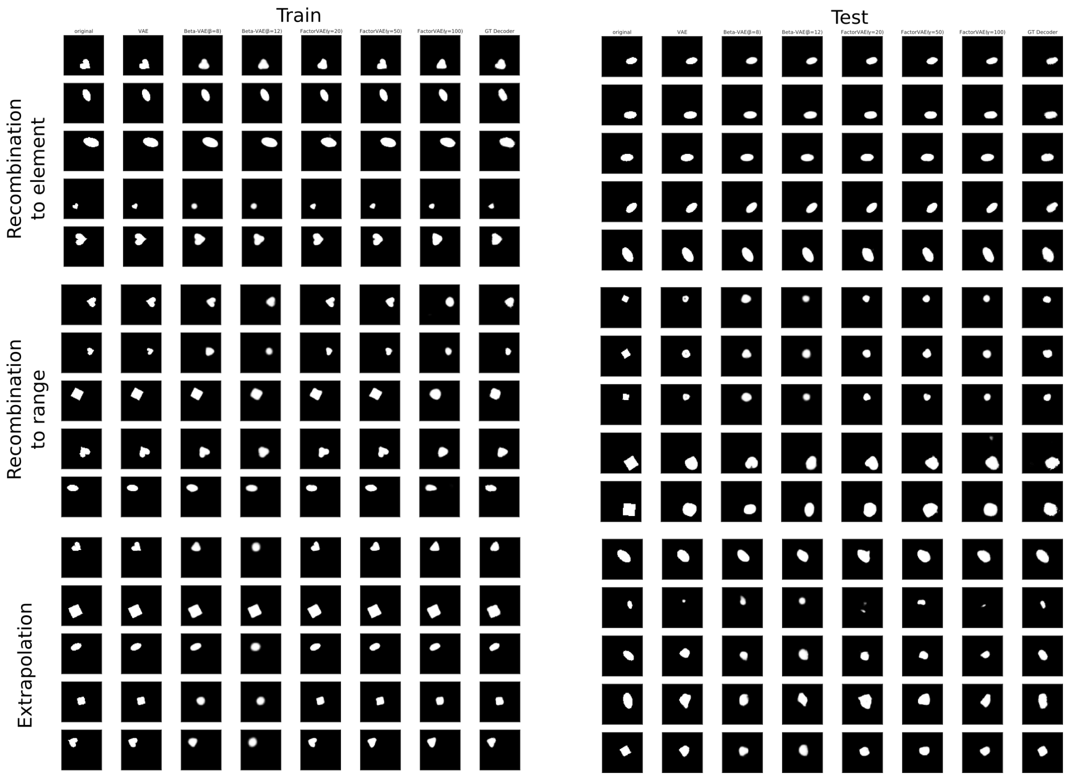

Figure 8: **Reconstructions for the dSprites dataset**. For each condition and model, these are some reconstruction examples for both training and testing. There is general success and failure in the Recombination to Element (top) and Extrapolation (bottom) conditions, respectively. For this last condition, the models seem to reproduce the closest instance they have seen, which tranlated to the middle of the image. For the Recombination to range (middle), the models tend to resort to generating a blob at the right location, minimising their pixel-level error.

