M2177.0043 Introduction to Deep Learning

Lecture 18: Disentanglement

Hyun Oh Song[1]

[1]Dept. of Computer Science and Engineering, Seoul National University

May 21, 2020

Last time

- Information theory
- Disentanglement

Outline

Info gain and VAE regularizer

Relationship between info gain and VAE regularizer

$$
\begin{aligned}
I(x;z) &= \int q(x,z) \log \frac{q(x,z)}{p(x)q(z)} dxdz \\
&= \int q(x,z) \log \frac{q(x,z)}{p(x)p(z)} dxdz + \int q(x,z) \log \frac{p(z)}{q(z)} dxdz \\
&= \int p(x)q(z \mid x) \log \frac{q(z \mid x)}{p(z)} dxdz + \int q(z) \log \frac{p(z)}{q(z)} dz \\
&= \underbrace{\mathbb{E}_{x \sim p(x)} D_{KL}(q(z \mid x) \parallel p(z))}_{\text{VAE regularizer}} - \underbrace{D_{KL}(q(z) \parallel p(z))}_{:=(*)} \qquad (1)
\end{aligned}
$$

Examine $D_{KL}(q(z) \parallel p(z))$

Examine (*)

$$
\begin{aligned}
D_{KL}(q(z) \parallel p(z)) &= \int q(z) \log \frac{q(z)}{p(z_1) \cdots p(z_d)} dz \\
&= \int q(z) \log \frac{q(z_1) \cdots q(z_d)}{p(z_1) \cdots p(z_d)} dz + \int q(z) \log \frac{q(z)}{q(z_1) \cdots q(z_d)} dz \\
&= \sum_{i=1}^{d} \int q(z_i) \log \frac{q(z_i)}{p(z_i)} dz_i + TC(z) \\
&= \sum_{i=1}^{d} D_{KL}(q(z_i) \parallel p(z_i)) + TC(z) \qquad (2)
\end{aligned}
$$

Putting Equation (1) and Equation (2) together

$$\underbrace{\mathbb{E}_{x\sim p(x)} D_{KL}(q(z \mid x) \parallel p(z))}_{\text{VAE regularizer}} = I(x; z) \; + \; TC(z) \; + \; \sum_i D_{KL}(q(z_i) \parallel p(z_i))$$

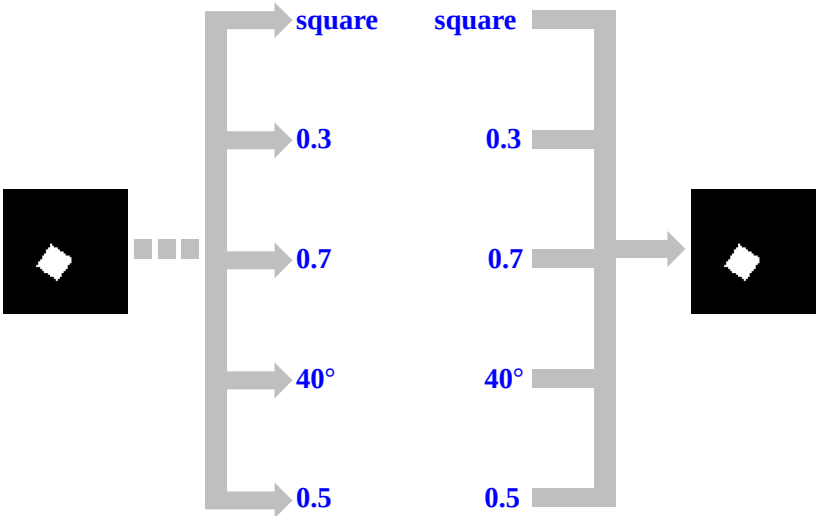
Interpretation:

- Minimizing the VAE regularizer decreases $TC(z)$, the redundancy of the learned representation z 
- However, the information gain between the data and the representation is decreased at the same time 
- Ideally, we want to increase $I(x; z)$ and decrease $TC(z)$
- Optimizing $TC(z)$ directly is hard 

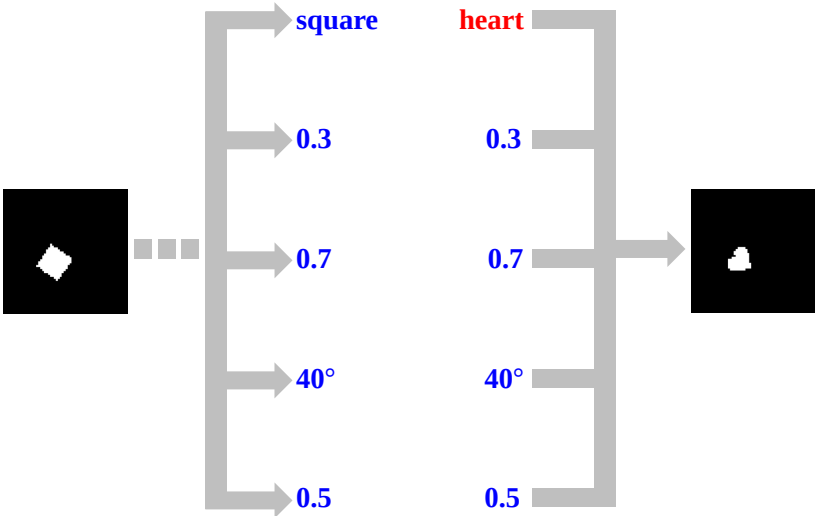
Unsupervised disentangled representation learning

- Learning to disentangle the underlying explanatory factors of data without supervision is a cruicial task for representation learning
- In a successfully disentangled representation, a single latent unit of representation should correspond to a change in a single generative factor of the data while being relatively invariant to others
- Penalizing total correlation encourages the model to learn statistically independent factors of data

