Appendix

A. Implicit Bias in InfoGAN

Practical implementation of InfoGAN loss and the resulting implicit bias. Let $P_{c,x}$ denote the joint distribution of the latent code c and the generated image $x = G(c, z)$. Two structural assumptions are enforced in (Chen et al., 2016) to make the maximization of $I(c; x)$ tractable. First, to make optimization of the mutual information tractable, all practical implementations of InfoGAN replace $I(c; x)$ with a variational lower bound $\max_{Q_{c|x}} \mathcal{L}_{\text{Info}}(G, Q)$. Here $Q_{c|x}$ is an auxiliary conditional distribution, which is maximized over the InfoGAN regularizer defined as

$$\mathcal{L}_{\text{Info}}(G, Q) \triangleq \mathbb{E}_{c\sim P_c, z\sim P_z, x\sim G(z,c)} \left[\log Q_{c|x}\right] , \tag{6}$$

where P_c and P_z denote the distributions chosen as priors. When this lower bound is maximized over $Q_{c|x}$, it acts as a surrogate for mutual information: rearranging the terms gives

$$\mathcal{L}_{\text{Info}}(G, Q) = I(c; x) - H(c) - \mathbb{E}_{x\sim P_x}[d_{\text{KL}}(P_{c|x} \| Q_{c|x})] ,$$

and $\max_{Q_{c|x}} \mathcal{L}_{\text{Info}}(G, Q) = I(c; x) - H(c)$ (see (Chen et al., 2016) for a derivation). However, this maximization is a functional optimization over an infinite dimensional function $Q_{c|x}$, which is intractable. To make this tractable, the optimization is done over a restricted family of Gaussian distributions in (Chen et al., 2016), which are factorized (or independent) Gaussian distributions (see $\mathcal{Q}$ in Remark 2). $Q_{c|x}$ can then be parametrized by the conditional means and variances, which is now finite dimensional functions, and one can use deep neural networks to approximate them. Note that the Shannon entropy $H(c)$ is a constant that does not depend on $G(\cdot, \cdot)$ or $Q_{c|x}$.

Next, if this maximum over $Q_{c|x}$ has been achieved, then notice that in the generator update, the generator tries to minimize $\min_G \mathcal{L}_{\text{Adv}}(G, D) - \lambda(I(c; x) - H(c))$. This is problematic as the G update will increase $I(c, x)$ unboundedly, tending to infinity (even if $Q_{c|x}$ is restricted to factorized Gaussian class). The maximum value of $I(c; x) = \infty$ is achieved, for example, if $Q_{c_i|x}$ has variance zero for some i. This problem is not just theoretical. In Appendix B, we confirm experimentally that training InfoGAN with an unfactorized $Q_{c|x}$ leads to training instability. To avoid such catastrophic failures, (Chen et al., 2016) forces $Q_{c|x}$ to have an identity covariance matrix. This restricts the class of $Q_{c|x}$ that we search over, and forces the $\mathcal{L}_{\text{Info}}(G, Q)$ to be bounded, and the G and $Q_{c|X}$ updates to be well-defined. In summary, for stability and efficiency of training, (Chen et al., 2016) restricted $Q_{c|X}$ to be a *factorized Gaussian* with *identity covariance*. We show next that this choice creates an implicit bias.

Remark 2. *If optimized over a class of factorized Gaussian conditional distributions with unit variances, i.e.* $\mathcal{Q} = \{Q_{c|x} \,|\, Q_{c|x} = Q_{c_1|x} \times \cdots \times Q_{c_k|x}\}$ *and* $Q_{c_i|x} = (1/\sqrt{2\pi}) \exp\{-(1/2)(c_i - \mu_i(x))^2\}$ *for some* $\mu_i(x) \in \mathbb{R}$ *for all* $i \in [k]$ *and* $x \in \mathcal{X}$*, the maximum of the InfoGAN loss in Eq.* (6) *has the following implicit bias:*

$$\max_{Q_{c|x}\in\mathcal{Q}} \mathcal{L}_{\text{Info}}(G, Q) = I(x; c) - H(c) - \underbrace{\mathbb{E}\Big[\log \frac{P_{c|x}}{N_{c_1|\nu_1(x)} \cdots N_{c_k|\nu_k(x)}}\Big]}_{\textit{implicit bias}} , \tag{7}$$

where $P_{c,x}$ *is the joint distribution between the latent code* c *and the generated image* $x = G(c, z)$*, and* $N_{c_i|\nu_i(x)} = 1/\sqrt{2\pi} \exp\{-(1/2)(c_i - \nu_i(x))^2\}$ *with* $\nu_i = \mathbb{E}_{P_{c_i|x}}[c_i]$ *is the best one-dimensional Gaussian approximation of* $P_{c_i|x}$*.*

We provide a proof in Appendix A.1. The above implicit bias keeps the loss bounded, so it is necessary. On the other hand, it might have undesired and unintended consequences in terms of learning a disentangled deep generative model. In this paper, we therefore introduce a new regularizer to explicitly encourage disentanglement during InfoGAN training.

Improving InfoGAN performance via stable training. Before introducing our proposed regularizer, note that several VAE-based architectures claim to outperform InfoGAN by significant margins (Kim & Mnih, 2018; Higgins et al., 2016; Chen et al., 2018). This series of empirical results has created a misconception that InfoGAN is fundamentally limited in achieving disentanglement, which has been reinforced in following literature (Jeon et al., 2018; Ansari & Soh, 2018), which refer to those initial results. We show that the previously-reported inferior empirical performance of InfoGAN is due to poor architectural and hyperparameter choices in training. We take the same implementation reported to have bad performance (disentanglement score of 0.59 in Table 1). We then apply recently-proposed (but now popular) techniques for stabilizing GAN training and achieve a performance comparable to the best reported results of competing approaches (disentanglement score of 0.83 in Table 1). We provide more supporting experimental results in Section 3.2. Concretely,

we start from the implementation in (Kim & Mnih, 2018). We then apply spectral normalization to the adversarial loss discriminator (Miyato et al., 2018), with a choice of cross entropy loss, and use two time-scale update rule (Heusel et al., 2017) with an appropriate choice of the learning rate. These choices are motivated by similar choices and successes of (Brock et al., 2018) in scaling GAN to extremely large datasets. Implementation details are in Appendix F.6, and we also submit the code for reproducibility. Hence, the starting point for our design is to achieve even better disentanglement than a properly-trained version of InfoGAN.

A.1. Proof of Remark 2

Notice that $\mathcal{L}_{\text{Info}}(G, Q) \leq I(c, G(z, c))$. To understand why this works, let us decompose this lower bound further:

$$\begin{aligned}
&\mathcal{L}_{\text{Info}}(G, Q) \\
&= \mathbb{E}_{c\sim P_c, z\sim P_z, x\sim G(z,c)}\left[\log Q_{c|x}\right] \\
&= \mathbb{E}_{(x,c)\sim P(x,c)}\left[\log Q_{c|x}\right] \\
&= I(x; c) - H(c) + \mathbb{E}_{(x,c)\sim P(x,c)}\left[\log \frac{Q_{c|x}P_x}{P_x P_{c|x}}\right] \\
&= I(x; c) - H(c) + \mathbb{E}_{(x,c)\sim P(x,c)}\left[\log \frac{Q_{c|x}}{P_{c|x}}\right] \\
&= I(c; x) - H(c) - \mathbb{E}_{x\sim P_x}[d_{\text{KL}}(P_{c|x}\|Q_{c|x})]
\end{aligned}$$

We can further simplify and maximize the last term, which is the only one dependent on Q as,

$$\begin{aligned}
&\min_{Q\in\mathcal{Q}} \ \mathbb{E}_{x\sim P_x}[d_{\text{KL}}(P_{c|x}\|Q_{c|x})] \\
&= \min_{Q\in\mathcal{Q}} \ \mathbb{E}_{c,x\sim P_{c,x}}\Big[\log\Big(\frac{P_{c|x}}{Q_{c|x}}\Big)\Big] \\
&= \min_{Q\in\mathcal{Q}} \ \mathbb{E}_{c,x\sim P_{c,x}}\Big[\log\Big(\frac{P_{c|x}}{N_{c_1|\nu_1(x)}\cdots N_{c_k|\nu_k(x)}}\Big) + \log\Big(\frac{N_{c_1|\nu_1(x)}\cdots N_{c_k|\nu_k(x)}}{Q_{c|x}}\Big)\Big] \\
&= \mathbb{E}_{c,x\sim P_{c,x}}\Big[\log\Big(\frac{P_{c|x}}{N_{c_1|\nu_1(x)}\cdots N_{c_k|\nu_k(x)}}\Big)\Big] \ + \ \min_{Q\in\mathcal{Q}} \ \mathbb{E}_{c,x\sim P_{c,x}}\Big[\log\Big(\frac{N_{c_1|\nu_1(x)}\cdots N_{c_k|\nu_k(x)}}{Q_{c|x}}\Big)\Big] \\
&= \mathbb{E}_{c,x\sim P_{c,x}}\Big[\log\Big(\frac{P_{c|x}}{N_{c_1|\nu_1(x)}\cdots N_{c_k|\nu_k(x)}}\Big)\Big] \ ,
\end{aligned}$$

where the last equality follows from the fact that any $Q \in \mathcal{Q}$ can be parametrized by $(\mu_1(x), \ldots, \mu_k(x))$ as $Q_{c|x} = Q_{c_1|x}\cdots Q_{c_k|x}$ with $Q_{c_i|x} = (1/\sqrt{2\pi})\exp\{-(1/2)(c_i - \mu(x))^2\}$, in which case

$$\begin{aligned}
\min_{Q\in\mathcal{Q}} \ \mathbb{E}_{c,x\sim P_{c,x}}\Big[\log\Big(\frac{N_{c_1|\nu_1(x)}\cdots N_{c_k|\nu_k(x)}}{Q_{c|x}}\Big)\Big] \ &= \ \min_{\{\mu_i(x)\}_{i=1}^k} \ \mathbb{E}_{x\sim P_x}\Big[\sum_{i\in[k]}\big(\mu_i(x) - \nu_i(x)\big)^2\Big] \\
&\geq \ 0 \ ,
\end{aligned}$$

and the minimum of zero can be achieved by $\mu_i(x) = \nu_i(x)$, where $\nu_i(x) = \mathbb{E}_{P_{c_i|x}}[c_i]$.

B. InfoGAN with Non-factorizing Decoder

In this section we verify that the InfoGAN trained with non-factorizing multi-variate Gaussian distribution $Q(c|x)$ is unstable. Specifically, we train an InfoGAN with a decoder distribution of $Q(c|x) = \mathcal{N}(\mu(x), \Sigma(x))$, where $\mathcal{N}$ is the multivariate Gaussian distribution with mean $\mu(x) \in \mathbb{R}^k$, and full covariance matrix $\Sigma(x) \in \mathbb{R}^{k\times k}$. These parameters of the distribution are modelled as neural network functions of x, where we explicitly enforce the positive semi-definiteness of $\Sigma(x)$. This is less restrictive than the factorizing decoder distribution, $Q(c|x) = \prod_{i\in[k]} Q(c_i|x) = \prod_{i\in[k]} \mathcal{N}(\mu_i(x), 1)$ with its fixed diagonal covariance matrix $\mathbb{I}$, in the standard InfoGAN (see Remark 2).

Figure 9 shows the degradation in disentanglement due to the non-factorizing decoder. Similar to the experiment in Figure 2, we first train the standard InfoGAN ($\lambda = 0.2$) with factorizing decoder distribution $Q(c|x) = \prod_{i\in[k]} \mathcal{N}(\mu_i(x), 1)$ on

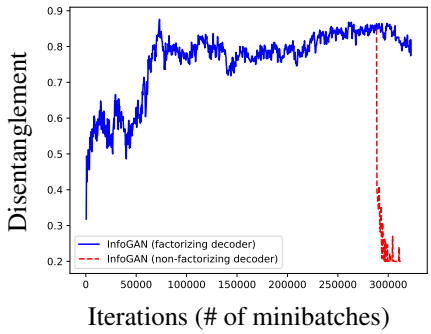

Figure 9: After 288,000 iterations, we continue training an InfoGAN with/without the non-factorizing decoder distribution $Q(c|x)$ (Section B). Non-factorizing decoder is high unstable and its training terminates early after reaching a low values of 0.2.

dSprites dataset for 25 epochs (288,000 batches) and from this point onwards we train two different versions of this model for 3 more epochs: one where we continue training the standard InfoGAN (blue solid curve) and another version where we replace the non-factorizing decoder distribution with the factorizing decoder distribution $Q(c|x) = \mathcal{N}(\mu(x), \Sigma(x))$ (red dashed curve). We plot the FactorVAE score, as defined in Section E.

We see that the non-factorizing decoder is highly unstable and its performance drops (from that of factorizing decoder) to 0.2 (minimum possible value for 5 latent codes) and its training terminates early because the learnt covariance matrix $\Sigma(x)$ becomes rank deficient.

