A Reproducibility

In this section, we provide the information required to reproduce our results reported in the main text. And we commit to making the code implementation and evaluating checkpoints public. Our experiments are run on a machine with an AMD Ryzen Threadripper 3970X 32-Core Processor and a GeForce RTX 3090 GPU.

VAE methods implementation. For the results on synthetic datasets, i.e., dSprites, Cars3D, SmallNORB, and Shapes3D, the disentanglement score is from the original logs of DisLib (34) [2]. In the released logs, each method has different training configurations, and our reported result is from the configuration with the highest average performance overall the provided random seeds. For the evaluation on the CelebA dataset, we follow an open-sourced implementation in Pytorch [3] and align the encoder architecture of all methods to be the same as described in Appendix A.1. For the results on Shapes3D, because DisLib does not release the pretrained checkpoints, we use the same open-sourced implementation to reproduce with the configuration indicated by DisLib. Parameters are kept as the default well-tuned version in the provided implementation. When the latent dimension is 1000, training of BetaTC VAE will collapse with the default hyperparameters, we have to decrease the β to 3.0 to work it around.

GAN methods implementation. Limited by the text length, we do not include the performance of GAN methods in the main text, but we will report some in the following appendix content. It is hard to include GAN methods' performance in the benchmark as the training is not always stable and the discriminator weights are usually not provided in many public codebases. When evaluating the methods on synthetic datasets, the FactorVAE scores of InforGAN, IB-GAN, and InfoGAN-CR are provided in the paper of Lin et al.. But the evaluation of other metrics in Lin et al. uses a not aligned settings with Locatello et al., so we check its officially release [4] to evaluate the provided implementation and model weights under the unified evaluation setup. We perform the same evaluation process for results on the CelebA dataset.

Energy-based Model (EBM). We refer to the implementation of ICE-BeeM (23) for this method. We use the officially released codebase for it [5]. The encoder implementation has been aligned with our default already. The only modification we make is to use the unconditional version instead of its default conditional version in loss computation to satisfy the fully unsupervised settings.

Contrastive Learning implementation. Our implementations are based on the public and official implementations of MoCo/MoCov2 [6], BYOL/SimSiam [7] and Barlow Twins [8]. The details of implementation are explained in Appendix A.1.

Evaluation Protocol. For MED, we first compute MI following the implementation of MIG by DisLib (34). Then we calculate the entropy disentanglement score in the same way as the DCI Disentanglement score in DisLib. For other disentanglement metrics evaluation, we use the implementation of DisLib. The settings of some important parameters are provided in Appendix A.2.

A.1 Implementation of contrastive learning model

Architecture. To make a fair comparison with previous methods, we follow the encoder architecture in Factor VAE (24). The pipeline details are shown in Table 2. After each convolutional layer in the figure, there is a ReLU activation layer and a group normalization (group number = 4) layer for BYOL. So, the encoder is a stack of (Conv-ReLU-GN) blocks. For other contrastive learning methods, we keep the default batch normalization to replace GN. By default, the final output channel number is 1000, i.e, $D = 1000$. For other details of contrastive learning methods, we follow the convention in their official implementations.

[2] https://github.com/google-research/disentanglement_lib

[3] https://github.com/AntixK/PyTorch-VAE

[4] https://github.com/fjxmlzn/InfoGAN-CR

[5] https://github.com/ilkhem/icebeem

[6] https://github.com/facebookresearch/moco

[7] https://github.com/lucidrains/byol-pytorch

[8] https://github.com/facebookresearch/barlowtwins

Besides the representation network (encoder), BYOL also has a projector network and a predictor network. Both of them consist of a pipeline "Linear $\longrightarrow$ BN $\longrightarrow$ ReLU $\longrightarrow$ Linear". The projection dimension is 256, and the hidden dimension of the projector is 4096. The predictor keeps a 256-dimensional feature vector in its pipeline.

Table 2: The encoder architecture for our implemented contrastive learning methods on synthetic datasets. Besides, there is a ReLU activation layer and a possible normalization layer following each convolutional layer to create a stack of (Conv-ReLU-Norm) blocks.

Encoder
input: 64×64 images
pipeline:
4×4 conv, stride 2, 32-channel
4×4 conv, stride 2, 32-channel
4×4 conv, stride 2, 64-channel
4×4 conv, stride 2, 64-channel
4×4 conv, stride 2, 128-channel
1×1 conv, stride 1, D-channel

Training settings. We make minor modifications to the training setting of default BYOL to apply to contrastive learning methods without negative samples. For training on all datasets, the images are resized to 64x64. For data preprocessing, we copy 1-channel images of dSprites and SmallNORB to be 3-channel. During the training stage, we use such a pipeline of augmentation (in *PyTorch*-style):

1. *RandomApply(transforms.ColorJitter(0.8, 0.8, 0.8, 0.2), p=0.3)*
2. *RandomHorizontalFlip()*
3. *RandomApply(transforms.GaussianBlur((3,3), (1.0, 2.0)), p=0.2)*
4. *RandomResizeCrop(size=(64, 64), scale=(0.6,1.0))*
5. normalization.

For the normalization, the pixel value of images from dSprites and SmallNORB is uniformly normalized from [0,255] to [0,1.0]. For Cars3D, Shapes3D, and CelebA, we adopt the commonly used Imagenet-statistic normalization for preprocessing the RGB image pixel values.

During training, we use Adam optimizer by default, whose learning rate is $3e-4$ without weight decay. The batch size is set to 512 by default. For evaluation on dSprites, Shapes3D, and CelebA, we select the weights after training for 15 epochs for evaluation. We select the weights after training for 140 epochs for evaluation on Cars3D and the weights of the 200th epoch on SmallNORB considering the small scale of these two datasets.

To decrease the influence of randomness, we train each model configuration multiple times with different random seeds (seed=0, 1, 2). We report the average and standard deviation. To be precise, as our implementation is based on Pytorch, we initialize the libraries of *numpy*, *torch*, *torch.cuda*, and *random* with the same random seeds.

Table 3: The factors on all the datasets we investigate the disentanglement on.

	dSprites	Shapes3D	Cars3D	SmallNORB	CelebA
Factors (# of values)	Shape (3) Scale (6) Orientation (40) Position X (32) Position Y (32)	Floor hue (10) Wall hue (10) Object hue (10) Scale (8) Orientation (15) Shape (4)	Elevation (4) Azimuth (24) Object id (183)	category (10) Elevation (9) Azimuth (18) Lighting (6)	40 attributes (2 for each)

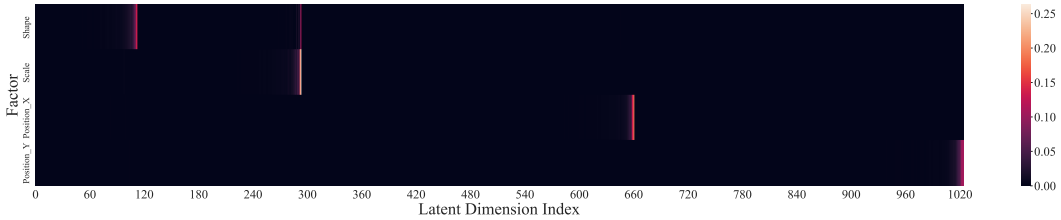

Figure 7: The importance distribution for the representation learned from BYOL on dSprites. Here, we follow the practice of DisLib to use a Gradient Boosting Tree (GBT) regressor to determine the importance matrix of each latent dimension in predicting each factor. Compared with the Mutual Information distribution shown in Figure 1a, the importance distribution is significantly more sparse. Sparsity is encouraged when constructing the GBT regressor. This makes it hard to study the true representation pattern.

