Machine Learning Department
School of Computer Science
Carnegie Mellon University

Variational Autoencoders (VAEs)

+

In-context Learning

Matt Gormley & Aran Nayebi
Lecture 9
Sep. 24, 2025

Q&A

Q: Is this correct?

3. Gaussian with a Gaussian mean has a Gaussian Conditional

$$Z \sim \mathcal{N}(\mu_z = X, \sigma_z^2) \Rightarrow P(Z \mid X) \sim \mathcal{N}(\cdot, \cdot)$$

4. But #3 does not hold if X is passed through a nonlinear function f

$$W \sim \mathcal{N}(\mu_z = f(X), \sigma_w^2) \not\Rightarrow P(W \mid X) \sim \mathcal{N}(\cdot, \cdot)$$

A: No.

Gaussians (an aside)

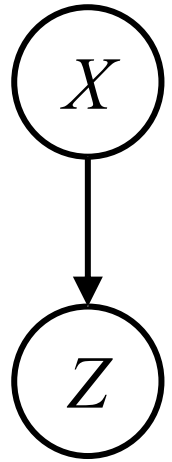
Let $X \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y^2)$

1. Sum of two Gaussians is a Gaussian. $X+Y \sim \mathcal{N}(\mu_x+\mu_y, \sigma_x^2+\sigma_y^2)$
2. Difference of two Gaussians is a Gaussian. $X-Y \sim \mathcal{N}(\mu_x-\mu_y, \sigma_x^2+\sigma_y^2)$

Let $Z \mid X \sim \mathcal{N}(\mu_z = g(X), \sigma_z^2)$

3. Gaussian with a Gaussian mean passed through a *linear/affine* function $g(\cdot)$ has a Gaussian marginal,

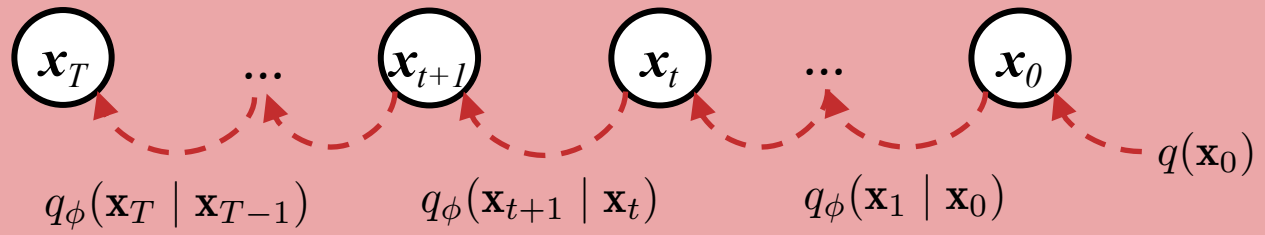
   $p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x)\,dx$ is a Gaussian density



4. Gaussian with a Gaussian mean passed through a *nonlinear* function $g(\cdot)$ does not have a Gaussian marginal,

   $p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x)\,dx$ is **not** a Gaussian density in general

Asymmetry of Forward/Reverse Processes



Forward Process:

$$q(\mathbf{x}_0) = \text{data distribution}$$

$$q_\phi(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \sim \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

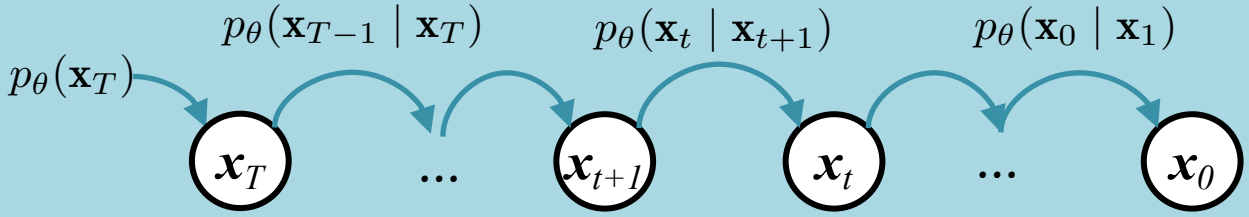
In the forward process, if we marginalize out the intermediate variables of the Markov chain, we get back a Gaussian:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \int \left[\prod_{s=1}^{t} q(\mathbf{x}_s \mid \mathbf{x}_{s-1})\right] d\mathbf{x}_{1:t-1}$$
$$= \mathcal{N}(\mathbf{x}_t \mid \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

(Learned) Reverse Process:

$$p_\theta(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \sim \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t))$$



In the reverse process, by contrast, if we marginalize out the intermediate variables of the Markov chain, we do NOT get a Gaussian.

$$p_\theta(\mathbf{x}_0) = \int \left[p_\theta(\mathbf{x}_T) \prod_{s=1}^{T} p_\theta(\mathbf{x}_{s-1} \mid \mathbf{x}_s)\right] d\mathbf{x}_{1:T}$$

$\neq$ a Gaussian density

Gaussians (an aside)

Let $X \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y^2)$

1. Sum of two Gaussians is a Gaussian. $X+Y \sim \mathcal{N}(\mu_x+\mu_y, \sigma_x^2+\sigma_y^2)$
2. Difference of two Gaussians is a Gaussian. $X-Y \sim \mathcal{N}(\mu_x-\mu_y, \sigma_x^2+\sigma_y^2)$

Let $Z \mid X \sim \mathcal{N}(\mu_z = g(X), \sigma_z^2)$

3. Gaussian with a Gaussian mean passed through a *linear/affine* function $g(\cdot)$ has a Gaussian marginal,

   $p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x)\,dx$ is a Gaussian density

   and a Gaussian posterior.