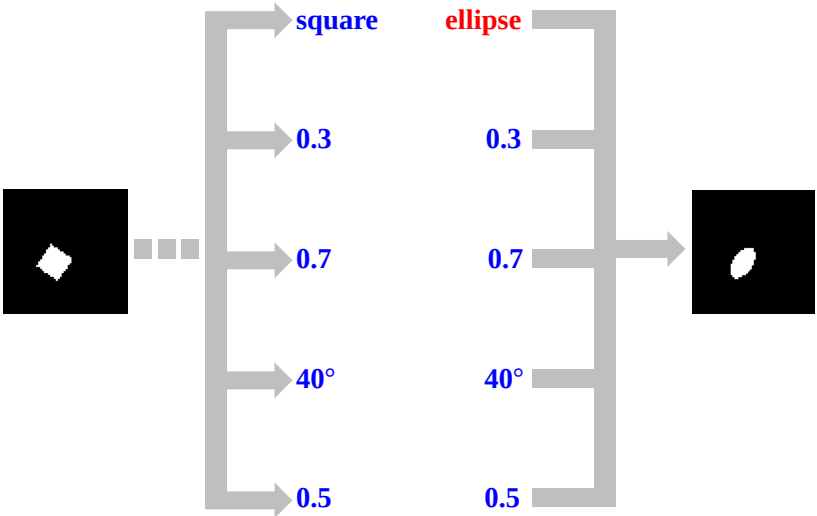
Disentanglement



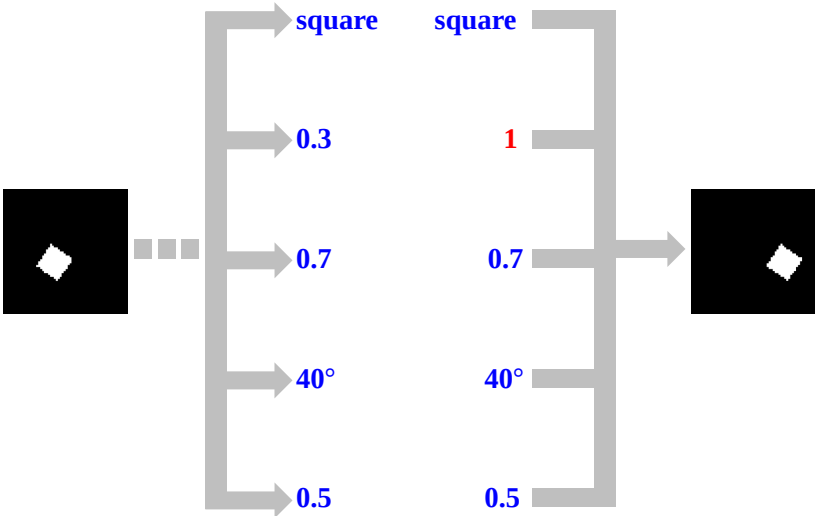
Disentanglement



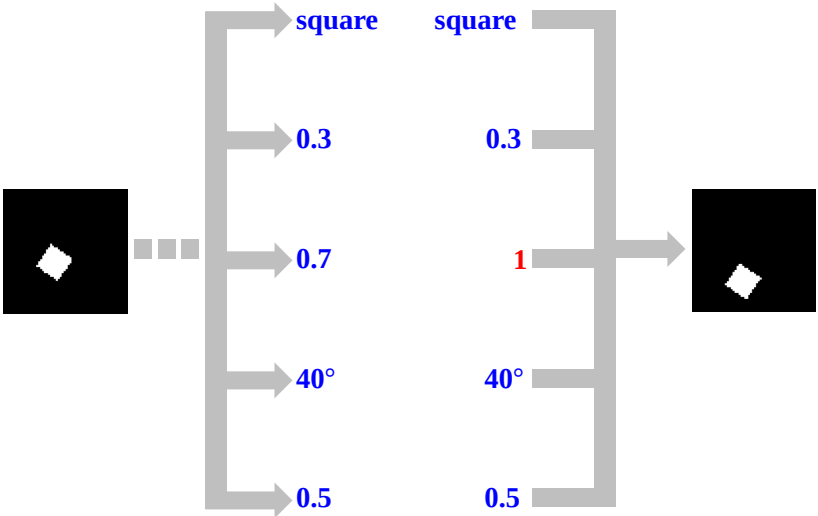
Disentanglement



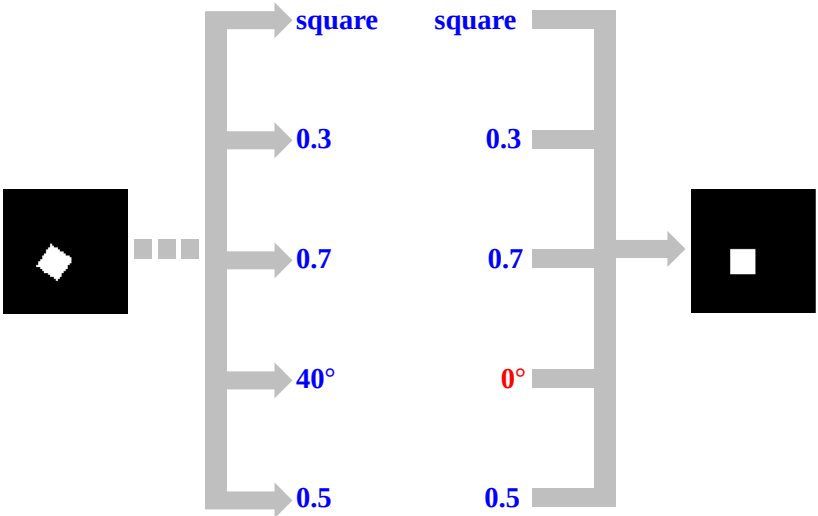
Disentanglement



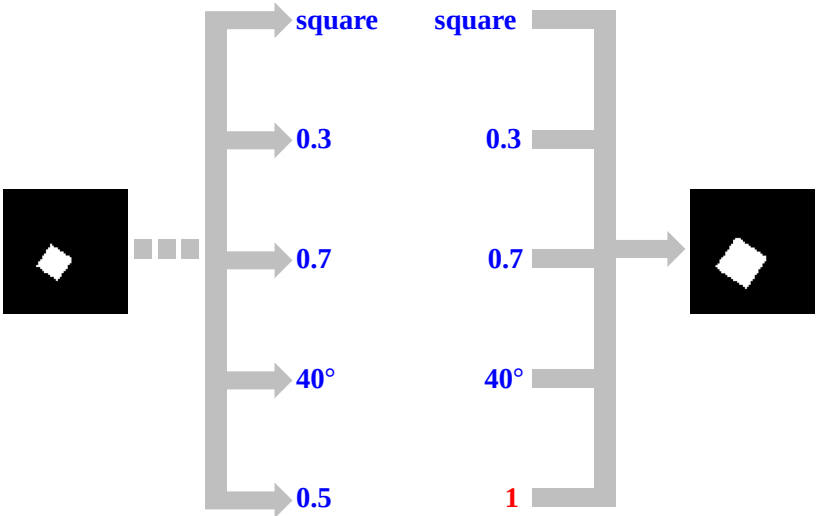
Disentanglement



Disentanglement



Disentanglement



β-VAE[1]

- If $\beta = 1$, objective same as VAE
- β-VAE sets $\beta > 1$ indirectly penalizing $TC(z)$ by increasing β coefficient to a high value (*i.e.* 10.0)

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{x \sim p(x)} \left[\mathbb{E}_{z \sim q_\phi(\cdot|x)} \log p_\theta(x \mid z) - \beta D_{KL}(q_\phi(z \mid x) \parallel p(z)) \right]$$

[1] Higgins, et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework" ICLR2017

β-VAE vs VAE

Figure: Traversal z_1, (Left) β-VAE, (Right) VAE

β-VAE vs VAE

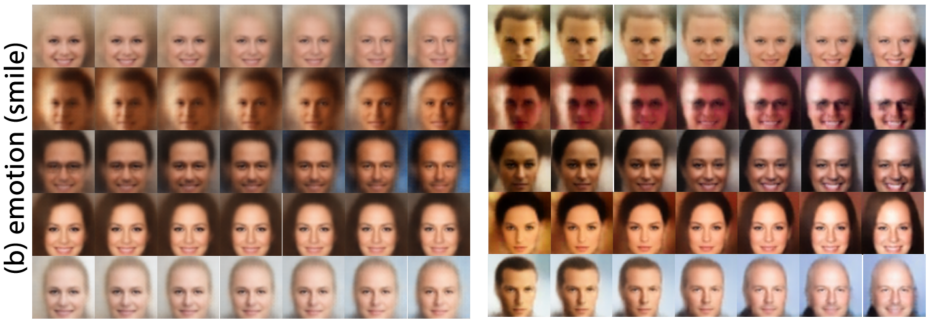


Figure: Traversal z_2, (Left) β-VAE, (Right) VAE

β-VAE vs VAE

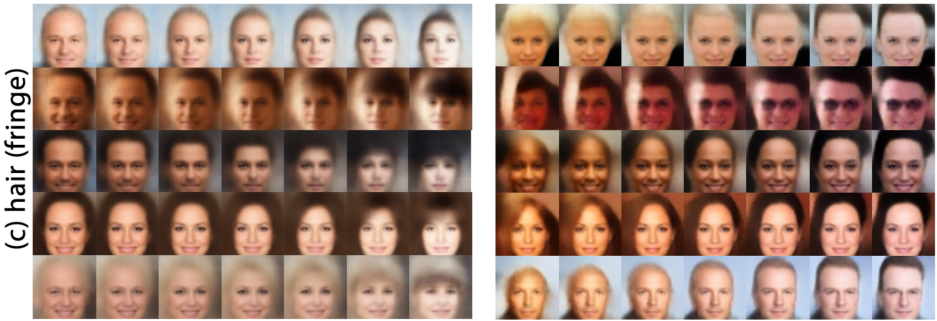


Figure: Traversal z_3, (Left) β-VAE, (Right) VAE

Adversarial density ratio trick[2]

- Given samples from p_x and p_y, we can estimate $D_{KL}(p_x \parallel p_y)$ using density-ratio trick

[2] Tschannen et al. "Recent advances in autoencoder-based representation learning" 2018.

Adversarial density ratio trick[2]

- Given samples from p_x and p_y, we can estimate $D_{KL}(p_x \parallel p_y)$ using density-ratio trick
- Trick is to express p_x and p_y as conditional distributions, conditioned on a label $c \in \{0, 1\}$, and reduce the task to binary classification

[2] Tschannen et al. "Recent advances in autoencoder-based representation learning" 2018.

Adversarial density ratio trick[2]

- Given samples from p_x and p_y, we can estimate $D_{KL}(p_x \parallel p_y)$ using density-ratio trick
- Trick is to express p_x and p_y as conditional distributions, conditioned on a label $c \in \{0, 1\}$, and reduce the task to binary classification
- Let $p_x(x) = p(x \mid c = 1), p_y(x) = p(x \mid c = 0)$, and consider a discriminator D trained to predict the probability that its input is a sample from distributions p_x rather than p_y, *i.e.* predict $p(c = 1 \mid x)$. The density ratio can be expressed as,

$$\frac{p_x(x)}{p_y(x)} = \frac{p(x \mid c = 1)}{p(x \mid c = 0)} = \frac{p(c = 1 \mid x)}{p(c = 0 \mid x)} \approx \frac{D(x)}{1 - D(x)},$$

where we assume $p(c = 1) = p(c = 0)$

[2] Tschannen et al. "Recent advances in autoencoder-based representation learning" 2018.