A.2 Evaluation Metrics

In the main text, we compare the evaluation metrics provided in the DisLib protocol with our proposed MED metric. Here we provide more details about them. Moreover, we would conduct evaluations under all of them in the next section.

BetaVAE Metrics. Introduced in Higgins et al. (15), BetaVAE score assumes each dimension corresponds to one category in a linear classifier. Representations are obtained after the generated samples with only one factor fixed. Then we calculate the summation of the divergence between different representations and put it into a linear classifier. The classifier is trained to predict the index k for the fixed data factor. The accuracy of this linear model is the value of the BetaVAE score.

FactorVAE Metrics. Kim and Mnih (24) argues the BetaVAE score has the tendency to fail into a spurious disentanglement and proposes a new metric based on a majority vote classifier. Representations are obtained after the generated samples with only the k-th factor fixed. Normalizing each dimension in representations in terms of standard deviation. The index of dimension with the lowest variances of normalized representation and the factor index k are the input and the output of the linear classifier. The accuracy of the classification is the FactorVAE score.

Mutual Information Gap. Chen et al. (6) assumes the disentanglement model has the property that most information of one specific factor is contained in one dimension or a group of certain dimensions. The mutual information gap is the summation of the difference between the highest and second-highest normalized mutual information between a fixed factor and the dimensions of the output representation vector. The formula can be illustrated below:

$$\frac{1}{K}\sum_{k=1}^{K}\frac{1}{H_{z_k}}(I(v_{j_k},z_k)-\max_{j\neq j_k}I(v_j,z_k)), \tag{5}$$

where K is the overall number of ground truth factors. v is the latent representation and z_k is the factors of latent variables and $j_k=\arg\max_j I(v_j,z_k)$.

DCI disentanglement. As Eastwood and Williams (9) suggest, the disentanglement can also be measured by the entropy of relative importance for each dimension in predicting factors. First, we have to know the importance of each dimension of the representation for predicting each factor. The importance is determined by a regressing model such as Lasso or Random Forest in the original DCI implementation (9) or Gradient Boosting Tree in DisLib implementation (34). We note the importance matrix R where R_{ij} is the importance of the i-th dimension in prediction the j-th factor. Then the disentanglement score for the i-th dimension is defined as $D_i=(1-H_K(P_i))$ where $H_K(P_i)=-\sum_{k=0}^{K-1}P_{ik}log_K P_{ik}$ denotes the entropy and $P_{ij}=R_{ij}/\sum_{k=0}^{K-1}R_{ik}$ denotes the normalized importance of the i-th dimension in prediction the j-th factor. Finally the overall disentanglement score is calculated as $D=\sum_i \rho_i D_i$ where $\rho_i=\sum_j R_{ij}/\sum_{ij}R_{ij}$ is the weighting factor of the each dimension's informativeness in representing factors.

SAP. Kumar et al. (27) proposes the Separated Attribute Predictability (SAP) score. SAP is computed with classification score of predicting j^{th} factors on i^{th} dimension as the ij^{th} entry. SAP is the mean of the difference between the highest and second-highest scores for each column.

We follow the implementation provided by DisLib (34) for the evaluation protocol. Despite exceptions, the evaluation batch size is 64, the *prune_dims.threshold* is 0.06. If a classifier is required to be trained

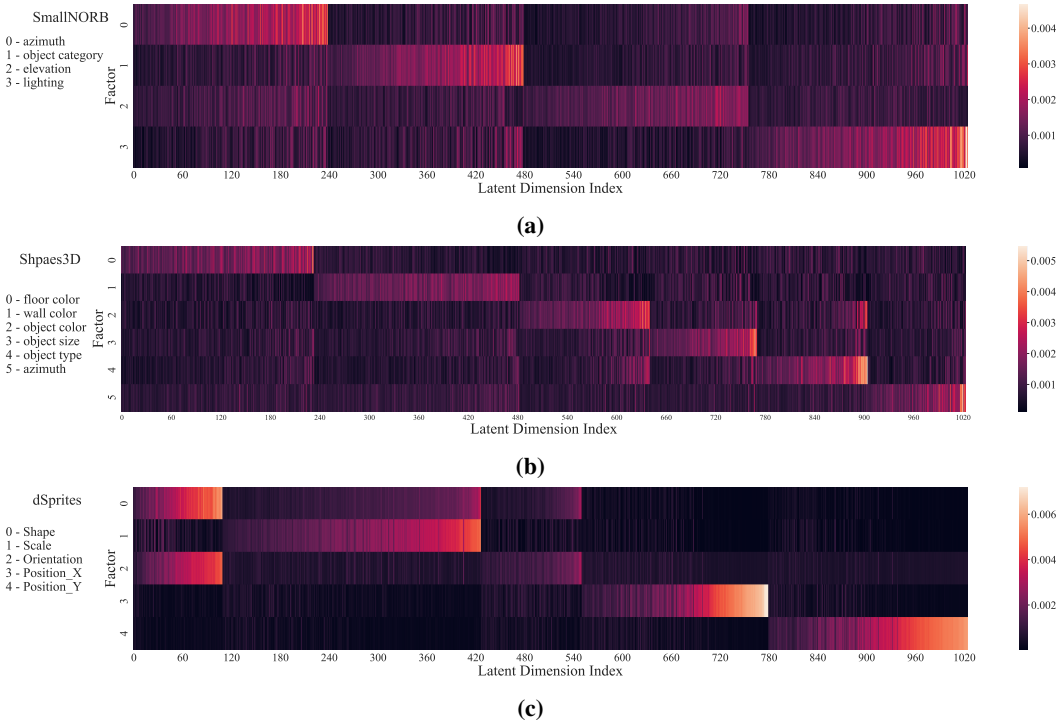

Figure 8: The mutual information distribution on SmallNORB(a), Shapes3D(b) , and dSprites including all factors(c).

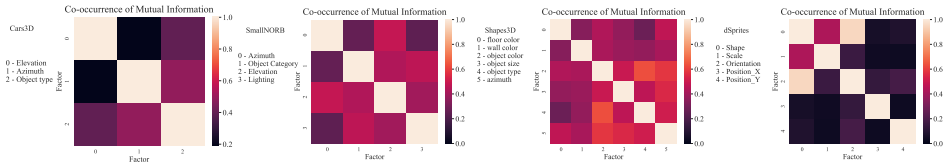

Figure 9: The co-occurrence of factors in the mutual information relationship among BYOL representations on Cars3D(a), SmallNORB(b) , Shapes3D(c), and dSprites including all factors(d).

during evaluation, *num_train* is 10000, and *num_eval* is 5000. For Mutual information computation, the discretizer function is the histogram discretizer, and the number of bins in the discretization is 20. For the evaluation of MIG and SAP on dSprites, SmallNORB, Cars3D, and Shapes3D, BYOL representation vectors are reduced to 10 dimensions by PCA to be aligned with other methods. For the evaluation of MIG and SAP on CelebA, to have a fair comparison, the representation vectors of all methods are reduced to 40 dimensions. For the implementation of our proposed MED, the basic logic is the same as DCI Disentanglement, but we replace the classifier output with the mutual information based scores.

