Universal Conditional Machine

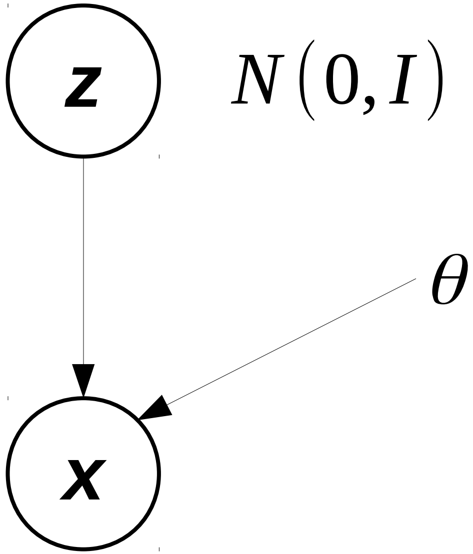
Oleg Ivanov

p(B|A)yesgroup.ru

Variational Autoencoder

$$p_{\theta}(\boldsymbol{x})$$

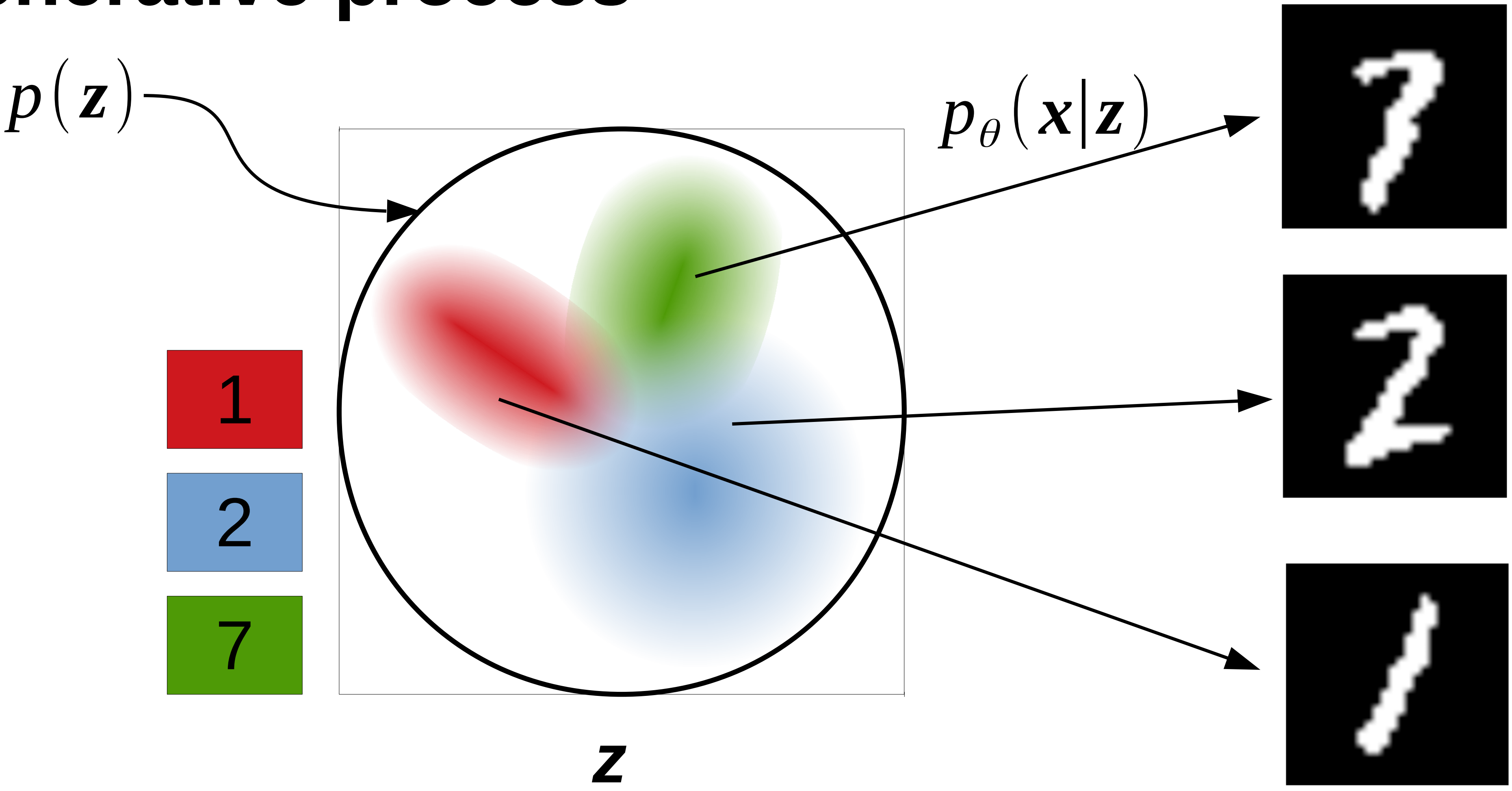
Generative process



$$p(\mathbf{z})$$

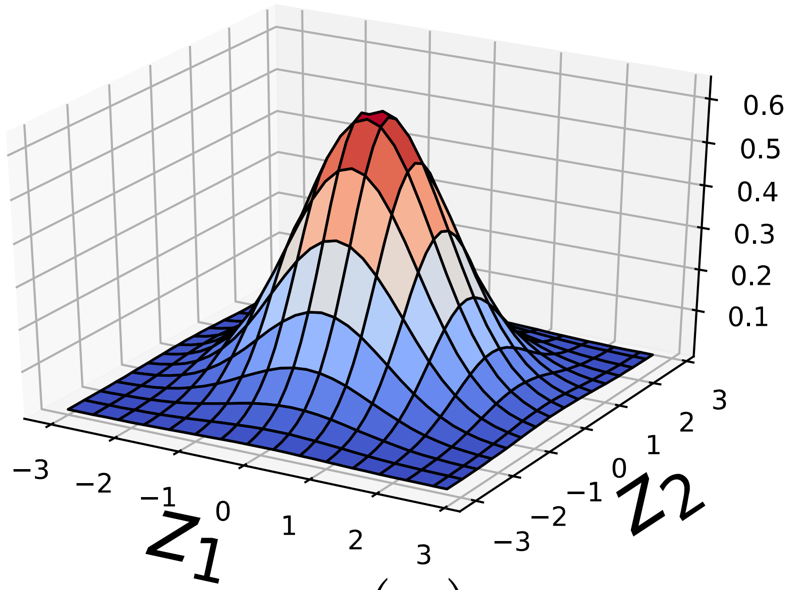
$$p_\theta(\mathbf{x}|\mathbf{z})$$

Generative process



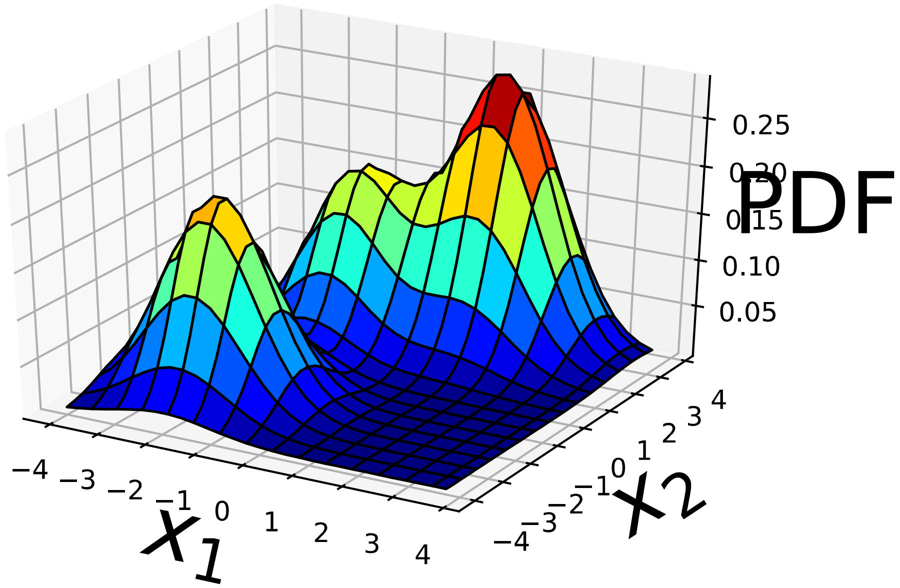
Generative process

Transform prior into distribution over objects:



$p(\boldsymbol{z})$

$p_\theta(\boldsymbol{x}|\boldsymbol{z})$



$$p_\theta(\boldsymbol{x}) = \mathop{\mathbf{E}}_{\boldsymbol{z} \sim p(\boldsymbol{z})} p_\theta(\boldsymbol{x}|\boldsymbol{z})$$

Model Log-Likelihood

$$p_\theta(\boldsymbol{x}) = \mathop{\mathbf{E}}_{\boldsymbol{z} \sim \boldsymbol{p}(\boldsymbol{z})} p_\theta(\boldsymbol{x}|\boldsymbol{z})$$

$$L(\theta) = \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \boldsymbol{p_d}(\boldsymbol{x})} \log p_\theta(\boldsymbol{x})$$

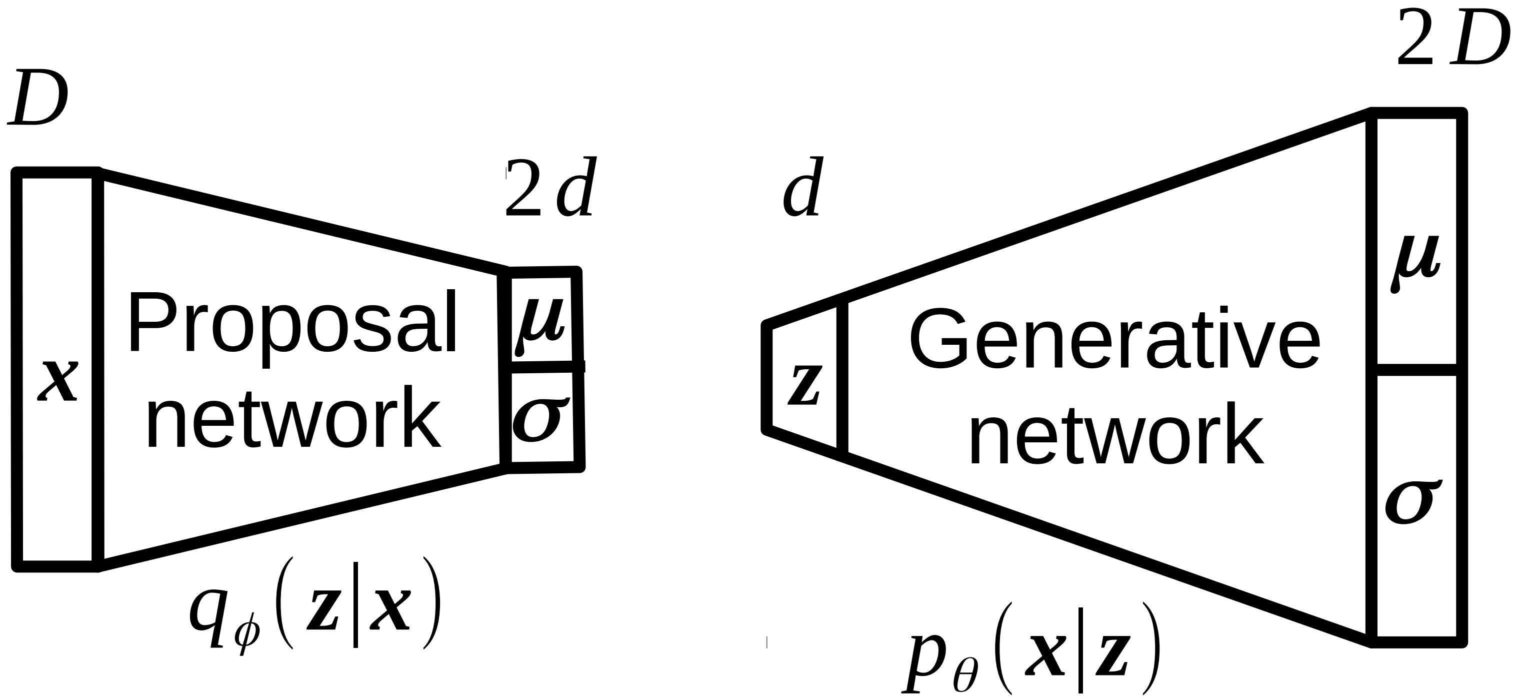
Variational Lower Bound

$$L(\theta) \geqslant L(\phi,\theta) =$$

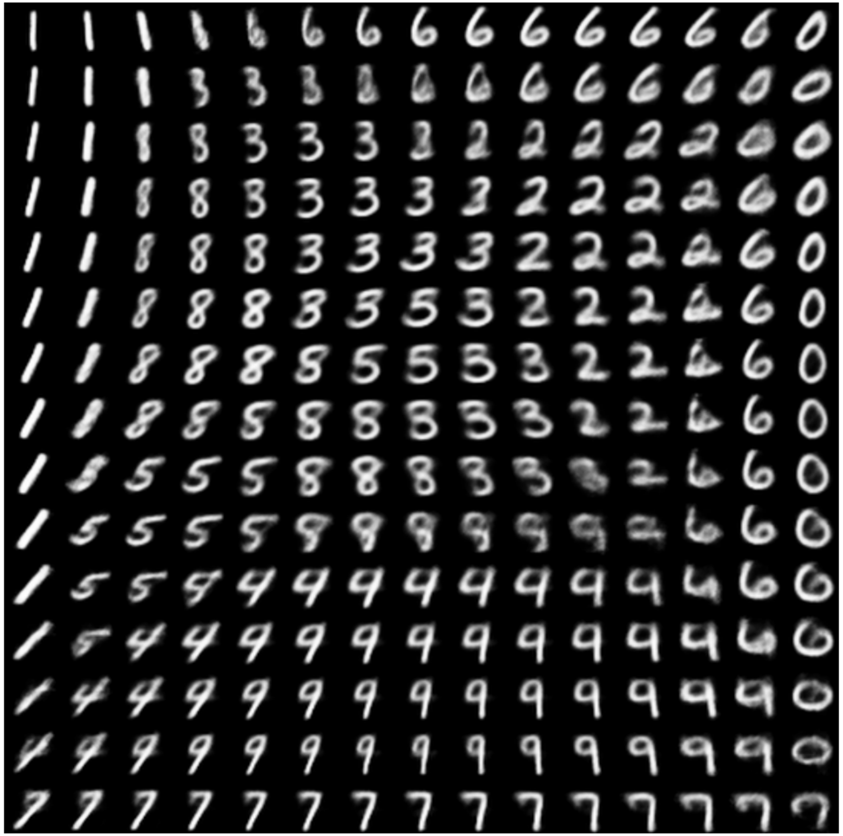
$$= \mathop{\mathbf{E}}_{\boldsymbol{x}\sim \boldsymbol{p}_{\boldsymbol{d}}(\boldsymbol{x})}\Bigg[\underbrace{\mathop{\mathbf{E}}_{\boldsymbol{z}\sim \boldsymbol{q}_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{x}|\boldsymbol{z})}_{\text{Reconstruction term}} - \underbrace{KL\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})\right)}_{\text{Regularization term}}\Bigg]$$