- Given N i.i.d. samples $\{x^{(i)}\}_{i=1}^{N}$ from p_x and a trained classifier D, one can estimate the KL-divergence by simple computing

$$D_{KL}(p_x \parallel p_y) = \mathbb{E}_p \log\left(\frac{p_x(x)}{p_y(x)}\right) \approx \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{D(x^{(i)})}{1 - D(x^{(i)})}\right)$$

- Note, however, the resulting objective does not necessarily lower-bound the marginal log-likelihood of the data

FactorVAE[3]

- FactorVAE approximately penalizes $TC(z)$ via density ratio trick

$$\frac{1}{N}\sum_{i=1}^{N}\left[\mathbb{E}_{q(z|x^{(i)})}[\log p(x^{(i)} \mid z)] - D_{KL}(q(z \mid x^{(i)}) \parallel p(z))\right] \\ \underbrace{-\gamma D_{KL}\left(q(z) \parallel \prod_{j=1}^{d} q(z_j)\right)}_{:=TC(z)}$$

[3] Kim and Minh "Disentangling by Factorising" ICML2018.

- We can sample from $q(z)$ efficiently by first choosing a datapoint $x^{(i)}$ uniformly at random and then sampling from $q(z \mid x^{(i)})$.
- We can also sample from $\bar{q}(z) := \prod_{j=1}^{d} q(z_j)$ by sampling a batch from $q(z)$ and then randomly permuting across the batch for each latent dimension (Algorithm 1). As long as the batch is large enough, the distribution of these samples will closely approximate $\bar{q}(z)$

Algorithm 1 `permute_dims`

Input: $\{z^{(i)} \in \mathbb{R}^d : i = 1, \ldots, B\}$
for $j = 1$ **to** d **do**
  $\pi \leftarrow$ random permutation on $\{1, \ldots, B\}$
  $(z_j^{(i)})_{i=1}^{B} \leftarrow (z_j^{(\pi(i))})_{i=1}^{B}$
end for
Output: $\{z^{(i)} : i = 1, \ldots, B\}$

- Having access to samples from both distributions ($q(z)$ and $\bar{q}(z)$) allows us to minimize the TC using the density ratio trick

- Suppose we have a trained discriminator network D that outputs an estimate of the probability $D(z)$ that its inputs is a sample from $q(z)$ rather than from $\bar{q}(z)$. Then we have

$$TC(z) = D_{KL}(q(z) \parallel \bar{q}(z)) = \mathbb{E}_{q(z)} \log \frac{q(z)}{\bar{q}(z)} \approx \mathbb{E}_{q(z)} \log \frac{D(z)}{1 - D(z)}$$

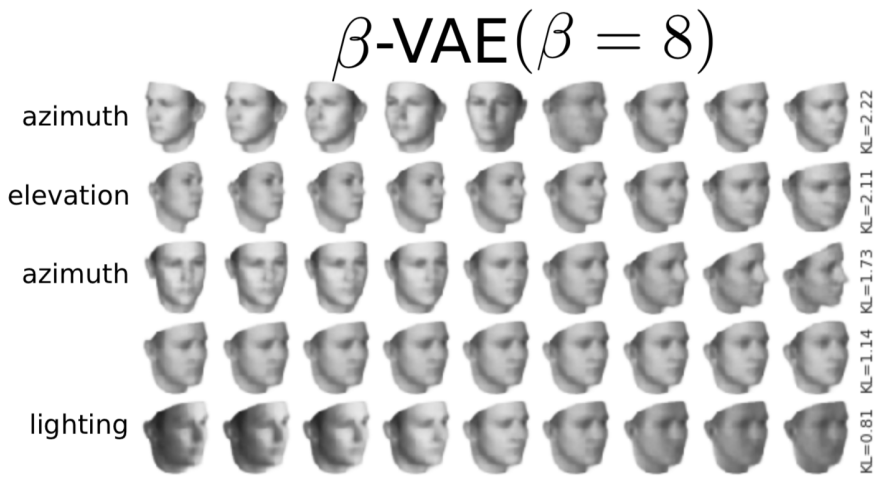
FactorVAE[4]

- Train VAE and the discriminator jointly. The discriminator is trained to classify between samples from $q(z)$ and $\bar{q}(z)$ thus learning to approximate the density ratio needed for estimating TC. See Algorithm 2

Algorithm 2 FactorVAE

Input: observations $(x^{(i)})_{i=1}^{N}$, batch size m, latent dimension d, γ, VAE/Discriminator optimisers: g, g_D
Initialize VAE and discriminator parameters θ, ψ.
repeat
  Randomly select batch $(x^{(i)})_{i \in \mathcal{B}}$ of size m
  Sample $z_\theta^{(i)} \sim q_\theta(z|x^{(i)})\ \forall i \in \mathcal{B}$
  $\theta \leftarrow g(\nabla_\theta \frac{1}{m} \sum_{i \in \mathcal{B}} [\log \frac{p_\theta(x^{(i)}, z_\theta^{(i)})}{q_\theta(z_\theta^{(i)}|x^{(i)})} - \gamma \log \frac{D_\psi(z_\theta^{(i)})}{1 - D_\psi(z_\theta^{(i)})}])$
  Randomly select batch $(x^{(i)})_{i \in \mathcal{B}'}$ of size m
  Sample $z_\theta'^{(i)} \sim q_\theta(z|x^{(i)})$ for $i \in \mathcal{B}'$
  $(z'^{(i)}_{perm})_{i \in \mathcal{B}'} \leftarrow \texttt{permute\_dims}((z_\theta'^{(i)})_{i \in \mathcal{B}'})$
  $\psi \leftarrow g_D(\nabla_\psi \frac{1}{2m} [\sum_{i \in \mathcal{B}} \log(D_\psi(z_\theta^{(i)}))$
    $+ \sum_{i \in \mathcal{B}'} \log(1 - D_\psi(z'^{(i)}_{perm}))])$
until convergence of objective.

[4] Kim and Minh "Disentangling by Factorising" ICML2018.


β-VAE($\beta = 8$)
azimuth
elevation
azimuth
lighting
KL=2.22
KL=2.11
KL=1.73
KL=1.14
KL=0.81

FactorVAE$(\gamma = 10)$

CascadeVAE[5]

Proposition 1

The mutual information between a single random variable and the rest can be factorized as

$$I(z_{1:i-1}; z_i) = TC(z_{1:i}) - TC(z_{1:i-1})$$

Proposition 2

The mutual information between x and partitions of $z = [z_1, z_2]$ can be factorized as,

$$I(x; [z_1, z_2]) = I(x; z_1) + I(x; z_2) - I(z_1; z_2)$$

~~See reference for the proofs.~~

[5] Jeong & Song "Learning Discrete and Continuous Factors of Data via Alternating Disentanglement" ICML2019.

- By telescoping sum and proposition1,

$$
\begin{aligned}
TC(z) &= \underbrace{TC(z_{1:2})}_{=I(z_1;z_2)} + \sum_{i=3}^{m} \left(\underbrace{TC(z_{1:i}) - TC(z_{1:i-1})}_{=I(z_{1:i-1};z_i)} \right) \\
&= \sum_{i=2}^{m} I(z_{1:i-1}; z_i).
\end{aligned}
$$

- By telescoping sum and proposition1,

$$\begin{aligned} TC(z) &= \underbrace{TC(z_{1:2})}_{=I(z_1;z_2)} + \sum_{i=3}^{m} \left(\underbrace{TC(z_{1:i}) - TC(z_{1:i-1})}_{=I(z_{1:i-1};z_i)} \right) \\ &= \sum_{i=2}^{m} I(z_{1:i-1}; z_i). \end{aligned}$$

- We aim at penalizing $TC(z)$ by sequentially penalizing the individual summand $I(z_{1:i-1}; z_i)$.

- By proposition2,

$$I(x; z_{1:i}) = I(x; z_{1:i-1}) + I(x; z_i) - I(z_{1:i-1}; z_i).$$
$$\uparrow$$

- This factorization motivates a maximization algorithm sequentially updating the left hand side $I(x; z_{1:i})$ for all $i = 2, \ldots, m$ which in turn minimizes each summand, $\boldsymbol{I(z_{1:i-1}; z_i)}$.