B More Qualitative Study

We provide more qualitative studies about the disentanglement property shown by the contrastive learning here. We still use BYOL as an example of the negative-free contrastive learning methods.

B.1 Importance Distribution by DCI

In the main text, we concisely talked about the potential variables introduced by the learnable model under some metrics. Here we show an example for the widely used DCI Disentanglement metric. We follow DisLib to use Gradient Boosting Tree to estimate the importance matrix between each factor and each latent dimension. All parameters are set the same as its default protocol. The visualization is shown in Figure 7. Compared with the mutual information distribution shown in

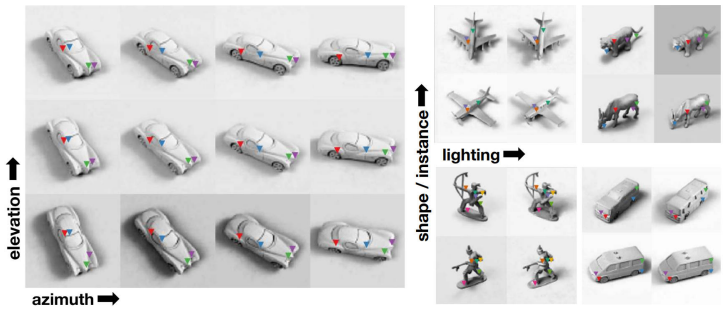

Figure 10: Some samples from SmallNORB dataset. The variance is controlled by the factor indicated on the axis. The image is from Jakab et al. (20).

Figure 1a, the importance distribution is much more sparse. This is because the construction of the GBT regressor encourage the sparsity of the output importance matrix. Such a sparsity can lead to the misunderstanding that the correlation between factors and latent dimensions is also sparse which is not true. By using pure measurement without involving additional adaptive models, such a problem will not be raised in the proposed MED metric.

B.2 Mutual Information Heatmaps

We compute MI between each latent dimension and each data factor and visualize them by heatmaps. The heatmaps offer us an intuitive picture of the learned representation space. For completeness, we show the MI heatmaps on SmallNORB, Shapes3D, and dSprites with all factors in Figure 8a, Figure 8b and Figure 8c respectively. We can see that the disentangled pattern described in the main text still emerges. There is a group of columns brighter than others in each row, and these groups do not overlap for most rows. However, we find that some latent dimensions emphasize more than one factor. We provide a more detailed analysis from the perspective of factor co-occurrence on this phenomenon in B.3 below.

B.3 Co-occurrence of Factors

To understand to what extent one dimension of the learned representation would respond to more than one factor, we make the co-occurrence of mutual information to factors on more datasets here. The visualizations are shown in Figure 9 for the results on Cars3D, SmallNORB, Shapes3D, and dSprites respectively. Moreover, we now analyze the definition of data factors of these datasets to discuss whether they are defined to be fully independent or not.

SmallNORB Though most non-diagonal entries have very low co-occurrence of mutual information, two pairs of factors show slightly higher co-occurrence. They are "azimuth-elevation" and "instance category-lighting". After investigating the dataset, we find the two pairs of factors are not fully independent. Figure 10 show some samples with corresponding factors manipulated. We could see that the elevation and azimuth are not fully independent. And the correlation between the instance category and the lighting factor is even more obvious because the lighting condition is sensibly related to the shadow around the object, whose distribution and shape are highly determined by the instance category.

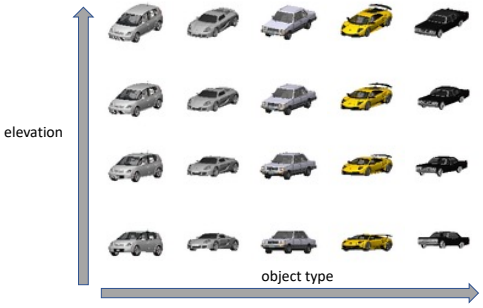

Figure 11: Samples from Cars3D. The variables of the object type and the elevation are controlled. It shows that the two factors are not independent.

Cars3D Only one pair of factors show some co-occurrence, i.e. "elevation-object type". We randomly selected samples from Cars3D by different object types and elevations, as shown in Figure 11. It shows that with the same value of

elevation, samples of different object types have different visual elevations. So these two factors are not fully independent. This explains the slightly higher co-occurrence of mutual information between this pair of factors.

Shapes3D The result shows relatively bad disentanglement. To be precise, some factor pairs show low mutual information co-occurrence as expected, such as the color factors of floor, wall, and object and the pair of "object color - azimuth". But the MI co-occurrence of "wall color - object size" and "object color - object size;" are higher than we expected as we did not recognize their high dependence. This result might relate to our model's relatively poor performance on Shapes3D.

dSprites We omit the factor *orientation* in the visualization at main text for clarity. Here we show the results including all factors. For dSprites dataset, the value of *orientation* is highly correlated with *shape*. This is because that different *shape* has different symmetry properties for rotation. As a result, dimensions that encode information of *shape* tend to capture information of *orientation* simultaneously. This explains that, in Figure 8c, some dimensions have two brighter entries at "shape" and "orientation" rows. Thus, it is not surprising that the "shape - orientation" cell is brighter than other off-diagonal elements in Figure 9d.

B.4 Manipulating Factors

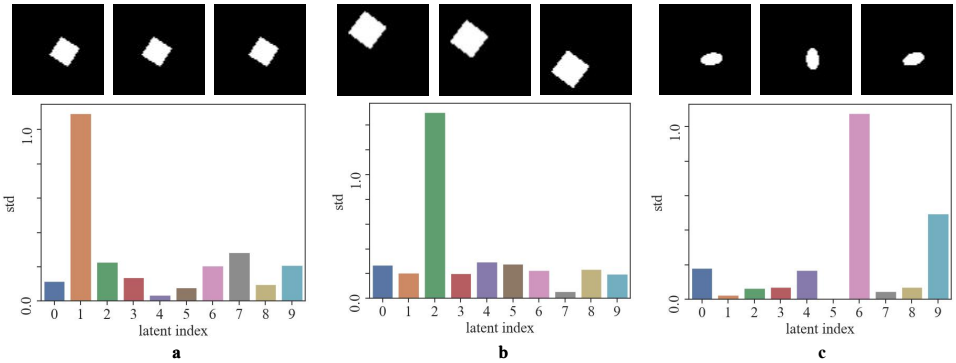

Figure 12: Representation variation when manipulating one factor only in the dimension-reduced version. In (a) and (b), *position_x* and *position_y* are manipulated respectively and only cause one dimension significantly variate. While, in (c), when manipulating the ill-defined factor *orientation*, two dimensions variate.

In the main paper, we studied the influence on the generated representation by manipulating the factors, where the representation is reduced by selecting dimensions as in calculating Top-k MED. Here, we do the qualitative study of the influence on representation by manipulating factors in another way but still on dSprites. To make the original high-dimensional representation space more compact, we use the unsupervised dimension reduction by PCA instead, which is more general when the data factor pattern is unknown. Here, we reduce the representation dimension by PCA to 10. Note that since the PCA operation mixes the original latent space with a linear combination, it might destroy the existing disentanglement property in the high dimensional space, or enhance the disentanglement if the original high dimensional space is a linear combination of the ground truth factors. But such influence is usually considered secondary to the disentanglement learned by a model. No matter which case, if the dimension-reduced representation shows disentangled properties, the original space at least captures linearly transformed ground truth factors, and the dimension reduction techniques such as PCA can make the representation more compact in a qualitative study.