$$L(\phi,\theta) \rightarrow \max_{\phi,\theta}$$

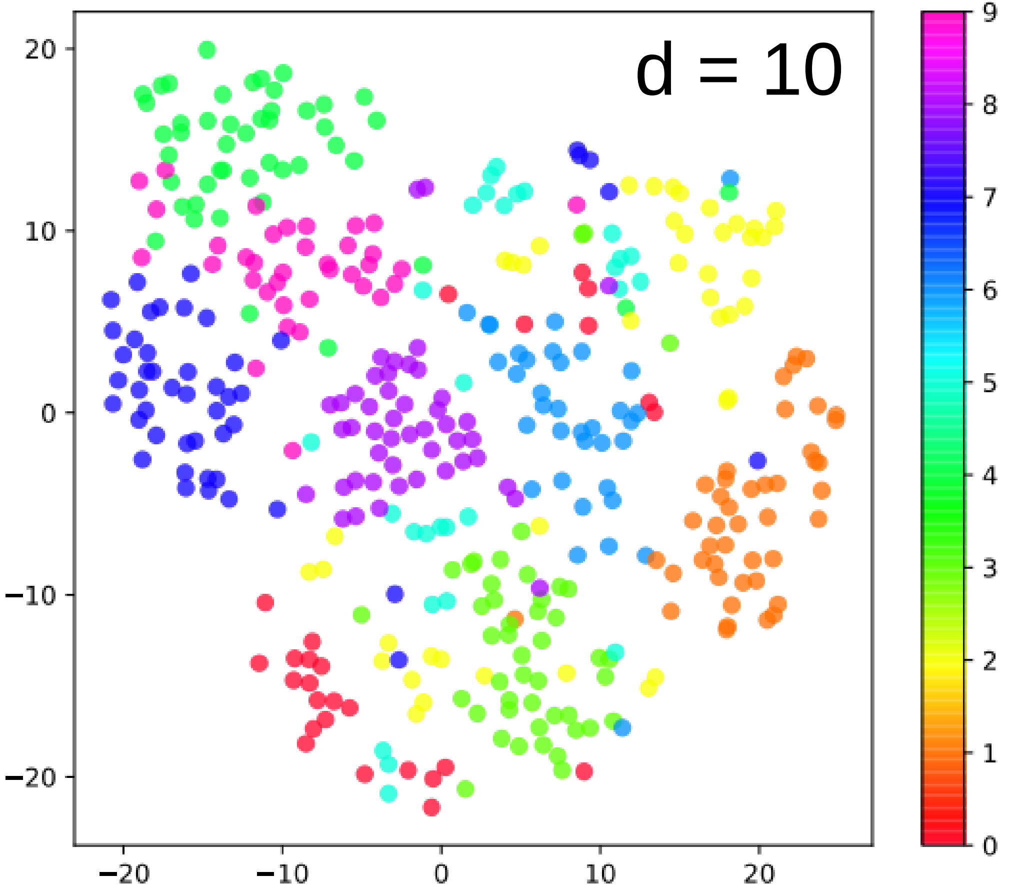
Variational Autoencoder



Latent space visualizations



d = 2



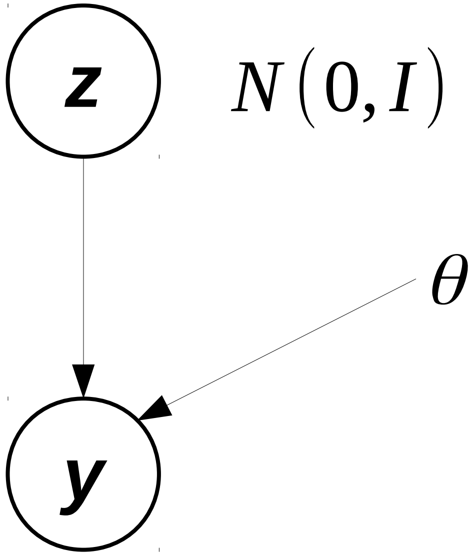
Variational Autoencoder

- Probabilistic generative model
  - Learns distribution $p_d(\boldsymbol{x})$
  - Works with both categorical and real features
  - No assumptions on the data distribution
- Semantic latent space
- Fast inference and generation

Conditional Variational Autoencoder

$$p_{\psi,\theta}(\boldsymbol{y}|\boldsymbol{x})$$

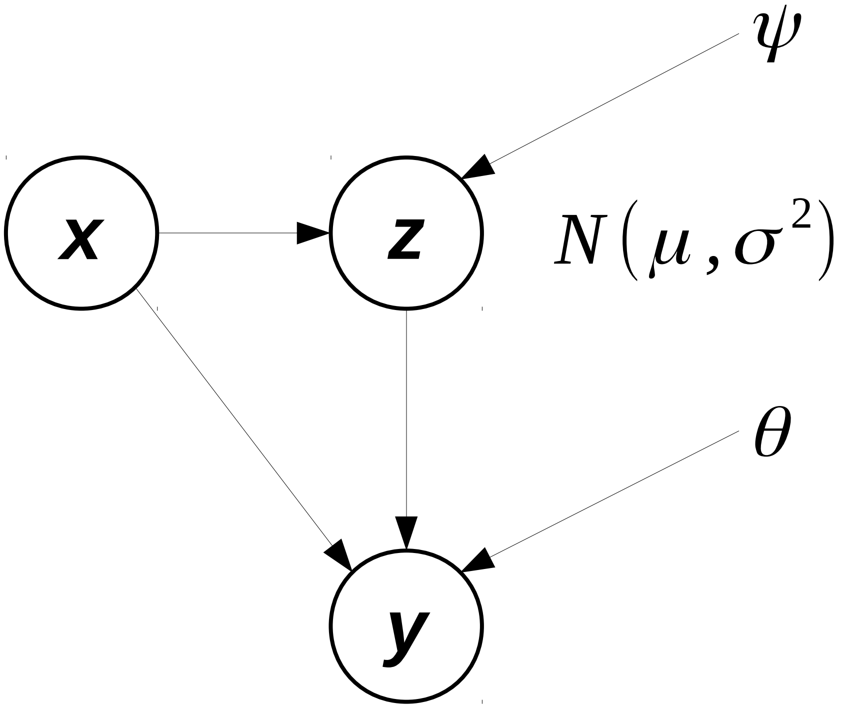
Generative process



$p(\boldsymbol{z})$

$p_\theta(\boldsymbol{y} | \boldsymbol{z})$

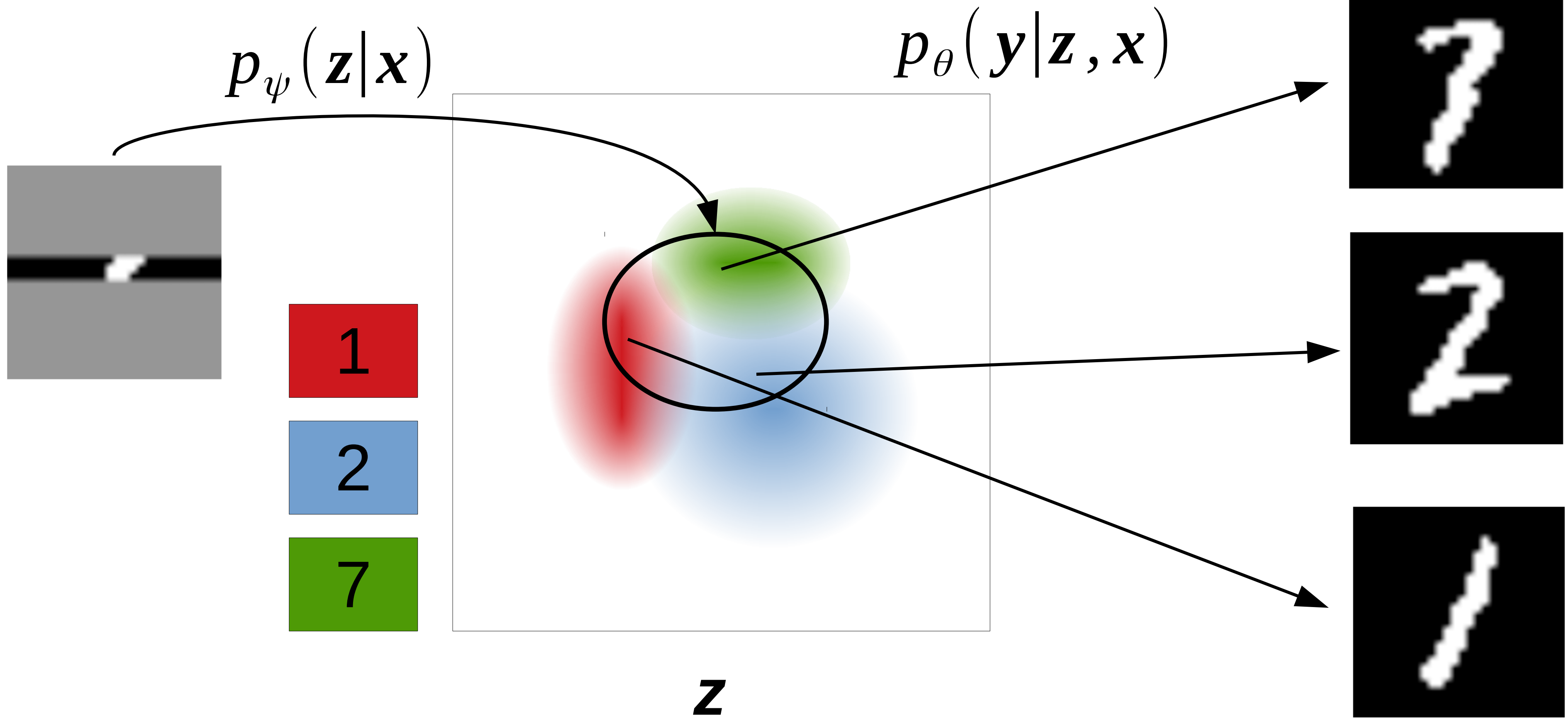
Generative process



$$p_{\psi}(\boldsymbol{z}|\boldsymbol{x})$$

$$p_{\theta}(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{x})$$

Generative process



Model Log-Likelihood

$$p_\theta(\boldsymbol{y}) = \mathop{\mathbf{E}}_{\boldsymbol{z}\sim \boldsymbol{p}(\boldsymbol{z})} p_\theta(\boldsymbol{x}|\boldsymbol{z})$$

$$L(\theta) = \mathop{\mathbf{E}}_{\boldsymbol{y}\sim \boldsymbol{p}_{\boldsymbol{d}}(\boldsymbol{y})} \log p_\theta(\boldsymbol{y})$$

Model Log-Likelihood

$$p_{\psi,\theta}(\mathbf{y}|\mathbf{x}) = \mathop{\mathbf{E}}_{\mathbf{z}\sim p_{\psi}(\mathbf{z}|\mathbf{x})} p_{\theta}(\mathbf{y}|\mathbf{z},\mathbf{x})$$

$$L(\psi,\theta) = \mathop{\mathbf{E}}_{\mathbf{x},\mathbf{y}\sim p_{d}(\mathbf{x},\mathbf{y})} \log p_{\theta}(\mathbf{y}|\mathbf{x})$$

Variational Lower Bound

Reconsruction loss

$$L(\theta) \geqslant L(\phi, \theta) =$$

$$= \mathop{\mathbf{E}}_{\boldsymbol{y} \sim \boldsymbol{p_d}(\boldsymbol{y})} \left[\mathop{\mathbf{E}}_{\boldsymbol{z} \sim \boldsymbol{q_\phi}(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{y}|\boldsymbol{z}) - \right.$$

$$\left. - KL(q_\phi(\boldsymbol{z}|\boldsymbol{y}) \| p(\boldsymbol{z})) \right]$$

