2. Variational AutoEncoders

M. Ravasi

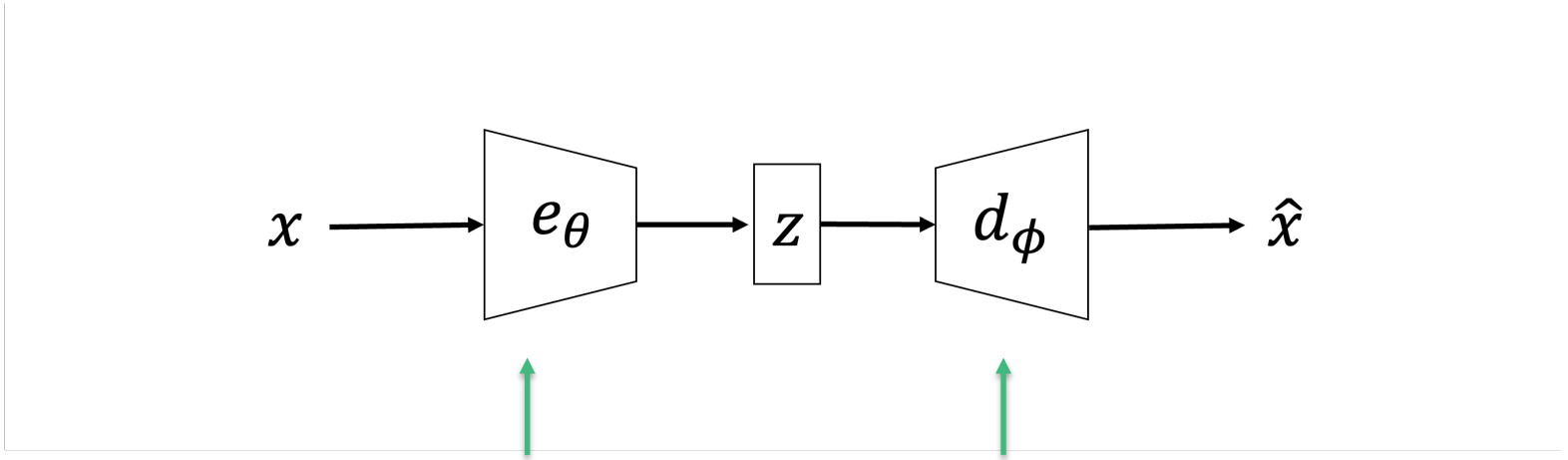
AI Summer School @ KAUST

Autoencoders

*Family of neural networks for which the **input is the same as the output**. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation.*

Autoencoders

*Family of neural networks for which the **input is the same as the output**. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation.*

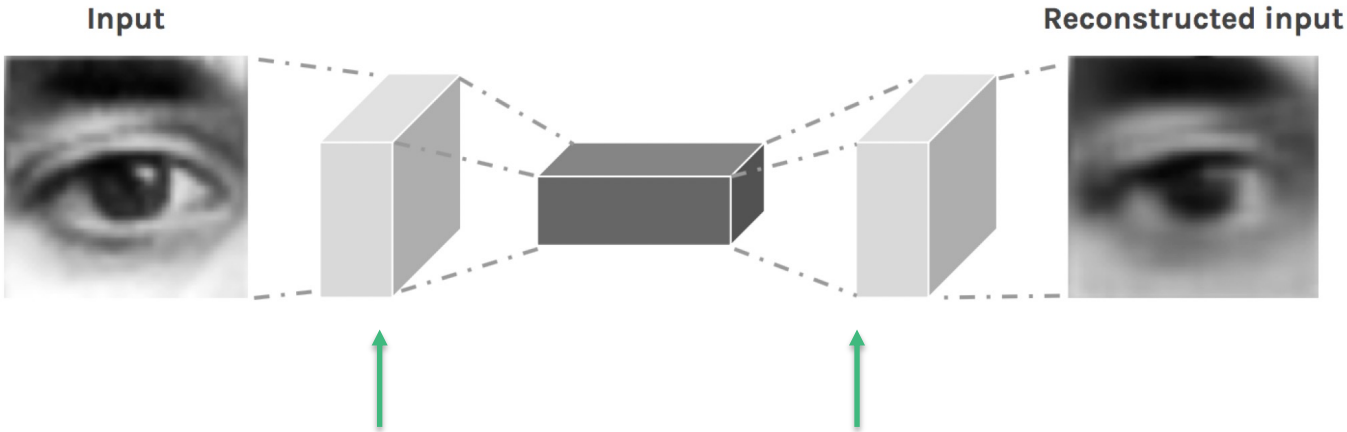


Encoder: project into to latent space

Decoder: reconstructs from latent code

Autoencoders

*Family of neural networks for which the **input is the same as the output**. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation.*



Encoder: project into to latent space

Decoder: reconstructs from latent code

Autoencoders

*Family of neural networks for which the **input is the same as the output**. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation.*

https://douglasduhaime.com/posts/visualizing-latent-spaces.html

Autoencoders

Mathematically described as:

$$\hat{\mathbf{x}} = d_\phi(e_\theta(\mathbf{x}))$$

The free-parameters of the 2 networks are optimized by minimizing the following loss function:

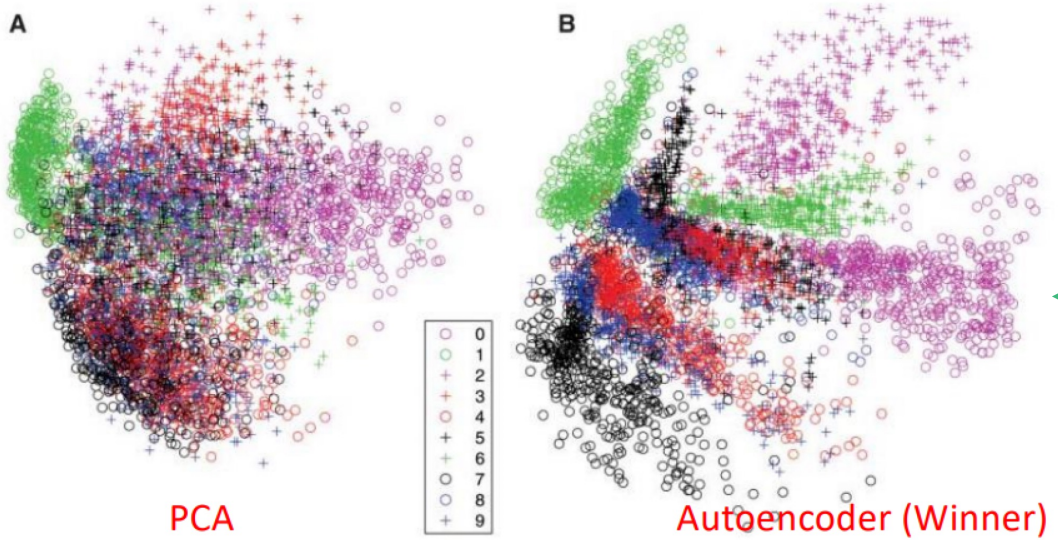
$$\hat{e}_\theta, \hat{d}_\phi = \underset{e_\theta, d_\phi}{\operatorname{argmin}} \frac{1}{N_s} \sum_i \mathcal{L}(\mathbf{x}^{(i)}, d_\phi(e_\theta(\mathbf{x}^{(i)})))$$

Autoencoders - applications

Data compression

Autoencoders - applications

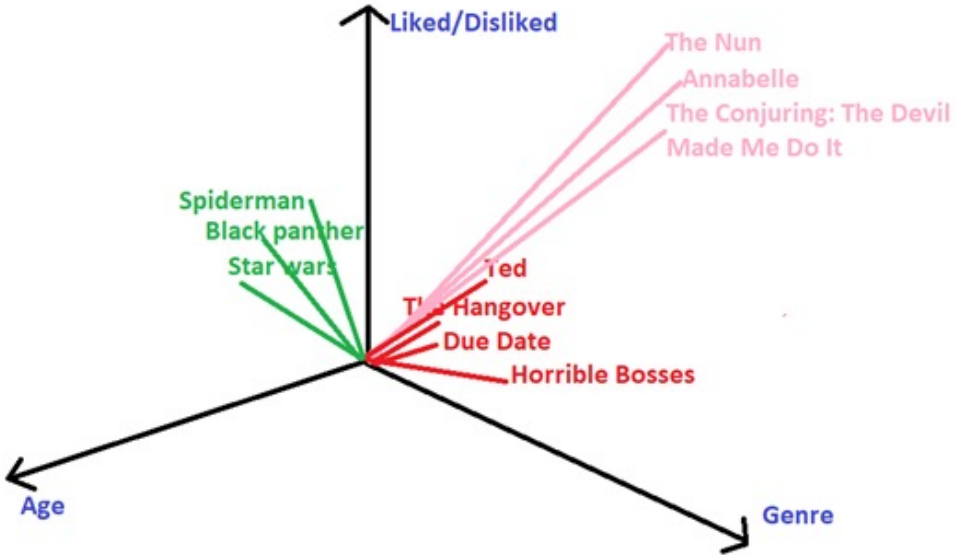
Learn robust features to be input for subsequent supervised task



Easier to do classification

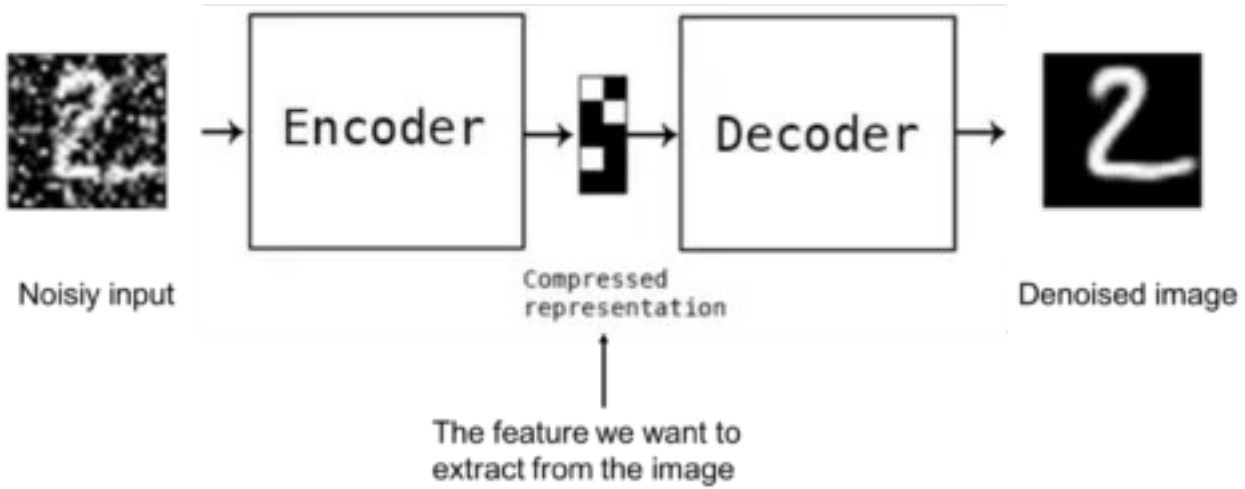
Autoencoders - applications

Perform vector math in latent space (easier to compare vectors…)