Figure 12 shows the result of representation vector variation when changing only one factor at once. Given three images with only one factor's value being different, we generate the 10-dim representation vectors from them. Then, we compute the variance across the three vectors, leading to 10 scalars. The larger the variance is, the more that dimension responds to the factor change. Figure 12(a) and (b) show how reduced representation vector changes when manipulating *position_x* and *position_y* factor respectively. It shows good disentanglement that only one representation dimension has high variation. However, in Figure 12(c) we show a failure mode of the ill-defined factor *orientation* that change of factor causes both the 6th and the 9th dimensions of reduced representation to have large

variations. From the results, we observe that manipulating one well-defined independent factor causes evident variance in only one dimension. And it shows that we could make the learned representation vector more compact by unsupervised dimension reduction.

C More Quantitative Results

C.1 Full Disentanglement Benchmark

In the main text, we evaluate the disentanglement under our proposed MED metric for full space and top-k subspace on multiple datasets. In this section, to provide a more complete understanding of the disentanglement property of contrastive learning without negatives, we report the disentanglement scores with other metrics, including FactorVAE score, BetaVAE score, MIG, SAP, and DCI Disentanglement.

For the results of VAE-based methods, as the large-scale benchmark of Locatello et al. (34) provides the original logs on dSprites, Cars3D, and SmallNORB datasets, we simply report the performance of the best configuration. The original logs on Shapes3D are not available, so we train and evaluate on Shapes3D by ourselves for the MED scores. For scores under other metrics, we report the median disentanglement scores. Some results are from Locatello et al. (35) but the std error is not available. The median performance of SlowVAE is from its original paper (26). For the results of CelebA, the result of InfoGAN-CR is from its officially released checkpoint without availability to the std error. For other methods, we report the mean value of our trained weights over three random seeds as default. Because the evaluation of DCI is extremely time-consuming, around 14 hours for a 1000-d model, we only take BYOL as an example here for negative-free contrastive learning methods. All results are combined and shown in Table 4. We also include traditional metrics results on their selected top-k subspace for contrastive learning methods.

Aligned with the analysis we provide in the main text, the results show significant disagreement among the existing metrics. To be precise, for those metrics (BetaVAE score, FactorVAE score, DCI Disentanglement) using a learnable model such as a regressor or a classifier, the high-dimensional BYOL model achieves a significant advantage. However, for the metrics relying on only one or two dimensions to reveal the connection between a latent dimension and a factor (MIG and SAP), BYOL's performance is not that impressive anymore.

Finally, the result on CelebA shows the great robustness of BYOL's learned representations to be disentangled on real-world datasets. Yet, the large gap between the score of CelebA and those on synthetic datasets emphasizes the difficulty of learning disentangled factors on real-world images. It is hard to empirically study whether it is the high dimension that gives BYOL advantages on some metrics because the nature of BYOL makes it hard to be trained with a small latent dimension to make a comparison.

In the last rows for each dataset, we show previous disentanglement metrics scores on the top-k subspaces of BYOL. We find that BYOL (top-k) has performed better than or on par with unsupervised methods with low latent dimensions. Note that Ada-GVAE, SlowVAE, and EBM are (weakly) supervised methods. The SOTA performance justifies our claim of the well-disentangled subspaces. MIG and SAP require a large gap between the two most vital representation dimension-factor responses, measured by mutual information and classification accuracy. But out of 1000 dimensions, we choose the k most relative ones. Hence, the gap ought to be minimal. Consequently, BYOL (top-k) receives lower MIG and SAP scores. Except for dSprites and CelebA, BYOL (top-k) have lower DCI scores. The primary cause of this is the massive information loss during the selection of the top-k dimensions, which completely obliterates the original encoding pattern. As a result, the gradient boosting tree (GBT) developed throughout the DCI process tends to "borrow" information from latent dimensions that emphasize additional relative factors to classify some complex factors. The "information-lending" dimensions then become less disentangled since they are also important for predicting those complex factors. Taking the Cars3D dataset as an example, we set $k = 3$ for top-k MED. To predict the factor *object type*, the GBT needs to classify 183 types of cars from only 9 dimensions with only 3 emphasizing *object type*. This forces GBT to utilize information from *elevation* since these two attributes are correlated (see Appendix B.3). However, we keep all they have learned from training for other low-dimensional methods. Therefore, the information loss problem does not apply to them.

Table 4: Evaluation results on multiple datasets with different disentanglement metrics. The best results of all models are **bolded**. The best results on low-dimensional space are underlined.