Regularization term

$$L(\phi, \theta) \rightarrow \max_{\phi, \theta}$$

Variational Lower Bound

Reconsruction loss

$$L(\psi,\theta) \geqslant L(\phi,\psi,\theta) =$$

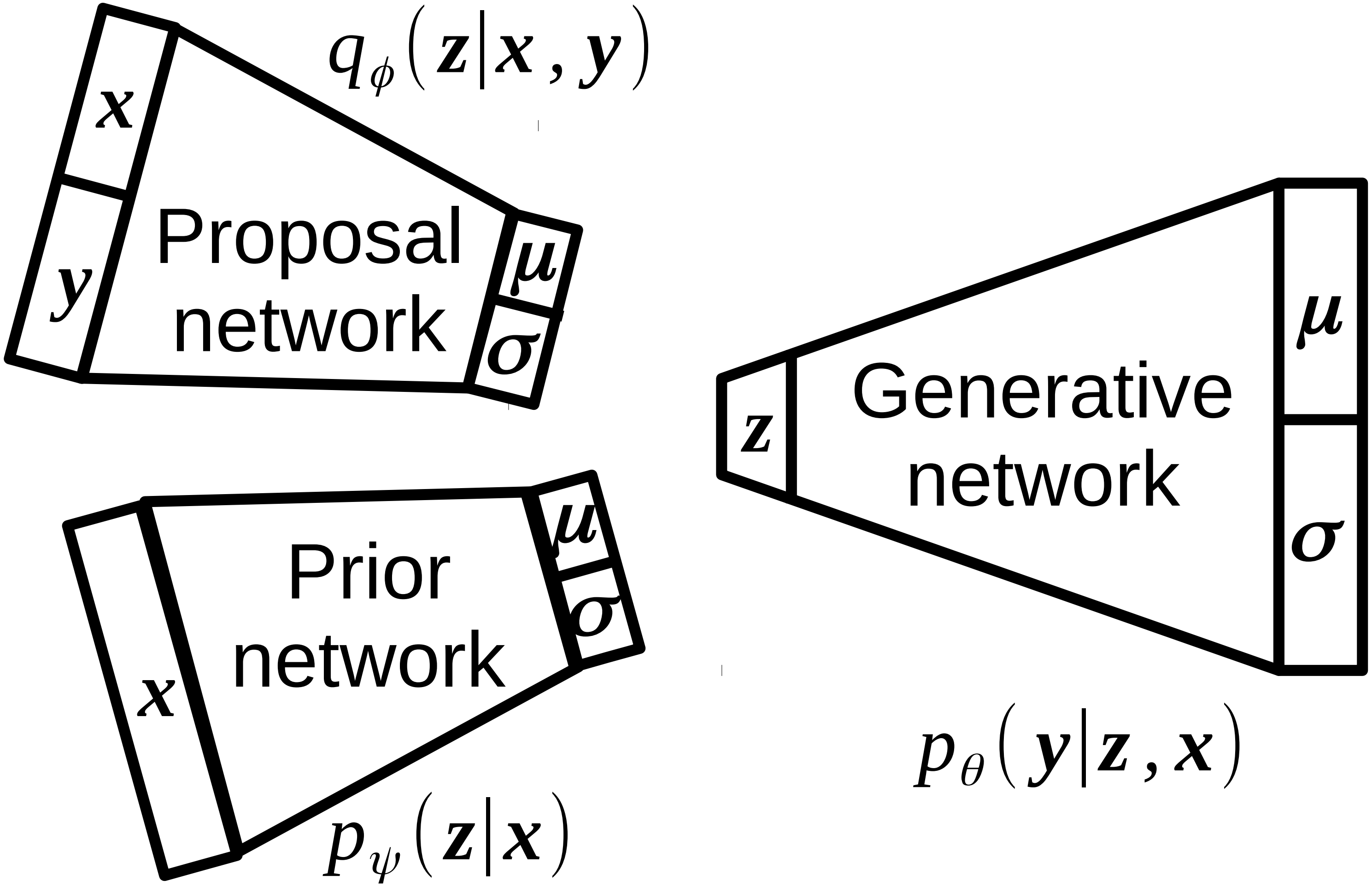
$$= \mathop{\mathbf{E}}_{\boldsymbol{x},\boldsymbol{y}\sim \boldsymbol{p}_d(\boldsymbol{x},\boldsymbol{y})} \left[\mathop{\mathbf{E}}_{\boldsymbol{z}\sim \boldsymbol{q}_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})} \log p_\theta(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{x}) - \right.$$

$$\left. - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) \| p_\psi(\boldsymbol{z}|\boldsymbol{x})) \right]$$

$$L(\phi,\psi,\theta) \rightarrow \max_{\phi,\psi,\theta}$$

Regularization term

Conditional Variational Autoencoder



Gaussian Stochastic Neural Network

Motivation:

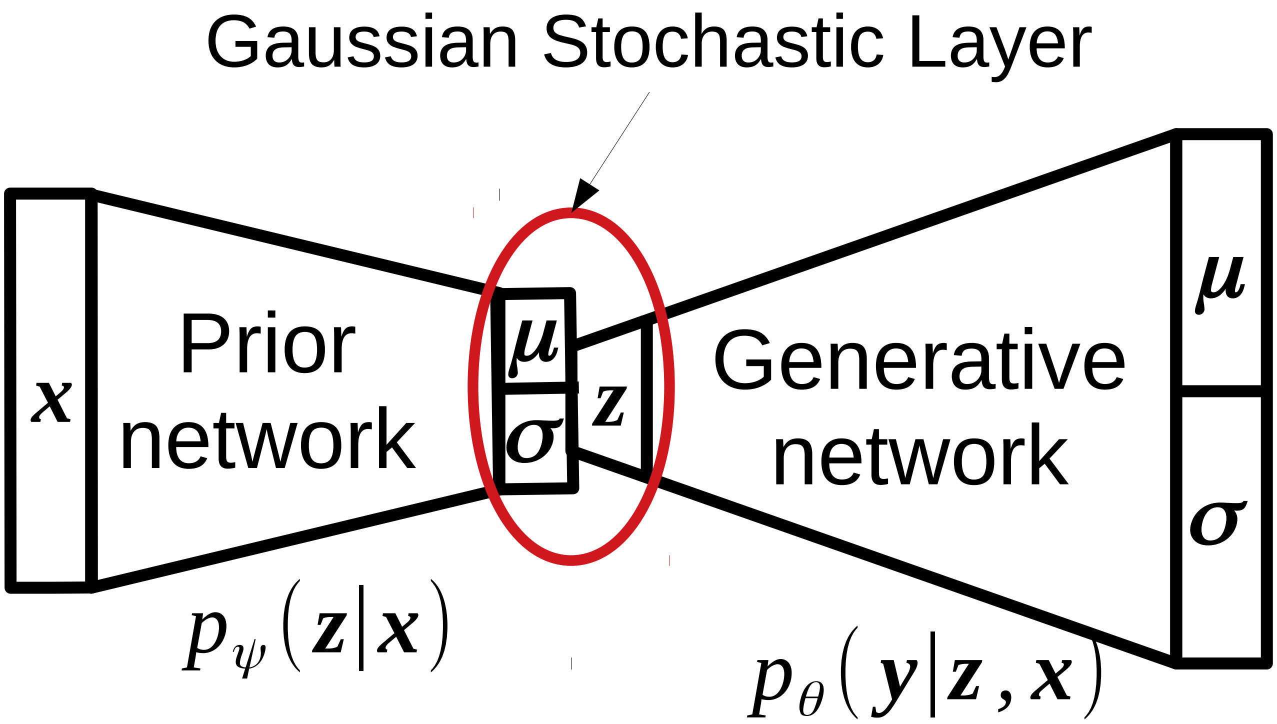
1. Train/test procedure inconsistency
2. Gaps in latent space
3. Better Monte-Carlo log-likelihood estimations

$$q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) = p_\psi(\boldsymbol{z}|\boldsymbol{x}_{\boldsymbol{1-b}},\boldsymbol{b})$$

$$L_{GSNN}(\phi,\psi,\theta) = \mathop{\mathbf{E}}_{\boldsymbol{x},\boldsymbol{y}\sim \boldsymbol{p_d}(\boldsymbol{x},\boldsymbol{y})} \mathop{\mathbf{E}}_{\boldsymbol{z}\sim \boldsymbol{p_\phi}(\boldsymbol{z}|\boldsymbol{x})} \log p_\theta(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{x})$$

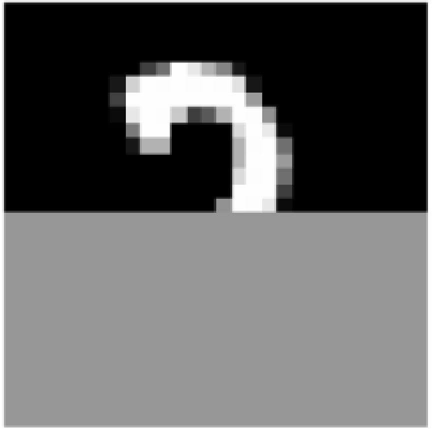
$$L_{hybrid} = \alpha L_{CVAE} + (1-\alpha) L_{GSNN}$$

Gaussian Stochastic Neural Network



Some motivation pictures

Input



Samples

Conditional Variational Autoencoder

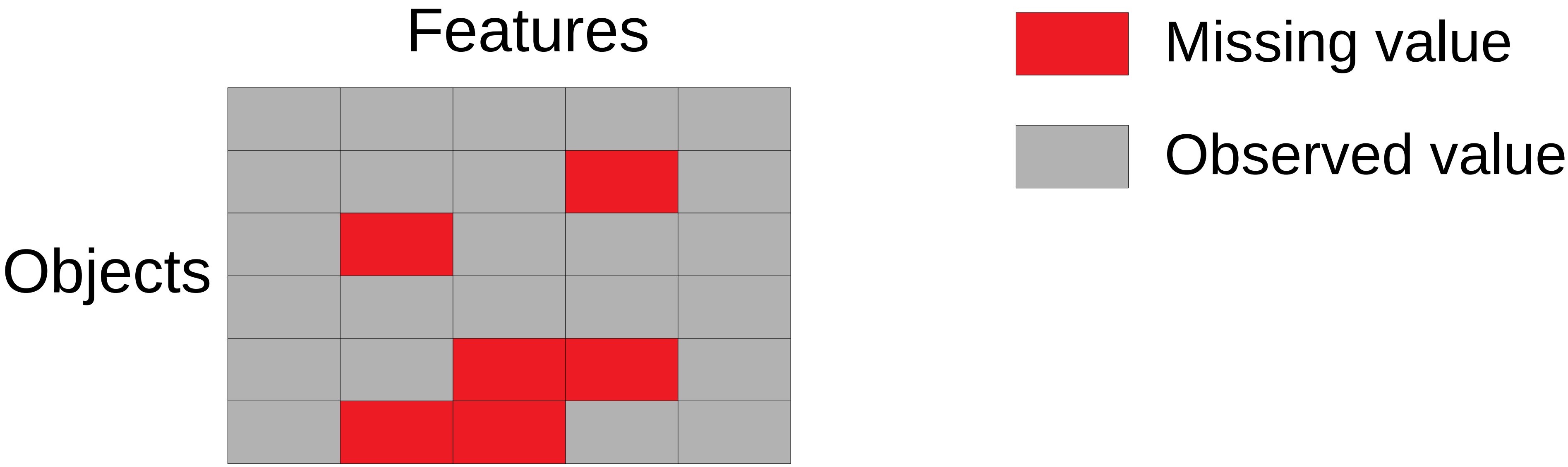
- Learns conditional distribution $p_d(\boldsymbol{y}|\boldsymbol{x})$
  - Essential when $p_d(\boldsymbol{y}|\boldsymbol{x})$ has several local maximums
- Obtains by conditioning Variational Autoencoder
  - Inherits the majority of its properties
- Has prior network to model prior latent distribution
- Modifications: GSNN, hybrid model

Universal Conditional Machine

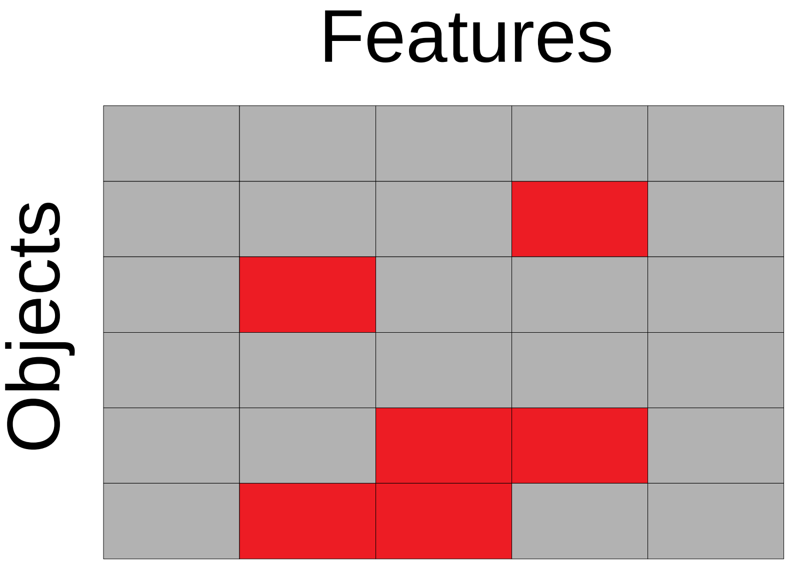
$$p_{\psi,\theta}\left(\boldsymbol{x}_I \middle| \boldsymbol{x}_{U\setminus I}\right)$$