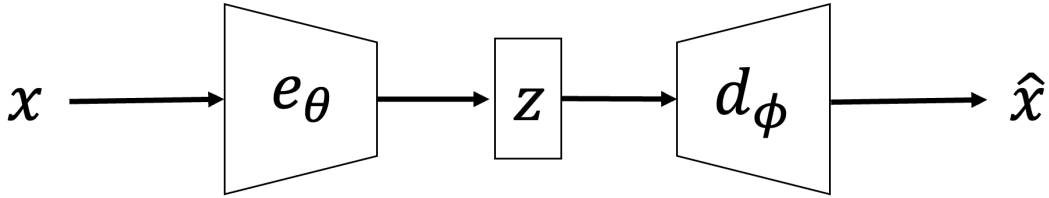


Autoencoders - applications

Denoising



Under- vs Overcomplete Autoencoders

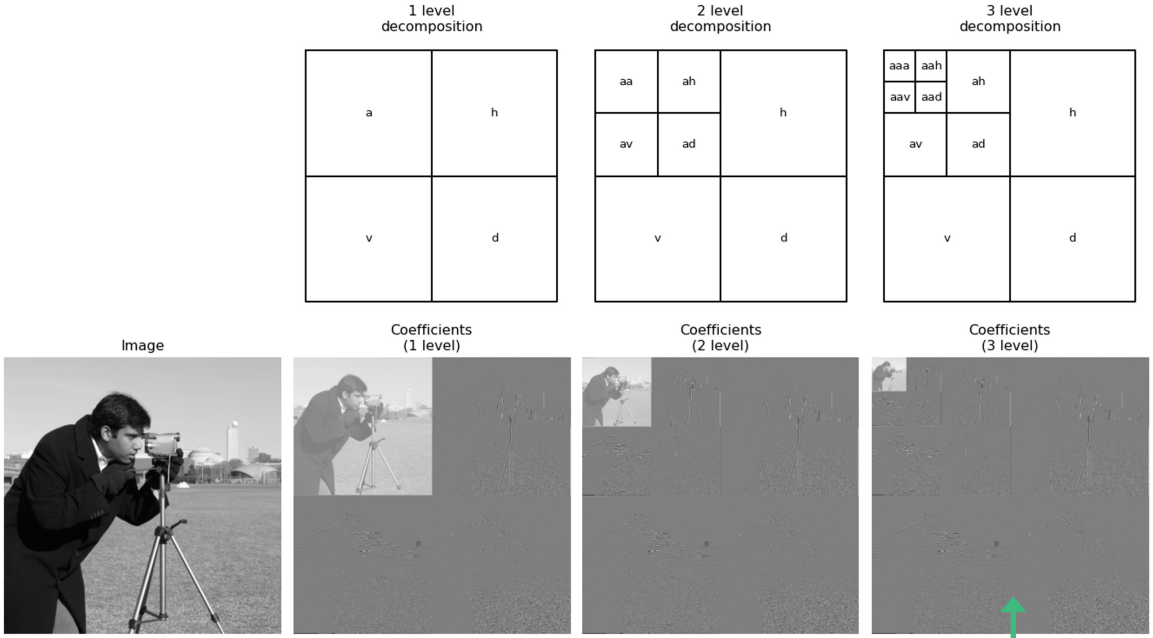


$N_z << N_x$ Undercomplete → risk of high compression loss

$N_x \approx N_z$ Overcomplete → risk of overfitting (learn identity)

Need for regularization

Overcomplete Transforms



Source: https://medium.com/@koushikc2000/2d-discrete-wavelet-transformation-and-its-applications-in-digital-image-processing-using-matlab-1f5c68672de3

Sparse representation

Overcomplete AEs

Add regularization to loss – **Sparse Autoencoders**

$$\mathcal{L}_r = \frac{1}{N_s} \sum_i \left(\mathcal{L}(\mathbf{x}^{(i)}, d_\phi(e_\theta(\mathbf{x}^{(i)}))) + \lambda R(\mathbf{x}^{(i)}; \theta, \phi) \right)$$

Overcomplete AEs

Add regularization to loss – **Sparse Autoencoders**

$$\mathcal{L}_r = \frac{1}{N_s} \sum_i \left(\mathcal{L}(\mathbf{x}^{(i)}, d_\phi(e_\theta(\mathbf{x}^{(i)}))) + \lambda R(\mathbf{x}^{(i)}; \theta, \phi) \right)$$

- L1 on latent space: $R(\mathbf{x}^{(i)}; \theta, \phi) = ||e_\theta(\mathbf{x}^{(i)})||_1$

Overcomplete AEs

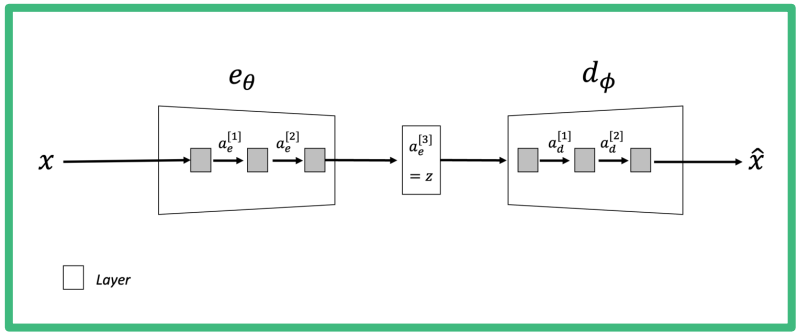
Add regularization to loss – **Sparse Autoencoders**

$$\mathcal{L}_r = \frac{1}{N_s} \sum_i \Big(\mathcal{L}(\mathbf{x}^{(i)}, d_\phi(e_\theta(\mathbf{x}^{(i)}))) + \lambda R(\mathbf{x}^{(i)}; \theta, \phi) \Big)$$

- L1 on latent space: $R(\mathbf{x}^{(i)}; \theta, \phi) = ||e_\theta(\mathbf{x}^{(i)})||_1$

- L1 on all activations:

$$R(\mathbf{x}^{(i)}; \theta, \phi) = \sum_j ||a_e^{[j](i)}||_1 + \sum_j ||a_d^{[j](i)}||_1$$



Overcomplete AEs

Add regularization to loss – **Contractive Autoencoders**

$$\mathcal{L}_r = \frac{1}{N_s} \sum_i \left(\mathcal{L}(\mathbf{x}^{(i)}, d_\phi(e_\theta(\mathbf{x}^{(i)}))) + \lambda R(\mathbf{x}^{(i)}; \theta, \phi) \right)$$