C. Proof of Remark 1

Proof. The solution to the following optimization problem is $H(x, x') = (1/Z_{x,x'}) \left[Q^{(1)}(x, x') \quad , \cdots , \quad Q^{(k)}(x, x')\right]$ with a normalizing constant $Z_{x,x'} = \sum_{i\in[k]} Q^{(i)}(x, x')$.

$$\begin{aligned} \arg\max_{H} \quad & \mathbb{E}_{I\sim\text{Unif}([k]),(x,x')\sim Q^{(I)}}[\langle I \, , \, \log H(x, x')\rangle] \, , \\ \text{subject to} \quad & \langle \mathbb{1}, H(x, x')\rangle = 1, \text{for all } (x, x') \, . \end{aligned}$$

This follows from the fact that the gradient of the Lagrangian is $Q^{(i)}(x, x')/H_i(x, x') - \mu_{x,x'}$ where $\mu_{x,x'}$ is the Lagrangian multiplier, and setting it to zero gives the desired maximizer. Plugging this back into the objective function, we get that

$$\begin{aligned} & \max_{||H(x,x')||_1=1} \mathbb{E}[\langle I \, , \, \log H(x, x')\rangle] \\ & = \frac{1}{k} \sum_{i\in[k]} \mathbb{E}_{(x,x')\sim Q^{(i)}} \left[\log \frac{Q^{(i)}(x, x')/k}{\sum_{j\in[k]} Q^{(j)}(x, x')/k} \right] \\ & = \frac{1}{k} \sum_{i\in[k]} d_{\text{KL}} \left(Q^{(i)} \, \middle|\middle| \, \frac{\sum_{j\in[k]} Q^{(j)}}{k} \right) - \log k \\ & = d_{\text{JS}}(Q^{(1)}, \cdots , Q^{(k)}) - \log k \, . \end{aligned}$$

□

D. Implementation Details of Contrastive Latent Sampling

The steps for sampling c^1 and c^2 given a contrastive gap g is as follows:

1. Sample I from Uniform$\{1, 2, ..., k\}$, the index of the fixed latent.

2. Sample the latent value $c_I^1 = c_I^2$ from Uniform$[-1, 1]$.
3. For any other index, j (not I):
   (a) Independently sample c_j^1 and c_j^2 from Uniform$[-1 + g/2, 1 - g/2]$.
   (b) If c_j^1 is greater than c_j^2, add $g/2$ to c_j^1, subtract $g/2$ from c_j^2; else subtract $g/2$ from c_j^1, add $g/2$ to c_j^2.

This ensures that we uniformly sample from the latent space where $c_I^1 = c_I^2$ and $\min_{j \in \{1,2,\ldots,k\} \setminus \{I\}} |c_j^1 - c_j^2| \geq g$.

E. Evaluating Performance

We use the following metrics to evaluate various aspects of the trained latent representation: disentanglement, independence, and generated image quality.

Disentanglement We use the popular disentanglement metric proposed in (Kim & Mnih, 2018). This metric is defined for datasets with known ground truth factors and is computed as follows. First, generate data points where the ith factor is fixed, but the other factors are varying uniformly at random, for a randomly-selected i. Pass each sample through the encoder, and normalize each resulting dimension by its empirical variance. Take the resulting dimension j with the smallest variance. In a setting with perfect disentanglement, the variance in the jth dimension would be 0. Each sample's encoding generates a 'vote' j; we take the majority vote as the final output of the classifier; if the classifier is correct, it should map to i. The disentanglement metric is the error rate of this classifier, taken over many independent trials of this procedure. In our experiment, for each fixed factor index i, we generate 100 groups of images, where each group has 100 images with the same value at the ith index.

One challenge is computing the disentanglement metric for FactorVAE when trained with more latent codes k than there are latent factors (let $\hat{k}$ denote the true number of factors). For instance, (Kim & Mnih, 2018) uses $c \in \mathbb{R}^{10}$ for datasets with only five latent codes. To account for this, (Kim & Mnih, 2018) first removes all latent codes that have collapsed to the prior (i.e., $Q_{c_j|x} = P_{c_j}$); they then use the majority vote on the remaining factors. However, this approach can artificially change the metric if the number of factors for which the posterior does not equal the prior does not equal $\hat{k}$. Hence to measure the metric on FactorVAE (or more generally, cases where $k > \hat{k}$) on 3DTeapots dataset, we first compute the $k \times \hat{k}$ metric matrix, find the maximum value of each row. We then take the top $\hat{k}$ among the k maximum row values, and sum them up.

We additionally compute the (less common) disentanglement metric of (Eastwood & Williams, 2018) (DCI). This metric first requires an estimate of the disentangled code $\tilde{c}$ from samples, for which we use our encoder. Next, we train a regressor f to predict ground truth code $\hat{c}$ from generated code $\tilde{c}$, so $\hat{c} = f(\tilde{c})$. These regressors must also provide a matrix of relative importance R, such that R_{ij} denotes the relative importance of $\tilde{c}_i$ in predicting $\hat{c}_j$. Because of this requirement, Eastwood and Williams propose using regressors that provide importance scores, such as LASSO and random forests. These restrictions limit the generality of the metric; nonetheless, we include it for completeness. Given the relative importance R, a disentanglement score can be computed (see (Eastwood & Williams, 2018) for details).

Eastwood and Williams disentanglement metric is computed using the random forest regressor (Eastwood & Williams, 2018)[2], as implemented in the `scikit-learn` library[3].For dSprites experiments, we use default values for all parameters, except for the `max_depth` parameter for which we use the values: 4, 2, 4, 2, and 2, for the latent factors: shape, scale, rotation, x-position, and y-position respectively, as used by the IB-GAN paper (Jeon et al., 2018) (as per a private communication with its authors). For 3DTeapots experiments, we use the default cross-validation version of random forest regressor.

The other metrics we compute are SAP (Kumar et al., 2017), Explicitness (Ridgeway & Mozer, 2018), Modularity (Ridgeway & Mozer, 2018), MIG (Chen et al., 2018), and BetaVAE (Higgins et al., 2016).

d-Variable Hilbert-Schmidt Independence Score (dHSIC) The dHSIC score is an empirical, kernel-based measure of the total correlation of a multivariate random variable from samples (Pfister et al., 2018). It is used in (Lopez et al., 2018) to enforce independence among latent factors. We use this metric to understand whether and how disentanglement is correlated with total correlation. Suppose we have $c \in \mathbb{R}^k$. We want to compute the distance of the distribution P_c from the product

[2]https://github.com/cianeastwood/qedr/blob/master/quantify.ipynb
[3]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

distribution of the marginals $\prod_i P_{c_i}$. The dHSIC score over n samples is computed as follows. Consider a Gaussian kernel (Aizerman, 1964) with a median heuristic for bandwidth:

$$K_h(x_i, x_j) \;=\; e^{-\frac{\|x_i - x_j\|}{h^2}} \;,$$

where $h = \mathrm{Median}(\{\|x_i - x_j\|\}_{i \neq j})$. When there are k latent codes $c_1, \ldots, c_k$ and n samples, we have

$$\begin{aligned} d\mathrm{HSIC}_n \;&=\; \frac{1}{n^2} \sum_{i,j \in [n]} \left(\prod_{\ell=1}^{k} K_{h_\ell}(c_{\ell i}, c_{\ell j}) \right) \\ &+\; \frac{1}{n^{2k}} \prod_{\ell=1}^{k} \sum_{i,j \in [n]} K_{h_\ell}(c_{\ell i}, c_{\ell j}) \;-\; \frac{2}{n^{k+1}} \sum_{i \in [n]} \prod_{\ell=1}^{k} \sum_{j} K_{h_\ell}(c_{\ell i}, c_{\ell j}) \;. \end{aligned}$$

Image Quality *Inception score* was first proposed in (Salimans et al., 2016) for labelled data, and is computed as $\exp(\mathbb{E}_x d_{\mathrm{KL}}(p(y|\boldsymbol{x})||p(y)))$, where d_{KL} denotes the Kullback-Liebler divergence of two distributions. The distribution $p(y|\boldsymbol{x})$ was originally designed to be used with the Inception network (Szegedy et al., 2016), but we instead use a pre-trained classifier for the dataset at hand. Notice that this metric does not require any information about the disentangled representation of a sample. Inception score is widely used in the GAN literature to evaluate data quality.

Intuitively, inception score measures a combination of two effects: mode collapse and individual sample quality. To tease apart these effects, we compute two additional metrics. The first is *reverse KL-divergence*, proposed in (Srivastava et al., 2017) to measure mode collapse in labelled data. It measures the KL-divergence between the generated label distribution and the true distribution. The second is *classifier confidence*, which we use as a proxy for sample quality. Classifier confidence is measured as the max of the softmax layer of a pre-trained classifier; the higher this value, the more confident the classifier is in its output, which suggests the image quality is better.

F. dSprites Dataset

We begin with the synthetic dSprites dataset (Matthey et al., 2017), commonly used to numerically compare disentangling methods. The dataset consists of 737,280 binary 64 $\times$ 64 images of 2D shapes generated from five ground truth independent latent factors: color, shape, scale, rotation, x and y position. All combinations of latent factors are present in the dataset; some examples are illustrated in Figure 10. Figure 1 illustrates the latent traversal for InfoGAN-CR. To generate this figure, we fix all latent factors except one c_i, and vary c_i from -1 to 1 in evenly-spaced intervals. We observe that each of the five empirically-learned factors captures one true underlying factor, and the traversals span the full range of possibilities for each hidden factor.

Figure 10: Example images from the dSprites dataset.

F.1. Details of Table 1 and Table 3

In Table 5 (combining Table 1 and Table 3), we show our main result of how InfoGAN-CR performs both (a) when hyper-parameters are tuned via supervised selection with oracle access to a synthetic dataset generated with ground truth disentangled latent codes available to us, and (b) with model selected with our proposed ModelCentrality. We use the following popular metrics: FactorVAE (Kim & Mnih, 2018), DCI (Eastwood & Williams, 2018), SAP (Kumar et al., 2017), Explicitness (Ridgeway & Mozer, 2018), Modularity (Ridgeway & Mozer, 2018), MIG (Chen et al., 2018), and BetaVAE (Higgins et al., 2016). For baseline scores, † indicates reported scores we copy from (Jeon et al., 2018), ‡ from (Chen et al., 2018), * from (Kumar et al., 2017), * from (Ansari & Soh, 2018), and $^\triangle$ from (Kim & Mnih, 2018). All the rest of the scores are run by us. The DCI metric uses random forest whose parameter is selected according to IB-GAN (Jeon et al., 2018). DCI metric of some FactorVAE runs has NaN values, which we ignore when computing the mean and standard error. UDR and our model selection randomly select 80% of other models for computing cross-metrics; and the results are averaged by 100 such random selection trials.

	Model	FactorVAE	DCI	SAP	Explicitness	Modularity	MIG	BetaVAE
VAE	VAE	$0.63 \pm 0.06^{\dagger}$	$0.30 \pm 0.10^{\dagger}$				$0.10^{\ddagger}$	
	β-TCVAE	$0.62 \pm 0.07^{\dagger}$	$0.29 \pm 0.10^{\dagger}$				$0.45^{\ddagger}$	
	HFVAE	$0.63 \pm 0.08^{\dagger}$	$0.39 \pm 0.16^{\dagger}$					
	β-VAE	$0.63 \pm 0.10^{\dagger}$	$0.41 \pm 0.11^{\dagger}$	$0.55^{\star}$			$0.21^{\ddagger}$	
	CHyVAE	0.77^{*}						
	DIP-VAE			$0.53^{\star}$				
	FactorVAE	$0.82^{\triangle}$					$0.15^{\ddagger}$	
	FactorVAE (1.0)	0.79 ± 0.01	0.67 ± 0.03	0.47 ± 0.03	0.78 ± 0.01	0.79 ± 0.01	0.27 ± 0.03	0.79 ± 0.02
	FactorVAE (10.0)	0.83 ± 0.01	0.70 ± 0.02	0.57 ± 0.00	0.79 ± 0.00	0.79 ± 0.00	0.40 ± 0.01	0.83 ± 0.01
	FactorVAE (20.0)	0.83 ± 0.01	0.72 ± 0.02	0.57 ± 0.00	0.79 ± 0.00	0.79 ± 0.01	0.40 ± 0.01	0.85 ± 0.00
	FactorVAE (40.0)	0.82 ± 0.01	0.74 ± 0.01	0.56 ± 0.00	0.79 ± 0.00	0.77 ± 0.01	0.43 ± 0.01	0.84 ± 0.01
	FactorVAE (UDR Lasso)	0.81 ± 0.00	0.70 ± 0.01	0.56 ± 0.00	0.79 ± 0.00	0.78 ± 0.00	0.40 ± 0.00	0.84 ± 0.00
	FactorVAE (UDR Spearman)	0.79 ± 0.00	0.73 ± 0.00	0.53 ± 0.01	0.79 ± 0.00	0.77 ± 0.00	0.40 ± 0.01	0.79 ± 0.00
	FactorVAE (our model selection)	0.84 ± 0.00	0.73 ± 0.01	0.58 ± 0.00	0.80 ± 0.00	0.82 ± 0.00	0.37 ± 0.00	0.86 ± 0.00
GAN	InfoGAN	$0.59 \pm 0.70^{\dagger}$	$0.41 \pm 0.05^{\dagger}$				$0.05^{\ddagger}$	
	IB-GAN	$0.80 \pm 0.07^{\dagger}$	$0.67 \pm 0.07^{\dagger}$					
	InfoGAN (modified)	0.82 ± 0.01	0.60 ± 0.02	0.41 ± 0.02	0.82 ± 0.00	0.94 ± 0.01	0.22 ± 0.01	0.87 ± 0.01
	InfoGAN-CR	0.88 ± 0.01	0.71 ± 0.01	0.58 ± 0.01	0.85 ± 0.00	0.96 ± 0.00	0.37 ± 0.01	0.95 ± 0.01
	InfoGAN-CR (UDR Lasso)	0.86 ± 0.01	0.68 ± 0.01	0.49 ± 0.01	0.84 ± 0.00	0.96 ± 0.00	0.30 ± 0.01	0.92 ± 0.01
	InfoGAN-CR (UDR Spearman)	0.84 ± 0.01	0.67 ± 0.01	0.53 ± 0.01	0.84 ± 0.00	0.96 ± 0.00	0.31 ± 0.01	0.90 ± 0.01
	InfoGAN-CR (our model selection)	$\mathbf{0.92 \pm 0.00}$	$\mathbf{0.77 \pm 0.00}$	$\mathbf{0.65 \pm 0.00}$	$\mathbf{0.87 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.45 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$

Table 5: Contrastive regularizer, together with ModelCentrality for model selection, achieves significant improvements over baseline approaches on benchmark datasets and on popular disentanglement metrics.