Problem statement

Test set



Missing features mask



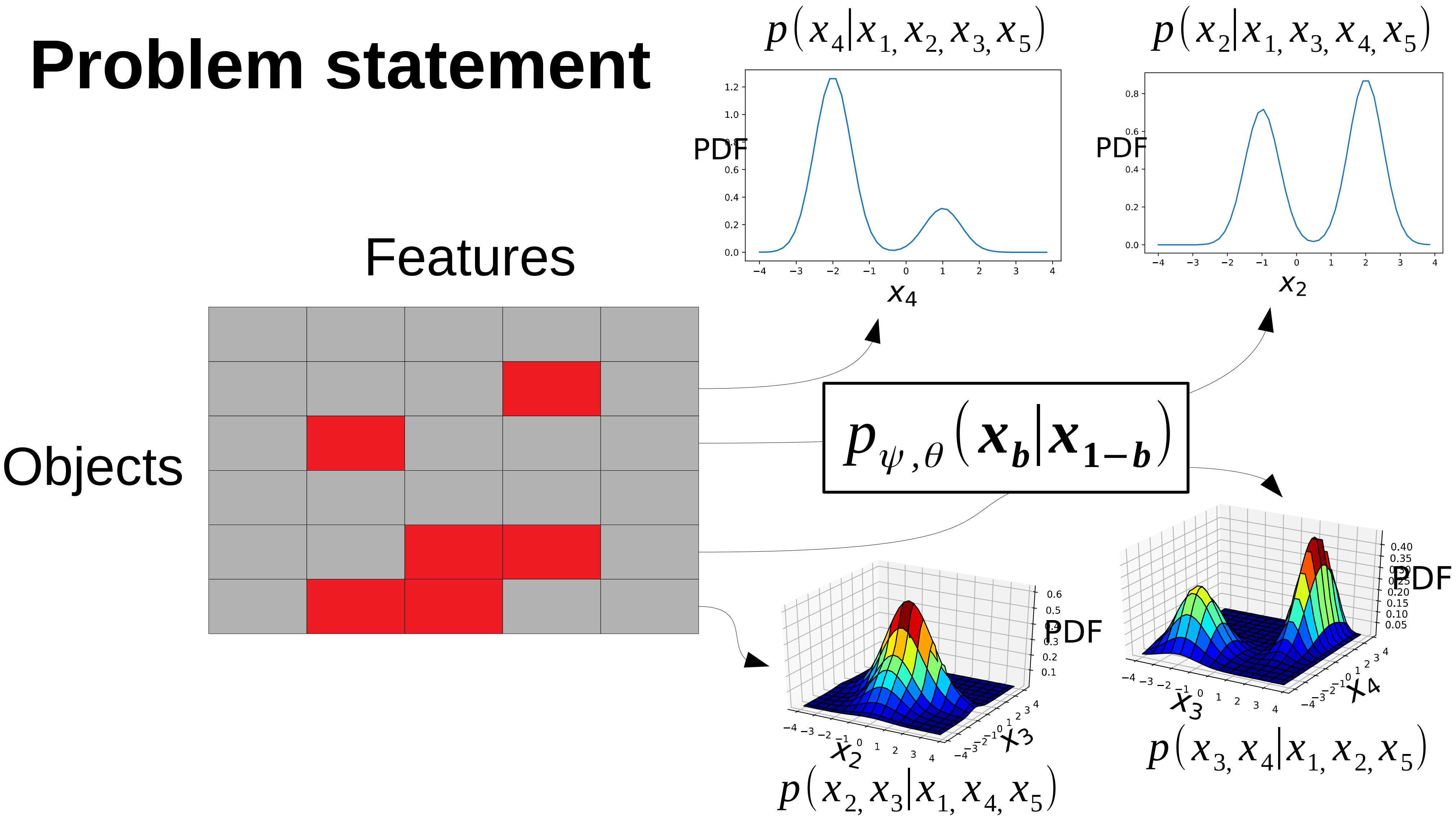
$\boldsymbol{b} \in \{0, 1\}^D$

(0, 0, 0, 0, 0)
(0, 0, 0, **1**, 0)
(0, **1**, 0, 0, 0)
(0, 0, 0, 0, 0)
(0, 0, **1**, **1**, 0)
(0, **1**, **1**, 0, 0)

Indexation example

$$\boldsymbol{b} = (0, 0, 1, 1, 0)$$
$$\boldsymbol{x}_{\boldsymbol{b}} = (x_{3,}\, x_4)$$
$$\boldsymbol{x}_{\boldsymbol{1}-\boldsymbol{b}} = (x_{1,}\, x_{2,}\, x_5)$$

Problem statement



Why distributions?

Input

Average

Most probable

Single imputation causes information loss

Why distributions?

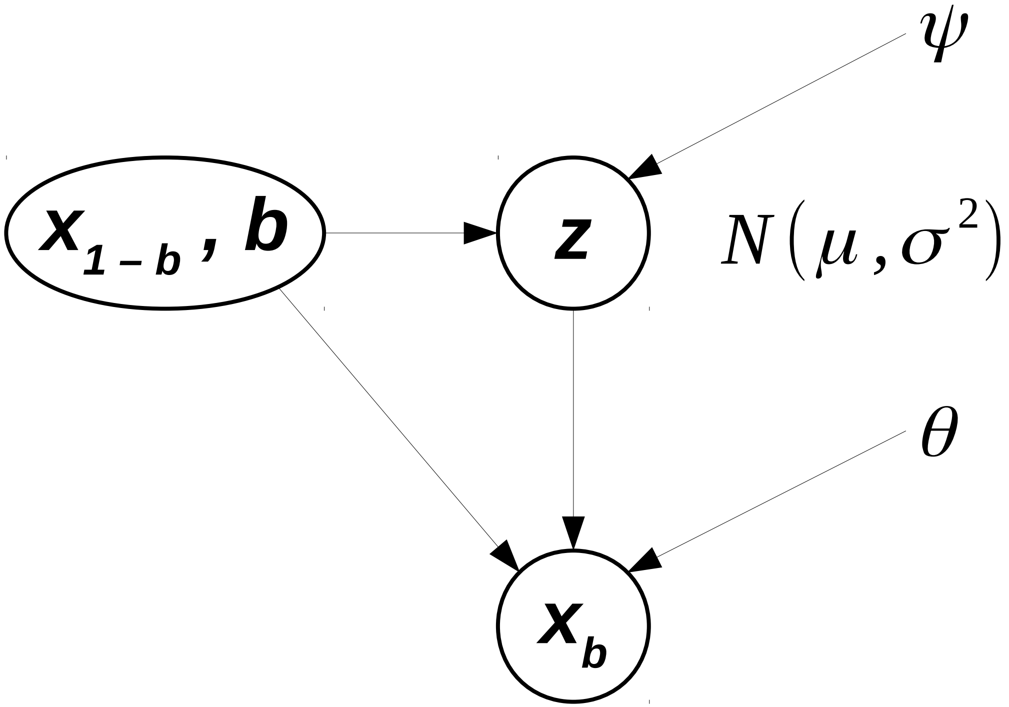
Input

Average

Most probable

Distribution samples

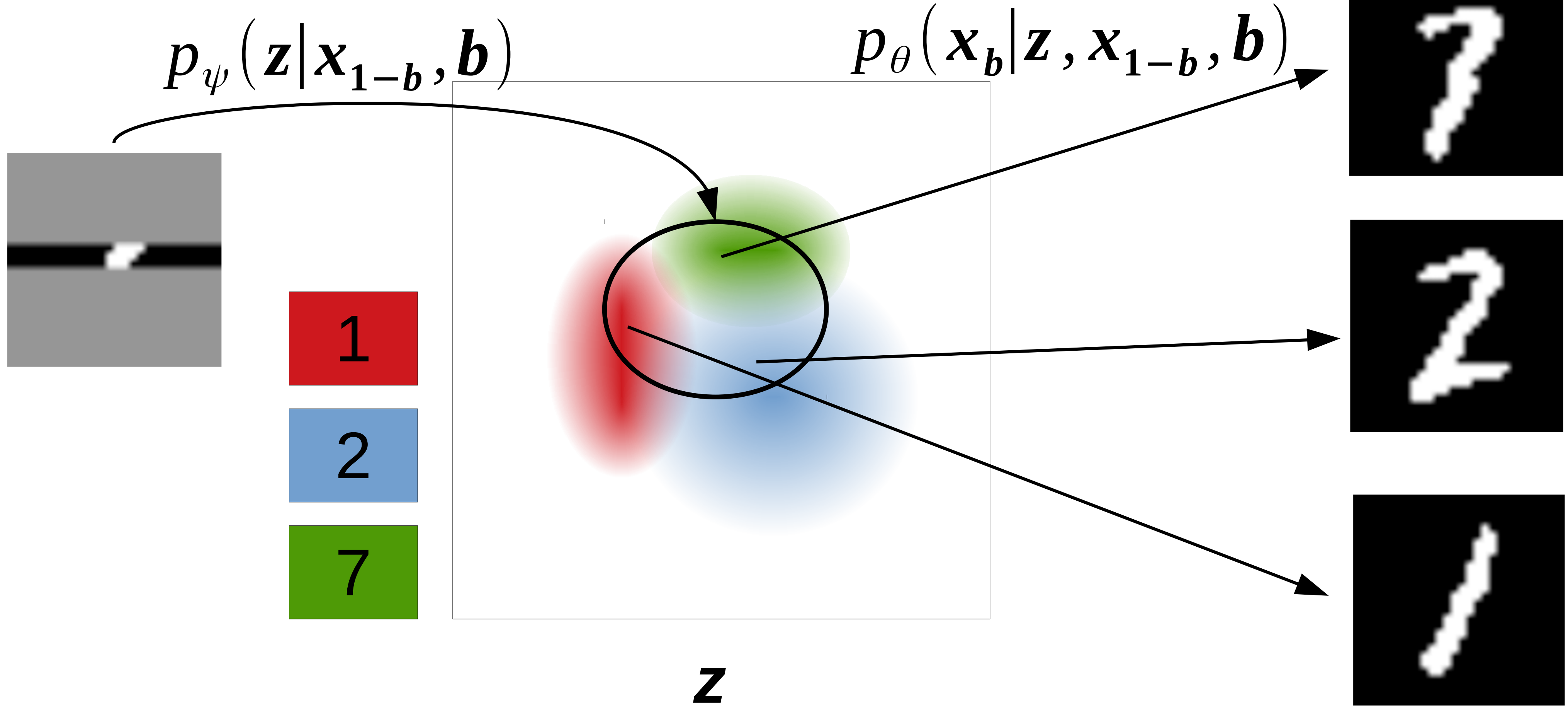
Generative process



$$p_{\psi}(\boldsymbol{z} | \boldsymbol{x}_{1-b}, \boldsymbol{b})$$

$$p_{\theta}(\boldsymbol{x}_{b} | \boldsymbol{z}, \boldsymbol{x}_{1-b}, \boldsymbol{b})$$

Generative process



Mask distribution and model likelihood

- $p_{\psi,\theta}(\boldsymbol{x_b}|\boldsymbol{x_{1-b}},\boldsymbol{b}) = \underset{\boldsymbol{z}\sim\boldsymbol{p_\psi}(\boldsymbol{z}|\boldsymbol{x_{1-b}},\boldsymbol{b})}{\mathrm{E}} p_\theta(\boldsymbol{x_b}|\boldsymbol{z},\boldsymbol{x_{1-b}},\boldsymbol{b})$
- User-defined mask distribution $p_b(\boldsymbol{b})$
- “Train set”:

$$(\boldsymbol{x}_i,\boldsymbol{b}_i)_{i=1}^N : \boldsymbol{x}\sim p_d(\boldsymbol{x}), \boldsymbol{b}\sim p_b(\boldsymbol{b})$$

- Model log-likelihood

$$L(\psi,\theta) = \underset{\substack{\boldsymbol{x}\sim\boldsymbol{p_d}(\boldsymbol{x}) \\ \boldsymbol{b}\sim\boldsymbol{p_b}(\boldsymbol{b})}}{\mathrm{E}} \log p_{\psi,\theta}(\boldsymbol{x_b}|\boldsymbol{x_{1-b}},\boldsymbol{b})$$