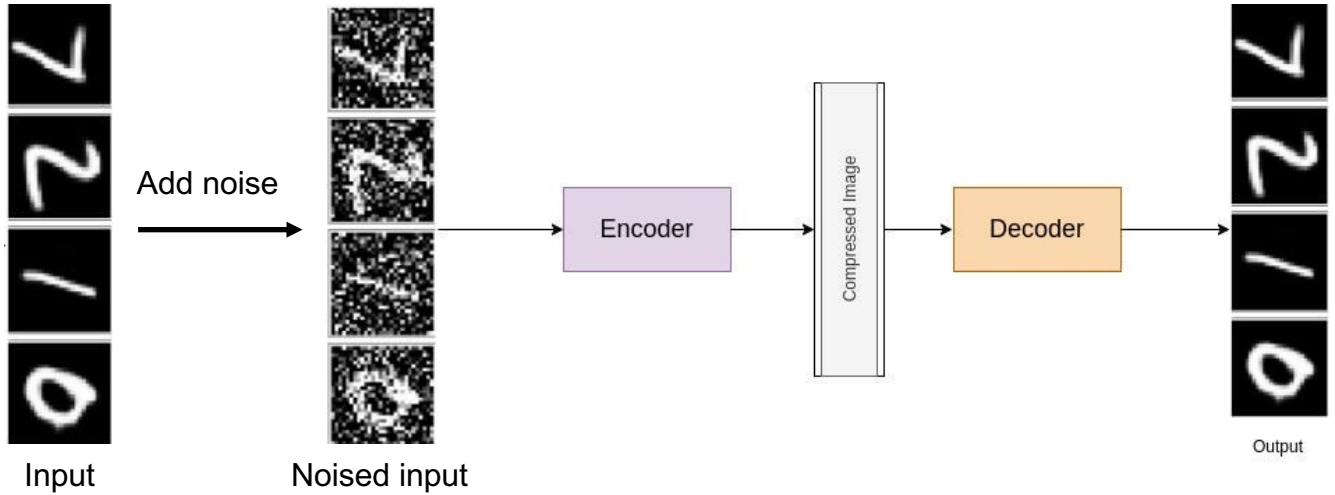
- Derivative of the latent vector over the input:

$$R(\mathbf{x}; \theta, \phi) = ||\nabla_{\mathbf{x}} \mathbf{z}||_F$$

*encourages robust latent vectors that have **small sensitivity to small perturbations of the input***

Overcomplete AEs

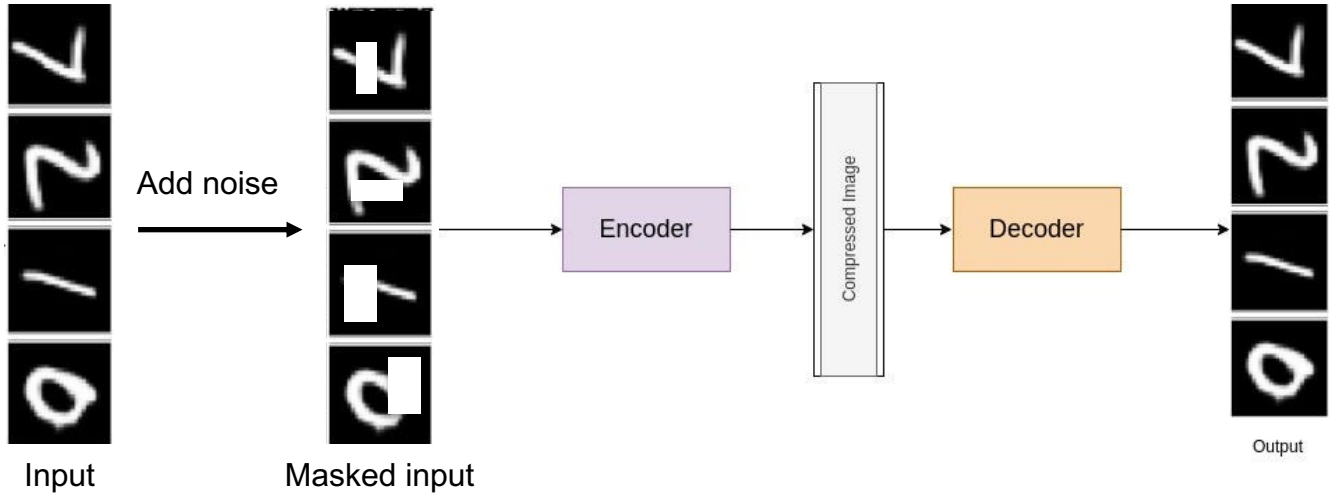
Corrupt input – **Denoising Autoencoders**



$$\tilde{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} + \mathbf{n}^{(i)} \quad \forall i$$

Overcomplete AEs

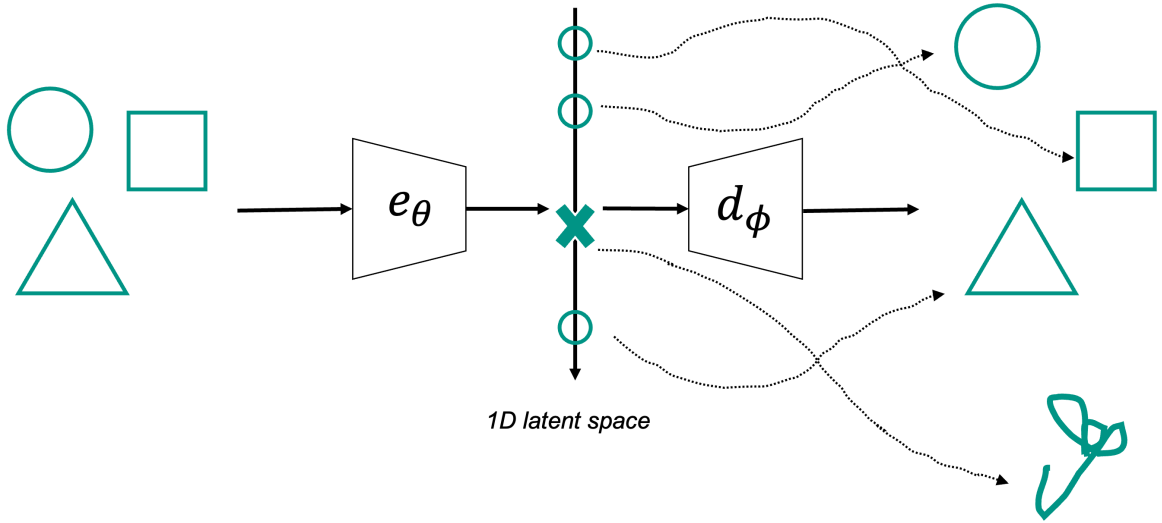
Corrupt input – **Denoising Autoencoders**



$$\tilde{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} \odot \mathbf{m}^{(i)} \quad \forall i$$