C EXTRA PLOTS FOR THE 3D SHAPES DATASET

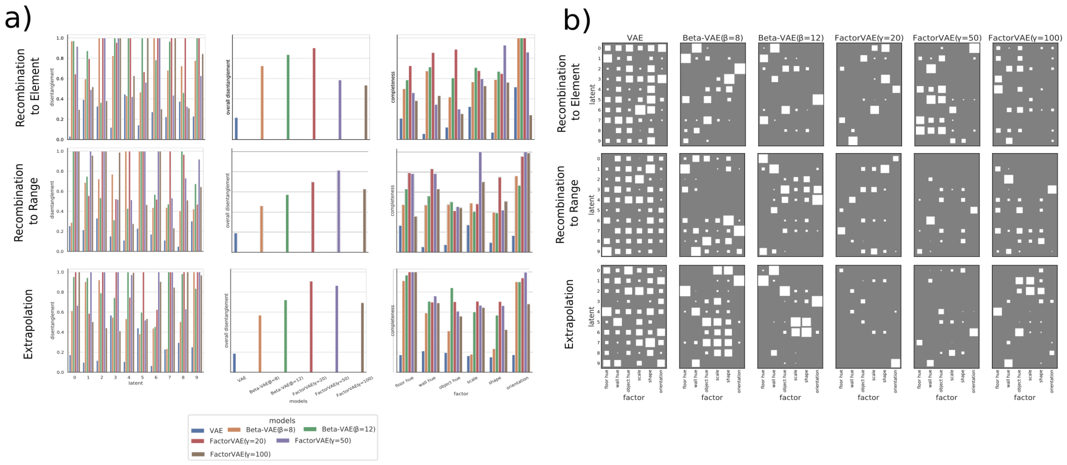

Figure 9: **Disentanglement analysis for Shapes3D**. The disentanglement analysis results for the 3D Shapes dataset. a) The scores for each of the metrics evaluated by the DCI framework (Eastwood & Williams, 2018): disentanglement (left), overall disentanglement (middle) and completeness (right) for each of the conditions. b) The matrices of coefficients computed by the framework plotted as Hinton Diagrams. As discussed in the main text, these matrices offer a qualitative view of how the model is disentangling. In general, sparse matrices indicate higher disentanglement. It is clear from these diagrams that the degree of disentanglement varies over a broad range for the tested models.

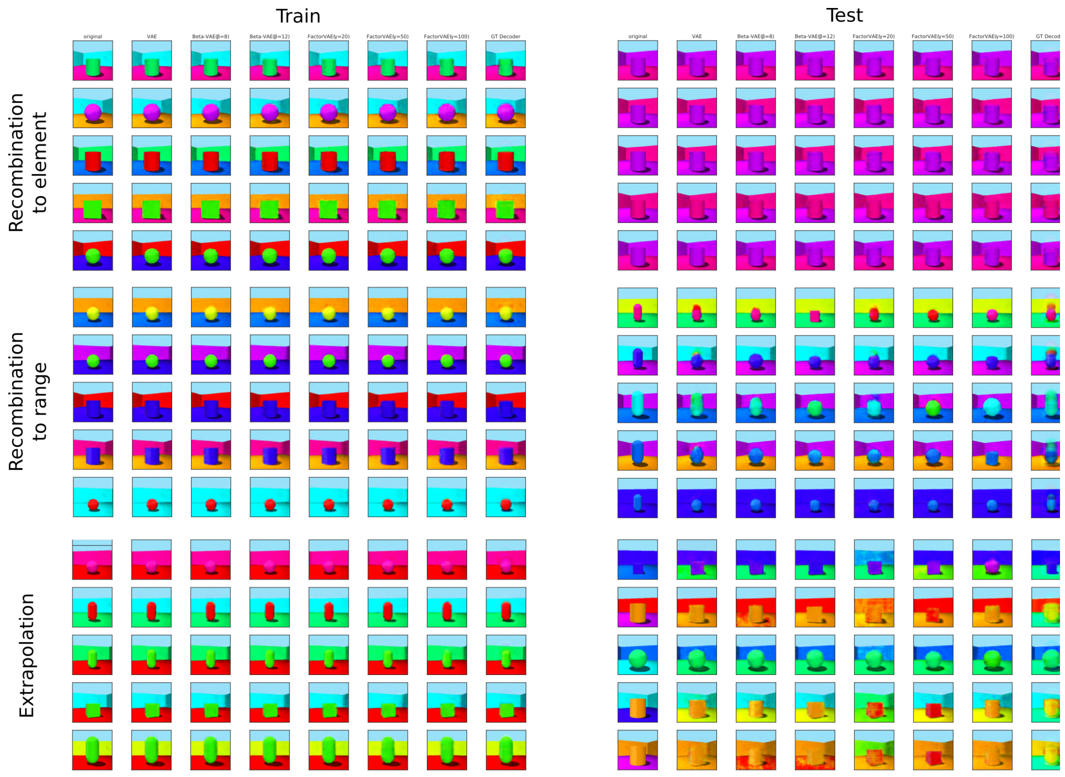

Figure 10: **Reconstructions for the Shapes3D dataset**. For each condition and model, these are some reconstruction examples for both training (left) and testing (right). In each case, the input image is shown in the left-most column and each subsequent column shows reconstruction by a different model. The test images always show a combination that was left out during training. All training images are successfully reproduced. However, reconstruction for test images only succeeds consistently for the Recombination-to-Element condition (top). All reconstructions fail in the Extrapolation condition (bottom) while most of them fail for the Recombination-to-Range condition (middle). There are occasional instances in Recombination-to-Range condition that seem to correctly reconstruct the input image. This seems to happen when the novel combination for color and shape is closest to the ones the model has experienced during training. For example, models are better when the oblong shape is paired with cyan (which is close to green, which it has seen) and worse on magenta.