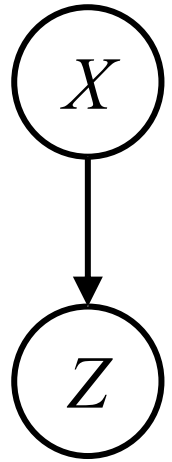
   $p_{X|Z}(x \mid z) = p_{Z|X}(z|x)p_X(x)/p_Z(z)$ is a Gaussian density

4. Gaussian with a Gaussian mean passed through a *nonlinear* function $g(\cdot)$ does not have a Gaussian marginal,

   $p_Z(z) = \int_x p_{Z|X}(z|x)p_X(x)\,dx$ is **not** a Gaussian density in general

   nor a Gaussian posterior.

   $p_{X|Z}(x \mid z) = p_{Z|X}(z|x)p_X(x)/p_Z(z)$ is **not** a Gaussian density in general



Properties of forward and *exact* reverse processes

Property #1:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$\text{where } \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

$\Rightarrow$ we can sample $\mathbf{x}_t$ from $\mathbf{x}_0$ at any timestep t efficiently in closed form

$\Rightarrow$ $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Property #2: Estimating $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is intractable because of its dependence on $q(\mathbf{x}_0)$. However, conditioning on $\mathbf{x}_0$ we can efficiently work with:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_t^2\mathbf{I})$$

$$\begin{aligned}\text{where } \tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\bar{\alpha}_t}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_t)}{1-\bar{\alpha}_t}\mathbf{x}_t \\ &= \alpha_t^{(0)}\mathbf{x}_0 + \alpha_t^{(t)}\mathbf{x}_t \\ \sigma_t^2 &= \frac{(1-\bar{\alpha}_{t-1})(1-\alpha_t)}{1-\bar{\alpha}_t}\end{aligned}$$

Property #3: Combining the two previous properties, we can obtain a different parameterization of $\tilde{\mu}_q$ which has been shown empirically to help in learning p_θ.

Rearranging $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ we have that:

$$\mathbf{x}_0 = \left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}\right) / \sqrt{\bar{\alpha}_t}$$

Substituting this definition of $\mathbf{x}_0$ into property #2's definition of $\tilde{\mu}_q$ gives:

$$\begin{aligned}\tilde{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) &= \alpha_t^{(0)}\mathbf{x}_0 + \alpha_t^{(t)}\mathbf{x}_t \\ &= \alpha_t^{(0)}\left(\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}\right) / \sqrt{\bar{\alpha}_t}\right) + \alpha_t^{(t)}\mathbf{x}_t \\ &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right)\end{aligned}$$

Reminders

- Quiz 2:
  - In-class: Mon, Sep 29
  - Everyone at the same time: 2:00 – 2:15pm
  - Lectures 5-9 (only the VAE part of Lecture 9)
- Homework 2: Generative Models of Images
  - Out: Mon, Sep 22
  - Due: Sat, Oct 4 at 11:59pm

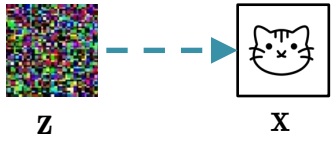
VARIATIONAL AUTOENCODERS

GAN → VAE → Diffusion (in 5 minutes)

GANs

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{x} \mid \mathbf{z}) =$ point mass on $G_\theta(\mathbf{z})$

Model:



MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^N \log p(\mathbf{x}^{(i)})$ NOPE!

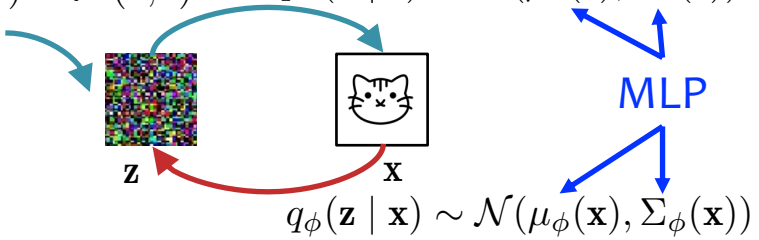
$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$

Learn: minimax game

VAEs

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$

Model:



$h = \tanh(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1)$
$\mu_\theta(\mathbf{z}) = \mathbf{W}_2 h + \mathbf{b}_2$
$\Sigma_\theta(\mathbf{z}) = (\mathbf{W}_3 h + \mathbf{b}_3)\mathbf{I}$

MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^N \log p(\mathbf{x}^{(i)})$ NOPE!

$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$

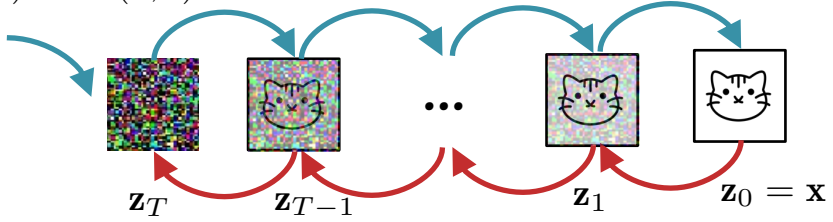
Learn: $\theta, \phi = \arg\min_{\theta,\phi} \mathrm{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z} \mid \mathbf{x}))$

where $p_\theta(\mathbf{z} \mid \mathbf{x})$ is true posterior of $p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})$

Diffusion

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t) \sim \mathcal{N}(\mu_\theta(\mathbf{z}_t), \Sigma_\theta(\mathbf{z}_t))$

Model:



$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \sim \mathcal{N}(\mu_\phi(\mathbf{z}_{t-1}), \Sigma_\phi(\mathbf{z}_{t-1}))$ $q(\mathbf{z}_0) =$ data dist.