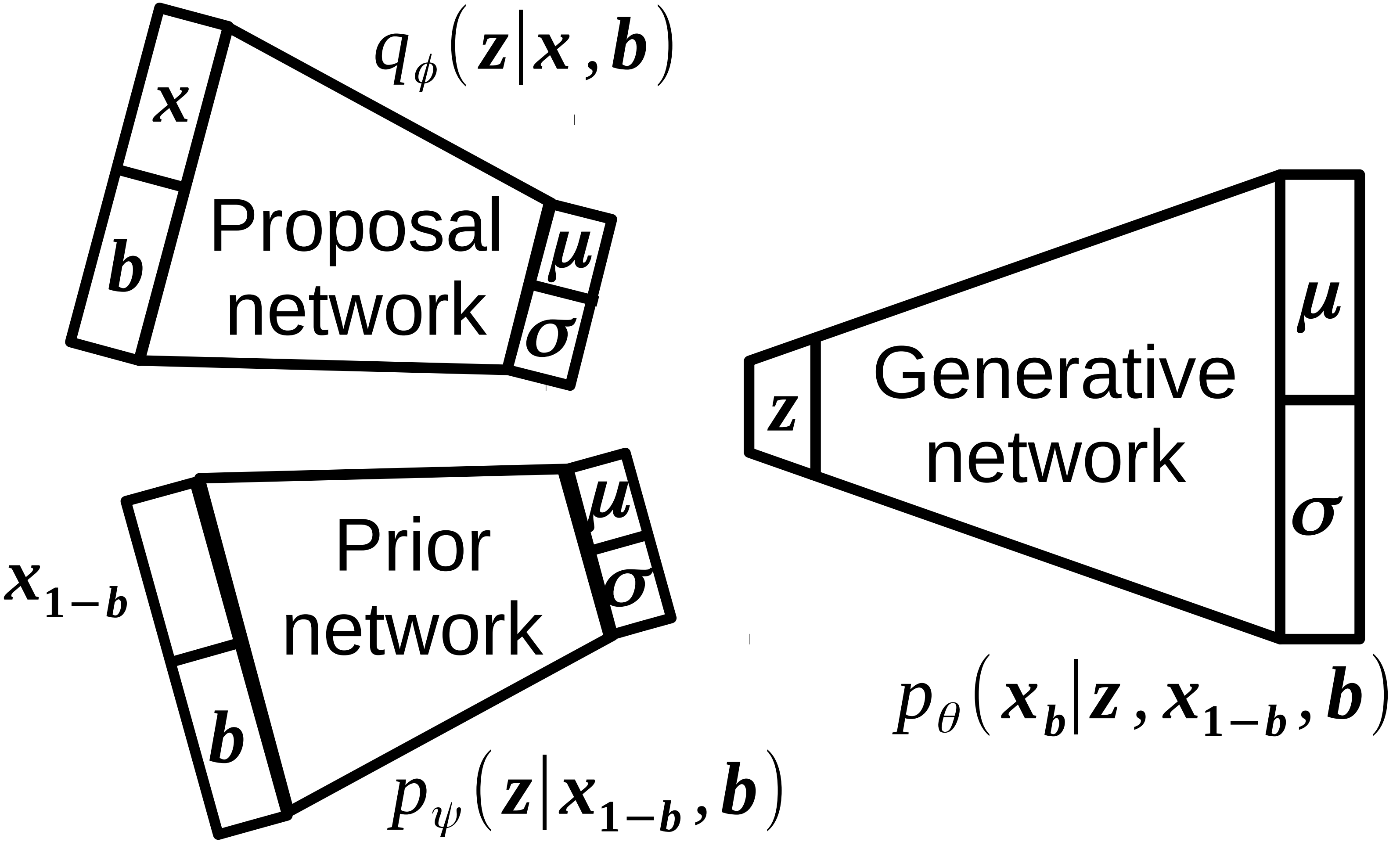
Variational Lower Bound

$$L(\psi,\theta) \geqslant L(\phi,\psi,\theta) =$$

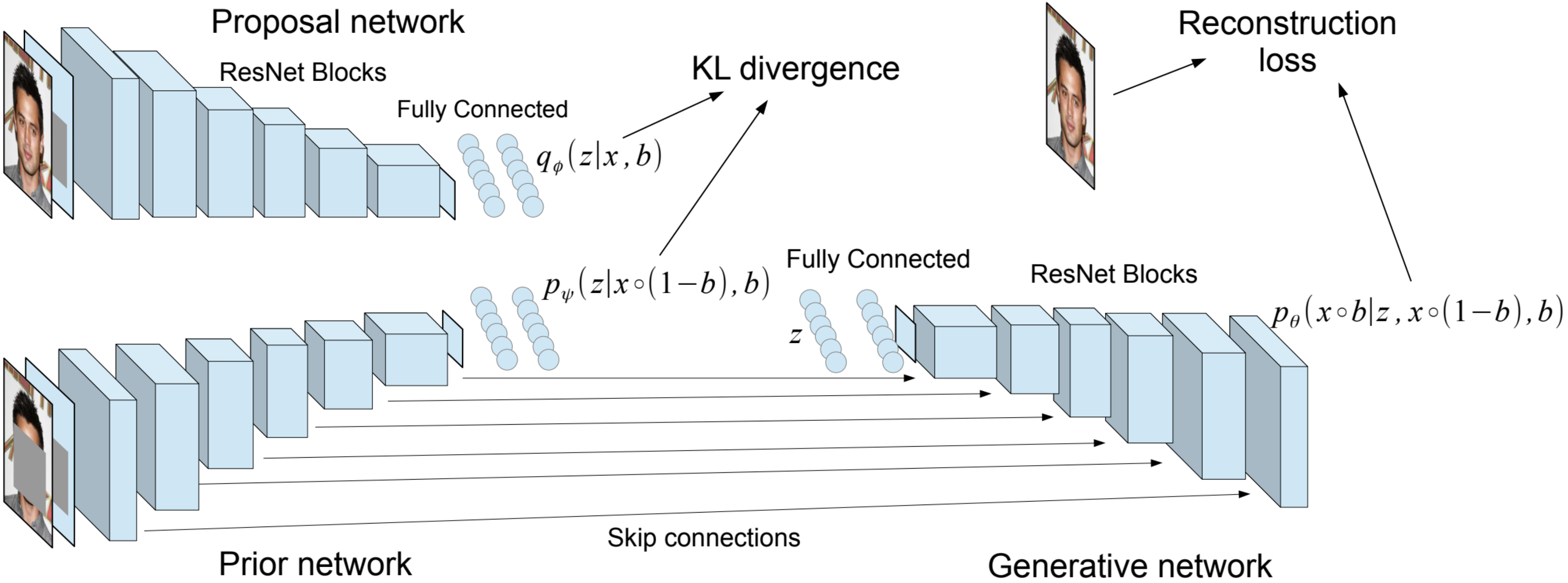
$$= \mathop{\mathbf{E}}_{\substack{\boldsymbol{x} \sim p_d(\boldsymbol{x}) \\ \boldsymbol{b} \sim p_b(\boldsymbol{b})}} \left[\mathop{\mathbf{E}}_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{b})} \log p_\theta(\boldsymbol{x_b}|\boldsymbol{z},\boldsymbol{x_{1-b}},\boldsymbol{b}) - \right.$$

$$\left. - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{b}) \| p_\psi(\boldsymbol{z}|\boldsymbol{x_{1-b}},\boldsymbol{b})) \right]$$

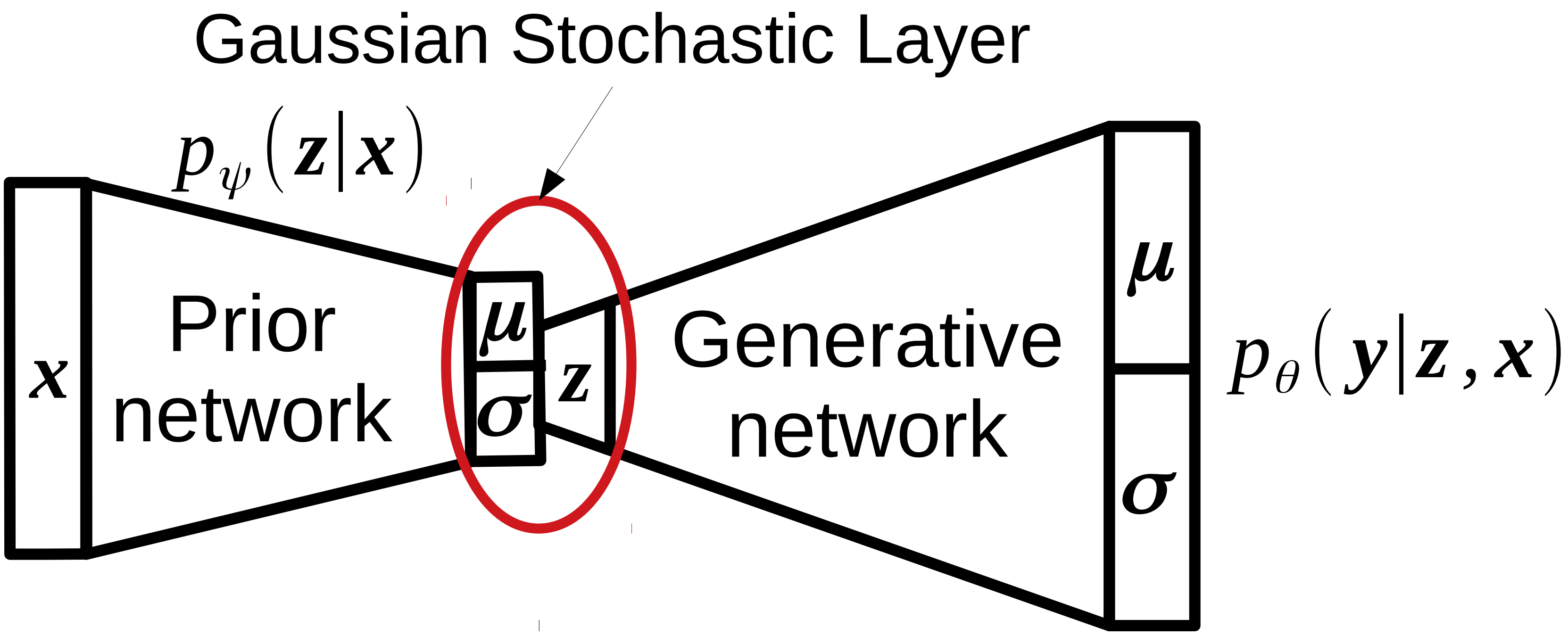
Universal Conditional Machine



Universal Conditional Machine



Gaussian Stochastic Neural Network



$$L_{hybrid} = \alpha L_{UCM} + (1-\alpha) L_{GSNN}$$

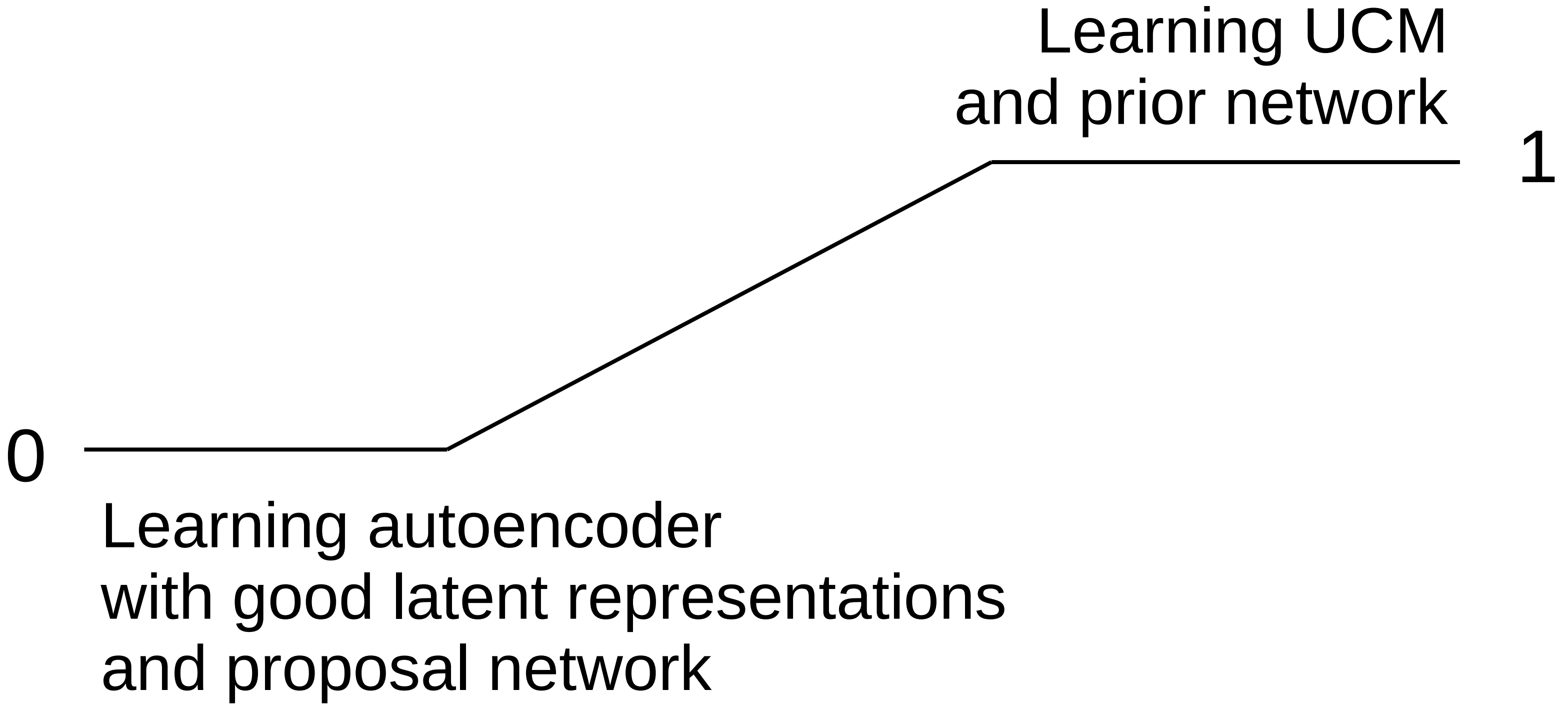
Missing features in train set

- Missing feature $x_i = \omega$
- Conditioned mask distribution $p_b(\boldsymbol{b}|\boldsymbol{x})$
- $x_i = \omega \;\Rightarrow\; p_b(b_i|\boldsymbol{x}) = 1$
- Missing features marginalization: $x_i = \omega \;\Rightarrow$

$$\log p_\theta(x_i|\boldsymbol{z}, \boldsymbol{x}_{\boldsymbol{1}-\boldsymbol{b}}, \boldsymbol{b}) \;=\; \log \int p_\theta(\hat{x}_i|\boldsymbol{z}, \boldsymbol{x}_{\boldsymbol{1}-\boldsymbol{b}}, \boldsymbol{b})\, d\,\hat{x}_i \;=\; \log 1 \;=\; 0$$