Autoencoders as generative models

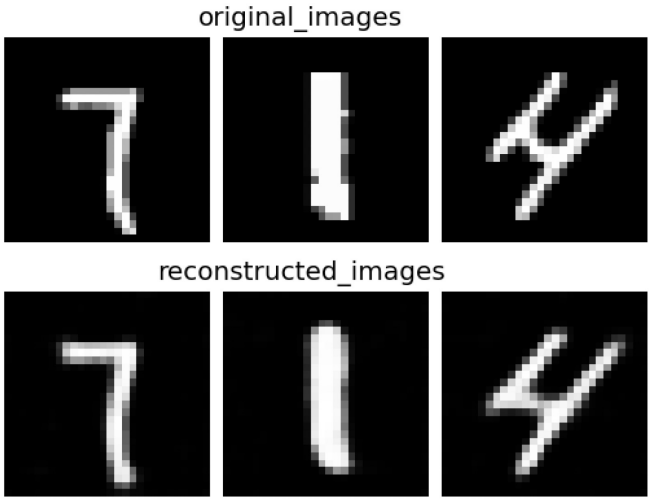
What if we **sample a new embedding vector** from z and then have the decoder reconstruct the "input" from it?

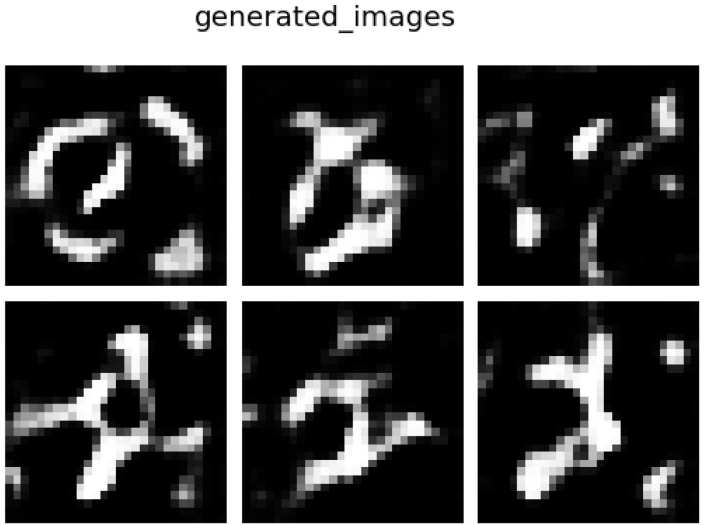


*Latent space is too **discontinous***

Autoencoders as generative models

What if we **sample a new embedding vector** from z and then have the decoder reconstruct the “input” from it?

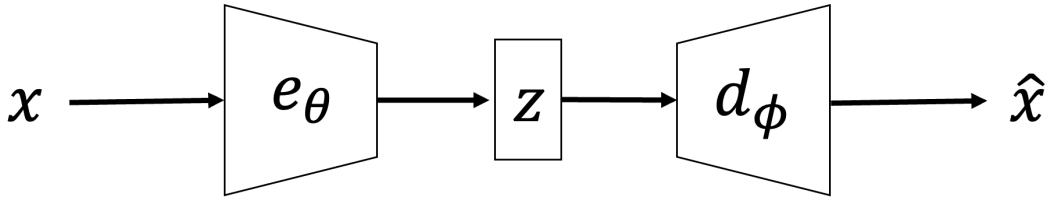




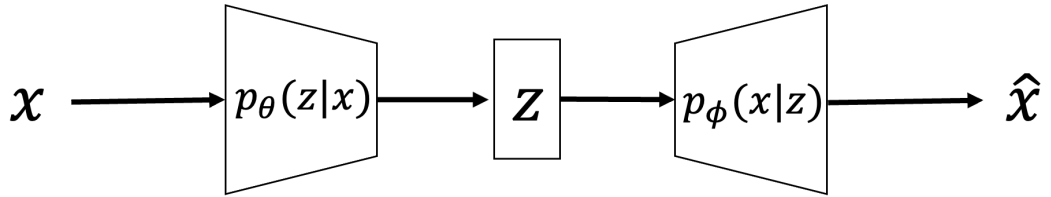
*Latent space is too **discontinous***

Variational Autoencoders (VAEs)

AE

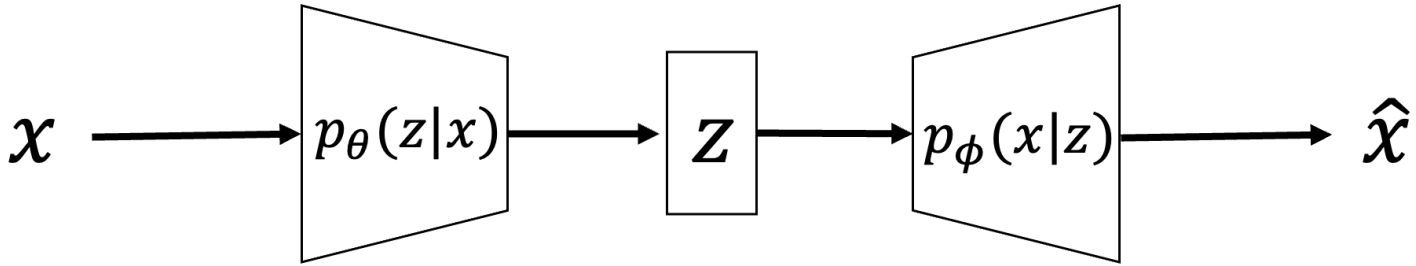


VAE



VAEs key addition: latent space sampling

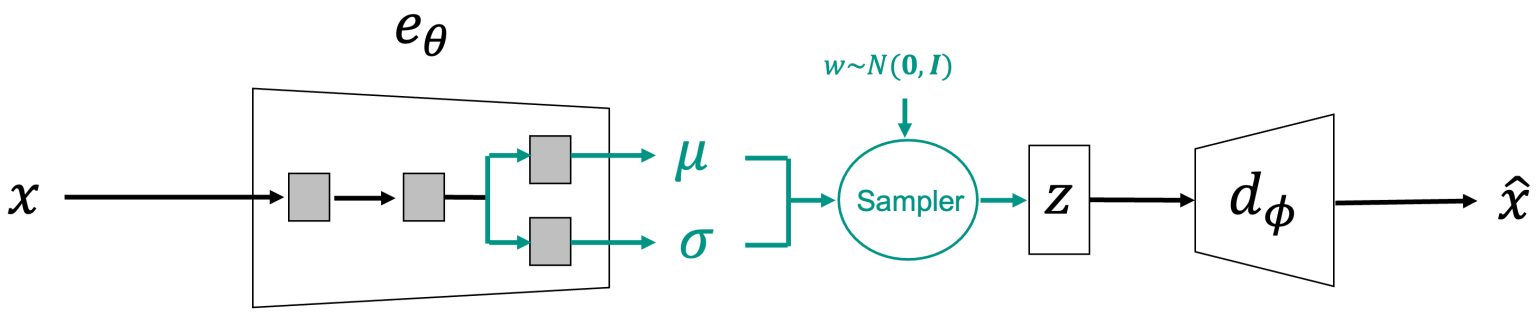
Variational AE add a probabilistic spin to AE: allow ***sampling the latent space***