F.2. Discussion about the Reproducibility of Table 1

An earlier version of out paper reports a slightly higher scores for both InfoGAN (modified) and InfoGAN-CR than the current Table 1 (e.g. the FactorVAE score for InfoGAN (modified) and InfoGAN-CR were 0.83 ± 0.03 and 0.90 ± 0.01). These results were averaged over 12 runs, which is not large enough. Later, we observed that rarely (but with a positive probability) the trained model has very bad disentanglement performance during the training of InfoGAN, and does not recover after CR is introduced. To make our results reproducible by anyone who uses our code from our github repo, we ran fresh 50 runs with exactly the same code and hyper-parameters. This is reported in the current Table 1 and Table 5. Also, the saved models of these 50 runs are available in our github repo, to help anyone verify the reproducibility of our experiments. The histograms of FactorVAE scores of InfoGAN (modified) and InfoGAN-CR are in Figure 11 and Figure 12.

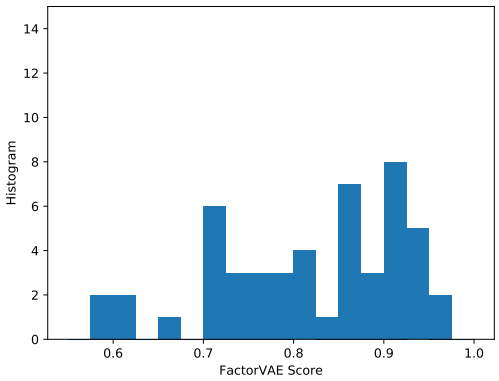

Figure 11: Histogram of FactorVAE scores of InfoGAN (modified).

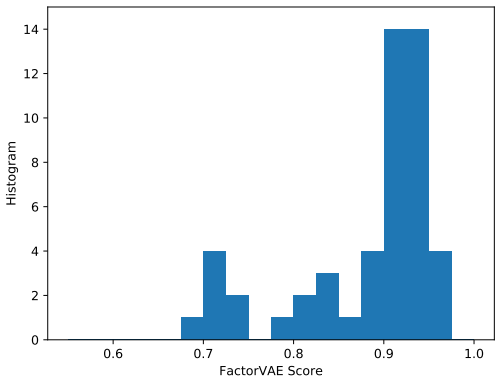

Figure 12: Histogram of FactorVAE scores of InfoGAN-CR.

F.3. Parameter Exploration

To better understand the parameter exploration in Figure 6, we generate similar plots, except representing image quality by mode-reversed KL-divergence (Figure 13) and classifier confidence (Figure 14).

For both reverse KL-divergence and classifier confidence, we observe similar trends to Figure 6; increasing α improves disentanglement to a point, whereas it appears to hurt both image quality metrics. One observation is that for $\alpha \in \{0, 1\}$, there is little noticeable change in the either image quality metric. This suggests that CR does not introduce mode collapse or substantial reductions in image quality for small α.

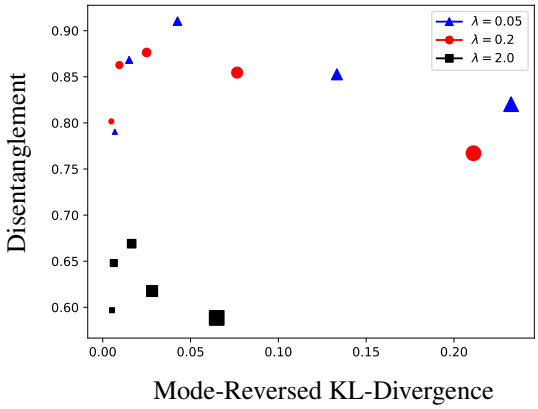

Figure 13: Achievable (mode-reversed KL-divergence, disentanglement) pairs for InfoGAN-CR, where disentanglement is measured as in (Kim & Mnih, 2018). We vary the contrastive regularizer α and the InfoGAN regularizer. The size of each point denotes α, ranging from smallest to largest for $\alpha \in \{0, 1, 2, 4, 8\}$.

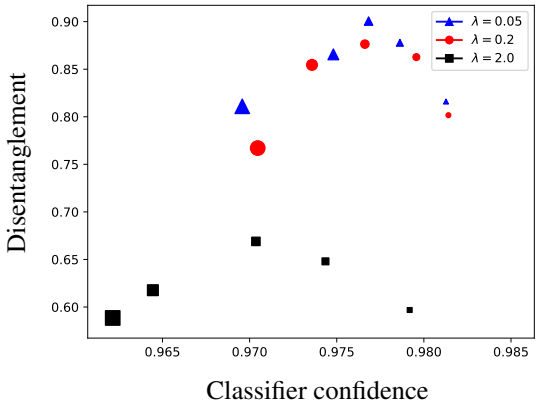

Figure 14: Achievable (classifier confidence, disentanglement) pairs for InfoGAN-CR, where disentanglement is measured as in (Kim & Mnih, 2018). We vary the contrastive regularizer α and the InfoGAN regularizer. The size of each point denotes α, ranging from smallest to largest for $\alpha \in \{0, 1, 2, 4, 8\}$.

F.4. Total Correlation

We claim that contrastive regularization is tailored to work well with GAN architectures. Similarly, we claim that total correlation regularization of FactorVAE is specifically tailored for VAE. To test this hypothesis, we have applied contrastive regularization (CR) to FactorVAE and total correlation (TC) regularization to InfoGAN-CR. Figure 15 left panel shows the disentanglement metric of each as a function of batch numbers. For FactorVAE, we introduce CR regularization of $\alpha = 20$ at batch 300,000. Note that the metric does not change perceptibly after adding CR. For InfoGAN-CR, we ran one set of trials with TC regularization from the beginning (red curve) and one set of trials without TC regularization (green curve). We use InfoGAN regularizer $\lambda = 0.7$ and TC coefficient $\beta = 1$ for the former. Notice that InfoGAN-CR has a lower disentanglement score than FactorVAE in this plot because we did not use the optimal λ for this dataset; this sensitivity is a weakness of InfoGAN-CR (as well as InfoGAN). In Figure 15, we observe that TC regularization actually reduces disentanglement compared to InfoGAN-CR without TC. The jumps in disentanglement for the InfoGAN-CR curves are due to progressive training; we change the contrastive gap from 1.9 to 0.0 at batch 120,000. The red line (InfoGAN-CR + TC) is averaged over 4 runs, the blue line (FactorVAE + CR) over 2 runs, and the green line (InfoGAN-CR) over ten. These results support (but do not prove) our hypothesis that CR is better-suited to GAN architectures, whereas TC is better-suited to VAE architectures. To further confirm this intuition, we show that disentanglement appears negatively correlated with a measure

of total correlation in the following.

To explore the relation between total correlation and disentanglement, Figure 15 plots the disentanglement score of (Kim & Mnih, 2018) as a function of dHSIC score while varying α for InfoGAN-CR. Each point represents a single model, and point size/color signifies the value of $\alpha \in \{0, 1, 2, 4, 8\}$. Larger points denote larger α. Since dHSIC approximates the total correlation between the latent codes, a lower dHSIC score implies a lower total correlation. Perhaps surprisingly, we find a noticeable positive correlation between dHSIC score and disentanglement. This suggests that TC regularization (i.e., encouraging small TC) actually hurts disentanglement for InfoGAN-CR. This may help to explain, or at least confirm, the findings in Figure 15, which show that adding TC regularization to InfoGAN-CR reduces the disentanglement score.

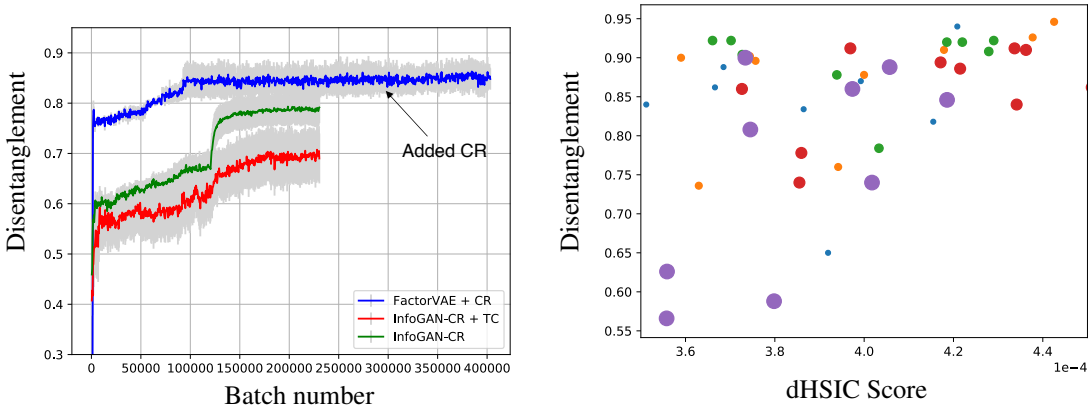

Figure 15: [left panel] Adding TC regularization to InfoGAN-CR does not improve disentanglement; neither does adding CR to FactorVAE. [right panel] (dHSIC score, disentanglement) pairs for various InfoGAN-CR models. Each point represents a single model, and point size/color signifies the value of $\alpha \in \{0, 1, 2, 4, 8\}$. Larger points denote larger α. Disentanglement is measured as in (Kim & Mnih, 2018). We observe a positive correlation between disentanglement and dHSIC score, suggesting that TC regularization does not help GANs disentangle better.

F.5. Contrastive Regularizer without InfoGAN Regularizer

To test if InfoGAN regularizer is necessary, we trained dSprites dataset without InfoGAN regularizer (i.e. $\lambda = 0$) and progressively increasing α. This new loss suffers from significant mode collapse, which can be significantly reduced with a recent technique for mitigating mode collapse known as PacGAN introduced in (Lin et al., 2017). The main idea is to life the adversarial discriminator to take m samples packed together, all from either real data or generated data. This provably introduces an implicit inductive bias towards penalizing mode collapse, which is mathematically defined in (Lin et al., 2017), in terms of binary hypothesis testing and type I and type II errors. The resulting metric are shown in Figure 16, where even with PacGAN we do not get the desired level of disentanglement without InfoGAN regularizer. We believe that InfoGAN and Contrastive regularizers play complementary roles in disentangling GANs.

F.6. Implementation Details

In dSprites experiments, we used a convolutional neural network for the FactorVAE encoder, InfoGAN discriminator, and CR discriminator, and a deconvolutional neural network for the decoder, and a multi-layer perceptron for total correlation discriminators. We used the Adam optimizer for all updates, whose learning rates are described below. For unmodified InfoGAN, we used the architecture described in Table 5 of (Kim & Mnih, 2018). This architecture is reproduced in Table 6 for completeness. As mentioned in Section 2, we make several changes to the training of InfoGAN to improve its disentanglement, including changing the Adam learning rate to 0.001 for the generator and 0.002 for the InfoGAN and CR discriminators (β_1 is still 0.5). The architectural changes are included in Table 7. We include in Table 8 the architecture of our CR discriminator, which is similar to the InfoGAN discriminator. Finally, Table 10 contains the architecture of FactorVAE, reproduced from (Kim & Mnih, 2018). We use a batch size of 64 for all experiments.