KL coefficient

Found necessary only for syntetic data



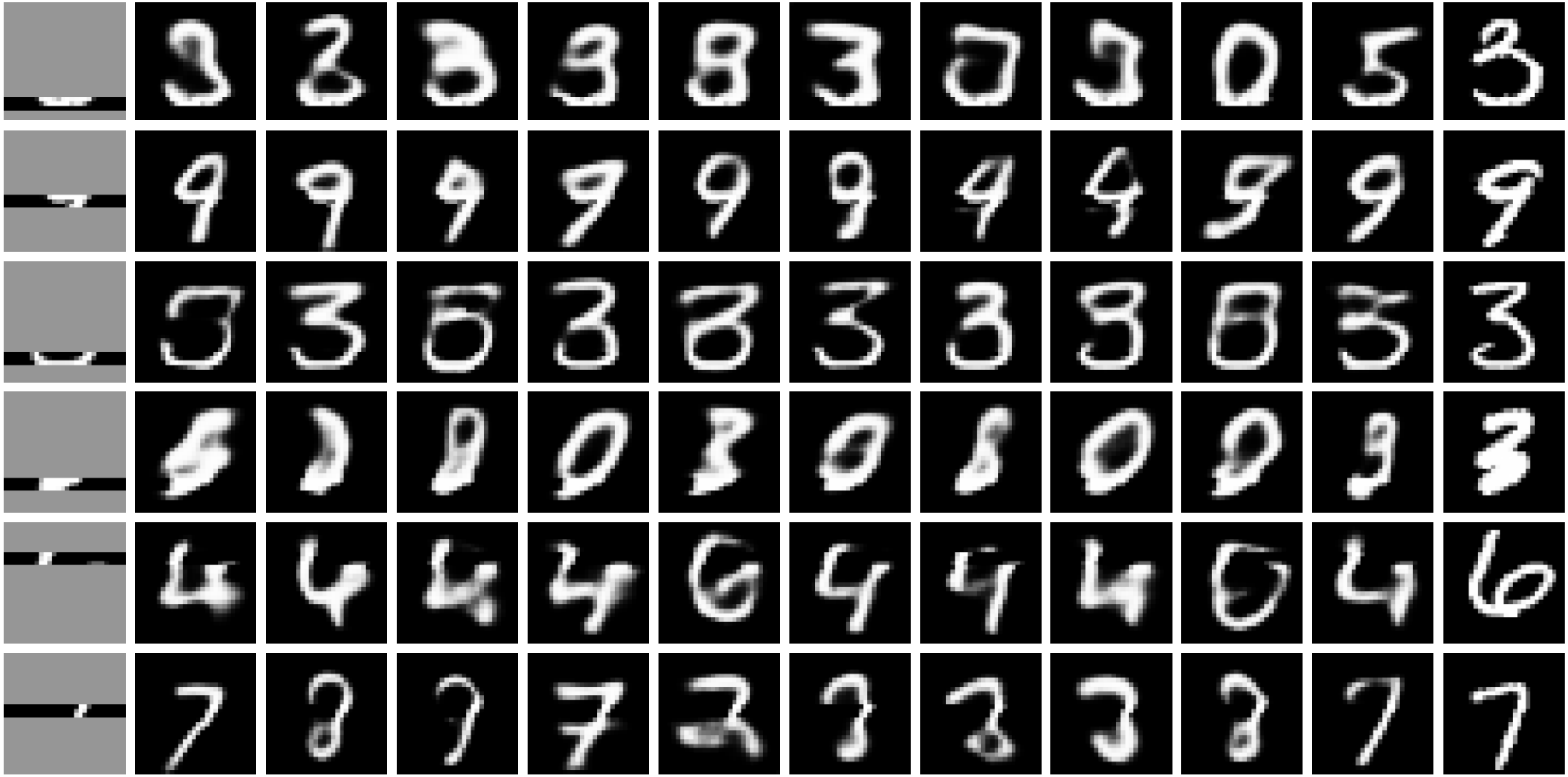
Experiment 1

Missing feature multiple imputation for supervised learning.
50% features of dataset are missed.
10 imputations for UCM, GSNN, MG.

Dataset	Average	XGBoost	UMC	GSNN	MG
Boston	0.505 ± 0.061	0.502 ± 0.056	$\mathbf{0.577} \pm 0.069$	0.563 ± 0.069	0.564 ± 0.055
Concrete	0.452 ± 0.042	0.458 ± 0.040	$\mathbf{0.494} \pm 0.032$	0.453 ± 0.030	0.484 ± 0.030
CASP	0.840 ± 0.002	0.842 ± 0.002	$\mathbf{0.856} \pm 0.002$	$\mathbf{0.856} \pm 0.002$	0.850 ± 0.003
Wine	0.230 ± 0.012	0.236 ± 0.008	0.232 ± 0.016	$\mathbf{0.243} \pm 0.014$	0.238 ± 0.011
Yeast	0.423 ± 0.025	0.426 ± 0.026	$\mathbf{0.436} \pm 0.019$	0.419 ± 0.025	0.430 ± 0.019