MLE?: NOPE!

Learn: $\theta = \arg\min_\theta \mathrm{KL}(q(\mathbf{z}_{1:T} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z}_{1:T} \mid \mathbf{x}))$

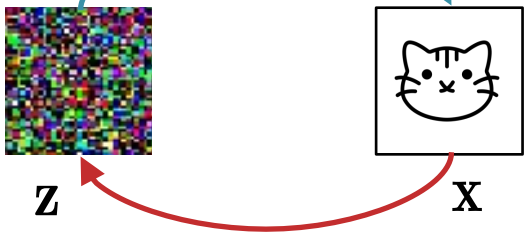
where $p_\theta(\mathbf{z}_{1:T} \mid \mathbf{x})$ is true posterior of $p(\mathbf{x} \mid \mathbf{z}_1) \cdots p(\mathbf{z}_{T-1} \mid \mathbf{z}_T) p(\mathbf{z}_T)$

Variational Autoencoder

VAEs

Model:

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$



$\mathbf{z}$ $\mathbf{x}$

MLP

$q_\phi(\mathbf{z} \mid \mathbf{x}) \sim \mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$

$$h = \tanh(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1)$$
$$\mu_\theta(\mathbf{z}) = \mathbf{W}_2 h + \mathbf{b}_2$$
$$\Sigma_\theta(\mathbf{z}) = (\mathbf{W}_3 h + \mathbf{b}_3)\mathbf{I}$$

MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)})$ NOPE!

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$$

Learn: $\theta, \phi = \arg\min_{\theta,\phi} \mathrm{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z} \mid \mathbf{x}))$

where $p_\theta(\mathbf{z} \mid \mathbf{x})$ is true posterior of $p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})$

Background for Variational Autoencoders (VAEs)

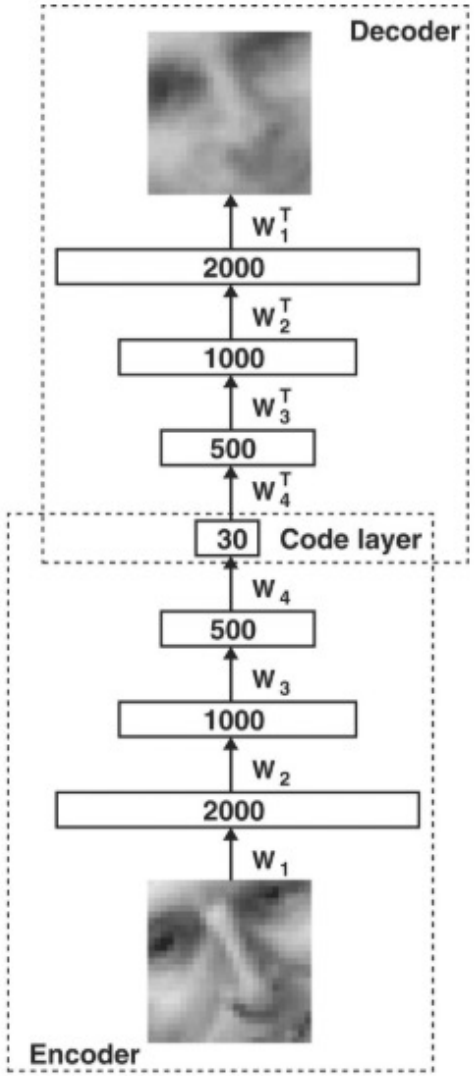
To talk about VAEs we need to lay some groundwork:

1. Autoencoders (not the variational kind)
2. KL divergence
3. Variational inference (using KL as an objective)
4. Monte Carlo estimation (for approximating expectations)
5. (Score function trick)
6. Reparameterization trick (for variance reduction)
7. Stochastic Gradient Variational Bayes

AUTOENCODERS
(NOT THE VARIATIONAL KIND)

Autoencoders

- **Data:** only unlabeled instances **x**
- **Model:** Autoencoders consist of two functions such that |x| = |x'|
  - **z** = ENCODER(**x**)
  - **x'** = DECODER(**z**)
- **Training:** minimize the reconstruction loss:
  $|| \mathbf{x} - \text{DECODER}(\text{ENCODER}(\mathbf{x})) ||^2$
- **Goal:** to learn a low-dimensional representation of the inputs