We implemented both FactorVAE and InfoGAN using the architectures described in (Kim & Mnih, 2018). Although

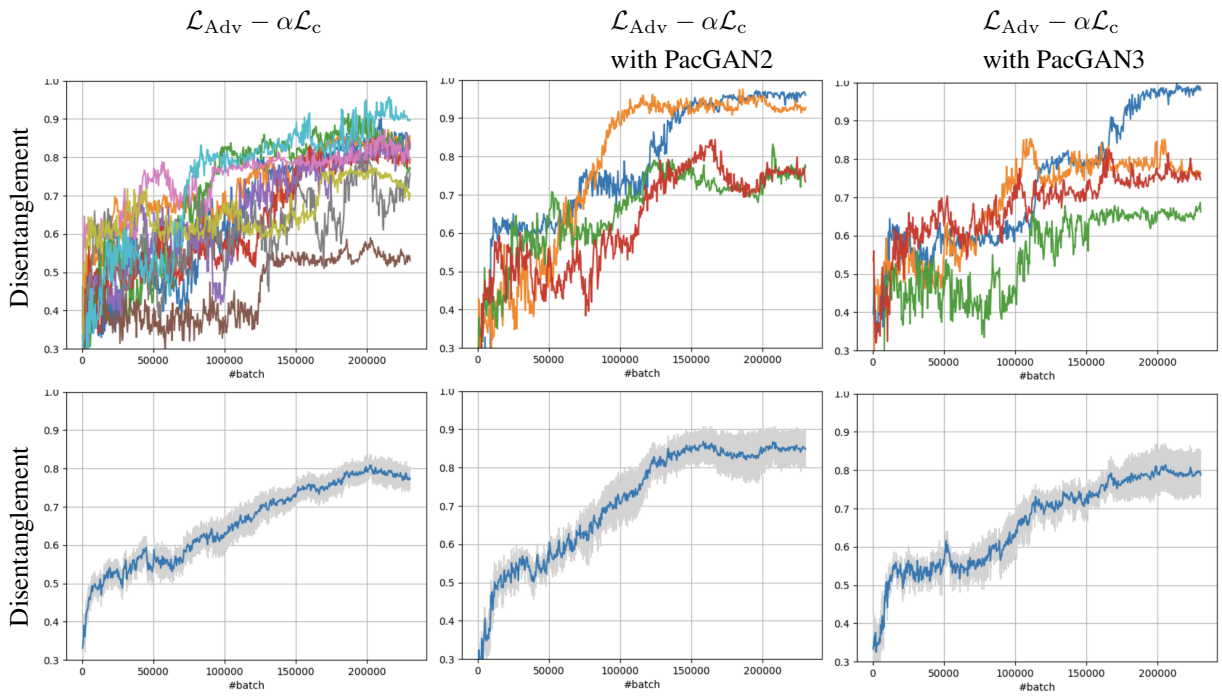

Figure 16: Training without InfoGAN loss suffers from severe mode collapse, which in turn results in poor disentanglement score (left). With PacGAN2 discriminator that takes 2 packed samples together each time, the mode collapse significantly decreases and disentanglement also improves (middle). With PacGAN3 discriminator that takes 3 packed samples together each time, the higher dimensionality of those packed samples results in poor performance (right). The training trajectory of each instance is shown on the top, and the average is shown on the bottom.

InfoGAN exhibits a reported disentanglement score of 0.59 ± 0.70 in (Kim & Mnih, 2018), we find that InfoGAN can exhibit substantially higher disentanglement scores (0.83 ± 0.03) through some basic changes to the architecture and loss function. In particular, in accordance with (Miyato et al., 2018), we changed the loss function from Wasserstein GAN to the traditional JSD loss. We also changed the generator's Adam learning rate to 0.001 and the InfoGAN and CR discriminators learning rates to 0.002; we used 5 continuous input codes, whereas (Kim & Mnih, 2018) reported using four continuous codes and one discrete one. We also used batch normalization in the generator, and spectral normalization in the discriminator. The effects of these changes are shown in the line 'InfoGAN (modified)' in Table 1. The progressive scheduling of InfoGAN-CR is to run InfoGAN ($\alpha = 0, \lambda = 0.05$) for 288000 batches, and then continue running it with CR ($\alpha = 2.0, \lambda = 0.05$, gap=0) for additional 34560 batches (so that the total number of epoches is 28). For FactorVAE, we used the architecture of (Kim & Mnih, 2018), which uses $k = 10$ latent codes in their architecture.

G. 3DTeapots Dataset

We ran InfoGAN-CR on the 3DTeapots dataset from (Eastwood & Williams, 2018), with images of 3DTeapots in various orientations and colors generated by the renderer in (Moreno et al., 2016). Images have five latent factors: color (red, blue, and green), rotation (vertical), and rotation (lateral). Colors are randomly drawn from $[0, 1]$. Rotation (vertical) is randomly drawn from $[0, \pi/2]$. Rotation (lateral) is drawn from $[0, 2\pi]$. We generated a dataset of 200,000 such images with each combination of latent factors represented. Table 2 shows the disentanglement scores of FactorVAE and InfoGAN compared to InfoGAN-CR. Since the 3DTeapots dataset does not have classes, we do not compute inception score for this dataset; however, the images generated by InfoGAN-CR appear sharper than those generated by FactorVAE.

We now discuss implementation details and additional experiments on the 3DTeapots dataset. We used an identical architecture to the dSprites dataset for InfoGAN and FactorVAE. For InfoGAN-CR, the CR discriminator architecture is changed slightly and is listed in Table 9. As with dSprites, FactorVAE used 10 latent codes, so we chose the best five to compute the disentanglement metric.

Discriminator D **/ Encoder** Q	**Generator** G
Input 64×64 binary image	Input $\in \mathbb{R}^{10}$
4×4 conv. 32 lReLU. stride 2	FC. 128 ReLU. batchnorm
4×4 conv. 32 lReLU. stride 2. batchnorm	FC. $4 \times 4 \times 64$ ReLU. batchnorm
4×4 conv. 64 lReLU. stride 2. batchnorm	4×4 upconv. 64 lReLU. stride 2. batchnorm
4×4 conv. 64 lReLU. stride 2. batchnorm	4×4 upconv. 32 lReLU. stride 2. batchnorm
FC. 128 lReLU. batchnorm (*)	4×4 upconv. 32 lReLU. stride 2. batchnorm
From *: FC. 1 sigmoid. (output layer for D)	4×4 upconv. 1 sigmoid. stride 2
From *: FC. 128 lReLU. batchnorm. FC 5 for Q	

Table 6: InfoGAN architecture for dSprites experiments from (Kim & Mnih, 2018). We used 5 continuous codes and 5 noise variables.

Discriminator D **/ Encoder** Q	**Generator** G
Input 64×64 binary (dSprites) or color (3DTeapots) image	Input $\in \mathbb{R}^{10}$
4×4 conv. 32 lReLU. stride 2. spectral normalization	FC. 128 ReLU. batchnorm
4×4 conv. 32 lReLU. stride 2. spectral normalization	FC. $4 \times 4 \times 64$ ReLU. batchnorm
4×4 conv. 64 lReLU. stride 2. spectral normalization	4×4 upconv. 64 lReLU. stride 2. batchnorm
4×4 conv. 64 lReLU. stride 2. spectral normalization	4×4 upconv. 32 lReLU. stride 2. batchnorm
FC. 128 lReLU. spectral normalization (*)	4×4 upconv. 32 lReLU. stride 2. batchnorm
From *: FC. 1 sigmoid. (output layer for D)	4×4 upconv. 1 (dSprites) or 3 (3DTeapots) sigmoid. stride 2
From *: FC. 128 lReLU. spectral normalization	
FC 5. spectral normalization (output layer for Q)	

Table 7: InfoGAN (modified) architecture for dSprites and 3DTeapots experiments. We used 5 continuous codes and 5 noise variables.

For InfoGAN-CR, we train InfoGAN with $\lambda = 0.2$ for 50000 batches, and then InfoGAN-CR with $\lambda = 0.2$, $\alpha = 3.0$, gap=1.9 for 35000 batches, and then InfoGAN-CR with $\lambda = 0.2$, $\alpha = 3.0$, gap=0.0 for 40000 batches. We use a batch size of 64 for all experiments.

We illustrate a latent traversal for the 3DTeapots dataset under InfoGAN-CR in Figure 18, which is from the best run of InfoGAN-CR with disentanglement metric of (Kim & Mnih, 2018) 1.0. To make the traversal easier to interpret, we have separated the color channels for each latent factor that captures color. This shows that each latent factor controls a single color channel, while the others are held fixed. A similar traversal is shown in Figure 19 for FactorVAE, which is from the best run of FactorVAE with $\beta = 40$ and disentanglement metric of (Kim & Mnih, 2018) 0.94. Although it is difficult to draw conclusions from qualitative comparison, we found that the sharpness of images was reduced in the FactorVAE images, though FactorVAE is able to learn a meaningful disentanglement with three color factors and three (one duplicate) rotation factors.

Building on Table 2, we also plot the disentanglement metric of (Kim & Mnih, 2018) during the training of InfoGAN-CR and FactorVAE in Figure 17. This plot shows that InfoGAN-CR achieves a consistently higher disentanglement score than FactorVAE throughout the training procedure, though FactorVAE comes close when $\beta = 40$.

CR Discriminator
Input $64 \times 64 \times 2$ (2 binary images)
4×4 conv. 32 lReLU. stride 2. spectral normalization
4×4 conv. 32 lReLU. stride 2. spectral normalization
4×4 conv. 64 lReLU. stride 2. spectral normalization
4×4 conv. 64 lReLU. stride 2. spectral normalization
FC. 128 lReLU. spectral normalization
FC 5. softmax

Table 8: CR discriminator architecture for dSprites experiments.

CR Discriminator
Input $64 \times 64 \times 6$ (2 color images)
4×4 conv. 32 lReLU. stride 2.
4×4 conv. 32 lReLU. stride 2. batchnorm
4×4 conv. 64 lReLU. stride 2. batchnorm
4×4 conv. 64 lReLU. stride 2. batchnorm
FC. 128 lReLU. batchnorm
FC 5. softmax

Table 9: CR discriminator architecture for 3DTeapots experiments.

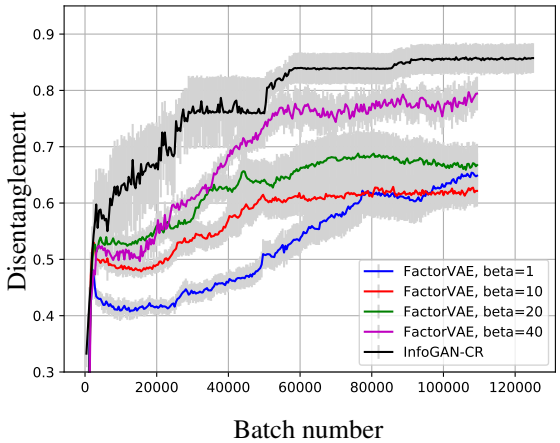

Figure 17: Disentanglement metric of (Kim & Mnih, 2018) as a function of batch number for InfoGAN-CR and FactorVAE on the 3DTeapots dataset. The final models from this training were used to generate Table 2.

Encoder	Decoder
Input 64×64 binary image	Input $\in \mathbb{R}^{10}$
4×4 conv. 32 ReLU. stride 2	FC. 128 ReLU.
4×4 conv. 32 ReLU. stride 2.	FC. $4 \times 4 \times 64$ ReLU.
4×4 conv. 64 ReLU. stride 2.	4×4 upconv. 64 ReLU. stride 2.
4×4 conv. 64 ReLU. stride 2.	4×4 upconv. 32 ReLU. stride 2.
FC. 128.	4×4 upconv. 32 ReLU. stride 2.
FC. 2×10.	4×4 upconv. 1. stride 2

Table 10: FactorVAE architecture for dSprites and 3DTeapots experiments, taken from (Kim & Mnih, 2018).

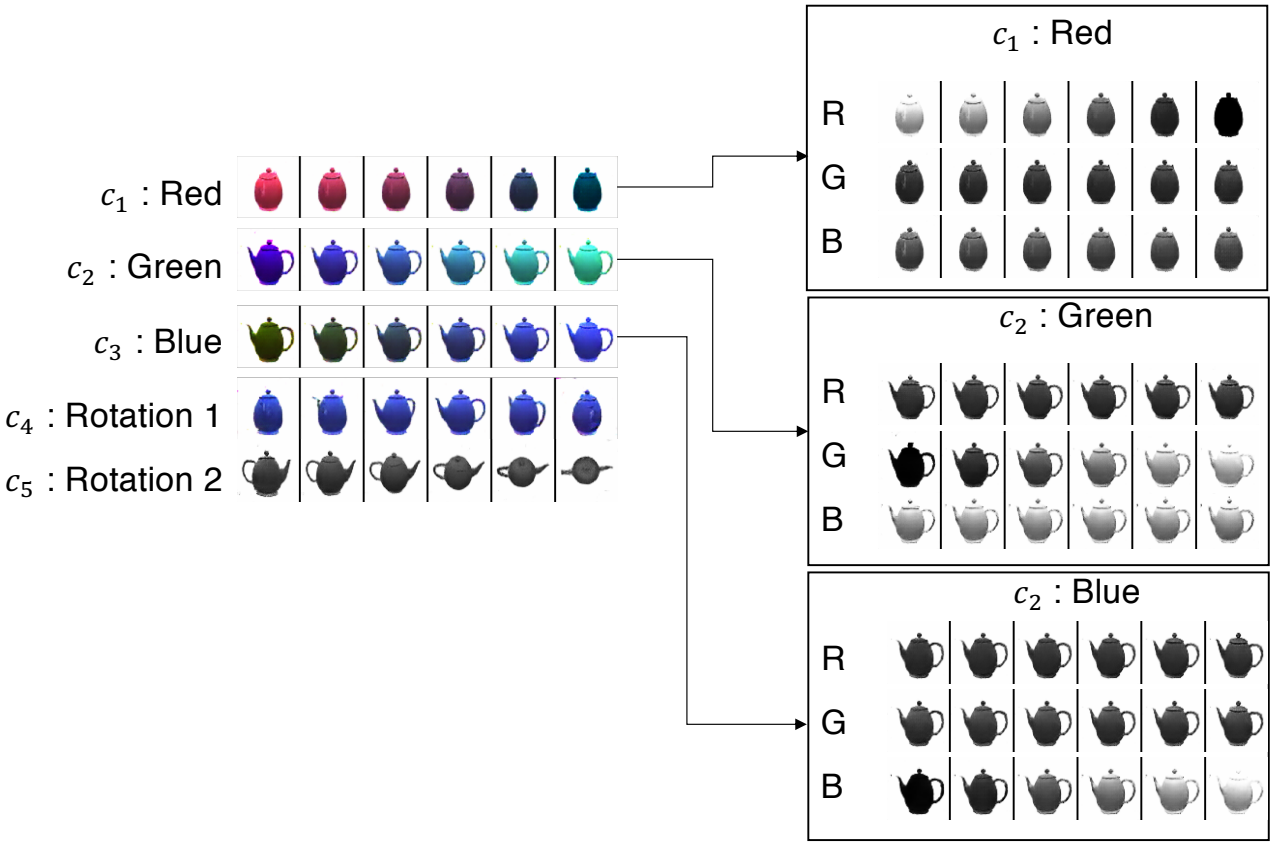

Figure 18: Latent traversal for 3DTeapots dataset with InfoGAN-CR. Red (R), blue (B), and green (G) channels are shown separately for the latent factors that correspond to color.