- By proposition2,

$$I(x; z_{1:i}) = I(x; z_{1:i-1}) + I(x; z_i) - I(z_{1:i-1}; z_i).$$
$$\uparrow \qquad\qquad \bullet \qquad\qquad \uparrow \qquad\qquad \downarrow$$

- This factorization motivates a maximization algorithm sequentially updating the left hand side $I(x; z_{1:i})$ for all $i = 2, \ldots, m$ which in turn minimizes each summand, $\boldsymbol{I(z_{1:i-1}; z_i)}$.

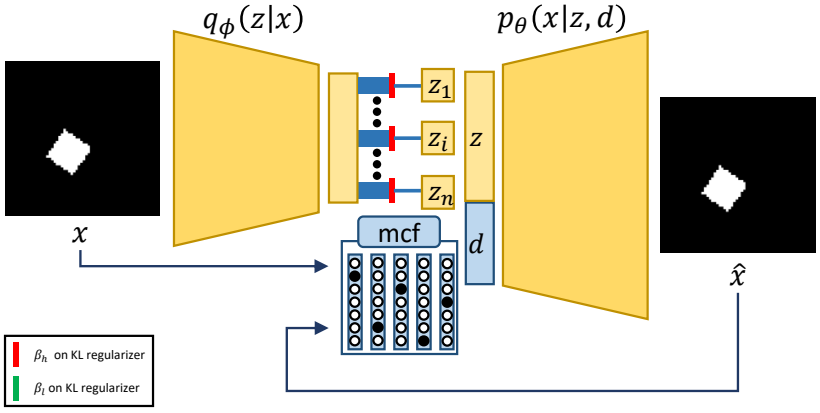
- By proposition2,

$$I(x; z_{1:i}) = I(x; z_{1:i-1}) + I(x; z_i) - I(z_{1:i-1}; z_i).$$
$$\uparrow \qquad\qquad \bullet \qquad\qquad \uparrow \qquad\qquad \downarrow$$

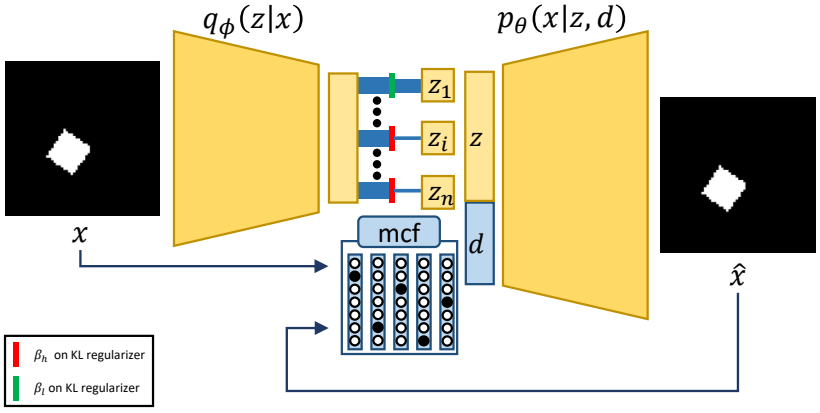
- This factorization motivates a maximization algorithm sequentially updating the left hand side $I(x; z_{1:i})$ for all $i = 2, \ldots, m$ which in turn minimizes each summand, $\boldsymbol{I(z_{1:i-1}; z_i)}$.

- Sequentially, we maximize $I(x; z_{1:i})$ by penalizing $z_{i+1:m}$ with high β $(:= \beta_h)$ and the others with small β $(:= \beta_l)$.

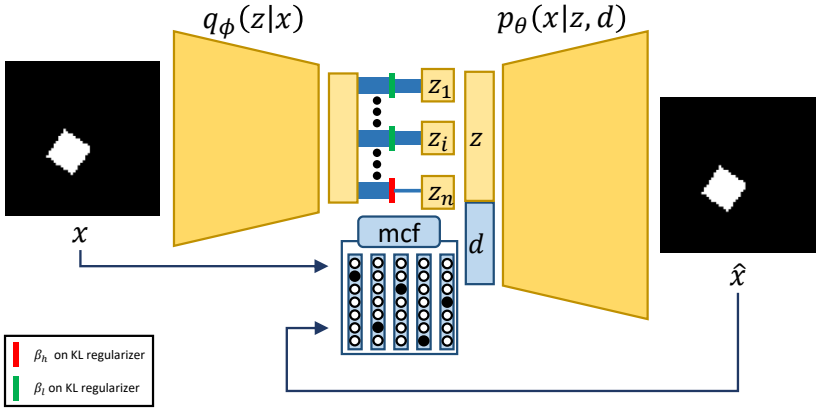
- The first term does not change as much since we are only applying gradient updates as opposed to direct maximization



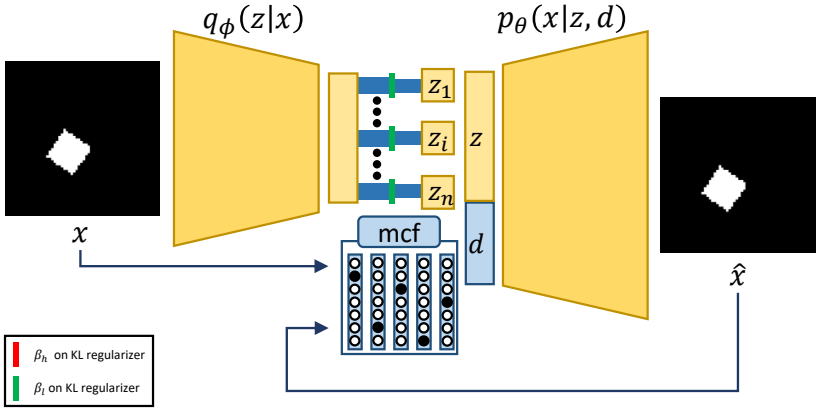
- Every latent dimensions are heavily penalized with β_h. Each penalty on latent dimension is sequentially relieved one at a time with β_l in a cascading fashion.



- Every latent dimensions are heavily penalized with β_h. Each penalty on latent dimension is sequentially relieved one at a time with β_l in a cascading fashion.



- Every latent dimensions are heavily penalized with β_h. Each penalty on latent dimension is sequentially relieved one at a time with β_l in a cascading fashion.



- Every latent dimensions are heavily penalized with β_h. Each penalty on latent dimension is sequentially relieved one at a time with β_l in a cascading fashion.

Joint modeling discrete and continuous generative factors

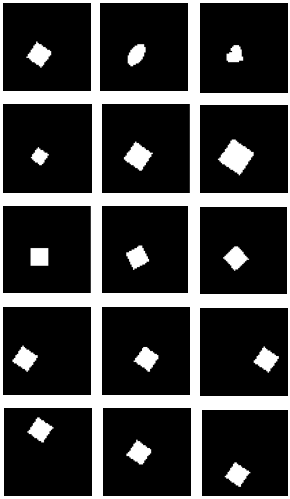
- Some of the generative factors are inherently discrete.
  i.e. Number/class/type of objects in the image, number of light sources in the image, number of speakers in a wave file, etc.