	Model	BetaVAE	FactorVAE	MIG	SAP	DCI	MED
dSprites	β-VAE	82.3 (7.6)	65.8 (9.2)	26.3 (11.0)	5.2 (2.7)	39.3 (13.2)	32.6 (10.0)
	β-TCVAE	86.7 (2.4)	76.6 (7.8)	23.8 (6.8)	6.9 (0.9)	36.3 (7.1)	31.8 (7.4)
	FactorVAE	84.9 (2.8)	75.3 (7.4)	18.4 (9.0)	6.8 (0.8)	28.8 (10.6)	32.5 (10.1)
	DIP-VAE-I	82.7 (3.3)	59.1 (4.8)	9.6 (5.1)	5.2 (2.6)	14.4 (4.6)	18.8 (5.6)
	DIP-VAE-II	81.5 (4.9)	58.6 (7.6)	7.4 (3.4)	3.6 (2.2)	12.3 (5.2)	14.7 (5.5)
	AnnealedVAE	86.5 (0.1)	60.1 (0.0)	**35.2 (1.3)**	7.6 (0.5)	37.9 (2.1)	35.8 (0.8)
	Ada-GVAE	88.0 (2.7)	73.1 (3.9)	17.3 (4.7)	6.6 (2.0)	32.3 (4.6)	–
	SlowVAE	87.0 (5.1)	75.2 (11.1)	28.3 (11.5)	4.4 (2.0)	47.7 (8.5)	–
	EBM	82.3 (2.0)	65.7 (12.5)	1.7 (0.5)	3.0 (1.2)	19.1 (1.8)	6.8 (4.0)
	InfoGAN-CR	85.5 (1.0)	88.0 (1.0)	19.8 (3.2)	6.0 (1.0)	14.0 (5.2)	–
	BYOL	**93.2 (0.4)**	**91.6 (0.8)**	29.3 (0.4)	**8.0 (0.4)**	**66.9 (0.2)**	31.3 (0.4)
	BYOL (top-2)	90.0 (0.9)	79.2 (2.3)	3.3 (0.9)	0.8 (0.2)	45.0(0.1)	**53.7 (0.7)**
Cars3D	β-VAE	**100.0 (0.0)**	89.3 (1.2)	11.7 (1.1)	1.4 (0.9)	38.7 (4.6)	29.0 (2.2)
	β-TCVAE	**100.0 (0.0)**	92.2 (2.7)	**15.5 (2.9)**	1.7 (0.3)	42.7 (3.5)	**33.0 (3.8)**
	FactorVAE	**100.0 (0.0)**	91.7 (4.1)	10.6 (2.2)	**2.0 (0.5)**	29.0 (6.7)	29.1 (3.0)
	DIP-VAE-I	**100.0 (0.0)**	90.5 (5.0)	5.9 (2.8)	1.9 (1.4)	22.6 (5.6)	19.4 (3.3)
	DIP-VAE-II	**100.0 (0.0)**	85.0 (6.1)	5.1 (2.7)	1.3 (0.8)	20.8 (5.4)	16.7 (4.1)
	AnnealedVAE	**100.0 (0.0)**	85.0 (4.3)	7.6 (1.0)	1.5 (0.5)	18.5 (4.3)	15.5 (2.5)
	SlowVAE	**100.0 (0.0)**	90.4 (0.5)	15.4 (2.2)	1.6 (0.5)	48.0 (2.4)	–
	BYOL	**100.0 (0.0)**	**95.8 (1.2)**	7.6 (0.9)	1.8 (0.7)	**48.5 (2.3)**	9.7 (0.5)
	BYOL (top-3)	**100.0 (0.0)**	95.2 (0.8)	3.8 (0.5)	1.1 (0.8)	15.8 (3.6)	31.8 (1.3)
SmallNORB	β-VAE	84.1 (2.7)	60.1 (2.4)	25.0 (1.1)	11.4 (1.1)	32.6 (0.6)	24.4 (0.7)
	β-TCVAE	84.5 (2.7)	60.3 (2.3)	25.4 (0.9)	11.7 (1.1)	35.2 (0.7)	25.0 (0.9)
	FactorVAE	80.8 (3.8)	62.5 (3.6)	23.9 (2.0)	10.2 (0.9)	33.4 (1.1)	**25.9 (1.2)**
	DIP-VAE-I	84.2 (3.2)	69.8 (4.6)	24.3 (2.7)	10.2 (1.4)	30.0 (2.1)	24.5 (2.1)
	DIP-VAE-II	85.2 (1.3)	58.4 (2.1)	**25.5 (1.5)**	**14.4 (0.4)**	32.3 (0.7)	24.4 (0.7)
	AnnealedVAE	60.8 (6.2)	50.0 (9.9)	9.1 (2.2)	6.8 (0.8)	15.7 (6.4)	5.5 (3.7)
	SlowVAE	78.2 (3.8)	47.0 (2.9)	23.8 (1.8)	7.8 (1.1)	28.7 (0.7)	21.8 (1.3)
	EBM	79.0 (4.4)	57.9 (3.5)	1.7 (0.5)	1.9 (0.1)	13.9 (2.2)	2.3 (1.7)
	BYOL	**97.0 (0.8)**	**81.0 (0.5)**	3.3 (0.9)	2.2 (0.3)	**51.0 (1.0)**	7.7 (0.2)
	BYOL (top-2)	86.7 (0.4)	65.6 (3.7)	3.3 (1.4)	1.5 (0.2)	13.6 (0.3)	25.7 (0.3)
Shapes3D	β-VAE	100.0 (0.0)	92.4 (4.5)	37.8 (16.0)	11.3 (3.2)	77.3 (3.2)	52.4 (9.4)
	β-TCVAE	100.0 (0.0)	90.5 (5.5)	46.4 (15.4)	12.4 (6.1)	78.4 (5.2)	53.2 (4.9)
	FactorVAE	98.1 (3.2)	90.6 (6.4)	48.2 (15.2)	11.1 (4.3)	71.8 (8.6)	55.9 (8.0)
	DIP-VAE-I	98.3 (3.3)	84.2 (11.3)	20.1 (8.4)	6.0 (1.1)	69.0 (3.6)	43.5 (3.7)
	DIP-VAE-II	99.6 (0.04)	94.9 (4.1)	22.1 (3.54)	8.2 (1.8)	49.8 (10.6)	52.6 (5.2)
	AnneledVAE	95.1 (4.4)	88.8 (4.6)	46.2(5.4)	8.5(1.6)	56.2(4.7)	**56.1 (1.5)**
	Ada-ML-VAE	**100.0**	**100.0**	50.9	12.7	94.0	–
	Ada-GVAE	**100.0**	**100.0**	56.2	**15.3**	**94.6**	–
	SlowVAE	**100.0 (0.1)**	97.3 (4.0)	**64.4 (8.4)**	5.8 (0.9)	82.6 (4.4)	–
	EBM	75.9 (11.2)	53.2 (8.7)	5.2 (2.2)	2.8 (1.1)	21.8 (11.0)	2.1 (2.6)
	BYOL	91.5 (3.9)	82.5 (2.4)	5.2 (1.7)	2.8 (0.3)	53.1 (1.5)	6.0 (0.5)
	BYOL (top-2)	95.5 (1.1)	83.0 (3.6)	2.8 (0.8)	1.4 (0.5)	27.2 (3.8)	19.7 (1.3)
CelebA	VAE	21.5 (3.2)	6.1 (3.8)	0.8 (0.1)	0.9 (0.2)	11.2 (2.3)	3.8 (0.2)
	β-VAE	19.1 (1.9)	5.8 (1.8)	0.1 (0.1)	0.6 (0.2)	8.7 (1.9)	3.3 (0.1)
	β-TCVAE	19.9 (2.3)	9.8 (2.4)	0.6 (0.2)	1.2 (0.3)	3.5 (1.1)	4.7 (0.1)
	FactorVAE	25.3 (3.0)	**12.0 (2.1)**	0.4 (0.1)	0.6 (0.2)	7.1 (0.7)	0.6 (0.6)
	DIP-VAE-I	21.0 (1.9)	9.3 (1.1)	0.2 (0.1)	0.9 (0.3)	13.8 (2.2)	3.6 (0.2)
	InfoGAN-CR	16.8	11.3	1.6	2.8	22.0	–
	BYOL	**35.7 (2.1)**	11.5 (1.1)	**2.6 (0.7)**	**8.2 (0.9)**	**41.0 (1.3)**	4.8 (0.4)
	BYOL (top-3)	25.8 (0.8)	9.7 (0.5)	0.7 (0.2)	0.2(0.02)	15.8(0.3)	**6.8 (0.7)**

D Ablation Study about the Normalization Choice

In this section, we put ablation studies here about the choice of normalization layers in the model architecture.

Table 5: Results of using different normalization strategies on dSprites.

normalization	w/o norm	BN	GN	LN	IN
MED	23.8 (0.6)	29.4 (0.5)	31.3 (0.4)	31.3 (0.8)	0.0 (0.0)

We experiment with five types of normalization layers in the encoder network on the dSprites dataset. The results are shown in Table 5. For the group normalization, we set the group number to 4. On the dSprites dataset, we find the commonly used BN decreases the disentanglement performance. By keeping the batch norm in the projector and the predictor, removing the batch norm in the encoder will not cause the model to collapse, which agrees with the observation in previous works (40). On the contrary, replacing batch norm in the encoder with group norm or layer norm will increase the representation disentanglement while achieving similar accuracy in downstream factor prediction. We notice that a similar phenomenon has been discovered before in supervised representation disentanglement. For example, Bau et al. (2) discovered that a network trained with batch normalization layers has less interpretable (disentangled) neurons. On the other hand, the instance norm (48) completely breaks the contrastive learning process. We still do not fully understand this behavior, but we hypothesize that it may be caused by the shared batch statistics that make it hard for a feature to be aligned with the ground truth factor.