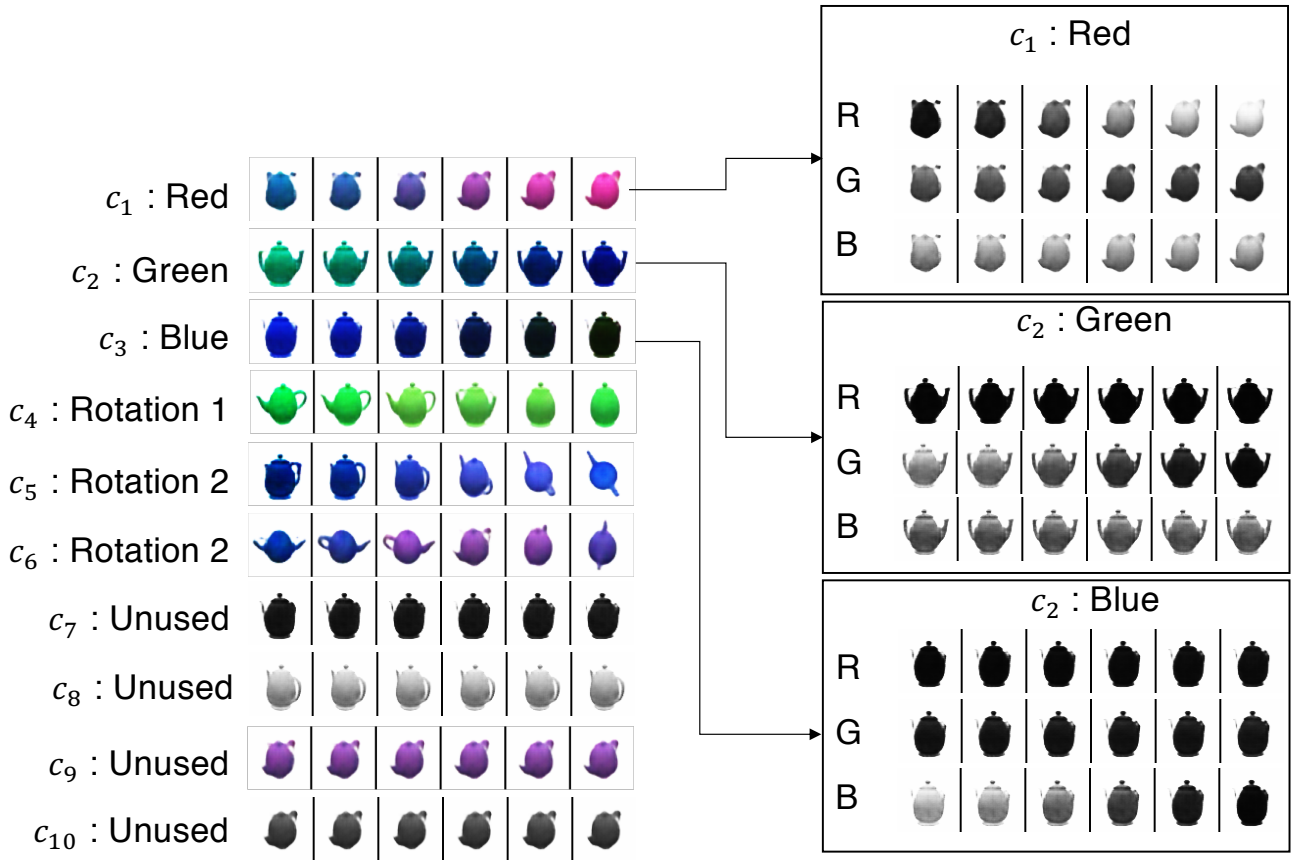

Figure 19: Latent traversal for 3DTeapots dataset with FactorVAE. Red (R), blue (B), and green (G) channels are shown separately for the latent factors that correspond to color.

G.1. Non-commutative and Ambiguous Coordinate Systems

To further push the boundaries of disentangled generation, it is important to understand when and why InfoGAN-CR fails; the 3DTeapots dataset gives a useful starting point. In particular, (Higgins et al., 2018a) observed that rotations about the canonical basis axes are not commutative. For example, suppose we apply two rotations of some angles, one about the x and one about the y axes in two different orders; in general, the resulting orientations of the object need not be the same.

Because of this, a disentangled GAN or VAE trained on a dataset where every possible orientation of teapot is represented should *not* recover the canonical basis for 3D space. Despite this fact, existing experiments (including our own) appear to recover the canonical axes of rotation. To explain this phenomenon, we observe that existing experiments on this dataset, including (Eastwood & Williams, 2018) and our own initial experiments, do not include all orientations of the object in the training data. For example, notice that none of the visualized images in Figure 18 show the bottom of the teapot. Because of the way the training data is selected, the canonical axes are indeed recovered.

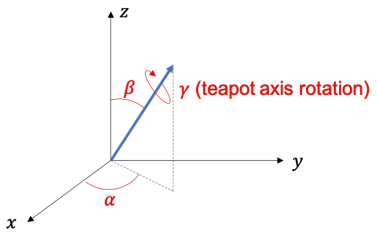

Figure 20: A commutative 3D rotational coordinate system.

We next ask what would happen if all orientations were present in the dataset. To answer this question, first note that in general, 3D rotations can indeed commute. For example, the rotational coordinate system described in Figure 20 commutes. In that coordinate system, γ represents the rotation along the axis of the teapot (orthogonal to the bottom of the teapot), while α and β represent the orientation of the teapot's principal axis. However, this coordinate system is not unique; we could choose a different z-axis, for instance, and construct a different commutative rotational coordinate system. Hence, even if we were to represent all orientations of the teapots in our training data, it is unclear what orientation we would recover.

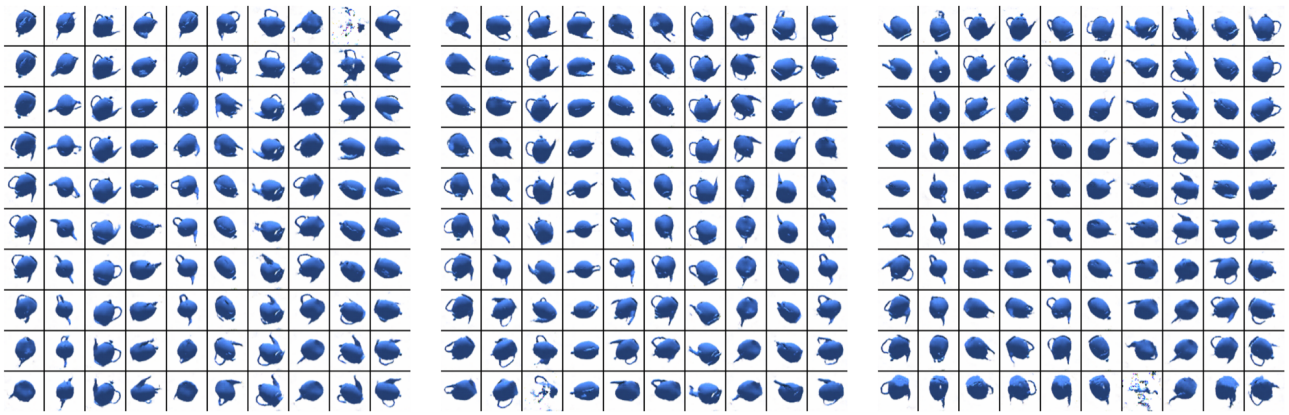

Figure 21: Latent traversal for InfoGAN-CR over the 3DTeapots dataset when every possible orientation of the teapot is included. We find that InfoGAN-CR does appear to learn a consistent coordinate axis, even though there is not a unique disentangled representation.

Figure 21 shows the result from a single trial of an experiment where every possible orientation of the teapot is included in the training data. We trained InfoGAN-CR with InfoGAN coefficient $\lambda = 0.2$ and CR coefficient $\alpha = 1.0$. As with our other experiments, we trained five latent factors and visualize the three most meaningful ones in Figure 21. To our surprise, we find that the system (roughly) recovers a similar coordinate system as the one depicted in Figure 20. Here c_1 appears to capture γ, c_2 appears to recover β, and c_3 recovers α. Even upon running multiple trials of this experiment, the InfoGAN-CR appears to learn (approximately) the same coordinate system from Figure 20. We hypothesize that this happens because of the illumination in the images. By default, the renderer from (Moreno et al., 2016) renders the teapots with an overhead light source. This may distinguish the vertical axis from the others, causing InfoGAN-CR to learn the vertical axis as a reference for the rotational coordinate system.

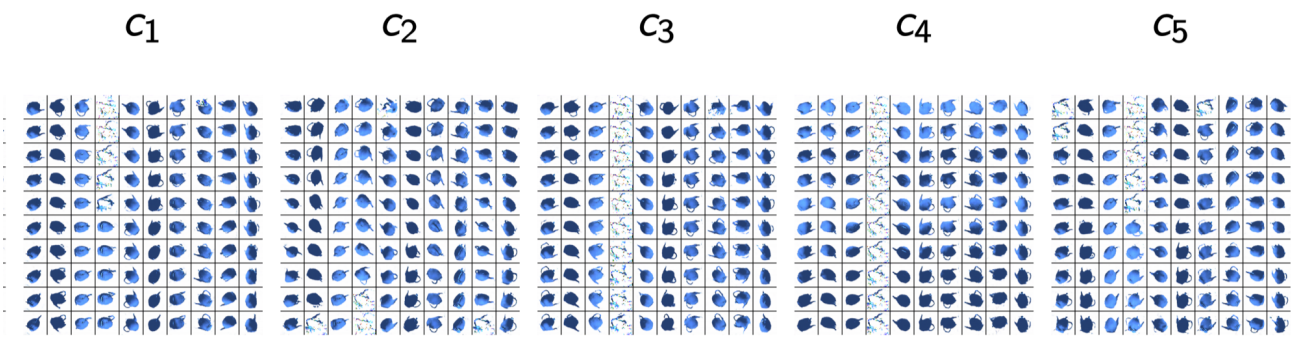

Figure 22: Latent traversal over the 3DTeapots dataset when the light source is chosen uniformly for each training image. We find that InfoGAN-CR does not appear to learn a consistent coordinate axis.

To test this hypothesis, Figure 22 shows an experiment in which we randomized the light source for each image. For this experiment, we again used an InfoGAN coefficient of $\lambda = 0.2$, and a disentangling coefficient of $\alpha = 1.0$; the experimental setup is identical to that of Figure 21, except for the light source in the training data. We find that in this case, InfoGAN-CR no longer appears to learn the vertical coordinate system. Indeed, it does not seem to learn *any* disentangled representation. We observed similar results for FactorVAE for a range of total correlation coefficients on this dataset, so we do not believe this effect is unique to InfoGAN-CR. Instead, it suggests that in settings where there is no single disentangled representation, current disentanglement methods fail.

H. Circular dSprites (CdSprites) Dataset

To further study the failure modes of our InfoGAN-CR and other state-of-the-art architectures for disentangling, we introduce the following experiments. Towards this purpose we generate a new synthetic dataset which we call Circular dSprites (CdSprites).