x', reconstruction

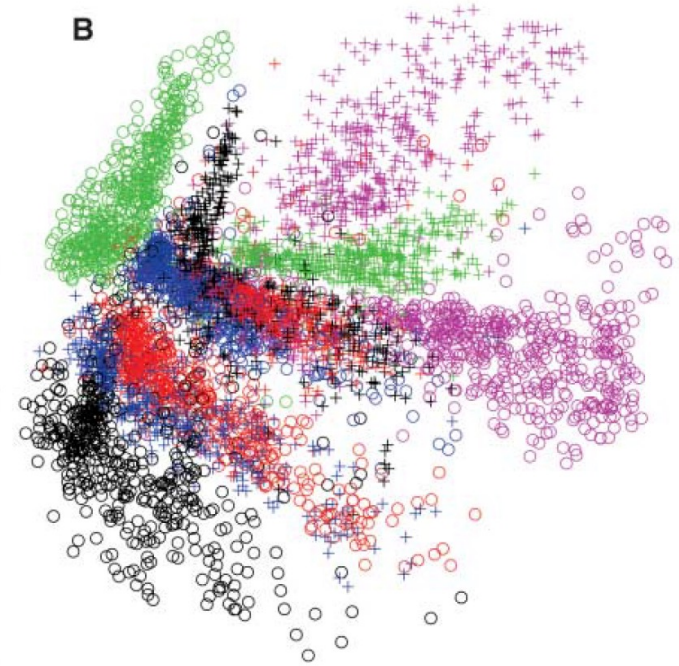
z, latent rep.

x, input

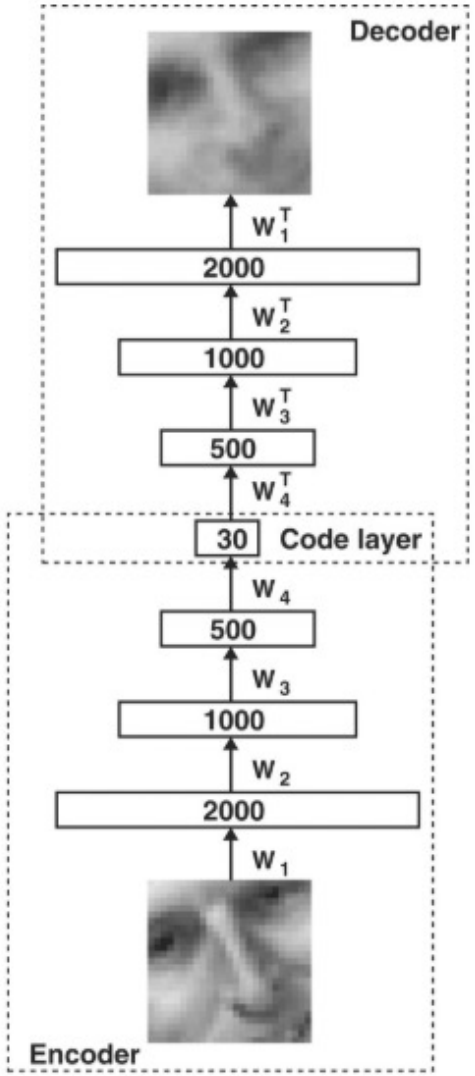
Figure from: https://www.science.org/doi/10.1126/science.1127647

Autoencoders

- **Problem:** the autoencoder does not define a probability distribution, so sampling from the latent space would effectively be sampling from the training instance reconstructions



autoencoder latent space



x', reconstruction

z, latent rep.

x, input

Figure from: https://www.science.org/doi/10.1126/science.1127647

KL DIVERGENCE

KL Divergence

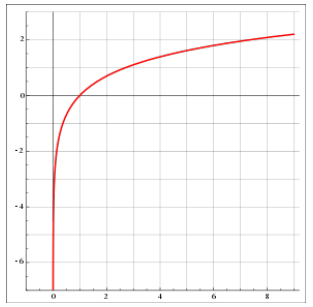
- *Definition*: for two distributions q(x) and p(x) over x ∈ $\mathcal{X}$, the **KL Divergence** is:

$$\mathrm{KL}(q||p) = E_{q(x)}\left[\log \frac{q(x)}{p(x)}\right] = \begin{cases} \sum_x q(x) \log \frac{q(x)}{p(x)} \\ \int_x q(x) \log \frac{q(x)}{p(x)} dx \end{cases}$$

- *Properties*:
  - KL(q || p) measures the **proximity** of two distributions q and p
  - KL is **not** symmetric: KL(q || p) ≠ KL(p || q)
  - KL is minimized when q(x) = p(x) for all x ∈ $\mathcal{X}$

$$\mathrm{KL}(q||p) = E_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]$$

KL Divergence



Understanding the Behavior of KL as an objective function

Example 1: Keeping all else constant, consider the effect of a particular x' on KL(q || p)

x'	q(x')	p(x')	q(x') log(q(x')/p(x'))	effect on KL(q || p)
1	0.4	0.4	0	no change
2	0.4	0.1	0.24	big increase
3	0.1	0.4	-0.06	little decrease
4	0.1	0.1	0	no change

KL **does** insist on good approximations for values that have **high** probability in q

KL **does not** insist on good approximations for values that have **low** probability in q

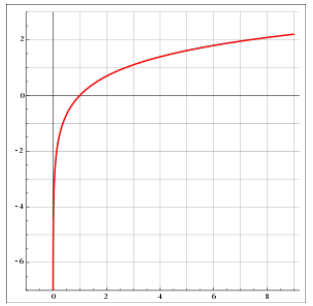
Example 2: Which q distribution minimizes KL(q || p)?

$$\mathbf{p} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \quad \mathbf{q}^{(1)} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \quad \mathbf{q}^{(2)} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \quad \mathbf{q}^{(3)} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

Q: If we're minimizing KL, why not return $q^{(3)}$?
A: Because it's not a distribution!