- Refer to "Learning Disentangled Joint Continuous and Discrete Representations, Dupont, Neurips 2018" and "Learning Discrete and Continuous Factors of Data via Alternating Disentanglement, Jeong & Song, ICML 2019" for details.

dSprites Dataset

- This dataset is used to assess the disentanglement properties of unsupervised learning methods.
- There are 6 ground truth independent latent factors (e.g. color, shape, scale, orientation, position x, position y).
- There are 737280 $(= 1 \times 3 \times 6 \times 40 \times 32 \times 32)$ images.

dSprites Dataset Example



- Shape (discrete) : square, ellipse, heart
- Scale: 6 values linearly spaced in $[0.5, 1]$
- Orientation: 40 values in $[0, 2\pi]$
- Position X: 32 values in $[0, 1]$
- Position Y: 32 values in $[0, 1]$

Quantitative disentanglement evaluation

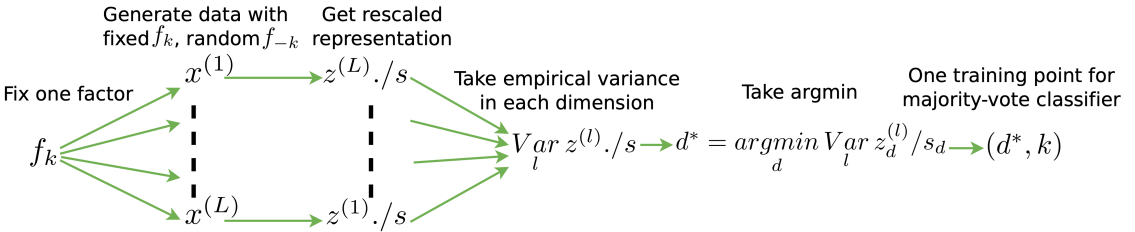
- dSprites allows users to generate $3 \times 6 \times 40 \times 32 \times 32 = 737,280$ real images

- Create a count matrix $C = 0_{d,k}$ where d is the number of latent dimensions (*i.e.* $\dim(z)$) and k is the number of true generative factors (*i.e.* $k = 6$ for dSprites) and repeat
  1. First, fix a real generative factor l (*i.e.* size) and randomly sample $L = 1,000$ real images. Call these $\{x^{(i)}\}_{i=1}^{L}$
  2. Encode each real images. Call these $\{z^{(i)}\}_{i=1}^{L}$
  3. Per each latent dimension, compute the variance among the $L = 1,000$ samples. Find the latent dimension with the least variance. $d^* = \text{argmin}_{j \in [d]} \mathbb{V}(z_j^{(1)}, \ldots, z_j^{(L)})$
  4. Cast a vote `C[d*][l] += 1`

- `accuracy` $= \|C\|_{\infty,1} / \|C\|_1$. The matrix norm in the numerator means first computing the row-wise ℓ_∞ norm and then computing ℓ_1 norm on the resulting vector.

Quantitative disentanglement evaluation[6]

- Overview diagram of the evaluation protocol



[6] Kim and Minh "Disentangling by Factorising" ICML2018.

Quantitative results on dSprites

Method	m	Mean (std)	Best
β VAE			
($\beta = 10.0$)	5	70.11 (7.54)	84.62
($\beta = 4.0$)	10	74.41 (7.68)	88.38
FactorVAE	5	81.09 (2.63)	85.12
	10	82.15 (0.88)	88.25
CascadeVAE-C			
($\beta_l = 0.7$)	5	81.69 (3.14)	88.38
($\beta_l = 1.0$)	10	81.74 (2.97)	87.38
JointVAE	6	74.51 (5.17)	91.75
	4	73.06 (2.18)	75.38
CascadeVAE			
($\beta_l = 1.0$)	6	90.49 (5.28)	**99.50**
($\beta_l = 2.0$)	4	**91.34 (7.36)**	98.62

- Disentanglement score is obtained from 10 different random seed each with the best hyperparameters.
- m is the number of continous latent dimensions.

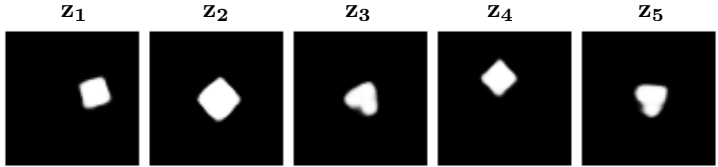
Quantitative results on dSprites

Method	m	Mean (std)	Best
JointVAE	6	44.79 (3.88)	53.14
	4	43.99 (3.94)	54.11
CascadeVAE	6	**78.84 (15.65)**	**99.66**
	4	76.00 (22.16)	98.72

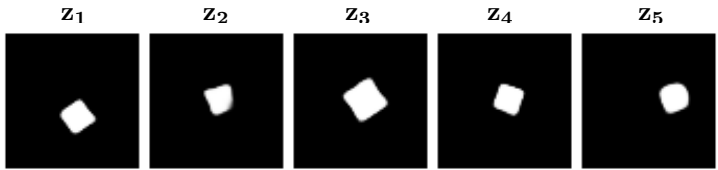
- Unsupervised classification accuracy when size of discrete variable is 3.
- Random chance : 33.33.