E Disagreement of Existing Metrics

Without a uniformly recognized definition of "disentanglement", the validity of existing metrics is backed up with their shared expectation of "disentanglement" and consistency of experimental results, as suggested in DisLib (34). However, the results in Figure 4 break the belief. The disagreement arises when the high-dimensional representation model joins the comparison. We provide an in-depth view of the disagreement by ranking different methods. The results are shown in Figure 13. If all metrics agree perfectly, there should be no relative ranking switch. However, the frequent switch, especially with BYOL being taken into comparison, strongly suggests their disagreement. Results in Table 4 further show that different metrics may make completely different rankings with large value gaps. We also provide logistic analysis in Appendix F and quantitative results in Appendix C. Given the fact that self-supervised representation learning always requires a high dimension to train, MED is necessary to extend the study of disentangled representation learning to complicated real-world datasets and self-supervised representation learning methods.

F The Superiority of MED

Here, we provide both experimental observations and theoretical analysis in synthetic scenarios to show the superiority of MED: in what cases other metrics outputs meaningless or even opposite results but MED can still perform a meaningful evaluation.

F.1 Experimental Observations

As we discuss throughout the paper, the five existing metrics we investigate, i.e. BetaVAE score, FactorVAE score, SAP, MIG, and DCI Disentanglement, are unfair to models of different dimensions. So we extend the results mentioned in Section 3.2 to show the superiority of MED by confirming its stability under different dimensions. We conduct a sanity check on randomly initialized models. For a good disentanglement metric, we do not expect a high score whatever the representation dimension is. The following results reveal the malfunction of these metrics.

- **MED vs MIG/SAP**: First of all, we emphasize that SAP/MIG is designed not just for *disentanglement* but also *completeness*. They encourage a strict one-to-one mapping between latent dimensions and factors. So they tend to reward models whose dimension is close to the factor number. On the popular synthetic datasets with only a limited number of factors, low-dimensional models have an unfair advantage with MIG/SAP metrics. We compare a well-trained 1000-d BYOL model and a 10-d randomly initialized encoder in Table 6. The results show that the 10-d randomly initialized model can achieve higher scores than the

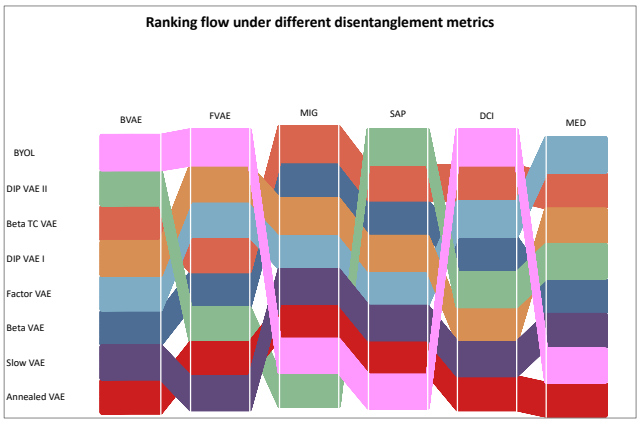

Figure 13: The flow chart of rankings of disentanglements scores. From top to down, the ranking decreases, indicating high to low disentanglement properties. If all metrics agree perfectly, there should be NO switch of relative ranking among these methods with different metrics. However, we notice obvious ranking variance, especially regarding the ranking of BYOL. With the analyses of the bias of existing metrics when the representation dimension varies, this figure emphasizes the necessity of proposing a new disentanglement metric that is fair to compare models of different dimensions.

1000-d BYOL model by SAP and MIG. But MED keeps a reasonable comparison that the trained high-dimensional model can still enjoy higher scores than low-dimensional random weights.

Table 6: The mean scores of different metrics on Shapes3D.

	MIG	SAP	MED
10-d randomly initialized	**6.7**	**3.0**	2.9
1000-d BYOL	5.2	2.8	**6.0**

- **MED vs BetaVAE/FactorVAE score**: BetaVAE and FactorVAE prefer high-dimensional models as they have more parameters to trick the learnable classifier. In Table 7, we find even randomly initialized high-dimensional (1000-d) model can achieve higher BetaVAE/FactorVAE score than a well-trained 10-d model (DIP-VAE-II). In contrast, MED provides results in line with our expectations.

Table 7: The mean scores of different metrics on dSprites.

	BetaVAE score	FactorVAE score	DCI	MED
10-d DIP-VAE-II	81.5	58.6	12.3	**14.7**
1000-d randomly initialized	**82.7**	**61.4**	**19.8**	3.8

- **MED vs DCI**: Similar to BetaVAE/FactorVAE scores, DCI also overestimate high-dimensional models as they have a higher chance to get lottery dimensions, especially with the sparsity encouraged by GBT. Also, another huge drawback of DCI is the computation efficiency: for a 1000-d model, DCI takes more than 14 hours for evaluation while MED only takes less than 20 seconds. The computation of MED is more than 2000 times faster than DCI.

Through the experimental observations, we find the abnormals from existing metrics when comparing models of different dimensions. Compared to them, the results of MED are consistent with human knowledge.

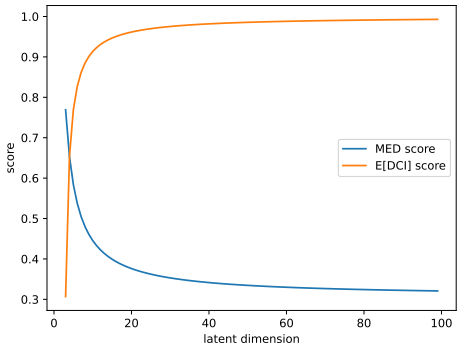

Figure 14: How MED score and DCI score change along the latent dimension of the model in our constructed linear scenarios.

F.2 Theoretical Analysis

In this part, we construct synthetic scenarios to investigate the validity of MED and existing metrics. We will show that in such situations, existing metrics will fail to reflect the real disentanglement quality but MED can show superiority to maintain reasonable evaluations. For the following cases, we assume there are two data factors F_0, F_1 on the data. We note their value as $v_0, v_1 \in \{0, 1\}$. And they are sampled independently and uniformly, i.e., $p(v_j = 1) = p(v_j = 0) = 0.5$. Then we construct linear cases to simplify the analysis that the value of latent dimensions is the combination of factor values.

F.2.1 MED vs MIG/SAP.

As mentioned, MIG and SAP require not just *disentanglement* but also *completeness*. So, for a 1000-d model whose latent code value is $c_i = v_{i \bmod 2}, 1 \leq i \leq D, D = 1000$. I.e. half of the latent code is the true first factor, and the other half latent code is the second factor. It is clear that each dimension perfectly correlates to a single factor. But its MIG/SAP scores turn out to be 0, which is significantly opposite to our intuition. But in this case, for MED we have

$$S_i = 1, \quad \rho_i = \frac{1}{D}. \tag{6}$$

So $\text{MED}(c) = 1$, indicating that every dimension is perfectly disentangled to a single factor and fully responsive to it. Also, this conclusion applies to different values of D, showing the robustness of MED with models of different dimensions.

F.2.2 MED vs DCI.