$$\mathrm{KL}(q||p) = E_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]$$

KL Divergence

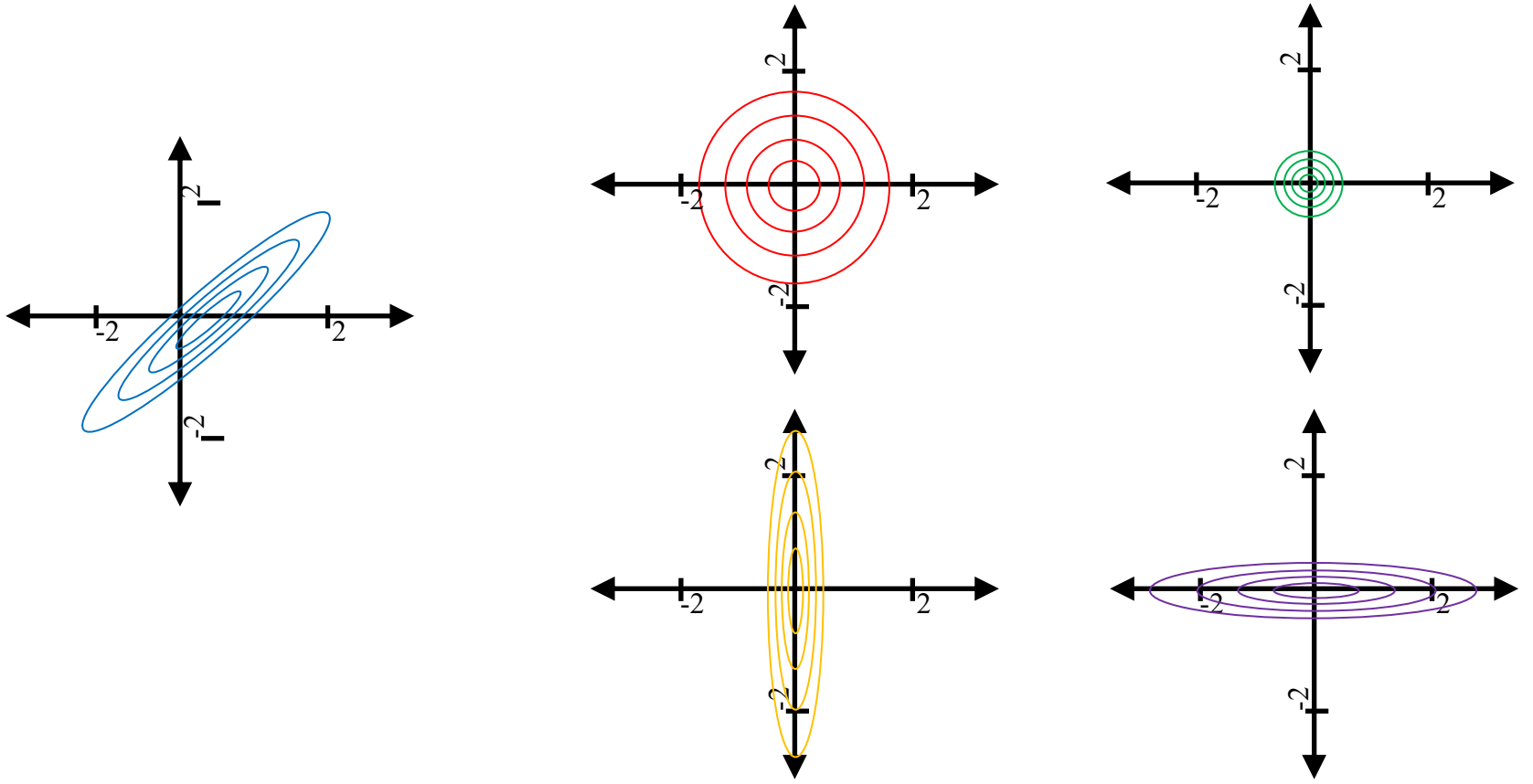


Understanding the Behavior of KL as an objective function

Example 3: Which q distribution minimizes KL(q || p)?

$$p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu} = [0, 0]^T, \boldsymbol{\Sigma})$$

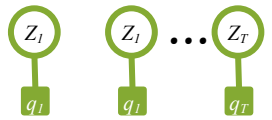
$$q(x_1, x_2) = \mathcal{N}_1(x_1 \mid \mu_1, \sigma_1^2)\mathcal{N}_2(x_2 \mid \mu_2, \sigma_2^2)$$



VARIATIONAL INFERENCE

Variational Inference Overview

1. *Goal*: estimate a distribution $p_\alpha(\mathbf{z} \mid \mathbf{x})$ given parameters α
   we assume this is intractable to compute exactly
2. *Variational Approximation*: approx. with another distribution
   $q_\theta(\mathbf{z} \mid \mathbf{x}) \approx p_\alpha(\mathbf{z} \mid \mathbf{x})$
   Example: the *mean field approximation* assumes
   $q_\theta(\mathbf{z} \mid \mathbf{x}) = \prod_t q_t(z_t \mid x; \theta)$ i.e., we decompose over variables



3. *Optimization Problem*: pick the q that minimizes KL(q || p)

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathrm{KL}(q_\theta(\mathbf{z} \mid \mathbf{x}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x}))$$

4. *Optimization Algorithm*: various options
   Example: gradient descent optimizes a surrogate objective
   ELBO(q_θ) to find θ

Optimizing KL Divergence

- *Question*: How do we minimize KL?