It contains a set of 1080, 64×64 8-bit gray-scale images generated as follows. Each image has a white (pixel value 255) circular (disc) shape of radius 5 pixels on a black background (pixel value 0). For the placement of the shape we construct a polar (2D) coordinate system, whose co-ordinates are radius $r \in [0, 32]$ and angle $\gamma \in [0, 2\pi)$, and its origin is the center of the image canvas: (32, 32). Then the circular shape is placed on the on a point (r, γ), such that radius r is uniformly selected from $\{0, 1, \ldots, 26\}$ (pixel unit) and angle γ is uniformly selected from $\{0, 2\pi\frac{1}{40}, 2\pi\frac{1}{40}, \ldots, 2\pi\frac{39}{40}\}$. Thus if the center of the circular shape is selected as (r, γ) then it will be place on the pixel $(32 + r\cos\gamma, 32 + r\sin\gamma)$. Thus there are 27×40 (1080) total images in the dataset. Fig. 23(a) shows some sample and their corresponding radius (r) and angle (γ) indices. Fig. 23(b) shows the overlap of all the images in the dataset which shows the circular region where the shape can be placed. We expect that a good disentangling representation should disentangle the radius and angle latent factors.

$r = 0, \gamma = 0$ $r = 26, \gamma = 33$ $r = 21, \gamma = 13$

(a) Samples (radius, angle)

(b) Overlap of all samples

Figure 23: Circular dSprites dataset.

We train FactorVAE, InfoGAN and InfoGAN-CAR models on this dataset. We use the same architecture as the one we use for the dSprites dataset for these models. However, we reduced number of latent factors to 2 for all the models, since there are only two factors to be learned.

In Fig. 24, we show the traversal of the dataset through the true latents. Each row corresponds to one one latent's traversal while the other is fixed. Each column has a different fixed value for the other fixed latent. As we traverse through a latent keeping the other fixed, for easy visualization, we increase the shade of the shape from the darkest to the brightest. In

Figs. 25, 26, and 27, where show the traversal through the learned latents of FactorVAE, InfoGAN, and InfoGAN-CR respectively. We see that the none of the models truly disentangle the true radius and angle factors and in fact they are mixed in the learned learned latents. We believe this dataset is hard for any current models to disentangle, and thus could be used as good baseline for future research.

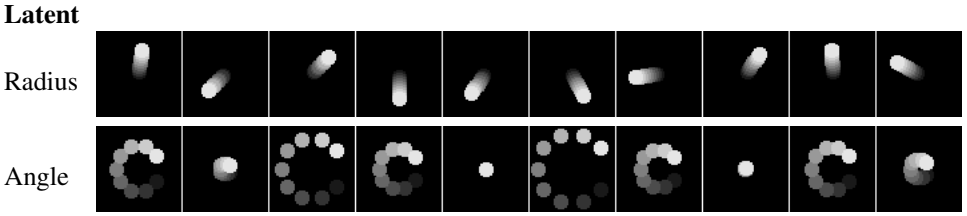

Figure 24: (CdSprites) True latent traversal (see Appendix H for explanation). We see that the top row changes position radially and the bottom row changes the position angluarly.

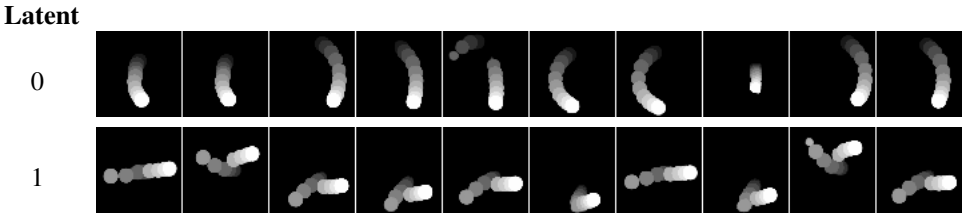

Figure 25: (CdSprites) FactorVAE latent traversal (see Appendix H for explanation). We see that the model does not disentangle the true radius and angle factors and in fact they are mixed.

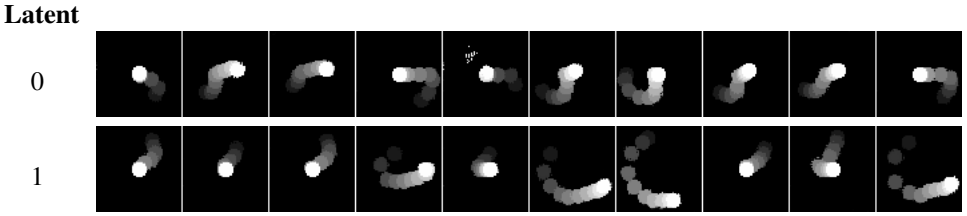

Figure 26: (CdSprites) InfoGAN latent traversal (see Appendix H for explanation). We see that the model does not disentangle the true radius and angle factors and in fact they are mixed.

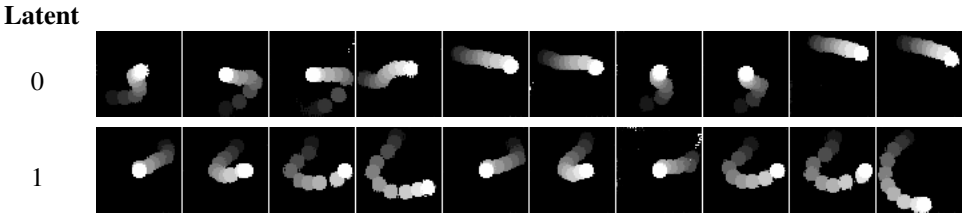

Figure 27: (CdSprites) InfoGAN-CR latent traversal (see Appendix H for explanation). We see that the model does not disentangle the true radius and angle factors and in fact they are mixed.

I. CelebA Dataset

I.1. Implementation Details

We crop the center 128×128 pixels from original CelebA images, and resize the images to 32 × 32 for training.

The architecture of generator, InfoGAN discriminator, CR discriminator in this experiment are shown in Table 11 and Table 12, which are in part motivated by an independent implementation of InfoGAN[4]. We used the Adam optimizer for all updates. Generator's learning rate is 2e-3, InfoGAN discriminator's and CR discriminator's learning rates are 2e-4. β_1 is 0.5, batch size is 128 for all components. We train InfoGAN with $\lambda = 2.0$ for 80000 batches, and then train InfoGAN-CR with $\lambda = 2.0$, $\alpha = 1.0$, gap $= 0.0$ for another 10173 batches (so that total number of epoch is 57). The samples are generated from the end of training of one run.

Discriminator D / Encoder Q	**Generator G**
Input 32×32 color image	Input $\in \mathbb{R}^{105}$
5×5 conv. 64 lReLU. stride 2. spectral normalization	FC. 1024 ReLU. batchnorm
5×5 conv. 128 lReLU. stride 2. spectral normalization	FC. $4 \times 4 \times 256$ ReLU. batchnorm
5×5 conv. 256 lReLU. stride 2. spectral normalization	5×5 upconv. 128 ReLU. stride 2. batchnorm
5×5 conv. 512 lReLU. stride 2. spectral normalization (*)	5×5 upconv. 64 ReLU. stride 2. batchnorm
From *: FC. 1 sigmoid. (output layer for D)	5×5 upconv. 3 tanh. stride 2
From *: FC. 5. spectral normalization (output layer for Q)	

Table 11: InfoGAN architecture for CelebA experiments. We used 5 continuous codes and 100 noise variables.

CR Discriminator
Input $32 \times 32 \times 6$ (2 color images)
5×5 conv. 64 lReLU. stride 2.
5×5 conv. 128 lReLU. stride 2. batchnorm
5×5 conv. 256 lReLU. stride 2. batchnorm
5×5 conv. 512 lReLU. stride 2. batchnorm
FC 5. softmax

Table 12: CR discriminator architecture for CelebA experiments.

J. Exploring Choices of CR

In the design of CR, there are 4 choices, based on how the two input images are generated, when assuming gap is always zero:

1. Same noise variables, and same latent codes except one
2. Same noise variables, and random latent codes except one
3. Random noise variables, and same latent codes except one
4. Random noise variables, and random latent codes except one

We tried all four settings in MNIST dataset, using architecture in (Chen et al., 2016), with 1 10-category latent codes, 2 continuous latent codes, and 62 noise variables. We had 8 runs for each setting, and visually inspect whether it correctly disentangle digit, rotation, and width. The disentanglement successful rate are recorded in table 13.

Based on this result, we choose to use option 2 as our starting point of designing CR.

K. Model Centrality

For all figure results of MC and UDR, each model is compared with all other models. For all numerical results of MC and UDR, in order to get more accurate comparison, each model is compared with randomly selected 80% of other models. Such random selection is done 100 times, and the mean and standard error are shown.

[4]https://github.com/conan7882/tf-gans

Choice	Rate of Successful Disentanglement
1	5/8
2	8/8
3	2/8
4	6/8

Table 13: Rate of successful disentanglement using four choices of CR

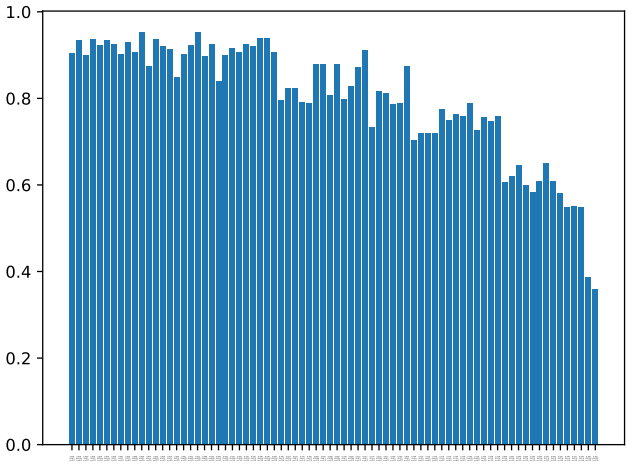

Figure 28: FactorVAE scores of InfoGAN-CR models (dSprites dataset). The models are sorted according to our model selection score.

K.1. Our Model Selection Approach on InfoGAN-CR Models (dSprites Dataset)

We run 8 independent trials for each of the following 9 hyperparameter settings $\{\alpha = 1, 2, 4\} \times \{\lambda = 0.05, 0.2, 2.0\}$, and an additional 4 independent trials of $(\alpha = 2, \lambda = 0.05)$. There are 76 models in total. The progressive scheduling of those 76 runs is the same as supervised experiments: run InfoGAN ($\alpha = 0$) for 288000 batches, and then continue running it with CR (gap=0) for additional 34560 batches (so that the total number of epoches is 28).

Figure 28, 29, 30,31,32,33,34 show the bar plots of supervised metrics, where the models are ordered according to our model selection approach. These figures show that our model selection approach correlates with other supervised metrics well.

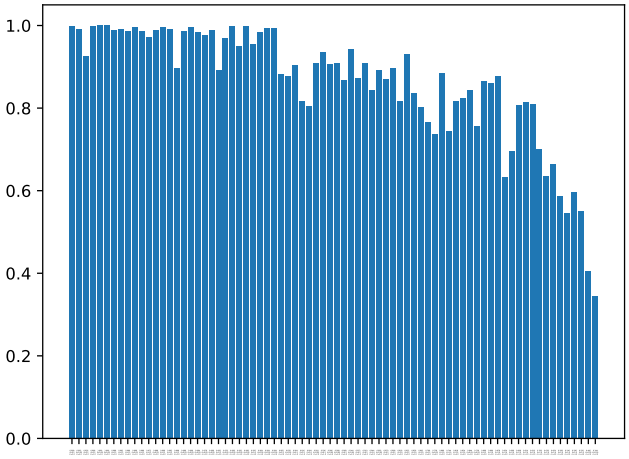

Figure 29: BetaVAE scores of InfoGAN-CR models (dSprites dataset). The models are sorted according to our model selection score.

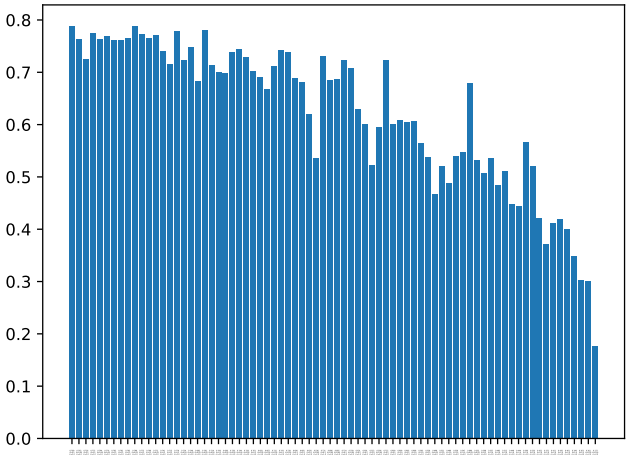

Figure 30: DCI scores of InfoGAN-CR models (dSprites dataset). The models are sorted according to our model selection score.

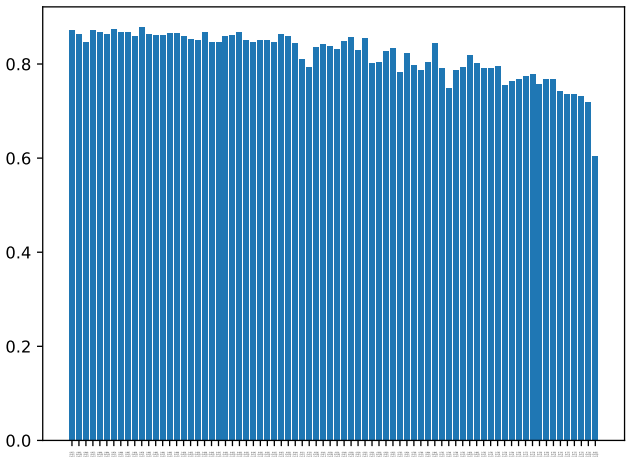

Figure 31: Explicitness scores of InfoGAN-CR models (dSprites dataset). The models are sorted according to our model selection score.