Latent dimension traversal in dSprites

- β-VAE

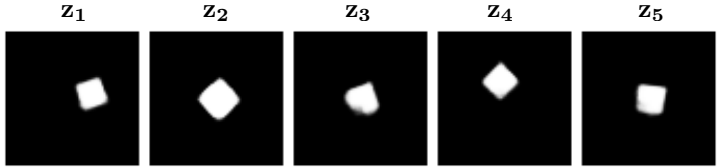


- FactorVAE

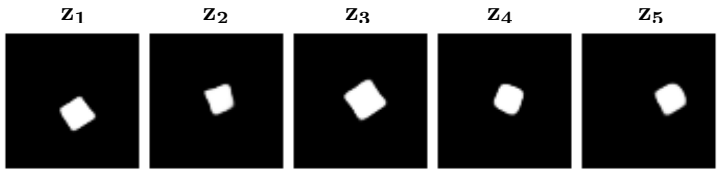


Latent dimension traversal in dSprites

- β-VAE

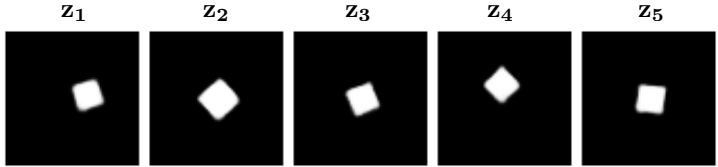


- FactorVAE

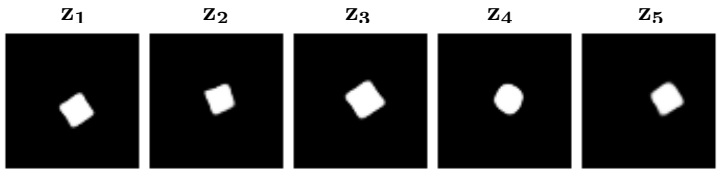


Latent dimension traversal in dSprites

- β-VAE

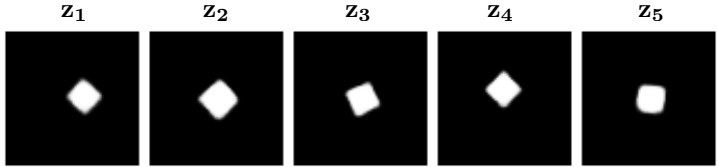


- FactorVAE

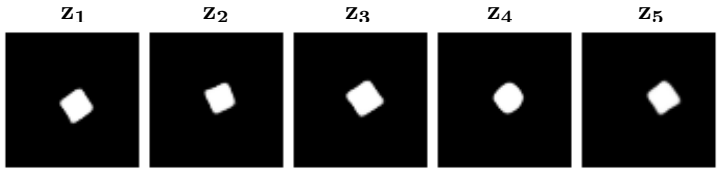


Latent dimension traversal in dSprites

- β-VAE

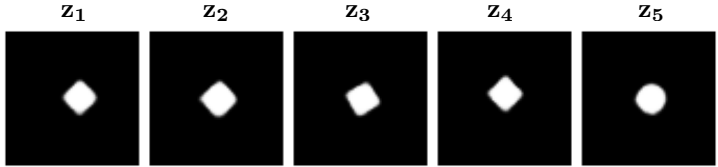


- FactorVAE

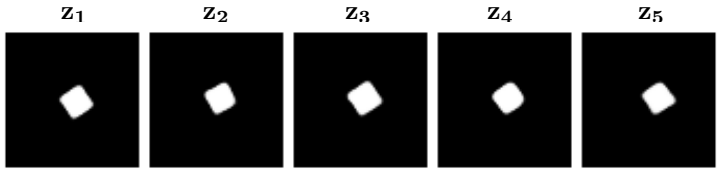


Latent dimension traversal in dSprites

- β-VAE

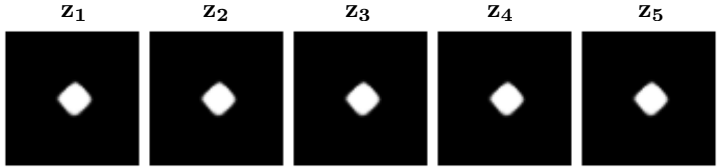


- FactorVAE

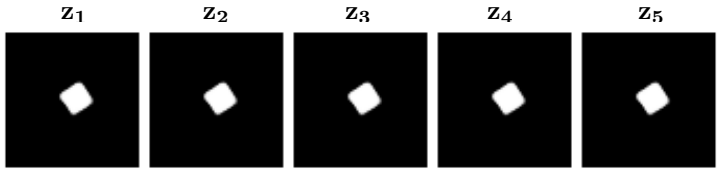


Latent dimension traversal in dSprites

- β-VAE

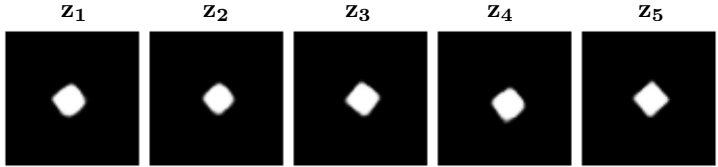


- FactorVAE

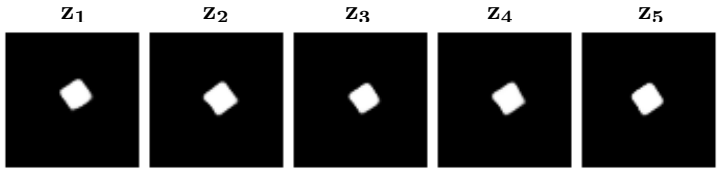


Latent dimension traversal in dSprites

- β-VAE

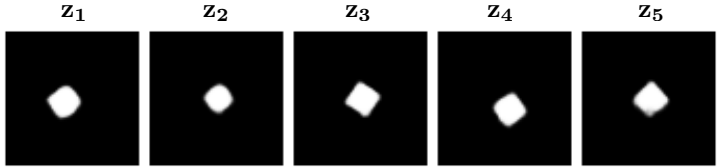


- FactorVAE

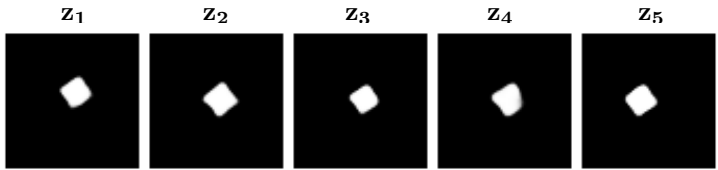


Latent dimension traversal in dSprites

- β-VAE