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathrm{KL}(q_\theta(\mathbf{z} \mid \mathbf{x}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x}))$$

- *Answer #1*: Oh no! We can't even compute this KL.

Why we can't compute KL…

$$\begin{aligned}
\mathrm{KL}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})) &= E_{q(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{q(\mathbf{z} \mid \mathbf{x})}{p(\mathbf{z} \mid \mathbf{x})}\right)\right] \\
&= E_{q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z} \mid \mathbf{x})\right] - E_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{z} \mid \mathbf{x})\right] \\
&= E_{q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z} \mid \mathbf{x})\right] - E_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}, \mathbf{z})\right] + E_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x})\right] \\
&= E_{q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z} \mid \mathbf{x})\right] - E_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}, \mathbf{z})\right] + \log p(\mathbf{x})
\end{aligned}$$

this expectation does not depend on q

we assumed this is intractable to compute!

Optimizing KL Divergence

- *Question*: How do we minimize KL?

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathsf{KL}(q_\theta(\mathbf{z} \mid \mathbf{x}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x}))$$

- *Answer #2*: We don't need to compute this KL
  We can instead maximize the ELBO (i.e. **E**vidence **L**ower **BO**und)

$$\mathsf{ELBO}(q_\theta) = E_{q_\theta(\mathbf{z}|\mathbf{x})}[\log p_\alpha(\mathbf{x}, \mathbf{z})] - E_{q_\theta(\mathbf{z}|\mathbf{x})}[\log q_\theta(\mathbf{z} \mid \mathbf{x})]$$

The ELBO for a DGM

Here is why…

$$\begin{aligned}
\theta &= \underset{\theta}{\operatorname{argmin}} \, \mathsf{KL}(q_\theta(\mathbf{z} \mid \mathbf{x}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x})) \\
&= \underset{\theta}{\operatorname{argmin}} \, E_{q_\theta(\mathbf{z}|\mathbf{x})}[\log q_\theta(\mathbf{z} \mid \mathbf{x})] - E_{q_\theta(\mathbf{z}|\mathbf{x})}[\log p_\alpha(\mathbf{x}, \mathbf{z})] + \log p_\alpha(\mathbf{x}) \\
&= \underset{\theta}{\operatorname{argmin}} \, E_{q_\theta(\mathbf{z}|\mathbf{x})}[\log q_\theta(\mathbf{z} \mid \mathbf{x})] - E_{q_\theta(\mathbf{z}|\mathbf{x})}[\log p_\alpha(\mathbf{x}, \mathbf{z})] \\
&= \underset{\theta}{\operatorname{argmax}} \, \mathsf{ELBO}(q_\theta)
\end{aligned}$$

dropping the intractable term gives the ELBO

ELBO as Objective Function

What does maximizing ELBO(q_θ) accomplish?

$$\text{ELBO}(q_\theta) = E_{q_\theta(\mathbf{z}|\mathbf{x})}\left[\log p_\alpha(\mathbf{x}, \mathbf{z})\right] - E_{q_\theta(\mathbf{z}|\mathbf{x})}\left[\log q_\theta(\mathbf{z} \mid \mathbf{x})\right]$$

1. The first expectation is high if q_θ puts probability mass on the same values of **z** that p_α puts probability mass

2. The second term is the entropy of q_θ and the entropy will be high if q_θ spreads its probability mass evenly

ELBO as lower bound

Theorem: For any q, $\log p(\mathbf{x}) \geq \text{ELBO}(q)$
i.e. $\text{ELBO}(q)$ is a lower bound for $\log p(\mathbf{x})$

Note:

$$\text{ELBO}(q_\theta) = E_{q_\theta(\mathbf{z}|\mathbf{x})}\left[\log p_\alpha(\mathbf{x}, \mathbf{z})\right] - E_{q_\theta(\mathbf{z}|\mathbf{x})}\left[\log q_\theta(\mathbf{z} \mid \mathbf{x})\right]$$

$$\text{KL}(q_\theta(\mathbf{z} \mid \mathbf{x}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x})) = E_{q_\theta(\mathbf{z}|\mathbf{x})}\left[\log q_\theta(\mathbf{z} \mid \mathbf{x})\right] - E_{q_\theta(\mathbf{z}|\mathbf{x})}\left[\log p_\alpha(\mathbf{x}, \mathbf{z})\right] + \log p_\alpha(\mathbf{x})$$

Proof #1:

1.
2.
3.

Takeaways:

1. in variational inference, we find the q that gives the **tightest bound** on the normalization constant for p(z | x)
2. maximizing the ELBO is equivalent to minimizing KL
3. maximizing the ELBO is maximizing a lower bound on the likelihood p(**x**)

MONTE CARLO ESTIMATION

Monte Carlo Estimation

Goal: Estimate the expectation of a function f of a (possibly high dimensional) random variable $\mathbf{X} \sim P(\mathbf{X})$:

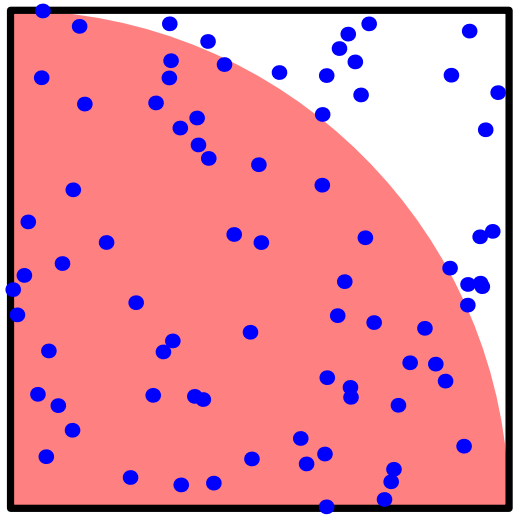
$$\phi = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}[f(\mathbf{x})] = \int_{\mathbf{x}} p(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x}$$

Monte Carlo Estimate: Approximate the expectation by drawing S samples from p and computing the average of the function f on the samples:

$$\hat{\phi} = \frac{1}{S} \sum_{s=1}^{S} f(\mathbf{x}^{(s)}) \text{ where } \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(S)} \overset{\text{i.i.d.}}{\sim} p(\mathbf{x})$$

Monte Carlo Estimation

Example: A Dumb Approximation of π



$$P(x,y) = \begin{cases} 1 & 0<x<1 \quad \text{and} \quad 0<y<1 \\ 0 & \text{otherwise} \end{cases}$$

$$\pi = 4 \iint \mathbb{I}\left((x^2+y^2)<1\right) P(x,y)\, \mathrm{d}x\, \mathrm{d}y$$

```
$ python monte-carlo-pi.py
S=100        pi ≈ 3.080000  error=0.061593
S=1000       pi ≈ 3.228000  error=0.086407
S=10000      pi ≈ 3.134400  error=0.007193
S=100000     pi ≈ 3.151360  error=0.009767
S=1000000    pi ≈ 3.140776  error=0.000817
```

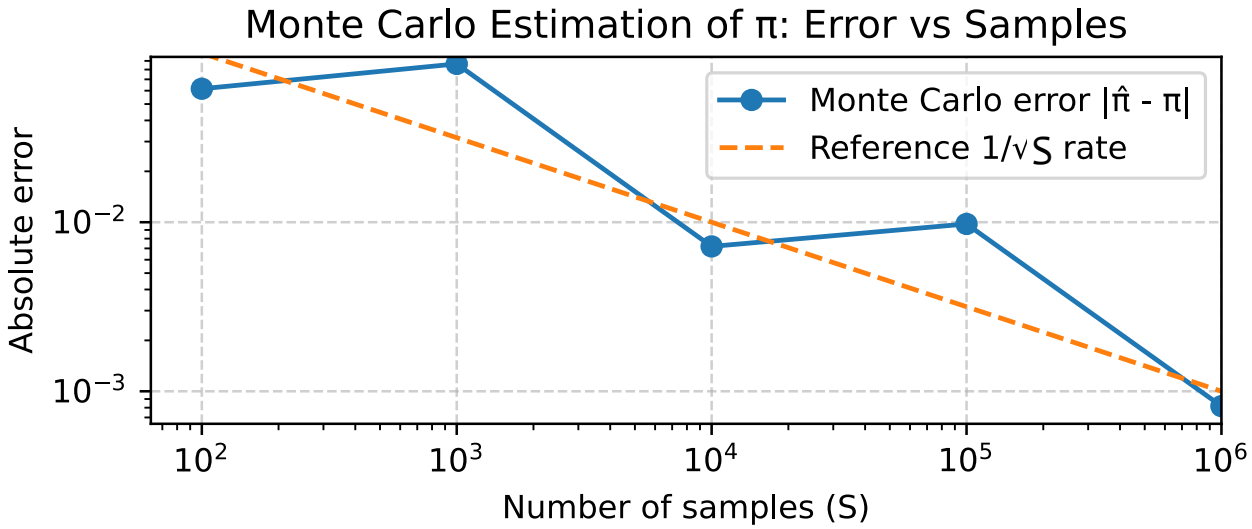


Figure / Example from Iain Murray https://homepages.inf.ed.ac.uk/imurray2/teaching/09mlss/slides.pdf

Monte Carlo Estimation

Properties:

The Monte Carlo estimate is an **unbiased estimator**:

$$\mathbb{E}[\hat{\phi}] = \mathbb{E}\left[\frac{1}{S}\sum_{s=1}^{S} f(\mathbf{x}^{(s)})\right] = \frac{1}{S}\sum_{s=1}^{S}\mathbb{E}[f(\mathbf{x}^{(s)})]$$
$$= \mathbb{E}[f(\mathbf{x})] = \phi$$

And the **variance** of $\hat{\phi}$ shrinks as σ^2/S where σ^2 is the variance of ϕ:

$$\text{Var}(\hat{\phi}) = \text{Var}\left(\frac{1}{S}\sum_{s=1}^{S} f(\mathbf{x}^{(s)})\right) = \frac{1}{S^2}\sum_{s=1}^{S}\text{Var}\left(f(\mathbf{x}^{(s)})\right)$$
$$= \frac{1}{S^2}\cdot S \cdot \text{Var}(f(\mathbf{x})) = \frac{\sigma^2}{S}$$

(So the **standard error** shrinks as $\sqrt{\sigma^2/S}$.)

Practical Challenges:

The convergence rate is independent of the dimensionality of x, which is great!

….but the variance constant σ^2 could be impractically large for high dimensional problems.