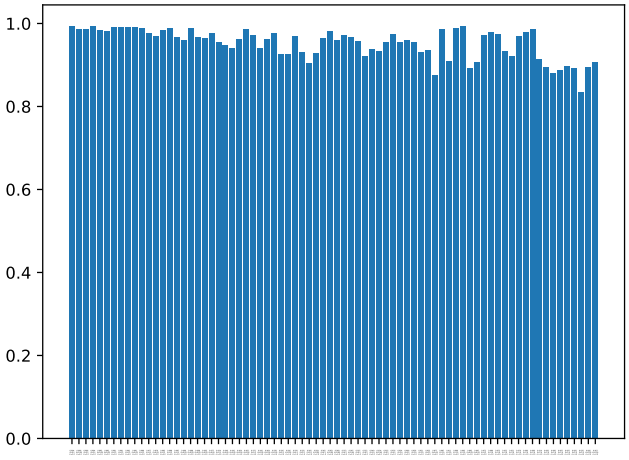

Figure 32: Modularity scores of InfoGAN-CR models (dSprites dataset). The models are sorted according to our model selection score.

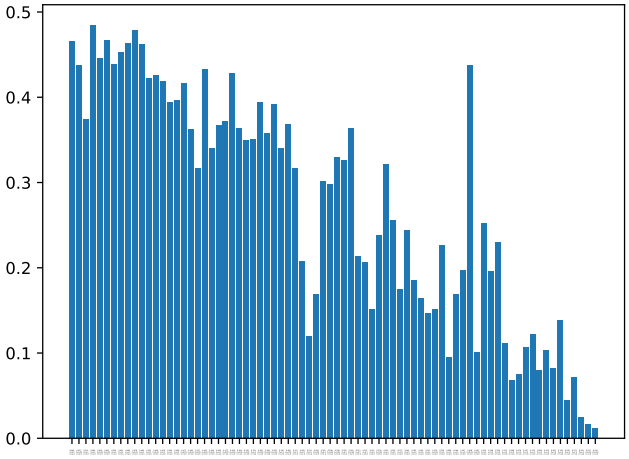

Figure 33: MIG scores of InfoGAN-CR models (dSprites dataset). The models are sorted according to our model selection score.

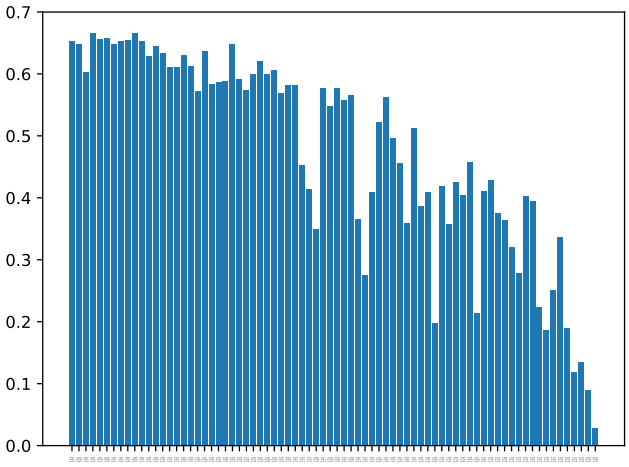

Figure 34: SAP scores of InfoGAN-CR models (dSprites dataset). The models are sorted according to our model selection score.

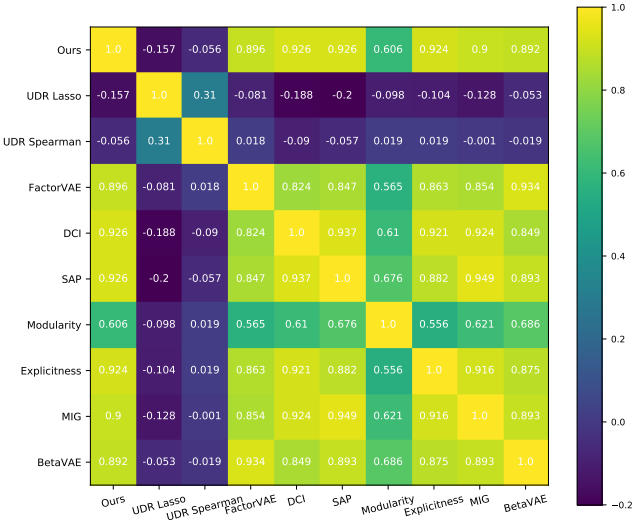

Figure 35: Spearman rank correlation of different metrics on FactorVAE models (dSprites dataset).

K.2. Our Model Selection Approach on FactorVAE Models (dSprites Dataset)

We run 10 independent trials for each of the following 4 hyperparameter settings $tc = \{1, 10, 20, 40\}$. There are 40 models in total. Each model is run for 27 epoches.

Figure 35 shows the correlations between different metrics (including UDR and our model selection approach). It is clear that the score given by our model selection approach is strongly and positively correlated with other supervised metrics. On the other hand, we can see that UDR methods (Lasso or Spearman) are weakly correlated with other metrics. This suggests that our model selection approach is very effective for model selection, whereas UDR performs badly.

Figure 36 shows the cross-model score given by our model selection approach. Generally we can see that the score between good models are larger. This suggests that our model selection approach can effectively select the good model pairs.

Figure 37, 38, 39,40,41,42,43 show the bar plots of supervised metrics, where the models are ordered according to our model selection approach. These figures show that our model selection approach correlates with other supervised metrics well.

Table 5 shows that the model selected with ModelCentrality outperforms UDR in most metrics.

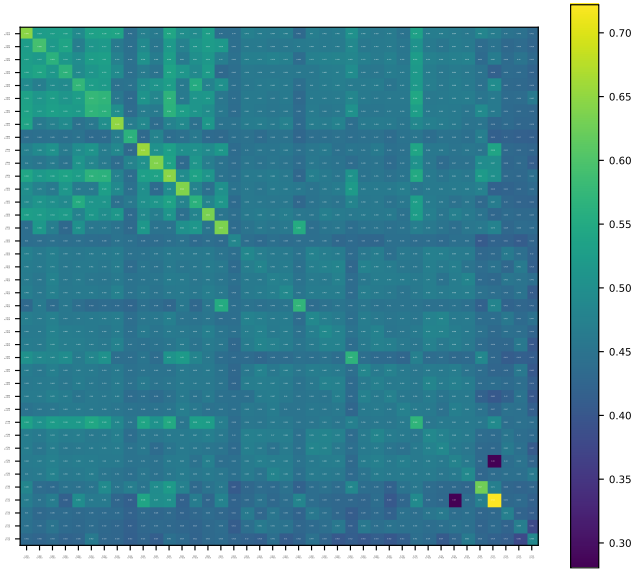

Figure 36: Our model selection approach on FactorVAE models (dSprites dataset). Each row/column corresponds to an FactorVAE model. The models are sorted according to FactorVAE metric. (i, j) entry represents the cross-evaluation score of i-th and j-th model using our model selection approach.

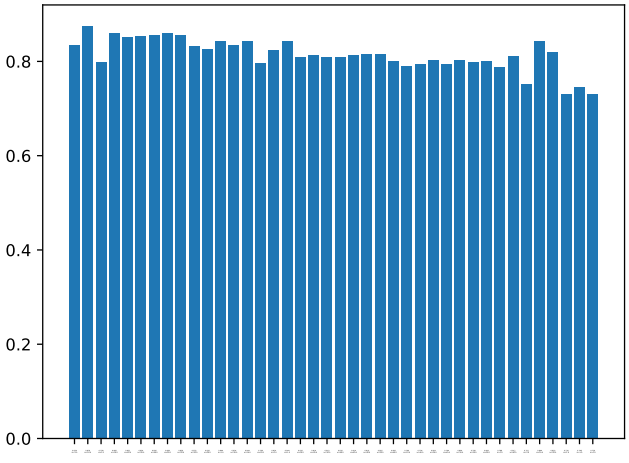

Figure 37: FactorVAE scores of FactorVAE models (dSprites dataset). The models are sorted according to our model selection score.

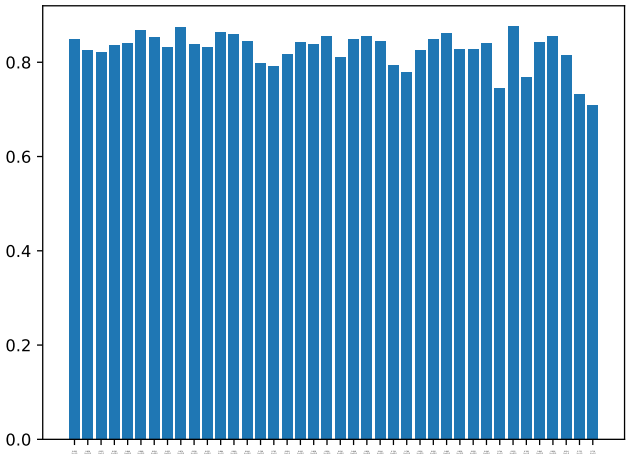

Figure 38: BetaVAE scores of FactorVAE models (dSprites dataset). The models are sorted according to our model selection score.

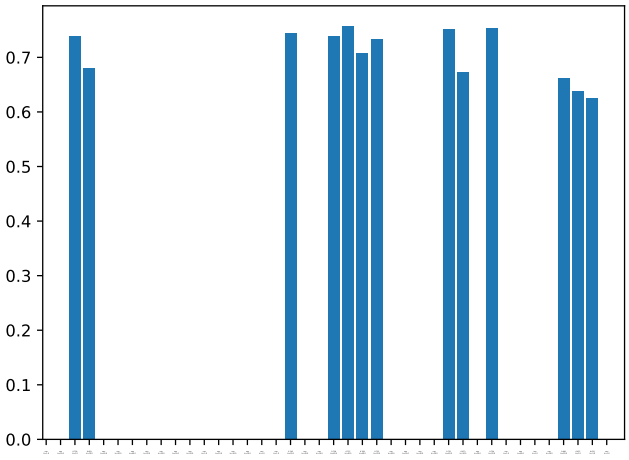

Figure 39: DCI scores of FactorVAE models (dSprites dataset). The models are sorted according to our model selection score. Note that some of the DCI score gives NaN values and therefore some bars are missing.

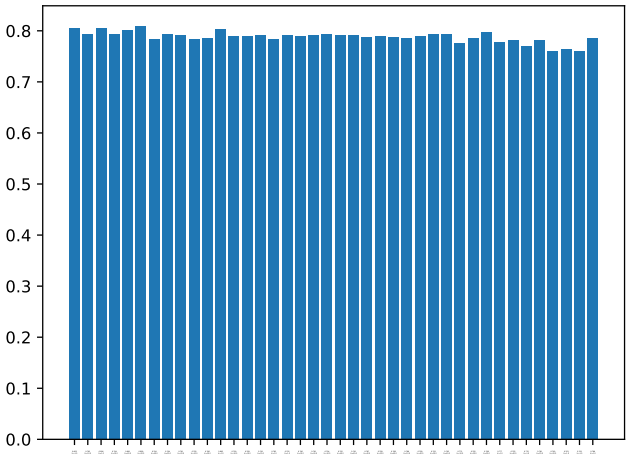

Figure 40: Explicitness scores of FactorVAE models (dSprites dataset). The models are sorted according to our model selection score.

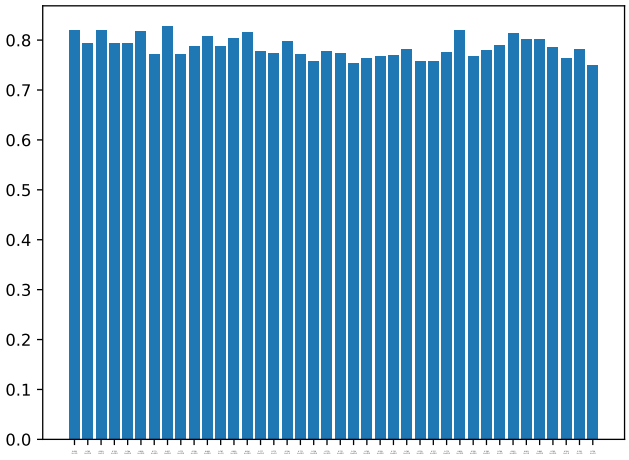

Figure 41: Modularity scores of FactorVAE models (dSprites dataset). The models are sorted according to our model selection score.

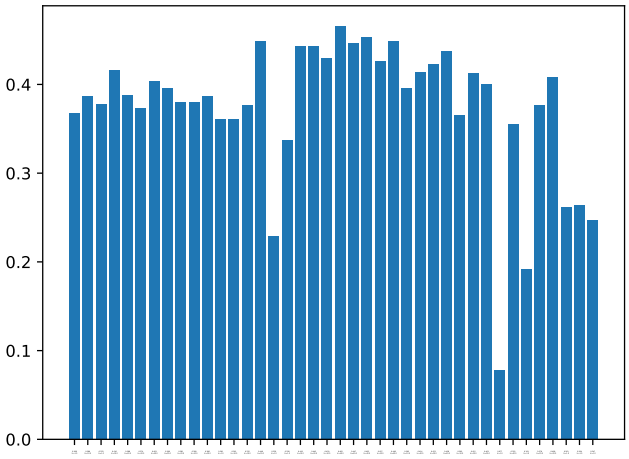

Figure 42: MIG scores of FactorVAE models (dSprites dataset). The models are sorted according to our model selection score.