With a constructed latent code as

$$c_i = \begin{cases} v_0 & , i = 0 \\ v_1 & , i = 1 \\ \frac{1}{2}(v_0 + v_1) & , i > 1 \end{cases}. \tag{7}$$

To simplify, we assume the importance matrix output by GBT in DCI metric is the absolute of first-order derivative, namely $|\frac{\partial c_i}{\partial v_j}|$ of the dimension i with respect to the factor F_j. GBT encourages sparsity in the output importance matrix. To reflect this, we assume the desired sparsity of the GBT is 2. Thus, we have the dimensions beyond the sparsity limit are given 0 importance. Then the importance matrix value R_{ij} is

$$R_{ij} = \begin{cases} 1 & , i = j, i \leq 1 \\ \frac{1}{2} & , i = d_j \\ 0 & , otherwise \end{cases}. \tag{8}$$

d_j is a randomly selected dimension between $[2, D-1]$. For the two factors F_0 and F_1, d_j is selected independently. So that chance that $d_0 = d_1$ is $p(d_0 = d_1) = \frac{1}{D-2}$.

(1) If $d_0 = d_1 = k$, we have

$$\rho_i = \begin{cases} \frac{1}{3} & , i = 0, 1, k \\ 0 & , otherwise \end{cases}, \qquad S_i = \begin{cases} 1 & , i = 0, 1 \\ 1 - log2 & , i = k \\ 0 & , otherwise \end{cases} \tag{9}$$

(2) If $d_0 \neq d_1$, we have

$$\rho_i = \begin{cases} \frac{1}{3} & , i = 0, 1 \\ \frac{1}{6} & , i = d_0, d_1 \\ 0 & , otherwise \end{cases}, \qquad S_i = \begin{cases} 1 & , i = 0, 1, d_0, d_1 \\ 0 & , otherwise \end{cases}. \tag{10}$$

And DCI Disentanglement score is $\text{DCI}(c) = \sum_{i=0}^{D-1} \rho_i S_i$. We have $\text{DCI}(c) = 1 - log2 = 30.7\%$ in situation (1) and $\text{DCI}(c) = 1$ in situation (2). So, in such a simplified version, the expectation of DCI Disentanglement score is

$$\mathbb{E}[\text{DCI}(c)] = \frac{1}{D-2}(1 - log2) + (1 - \frac{1}{D-2}) = 1 - \frac{log2}{D-2}. \tag{11}$$

On the other hand, similarly we can derive the MED score is

$$\text{MED}(c) = 1 - \frac{D-2}{D} log2. \tag{12}$$

which is quite consistent along the change of D. Note that the drastic difference between MED and DCI is caused by the GBT used in DCI. The GBT leads to a sparse importance matrix M, while the mutual information in MED won't. This difference doesn't make much difference in low-dimension latent codes, but DCI will be high even if the latent code doesn't disentangle at all in the high-dimensional case.

We plot the curves of MED score and the expectation of DCI score in Figure 14. It shows that when the latent dimension D increases, the expectation of the DCI score increases and approaches 1. For instance, if $D = 3$, we have $\mathbb{E}[\text{DCI}(c)] = 30.7\%$; if $D = 1000$, we have $\mathbb{E}[\text{DCI}(c)] = 99.9\%$. Such an abnormality from DCI Disentanglement alerts us that it can not give a meaningful and fair comparison between methods of different dimensions. However, MED is much more robust with the change of model dimension. For instance, if $D = 3$, we have $\text{MED}(c) = 76.9\%$ and if $D = 1000$, we have $\text{MED}(c) = 30.8\%$.

F.2.3 MED vs BetaVAE/FactorVAE score.

Situation 1 We construct a situation where the first two dimensions are a weighted combination of the factors and the value of all following dimensions are the average of the factor values. Thus we have the latent code as

$$c_i = \begin{cases} \frac{1}{3}v_0 + \frac{2}{3}v_1 & , i = 0 \\ \frac{1}{3}v_1 + \frac{2}{3}v_0 & , i = 1 \\ \frac{1}{2}(v_0 + v_1) & , i > 1 \end{cases}. \tag{13}$$

We can justify it as a very entangled representation. But the latent code value change with respect to the factor value is what BetaVAE and FactorVAE scores care about.

FactorVAE calculates the variance of latent code, i.e. $Var(c)$, when changing the factors to represent the factor-dimension response degree. During calculating the FactorVAE score, one factor is always fixed. So we have

$$Var_{\text{fix}v_0}(c_i) = \begin{cases} \frac{1}{9} & , i = 0 \\ \frac{1}{36} & , i = 1 \\ \frac{1}{16} & , i > 1 \end{cases}, \quad Var_{\text{fix}v_1}(c_i) = \begin{cases} \frac{1}{36} & , i = 0 \\ \frac{1}{9} & , i = 1 \\ \frac{1}{16} & , i > 1 \end{cases}, \tag{14}$$

so for each factor, we have a single dimension that gives the minimum variance. In this case, the FactorVAE score uses a majority vote to determine the dimension with the smallest variance with respect to the value change of a factor as the "most disentangled" dimension to it. And the vote accuracy is regarded as the FactorVAE score. So, it is clear that because of the existence of the lucky dimension $j = i$, the FactorVAE score can be expected to approach 100, indicating "perfectly disentangled"! With the increase of model dimension, the chance of having such a lucky dimension increases as well in real cases. The bias is that FactorVAE only needs one dimension which has a low response to all other factors except for one factor to get high scores.

Table 8: *Logistic Regression* accuracy results of the full representation space and top-k subspace. Methods in gray are contrastive self-supervised learning methods.The best results of all models are **bolded**. The best results on low-dimensional space are underlined.

Space	Model	dSprites	Shapes3D	Cars3D	SmallNORB	CelebA
Full Space	β-VAE	34.5 (3.0)	35.6 (2.6)	65.0 (1.6)	48.6 (0.5)	82.1 (0.1)
	β-TCVAE	32.1 (3.4)	40.3 (2.1)	68.3 (1.4)	47.4 (0.7)	**85.2 (0.1)**
	FactorVAE	30.4 (2.5)	20.7 (2.3)	56.1 (1.3)	42.6 (1.3)	80.4 (0.2)
	DIP-VAE-I	30.3 (0.9)	37.6 (1.5)	68.5 (1.2)	49.6 (0.5)	**85.2 (0.1)**
	DIP-VAE-II	30.9 (1.9)	–	67.2 (1.7)	48.5 (0.5)	–
	MoCo	46.3 (1.3)	78.3 (2.1)	61.8 (1.4)	56.3 (0.7)	**85.2 (0.3)**
	MoCov2	52.0 (0.3)	76.5 (0.3)	63.1 (1.2)	53.8 (0.5)	84.9 (0.1)
	BarlowTwins	41.7 (0.2)	74.5 (0.8)	79.0 (0.2)	**62.0 (0.1)**	83.1 (0.2)
	SimSiam	40.3 (0.1)	**80.9 (3.4)**	45.3 (4.6)	54.5 (0.5)	84.0 (0.2)
	BYOL	**54.6 (0.2)**	71.5 (1.3)	**81.2 (0.4)**	57.2 (1.1)	81.9 (0.2)
Top-k Subspace	MoCo	22.7 (1.4)	43.7 (1.1)	38.1 (1.2)	34.5 (0.2)	85.1 (0.2)
	MoCov2	26.0 (1.0)	33.4 (0.2)	32.6 (0.8)	31.4 (1.1)	84.9 (0.1)
	BarlowTwins	29.1 (0.7)	44.4 (1.3)	43.1 (1.4)	34.5 (0.4)	82.9 (0.2)
	SimSiam	38.0 (0.6)	61.8 (1.0)	29.4 (3.6)	37.7 (2.5)	83.8 (0.2)
	BYOL	37.9 (0.6)	46.3 (0.5)	43.3 (0.9)	40.2 (0.8)	81.9 (0.2)

On the other hand, the BetaVAE score calculates the latent code value change to represent the response between factors and dimensions. This is represented by the absolute value of the first-order derivative. So for the i-th dimension with respect to the j-th factor as R_{ij}, we have

$$R_{ij} = \begin{cases} \frac{1}{3} & , i = j, i \leq 1 \\ \frac{2}{3} & , i \neq j, i \leq 1 \\ \frac{1}{2} & , otherwise \end{cases} . \tag{15}$$

BetaVAE score uses a learnable linear classifier to determine the dimension. In this case, prediction j from $R_{\cdot j}$ can achieves 100% accuracy. So similarly, it also requires one "lucky" dimension to make the prediction, offering an advantage to high-dimensional models. Moreover, because of the powerful capacity of a learnable linear classifier, the BetaVAE score is close to 100 for many cases as in Table 4. It thus has another flaw of low distinguishability of methods' disentanglement properties.