The two networks learn to sample some conditional probabilities…

VAEs key addition: latent space sampling

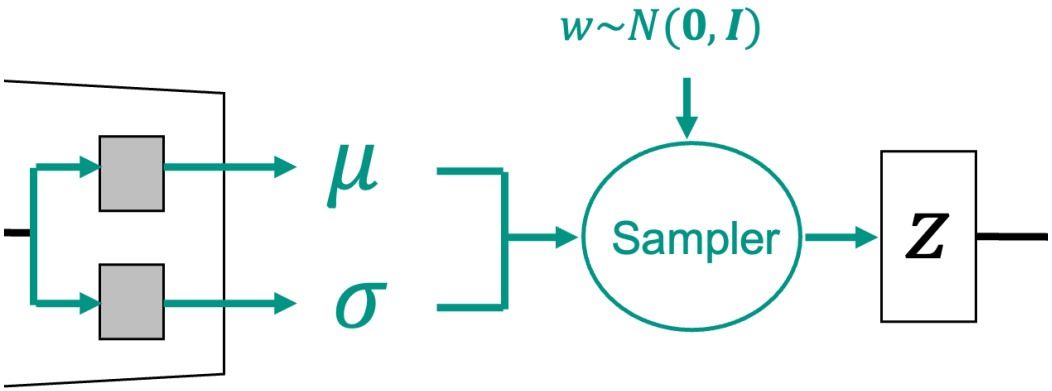
Variational AE add a probabilistic spin to AE: allow ***sampling the latent space***



2 key changes:
- Encoding process
- Learning process (i.e., loss)

VAEs key addition: latent space sampling

Variational AE add a probabilistic spin to AE: allow ***sampling the latent space***

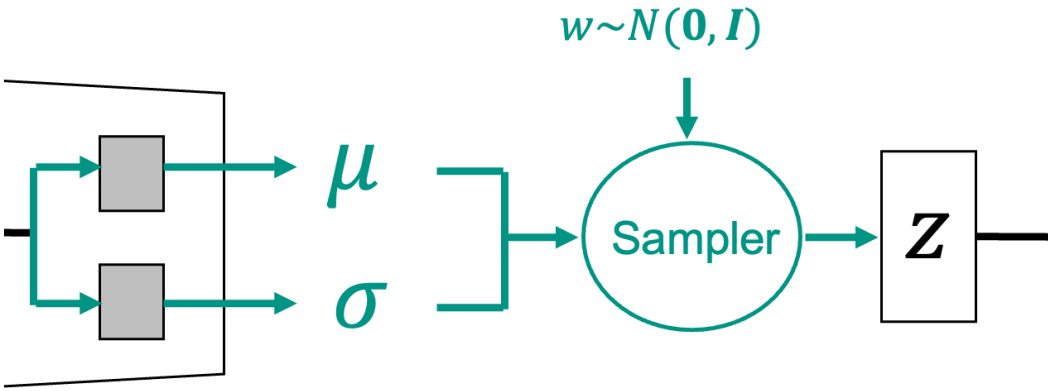


$q_\theta(\mathbf{z}) \approx p(\mathbf{z}|\mathbf{x})$: Proposal distribution

$p(\mathbf{z})$: Prior distribution

VAEs key addition: latent space sampling

Variational AE add a probabilistic spin to AE: allow ***sampling the latent space***



$q_\theta(\mathbf{z}) \approx p(\mathbf{z}|\mathbf{x})$: Proposal distribution

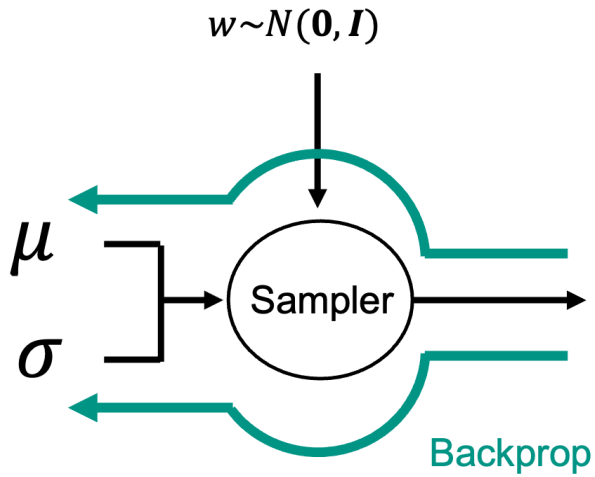
The learning process for θ changes…

$p(\mathbf{z})$: Prior distribution

Free to choose ourselves
(usually normal distribution)

Reparametrization trick

Something is required to sample a distribution, whilst still being able to do backprop



$$z = \mu + \sigma \cdot w$$

$$\frac{\partial z}{\partial \mu} = 1$$

$$\frac{\partial z}{\partial \sigma} = w$$

VAEs key addition: loss function

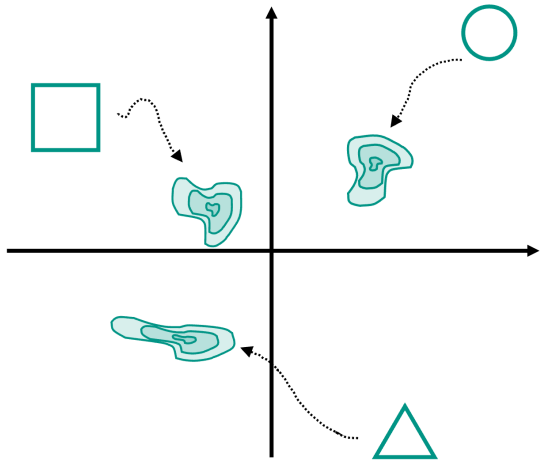
Add ‘probability’ regularization to loss

$$\mathcal{L}_r = \frac{1}{N_s} \sum_i \left(\mathcal{L}(\mathbf{x}^{(i)}, d_\phi(e_\theta(\mathbf{x}^{(i)}))) + KL(p(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})||p(\mathbf{z}^{(i)})) \right)$$