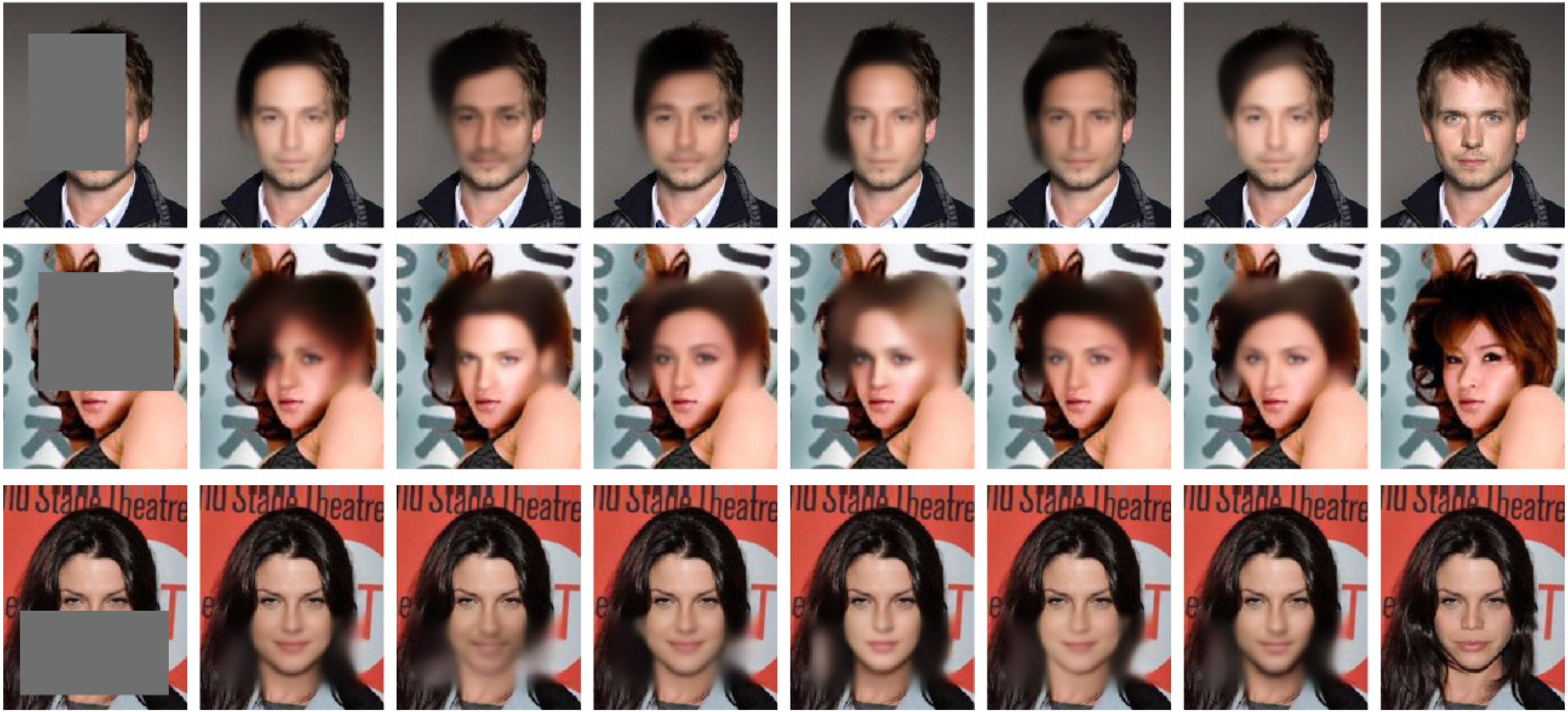
MG – Multivariate Gaussian

Experiment 2: image inpainting, MNIST

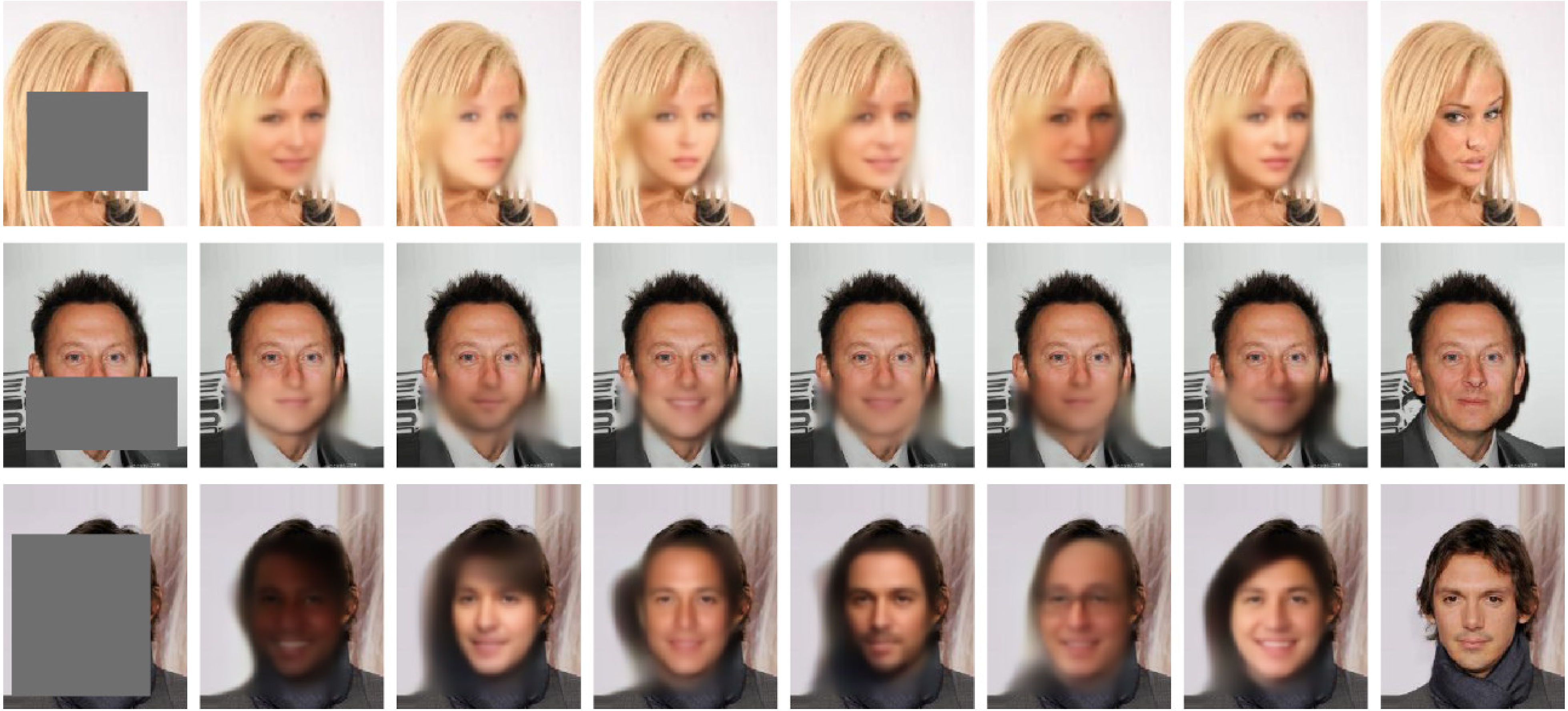


Experiment 2: image inpainting, Omniglot

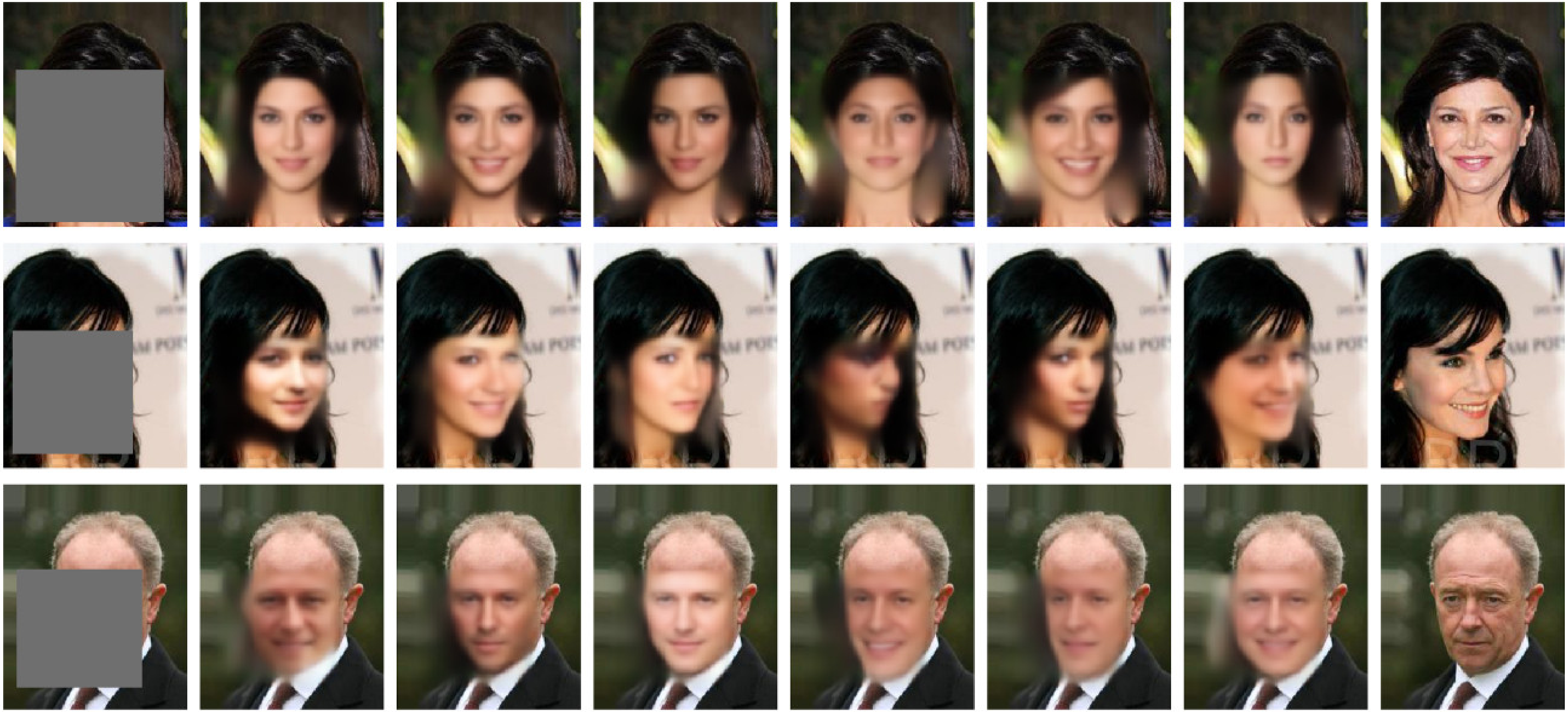
Experiment 2: image inpainting, CelebA



Experiment 2: image inpainting, CelebA



Experiment 2: image inpainting, CelebA



Universal Conditional Machine

- Learns all conditional distributions $p_d\left(\boldsymbol{x_b} \middle| \boldsymbol{x_{1-b}}\right)$
  - The importance of the conditioning is given by $p_b\left(\boldsymbol{b}\right)$
- Further extension of VAE and CVAE
  - Same conditioning technique as in CVAE
  - Lots of slight modifications
- It works!
  - As preprocessing (multiple imputation) for datasets with missing data
  - For image inpainting
  - TBD: image colourization

Saga of hybrid model

$$\alpha L_{CVAE} + (1 - \alpha) L_{GSNN}$$

GSNN – good or evil?

$$L_{hybrid} = \alpha L_{UCM} + (1 - \alpha) L_{GSNN}$$

Motivation:

1. Train/test procedure inconsistency
2. Gaps in latent space
3. Better Monte-Carlo log-likelihood estimations

Log-likelihood estimations

$$\log p_{\psi,\theta}(\boldsymbol{x_b}|\boldsymbol{x_{1-b}},\boldsymbol{b}) = \log \underset{\boldsymbol{z}\sim \boldsymbol{p_\psi}(\boldsymbol{z}|\boldsymbol{x_{1-b}},\boldsymbol{b})}{\mathbf{E}} p_\theta(\boldsymbol{x_b}|\boldsymbol{z},\boldsymbol{x_{1-b}},\boldsymbol{b})$$

Monte-Carlo

$$\approx \log \frac{1}{S}\sum_{i=1}^{S} p_\theta(\boldsymbol{x_b}|\boldsymbol{z_i},\boldsymbol{x_{1-b}},\boldsymbol{b})$$

$$\boldsymbol{z_i} \sim p_\psi(\boldsymbol{z}|\boldsymbol{x_{1-b}},\boldsymbol{b})$$

Importance Sampling

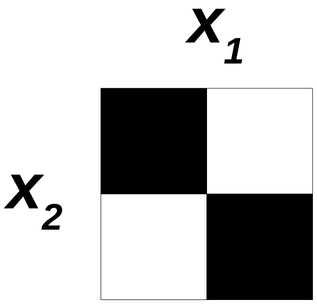
$$\approx \log \frac{1}{S}\sum_{i=1}^{S} \frac{p_\theta(\boldsymbol{x_b}|\boldsymbol{z_i},\boldsymbol{x_{1-b}},\boldsymbol{b})\, p_\psi(\boldsymbol{z_i}|\boldsymbol{x_{1-b}},\boldsymbol{b})}{q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{b})}$$

$$\boldsymbol{z_i} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{b})$$

Log-likelihood estimations

Method	MNIST	Omniglot	CelebA
UMC IS-10^2	$\mathbf{83} \pm 2$	$\mathbf{275} \pm 17$	$\mathbf{34035} \pm 1609$
UMC MC-10^4	98 ± 4	1452 ± 109	41513 ± 2163
UMC MC-10^2	135 ± 6	2203 ± 150	53904 ± 3121
GSNN MC-10^4	139 ± 3	1199 ± 62	53427 ± 2208
GSNN MC-10^2	139 ± 3	1200 ± 62	53486 ± 2210
Naive Bayes	205	2490	269480

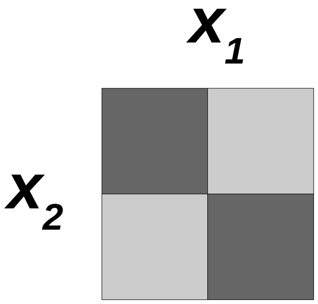
Gaps & overlapping in latent space: XOR



True distribution

$$x_1, x_2 \in \{0, 1\}$$

$$p_d(x_1, x_2) = 0.5(x_1 \oplus x_2)$$

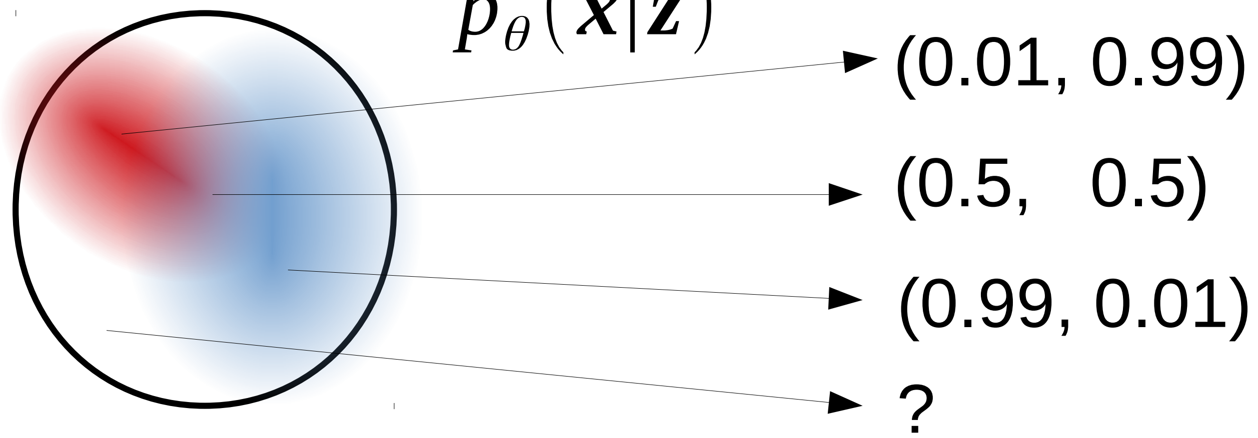


$p_\theta(x_1, x_2)$

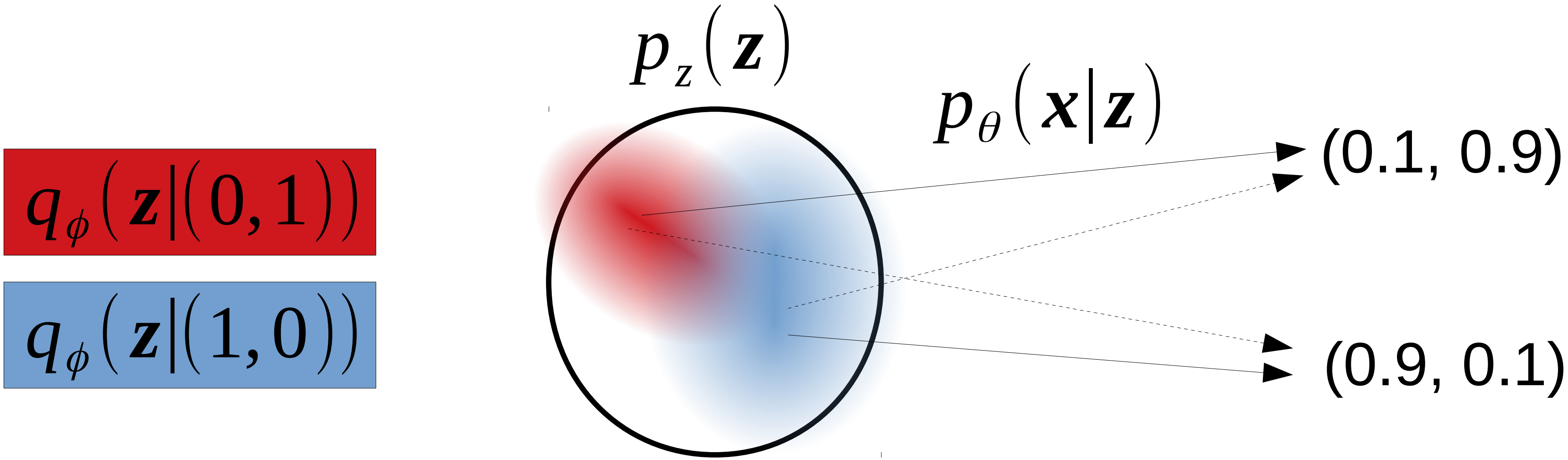
$p_z(\mathbf{z})$

$q_\phi(\mathbf{z}|(0,1))$

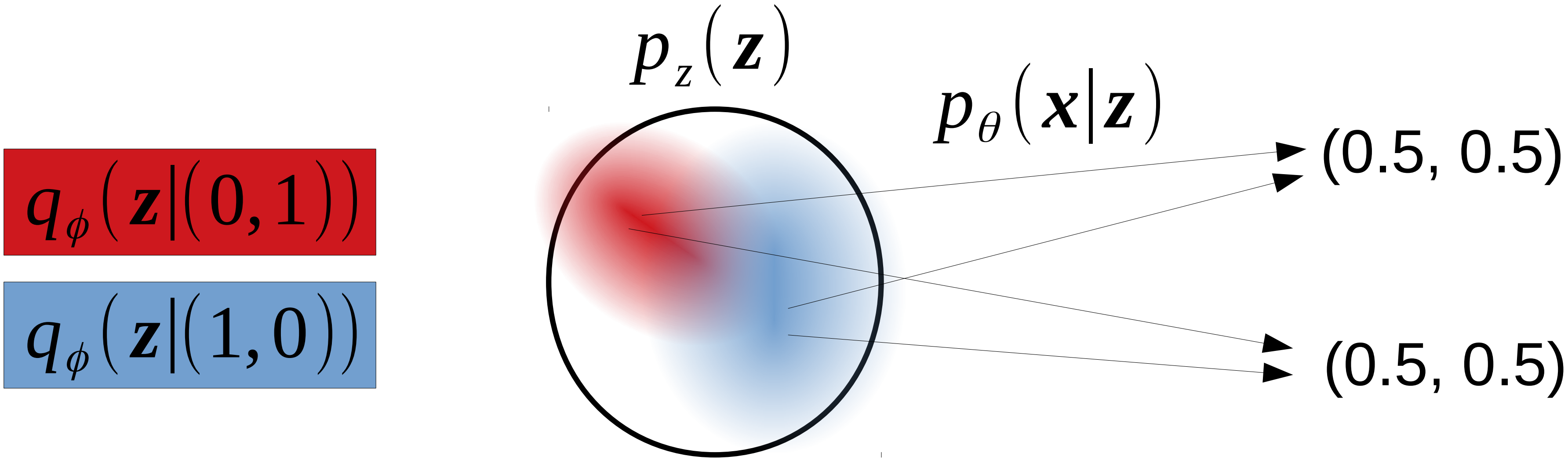
$q_\phi(\mathbf{z}|(1,0))$



Why don't use Monte-Carlo?