On the other hand, we could now calculate the MED score for this situation. The results turns out to be MED$(c) = 1 - log2 = 30.7\%$ for any value of D. This is a relatively low score and we think it makes better sense in such entangled cases compared to the falsely high score from BetaVAE and FactorVAE scores.

Situation 2 Let us build another case as in the discussion of DCI above

$$c_i = \begin{cases} v_0 & , i = 0 \\ v_1 & , i = 1 \\ \frac{1}{2}(v_0 + v_1) & , i > 1 \end{cases} . \tag{16}$$

FactorVAE and BetaVAE scores can still give a "perfect" score over such a representation. However, when the dimension D is very large, only a small subset of the representation dimensions is disentangled while most are highly entangled. As a higher-dimensional model has a better chance to get a lucky dimension, maintaining the perfect scoring with increasing dimensions is not aligned with our intuition and expectation. From this sense, Top-K MED has its special use. When $D = 1000$, we know MED$(c) = 30.8\%$, but we can still get a subset of the 1000 dimensions to get a high Top-K MED score. Such a subset of representations is more compact and has better explainability to the factor variance.

With all the situations constructed and analyzed above, we could notice the failure of existing metrics to achieve evaluation results (1) meaningful, (2) fair, and (3) aligned with the human sense to disentanglement in some scenarios. On the other hand, our proposed MED always keeps the evaluation and comparison reasonable and aligned with the expectation from the human high-level institution.

G Properties of the Top-k Subspace

We propose the top-k selection process to study the partial disentangle properties. Here we provide more details. We first visualize the top-k selection process based on the heatmap of the importance matrix. Then we will discuss the effects of k and the downstream performance comparison of the top-k subspace and full space.

Visualization of the top-k Process We describe the top-k process in the Section 3.3. Correspondingly, here we colored the picked dimensions yellow in the Figure 15. It shows that most picked dimensions only have one entry much brighter than the others, indicating they are well-disentangled. Since *orientation* is ill-defined in dSprites, BYOL tends to encode it with *shape*. Hence the most disentangled dimensions for *shape* also capture information of *orientation*.

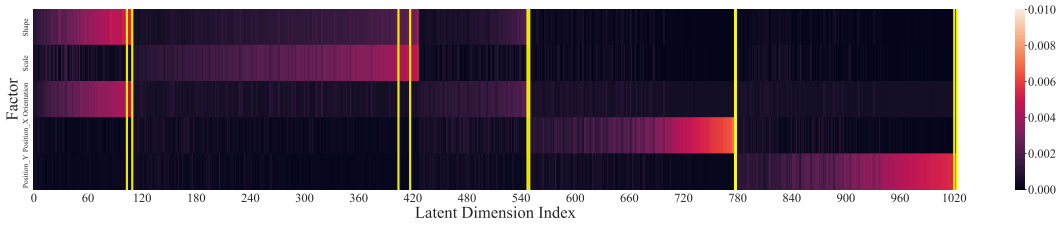

Figure 15: The visualization of the top-k process. Based on Figure 8c, the selected dimensions are highlighted with yellow boxes. For dSprites we set $k = 2$. The indices picked are 103, 109, 404, 417, 547, 549, 777, 778, 1020, and 1023.

Ablation on k The parameter k is designed as an evaluation choice. For analytical purposes, we choose the value of k to match the dimension of VAE methods in the main text. To investigate the influence of k, we set $k = 1, 2, 10, 50, 100$ and evaluate the top-k MED scores of 1000-d VAEs and BYOL on dSprites. The results are shown in Figure 16. We find that with different values of k, the ranks of top-k MED scores remain unchanged. Since the top-k process is a supervised greedy selection, as k grows the scores decrease. In conclusion, top-k MED can measure the subspace disentangle properties with a wide range of k. Therefore we can choose k according to actual needs.

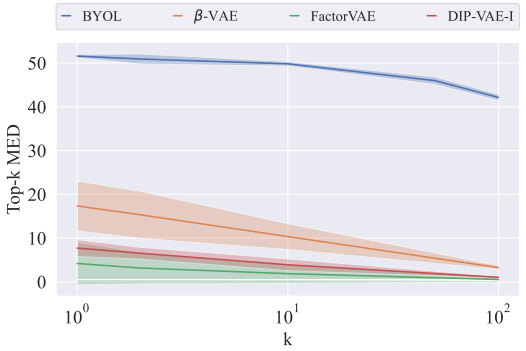

Figure 16: The influence of k in top-k MED on dSprites dataset. The evaluation results are consistent with multiple choices of k.

Downstream performance of top-k subspace We study the disentangle properties of the full space and top-k subspace in 5.3. Here we analyze the downstream performance of these spaces. We train a *Logistic Regressor* to predict each factor from the full representation or the top-k picked representation. Then we take the mean prediction accuracy of all factors as the measure of downstream performance. We compare the downstream performance of the full space of VAEs and BYOL, and top-k subspace of BYOL on multiple datasets. Here we set $k = 2$ for dSprites, Shape3D, and SmallNORB, and $k = 3$ for Cars3D and CelebA to match the dimension of VAEs. The results are in Table 8. We note that, except on CelebA, performance drops non-negligibly when completing downstream tasks with the top-k representations. This suggests that the top-k subspace is well-disentangled but other

dimensions also contain factor-related information. It is unsurprising because we cut down the dimension drastically (from 1000 to 10 on synthetic datasets and from 1000 to 100 on CelebA). However, we find that the top-k spaces learned by contrastive methods achieve better or compatible performance compared with the full spaces of VAEs on 3 out of 5 datasets (dSprites, Shapes3D, and CelebA). It is reasonable that a high-dimensional model can be better for downstream tasks as the downstream model can be trained to leverage each little piece of information from the input representations. But we get a message from the results that by drawing a subspace from the powerful CL-trained model, its performance on downstream tasks can be comparable to or superior to the low-dimensional models. And the selected low-dimensional model can have much better compactness and interpretability.

H Limitations

Our work still has some limitations. Firstly, the design of contrastive learning methods still depends on empirical practice. Therefore there lacks a widely accepted theoretical framework to analyze its disentanglement properties. As for the experiment setups, the inductive biases of different methods may potentially affect the results (such as normalization in Section D). Instead of an exhaustive search, we inherit the available best settings, i.e., the settings from DisLib (34) and the public official implementations of other methods. All hyperparameters and details have been indicated in Section A.