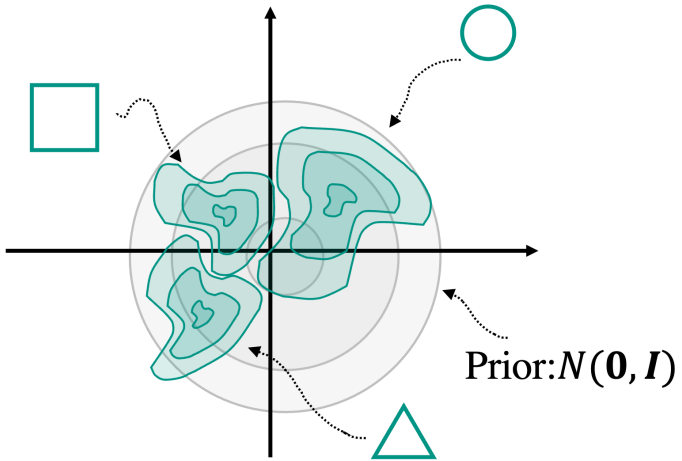
Kullback-Leibler divergence

VAEs key addition: loss function

Add ‘probability ’regularization to loss



AE latent space



VAE latent space

Kullback-Leibler divergence

Measure of closeness between two probability distributions

$$D_{KL}(Q||P) = E_{x\sim Q}\left[log\frac{q(x)}{p(x)}\right] = \int q(x)log\frac{q(x)}{p(x)}dx$$

- **Assymetric metric**, not really a distance metric.
- **KL = 0:** similar, if not the same, behavior of two different distributions.
- **KL = 1:** two distributions behave in such a different manner that the expectation given the first distribution approaches zero.

Kullback-Leibler divergence

Measure of closeness between two probability distributions

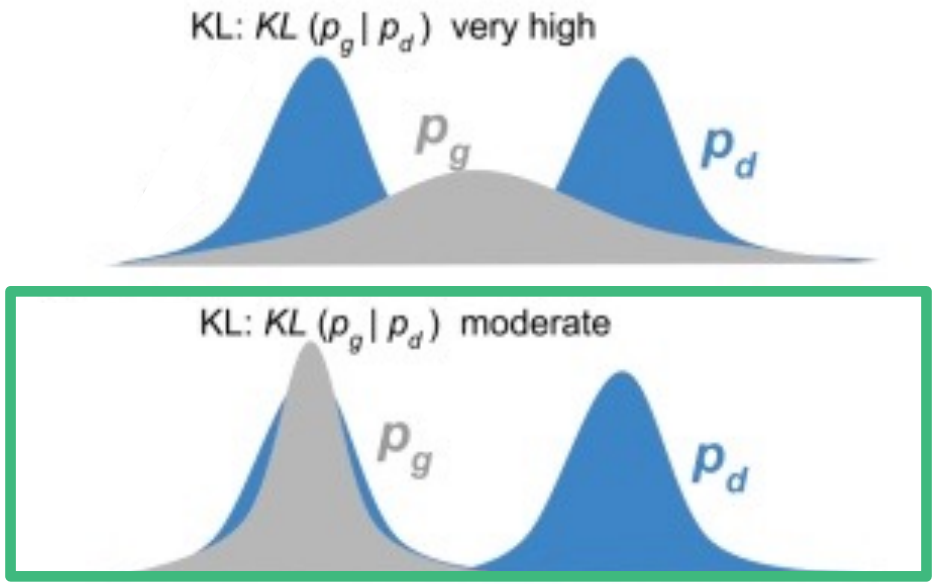
$$D_{KL}(Q||P) = E_{x\sim Q}[logq(x)] - E_{x\sim Q}[logp(x)]$$

- **Assymetric metric**, not really a distance metric.
- **KL = 0:** similar, if not the same, behavior of two different distributions.
- **KL = 1:** two distributions behave in such a different manner that the expectation given the first distribution approaches zero.

Kullback-Leibler divergence

Measure of closeness between two probability distributions

p_d: target distr.
p_g: matching distr.



This is what we will aim to achieve…

Variational Inference

Mathematical framework to **optimize parameters of one distribution to match another distribution** (usually impossible to sample and evaluate)

$$\underset{\theta}{\operatorname{argmin}}\ KL(q_\theta(\mathbf{z})||p(\mathbf{z}|\mathbf{x}))$$

Bayes Rule: $$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$$

Variational Inference (derivation)

$$\underset{\theta}{\operatorname{argmin}}\ KL(q_\theta(\mathbf{z})||p(\mathbf{z}|\mathbf{x}))$$

$$\underset{\theta}{\operatorname{argmin}}\ E_{\mathbf{z}\sim q_\theta}\left[log\left(\frac{q_\theta(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}\right)\right] =$$

$$\underset{\theta}{\operatorname{argmin}}\ E_{\mathbf{z}\sim q_\theta}[logq_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{z}|\mathbf{x})] =$$

$$\underset{\theta}{\operatorname{argmin}}\ E_{\mathbf{z}\sim q_\theta}[logq_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q_\theta}\left[log\left(\frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}\right)\right] =$$

Evidence: No dependance on θ

$$\underset{\theta}{\operatorname{argmin}}\ E_{\mathbf{z}\sim q_\theta}[logq_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{x}|\mathbf{z})] - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{z})] + \cancel{E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{x})]} =$$

$$\underset{\theta}{\operatorname{argmin}}\ KL(q_\theta(\mathbf{z})||p(\mathbf{z})) - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{x}|\mathbf{z})]$$