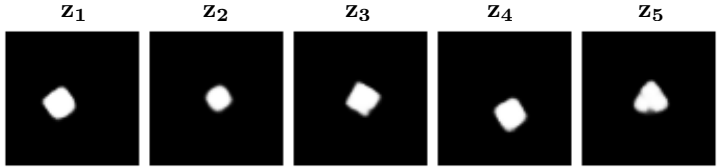


- FactorVAE

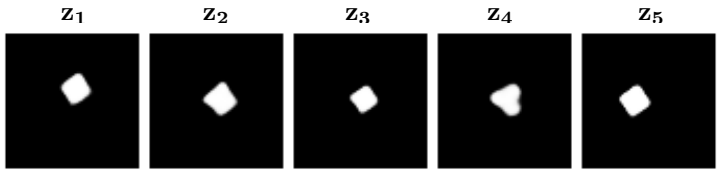


Latent dimension traversal in dSprites

- β-VAE

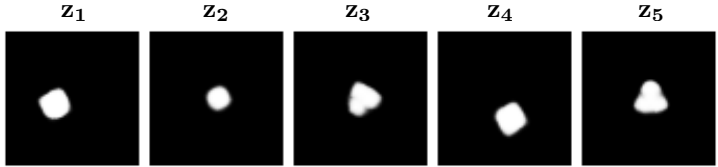


- FactorVAE

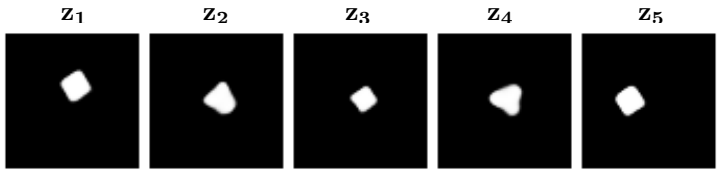


Latent dimension traversal in dSprites

- β-VAE

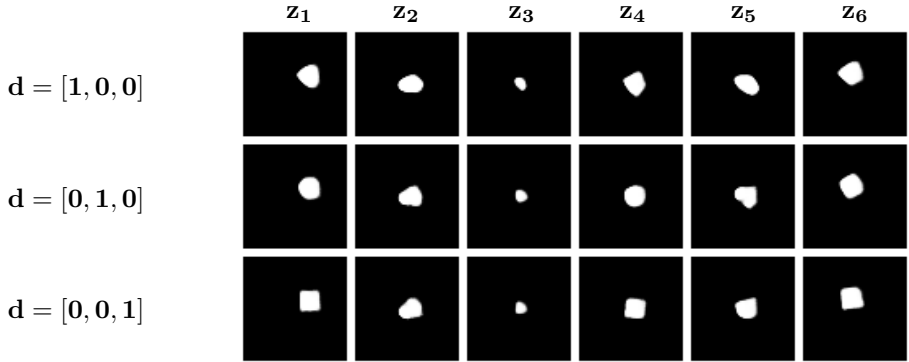


- FactorVAE



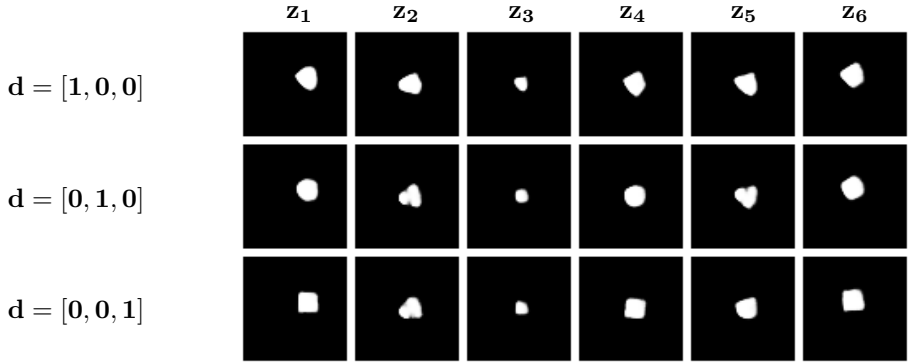
Latent dimension traversal in dSprites

- JointVAE



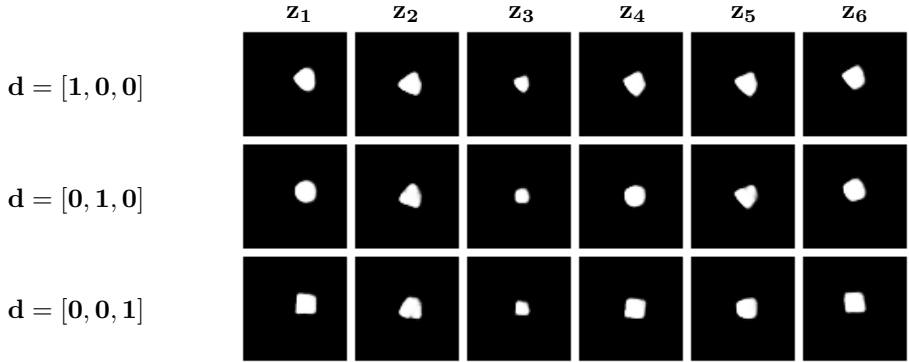
Latent dimension traversal in dSprites

- JointVAE



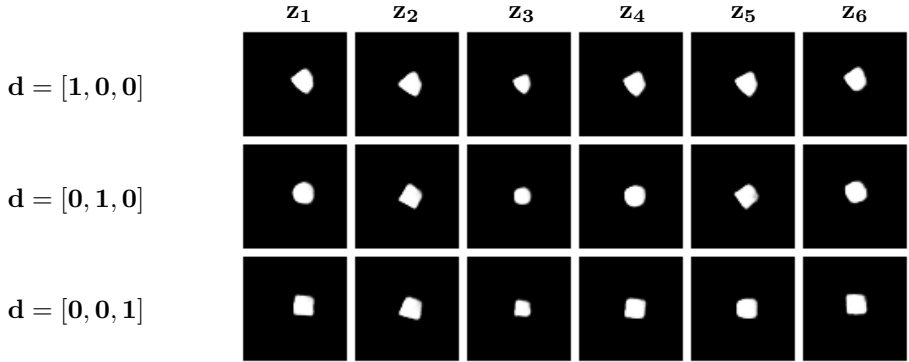
Latent dimension traversal in dSprites

- JointVAE



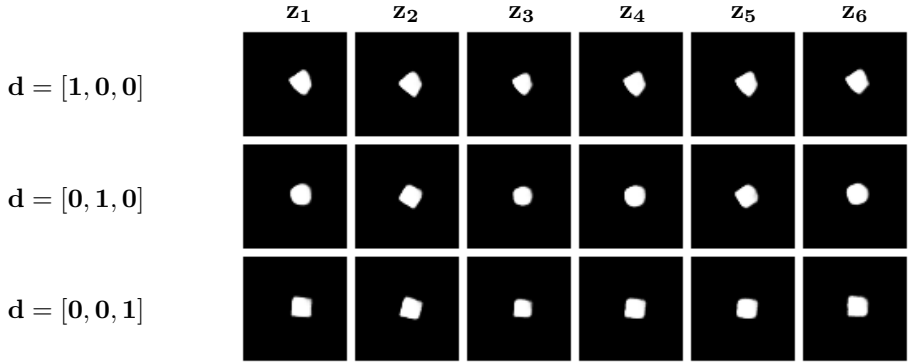
Latent dimension traversal in dSprites

- JointVAE



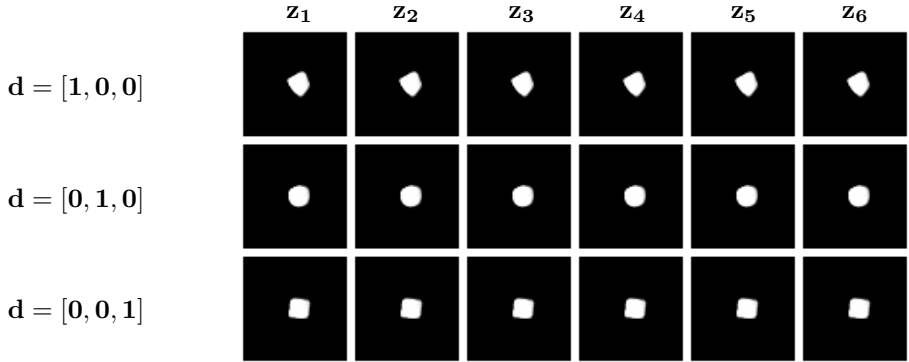
Latent dimension traversal in dSprites

- JointVAE



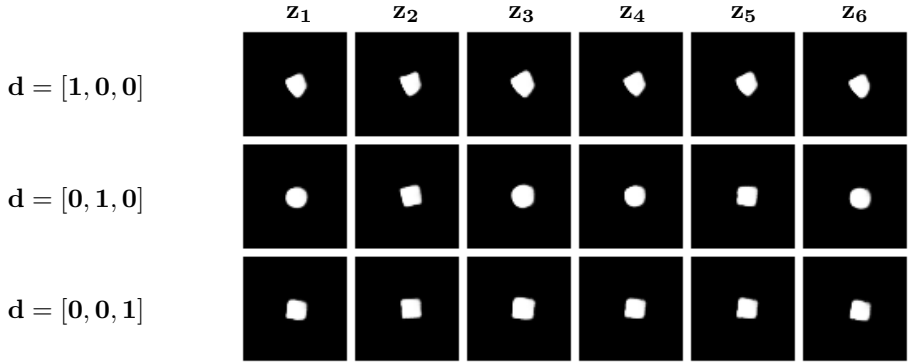
Latent dimension traversal in dSprites

- JointVAE



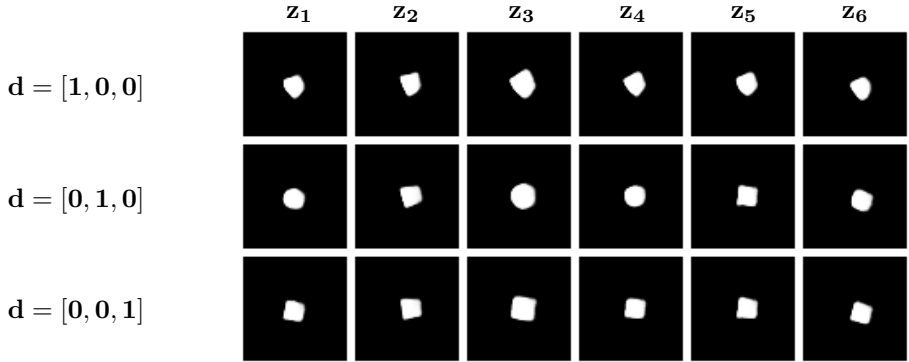
Latent dimension traversal in dSprites

- JointVAE



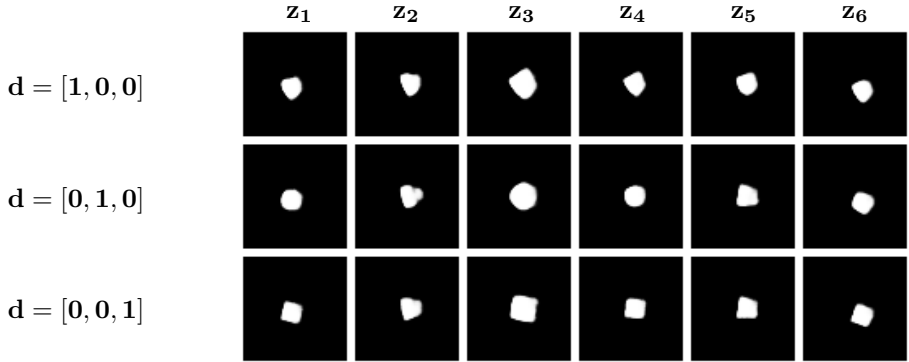
Latent dimension traversal in dSprites