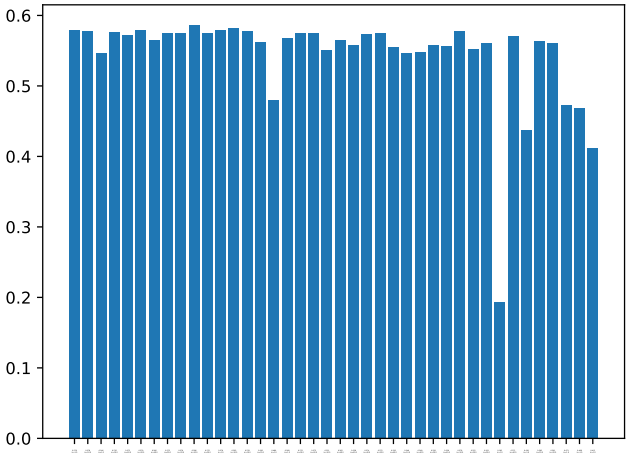

Figure 43: SAP scores of FactorVAE models (dSprites dataset). The models are sorted according to our model selection score.

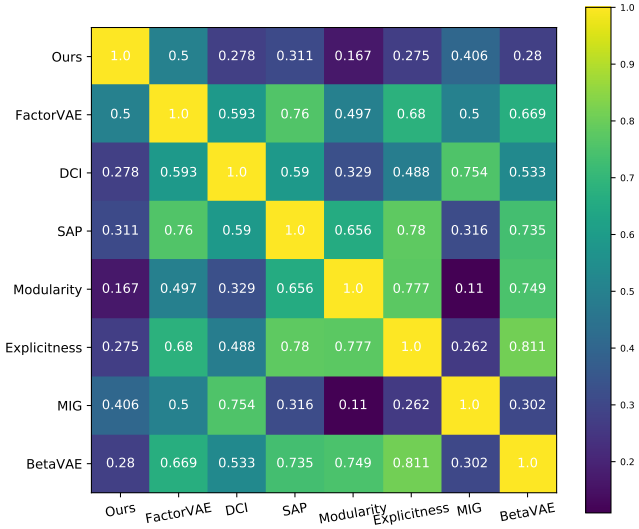

Figure 44: Spearman rank correlation of different metrics on InfoGAN-CR models (3DTeapots dataset).

K.3. Our Model Selection Approach on InfoGAN-CR Models (3DTeapots Dataset)

We run 10 independent trials for each of the following 9 hyperparameter settings $\{\alpha = 1.5, 3.0, 6.0\} \times \{\lambda = 0.05, 0.2, 2.0\}$. There are 90 models in total. The progressive scheduling is the same as the supervised experiments: we train InfoGAN for 50000 batches, and then InfoGAN-CR with gap=1.9 for 35000 batches, and then InfoGAN-CR with gap=0.0 for 40000 batches.

Figure 44 shows the correlations between different metrics (including UDR and our model selection approach). It is clear that the score given by our model selection approach is positively correlated with other supervised metrics. Though the correlation is not as positive as in dSprites dataset, we see that the correlations between other supervised metrics are also weaker than the ones in dSprites dataset. This may because this dataset is harder to interpret and evaluate for those metrics than dSprites.

Figure 45 shows the cross-model score given by our model selection approach. Generally we can see that the score between good models are larger, but the result is much more noisy than the one in dSprites. Again, the reason might be that this dataset is more difficult.

Figure 46, 47, 48,49,50,51,52 show the bar plots of supervised metrics, where the models are ordered according to our model selection approach. These figures show that our model selection approach roughly correlates with other supervised metrics, but is more noisy than the results in dSprites. Again, the reason might be that this dataset is more difficult.

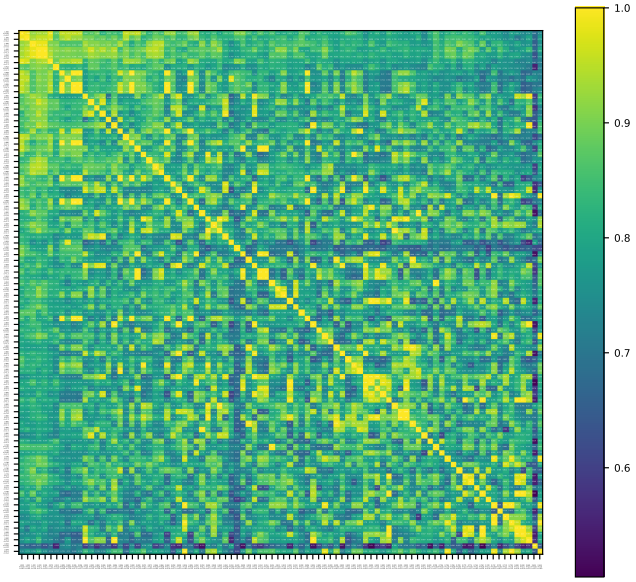

Figure 45: Our model selection approach on InfoGAN-CR models (3DTeapots dataset). Each row/column corresponds to an InfoGAN-CR model. The models are sorted according to FactorVAE metric. (i, j) entry represents the cross-evaluation score of i-th and j-th model using our model selection approach.

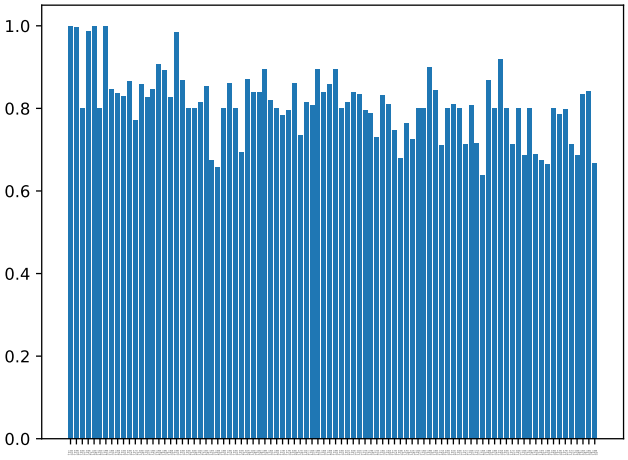

Figure 46: FactorVAE scores of InfoGAN-CR models (3DTeapots dataset). The models are sorted according to our model selection score.

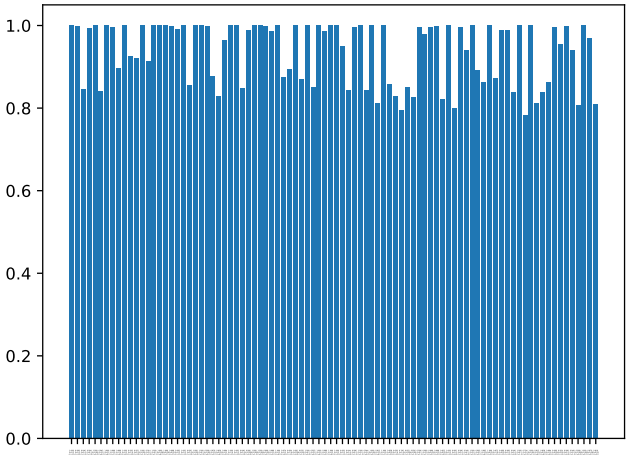

Figure 47: BetaVAE scores of InfoGAN-CR models (3DTeapots dataset). The models are sorted according to our model selection score.

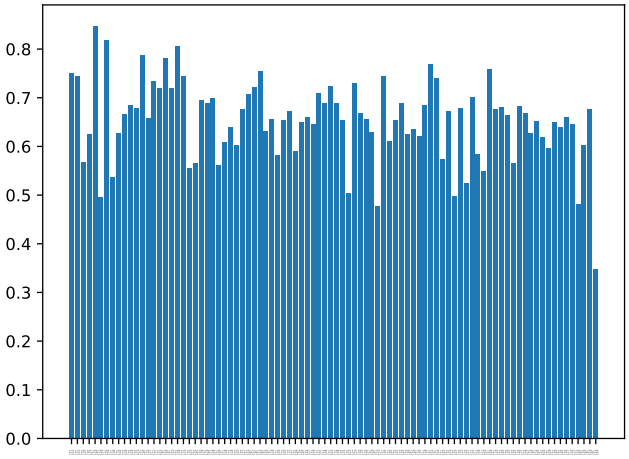

Figure 48: DCI scores of InfoGAN-CR models (3DTeapots dataset). The models are sorted according to our model selection score.

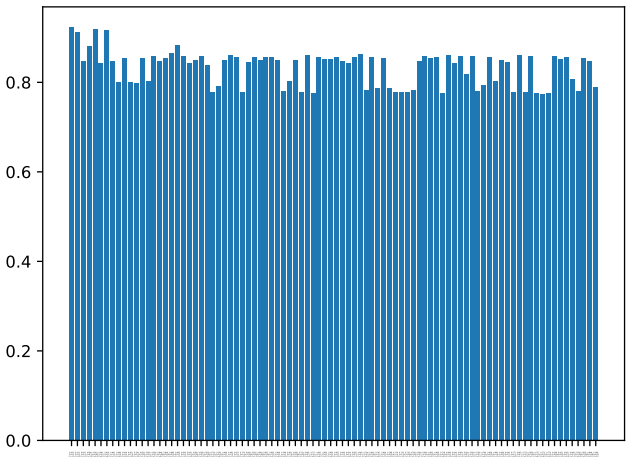

Figure 49: Explicitness scores of InfoGAN-CR models (3DTeapots dataset). The models are sorted according to our model selection score.

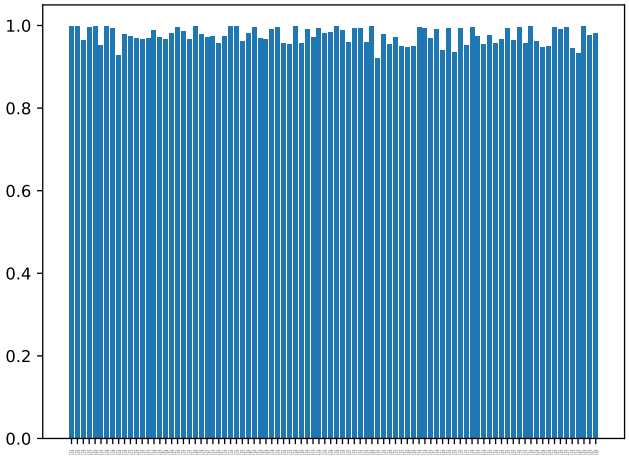

Figure 50: Modularity scores of InfoGAN-CR models (3DTeapots dataset). The models are sorted according to our model selection score.

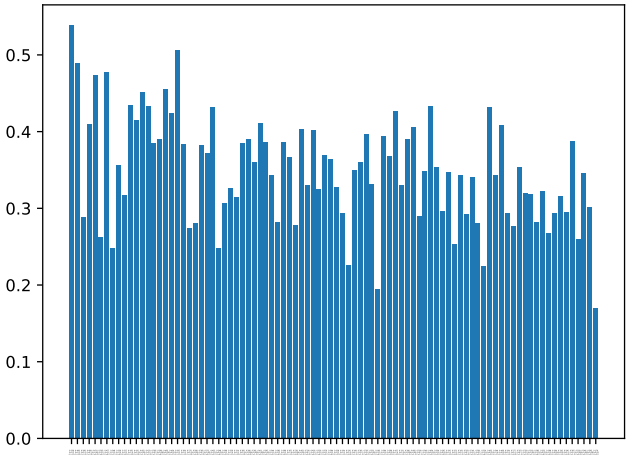

Figure 51: MIG scores of InfoGAN-CR models (3DTeapots dataset). The models are sorted according to our model selection score.

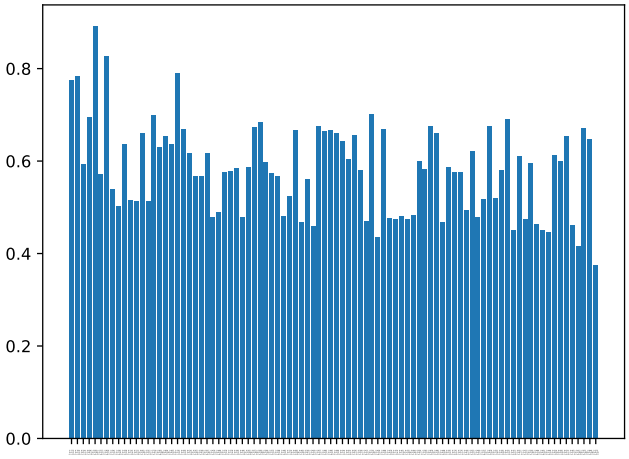

Figure 52: SAP scores of InfoGAN-CR models (3DTeapots dataset). The models are sorted according to our model selection score.