Variational Inference (derivation)

$$\underset{\theta}{\operatorname{argmin}}\ KL(q_\theta(\mathbf{z})||p(\mathbf{z}|\mathbf{x}))$$

$$\underset{\theta}{\operatorname{argmin}}\ E_{\mathbf{z}\sim q_\theta}\left[log\left(\frac{q_\theta(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}\right)\right] =$$

$$\underset{\theta}{\operatorname{argmin}}\ E_{\mathbf{z}\sim q_\theta}[logq_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{z}|\mathbf{x})] =$$

$$\underset{\theta}{\operatorname{argmin}}\ E_{\mathbf{z}\sim q_\theta}[logq_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q_\theta}\left[log\left(\frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}\right)\right] =$$

Evidence: No dependance on θ

$$\underset{\theta}{\operatorname{argmin}}\ E_{\mathbf{z}\sim q_\theta}[logq_\theta(\mathbf{z})] - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{x}|\mathbf{z})] - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{z})] + \cancel{E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{x})]} =$$

$$\underset{\theta}{\operatorname{argmin}}\ KL(q_\theta(\mathbf{z})||p(\mathbf{z})) - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{x}|\mathbf{z})]$$

KL on prior (tractable)

Classic ML estimator on decoder (tractable)

Evidence Lower BOund (ELBO)

A different interpretation of learning process

$$logp(\mathbf{x}) - KL(q_\theta(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{x}|\mathbf{z})] - KL(q_\theta(\mathbf{z})||p(\mathbf{z}))$$

Probability we want to maximize in generative modelling

What we actually maximise

Since KL is always positive (lower bound on evidence):

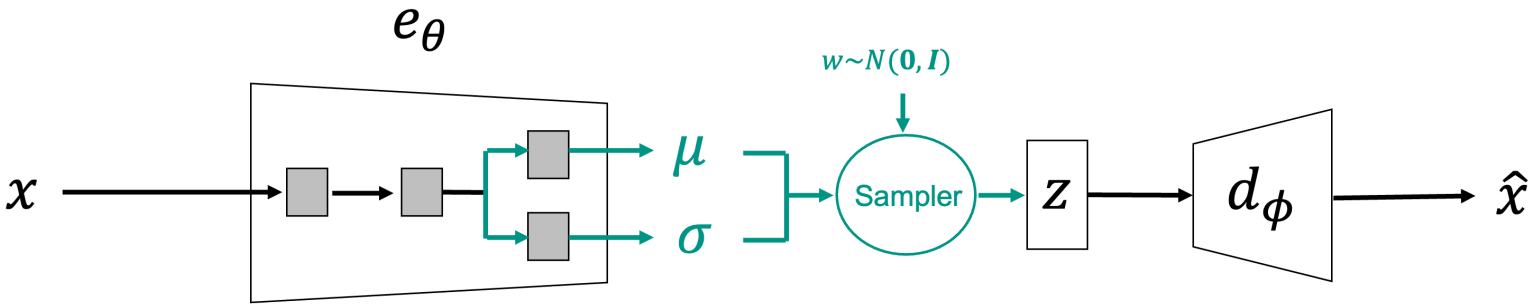
$$logp(\mathbf{x}) - KL(q_\theta(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) \leq logp(\mathbf{x})$$

VAE in practice

$$\underset{\theta}{\operatorname{argmin}}\ KL(q_\theta(\mathbf{z})||p(\mathbf{z})) - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{x}|\mathbf{z})]$$

$$q_\theta(\mathbf{z}) \sim \mathcal{N}(e_{\theta,\mu}(\mathbf{x}), diag(e_{\theta,\sigma}(\mathbf{x})))$$

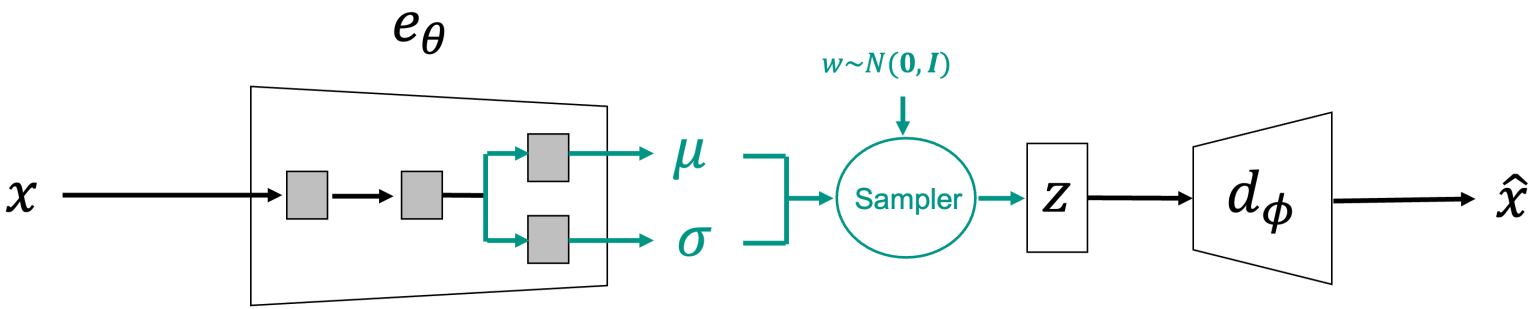
$$p(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$



VAE in practice

$$\underset{\theta}{\operatorname{argmin}}\ KL(q_\theta(\mathbf{z})||p(\mathbf{z})) - E_{\mathbf{z}\sim q_\theta}[logp(\mathbf{x}|\mathbf{z})]$$

$$p(\mathbf{x}|\mathbf{z}) \sim \mathcal{N}(d_\phi(\mathbf{z}), \mathbf{I}) \longrightarrow -logp(\mathbf{x}|\mathbf{z}) = \|\mathbf{x} - d_\phi(\mathbf{z})\|_2^2$$



VAE in practice

Learning Algorithm

1. Select a batch of training samples and feed to encoder
2. Divide the output of the encoder into two parts: mean and std (each of them has the same number of samples as the input batch)
3. Sample a latent space realization per training sample
4. Feed each latent vector into the decoder
5. Compute the loss: 1) MSE of reconstruction, 2) KL between prior and proposal using the mean and std obtained in step2. + backprop.

VAE in practice

Sampling Algorithm

1. Sample a latent space realization
2. Feed each latent vector into the decoder

Encoder is not used in inference!