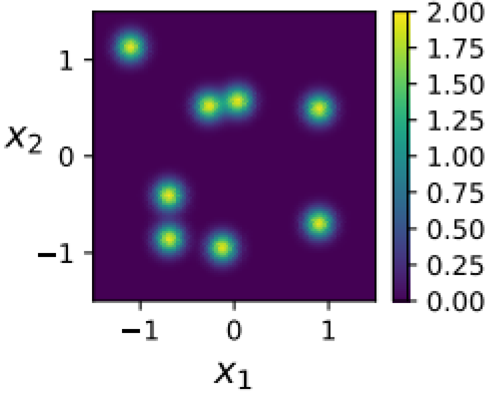


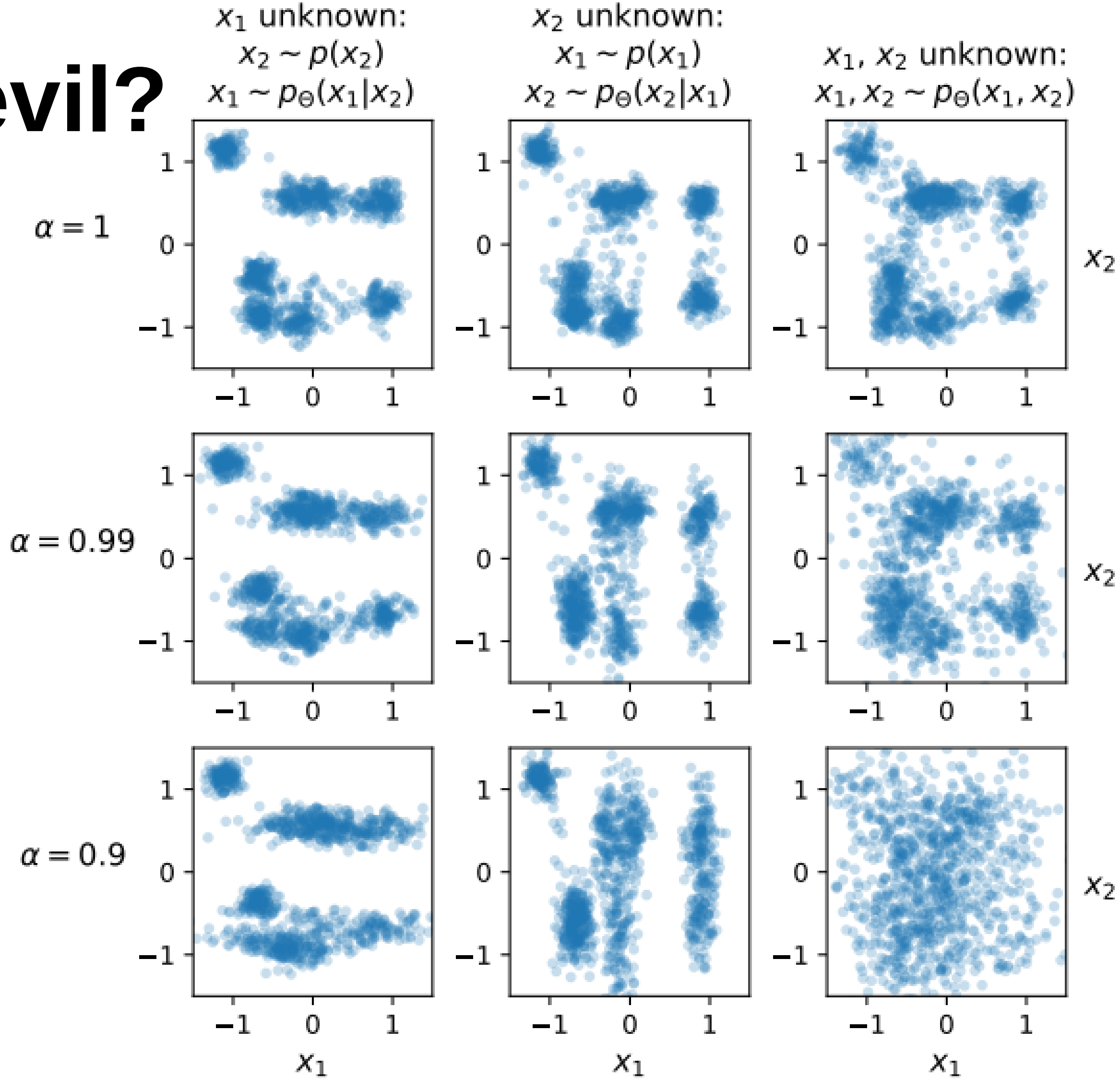
Why don't use Monte-Carlo?



GSNN – good or evil?

True distribution





Saga of hybrid model

- Gaps and overlapping are problems for Gaussian latent space
  - Might be with normalizing flows, etc
- GSNN Monte-Carlo estimations with a few samples are better
  - Monte-Carlo estimator is not precise
    - Needs too many samples to find the region in latent space suitable for the given object
- GSNN can’t learn multimodal distribution
  - GSNN might work better with big S at the training stage
- True log-likelihood is better for UCM without any GSNN

Universal Marginalizer

Improved version

$$p_\theta\left(\boldsymbol{x}_{\boldsymbol{i}} \middle| \boldsymbol{x}_{U \setminus \boldsymbol{I}}\right)$$

Mask distribution and model likelihood

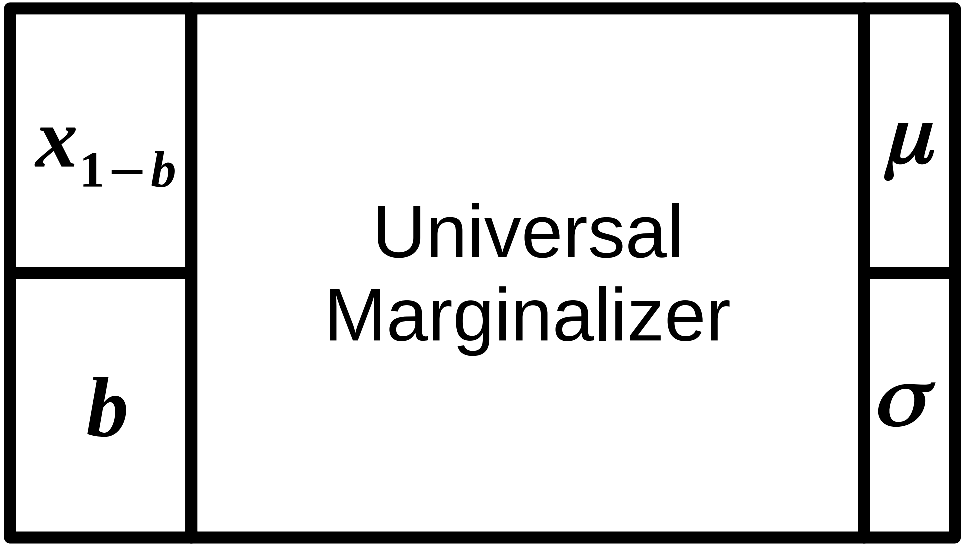
- User-defined mask distribution $p_b(\boldsymbol{b})$
- "Train set":

$$(\boldsymbol{x}_i, \boldsymbol{b}_i)_{i=1}^{N} : \boldsymbol{x} \sim p_d(\boldsymbol{x}), \boldsymbol{b} \sim p_b(\boldsymbol{b})$$

- Model log-likelihood

$$L(\theta) = \mathop{\mathbf{E}}_{\substack{\boldsymbol{x} \sim \boldsymbol{p_d}(\boldsymbol{x}) \\ \boldsymbol{b} \sim \boldsymbol{p_b}(\boldsymbol{b})}} \sum_{i=1}^{D} b_i \log p_\theta(x_i | \boldsymbol{x}_{\mathbf{1}-\boldsymbol{b}}, \boldsymbol{b})$$

Universal Marginalizer



$$p_\theta\left(x_i \middle| \boldsymbol{x}_{\mathbf{1}-\boldsymbol{b}}, \boldsymbol{b}\right)$$

Joint distribution: chain rule

- Choose $i \in \boldsymbol{b}$
- Sample $x_i \sim p_\theta\left(x_i | \boldsymbol{x}_{\boldsymbol{1}-\boldsymbol{b}}, \boldsymbol{b}\right)$
- Update $\boldsymbol{b} \Leftarrow \boldsymbol{b} - \boldsymbol{e}_{\boldsymbol{i}}$
- Repeat while $\boldsymbol{b} \neq \boldsymbol{0}$
- Log-likelihood: don’t sample x_i, but compute the product of conditional probabilities instead

Here ends the original paper

Joint distribution

- Choose $i \in \boldsymbol{b}$
  - Sequential left to right:

$$\boldsymbol{e}_{1..i} = (\overbrace{0, 0, \ldots, 0}^{i}, 1, 1, \ldots, 1)$$

$$p_\theta(\boldsymbol{x}_{\boldsymbol{b}} | \boldsymbol{x}_{\boldsymbol{1}-\boldsymbol{b}}, \boldsymbol{b}) = \prod_{i \in \boldsymbol{b}} p_\theta(x_i | \boldsymbol{x}_{\boldsymbol{1}-\boldsymbol{b} \wedge \boldsymbol{e}_{1..i}}, \boldsymbol{b} \wedge \boldsymbol{e}_{1..i})$$

Joint distribution

- Choose $\boldsymbol{i} \in \boldsymbol{b}$
  - Sequential left to right
  - At random uni-probable

Joint distribution

- Choose $i \in \boldsymbol{b}$
  - Sequential left to right
  - At random uni-probable
- The distribution over $\boldsymbol{b}$ at test stage is not $p_b(\boldsymbol{b})$
- This inconsistency ruins everything
- Need generative process for induced $\hat{p}_b(\boldsymbol{b})$

Consistent mask generative process

Generative process for $\hat{p}_b(\boldsymbol{b})$

Choose $i \in \boldsymbol{b}$

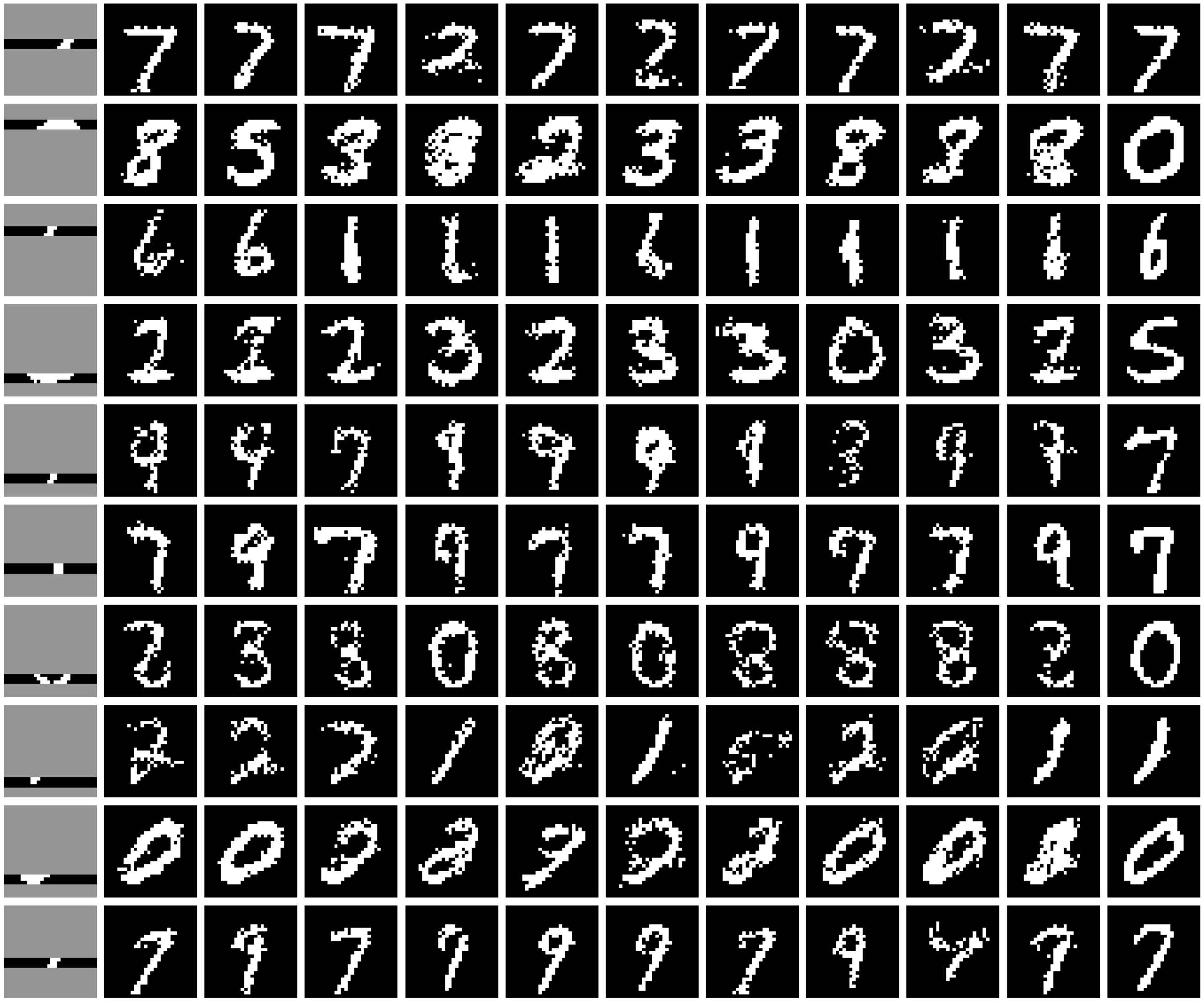
Sequential left to right

- $\boldsymbol{b} \sim p_b(\boldsymbol{b})$
- $u \sim U_D[0, 1, \ldots, \sum \boldsymbol{b}]$
- j: u[th] 1 in $\boldsymbol{b}$
- $\hat{\boldsymbol{b}} = \boldsymbol{b} \wedge \boldsymbol{e}_{1..j}$

At random uni-probable

- $\boldsymbol{b} \sim p_b(\boldsymbol{b})$
- $u \sim U[0, 1]$
- $\boldsymbol{b_0} \sim Bernoulli(u)$
- $\hat{\boldsymbol{b}} = \boldsymbol{b_0} \circ \boldsymbol{b}$

MNIST inpaintings



Universal Marginalizer

- Needs fast generative process for induced $\hat{p}_b(\boldsymbol{b})$ for given $p_b(\boldsymbol{b})$
  - Allows to keep the training speed for one epoch
- Needs O(D) time to generate sample or estimate likelihood
- Takes into account local dependencies
- The relation to UCM is similar to the relation between VAE and PixelCNN