- JointVAE



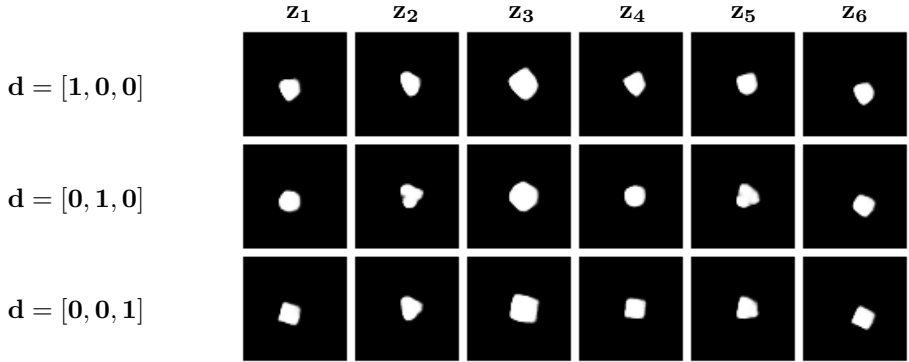
Latent dimension traversal in dSprites

- JointVAE



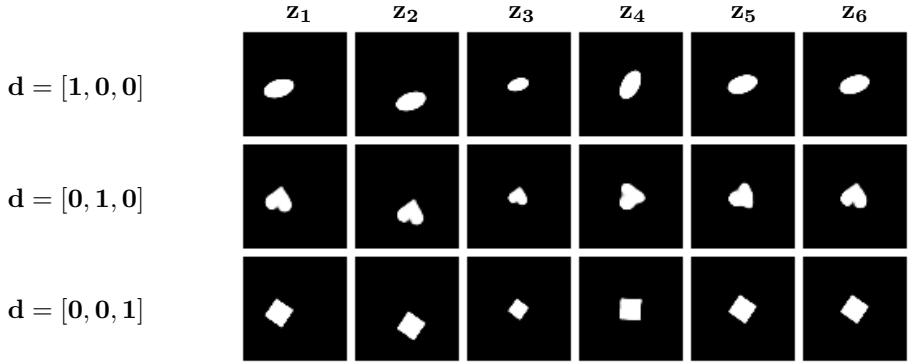
Latent dimension traversal in dSprites

- JointVAE



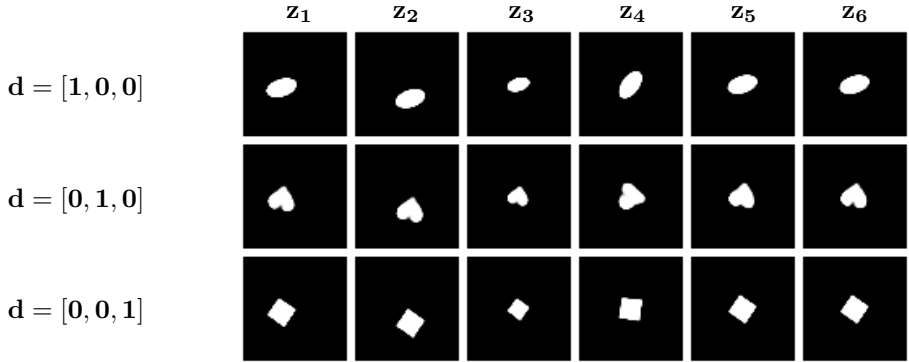
Latent dimension traversal in dSprites

- CascadeVAE



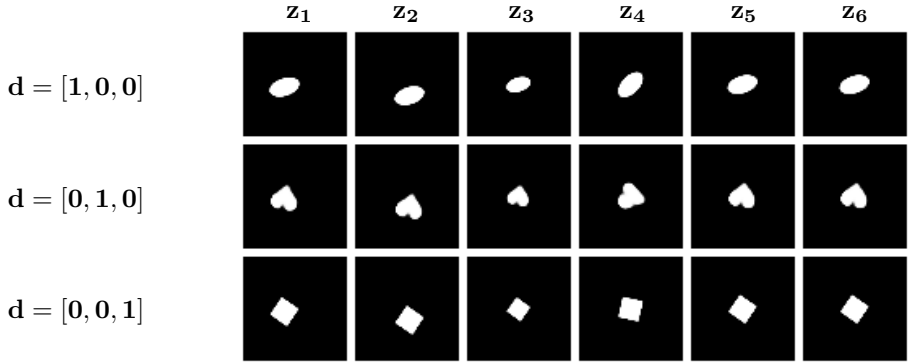
Latent dimension traversal in dSprites

- CascadeVAE



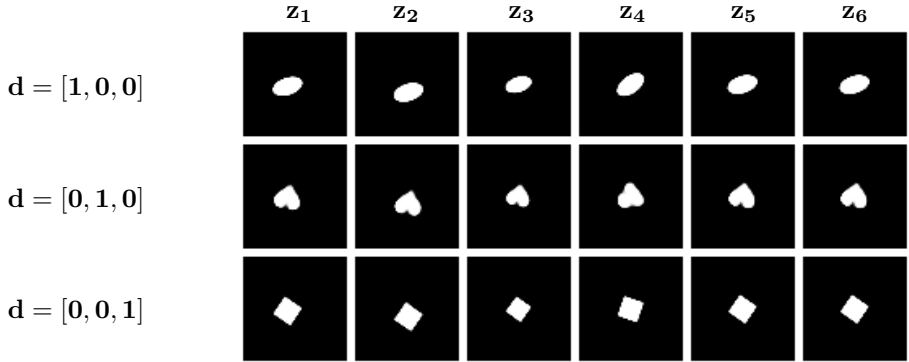
Latent dimension traversal in dSprites

- CascadeVAE



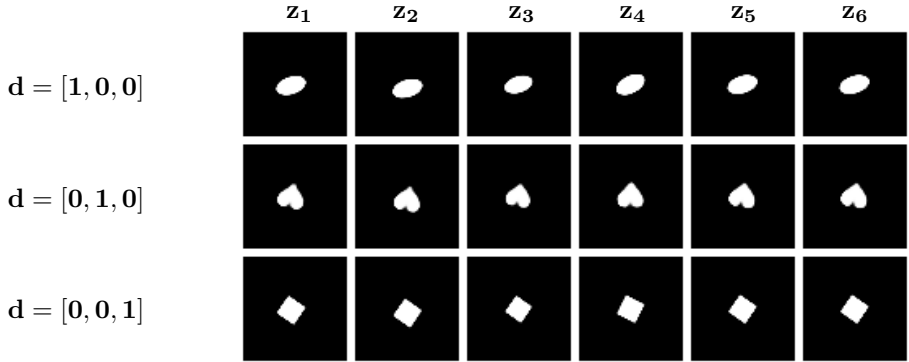
Latent dimension traversal in dSprites

- CascadeVAE



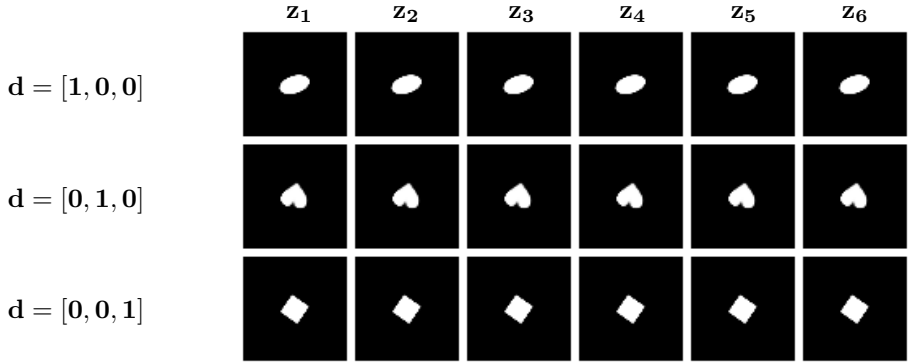
Latent dimension traversal in dSprites

- CascadeVAE



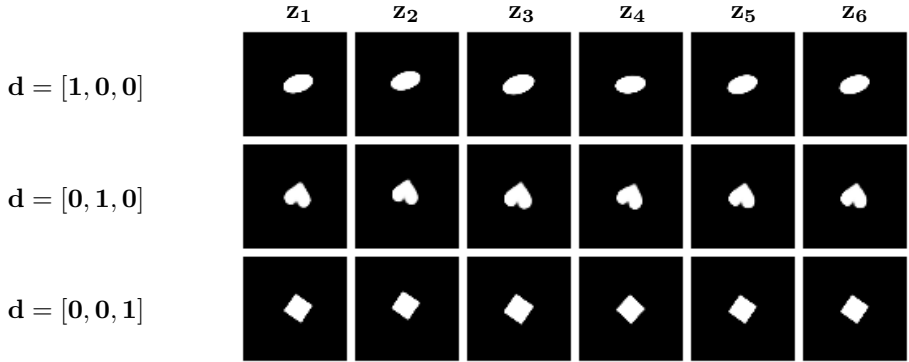
Latent dimension traversal in dSprites

- CascadeVAE



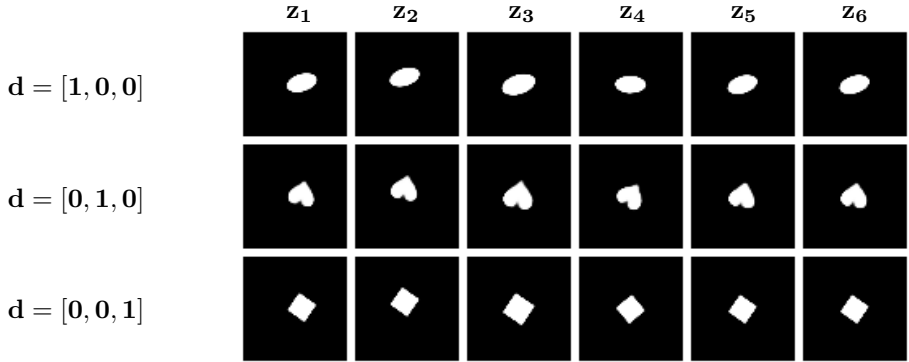
Latent dimension traversal in dSprites

- CascadeVAE



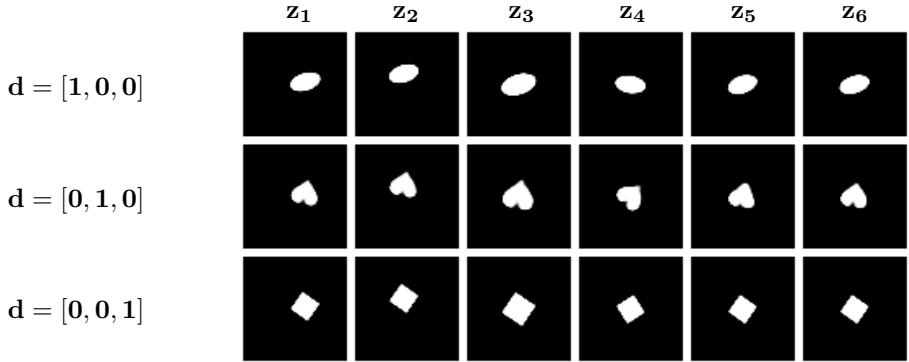
Latent dimension traversal in dSprites

- CascadeVAE



Latent dimension traversal in dSprites

- CascadeVAE



Latent dimension traversal in dSprites

- CascadeVAE