REPARAMETERIZATION TRICK

The Reparameterization Trick

- **Problem:** you want to backprop through a sample from a Gaussian, but you can't (at first glance) because it's stochastic
- **Solution:** reparametrize the sampling process

  Suppose the variational distribution is

$$q_\phi(z \mid x) = \mathcal{N}\big(z\,;\, \mu_\phi(x), \sigma^2_\phi(x)\big).$$

  Instead of sampling z directly, we reparameterize as

$$z = \mu_\phi(x) + \sigma_\phi(x)\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, 1).$$

  This expresses z as a differentiable function of parameters $\mu_\phi(x), \sigma_\phi(x)$ and independent noise ϵ, allowing gradients to flow through z during training.

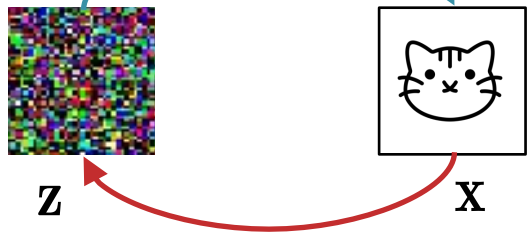
VARIATIONAL AUTOENCODERS

Variational Autoencoder

VAEs

Model:

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$



MLP

$q_\phi(\mathbf{z} \mid \mathbf{x}) \sim \mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$

$h = \tanh(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1)$

$\mu_\theta(\mathbf{z}) = \mathbf{W}_2 h + \mathbf{b}_2$

$\Sigma_\theta(\mathbf{z}) = (\mathbf{W}_3 h + \mathbf{b}_3)\mathbf{I}$

MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)})$ NOPE!

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$$

Learn: $\theta, \phi = \arg\min_{\theta,\phi} \mathrm{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z} \mid \mathbf{x}))$

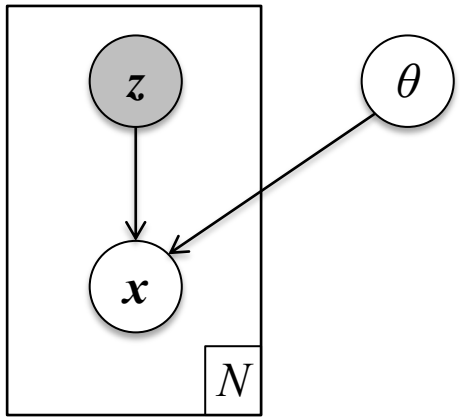
where $p_\theta(\mathbf{z} \mid \mathbf{x})$ is true posterior of $p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})$

Variational Autoencoder

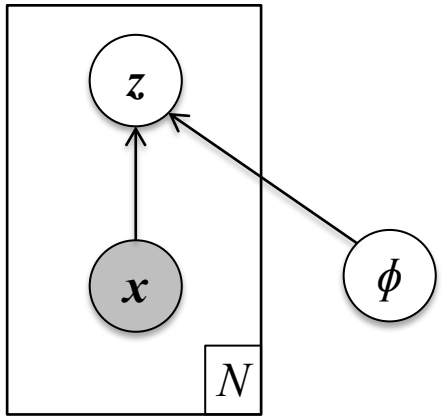
Graphical Model Perspective

- We can view a variational autoencoder (VAE) as consisting of two graphical models; each of which is defined by a neural network
- VAEs define two distributions:
  - **encoder:** $q_\phi(\mathbf{z} \mid \mathbf{x})$
  - **decoder:** $p_\theta(\mathbf{x} \mid \mathbf{z})$
- Parameters θ and ϕ are neural network parameters (i.e. θ are not the variational parameters)

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



$q_\phi(\mathbf{z} \mid \mathbf{x})$

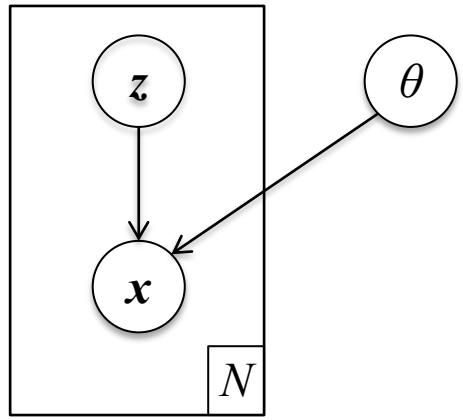


Variational Autoencoder

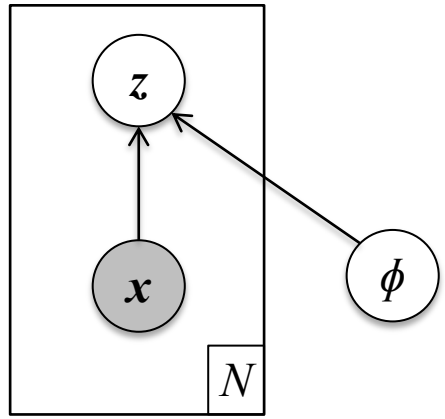
Graphical Model Perspective

- We can view a variational autoencoder (VAE) as consisting of two graphical models; each of which is defined by a neural network
- VAEs define two distributions:
  - **encoder**: $q_\phi(\mathbf{z} \mid \mathbf{x})$
  - **decoder**: $p_\theta(\mathbf{x} \mid \mathbf{z})$
- Parameters θ and ϕ are neural network parameters (i.e. θ are not the variational parameters)
- The decoder is actually a joint model over both **x** and **z**.
  - **joint decoder model**: $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})$

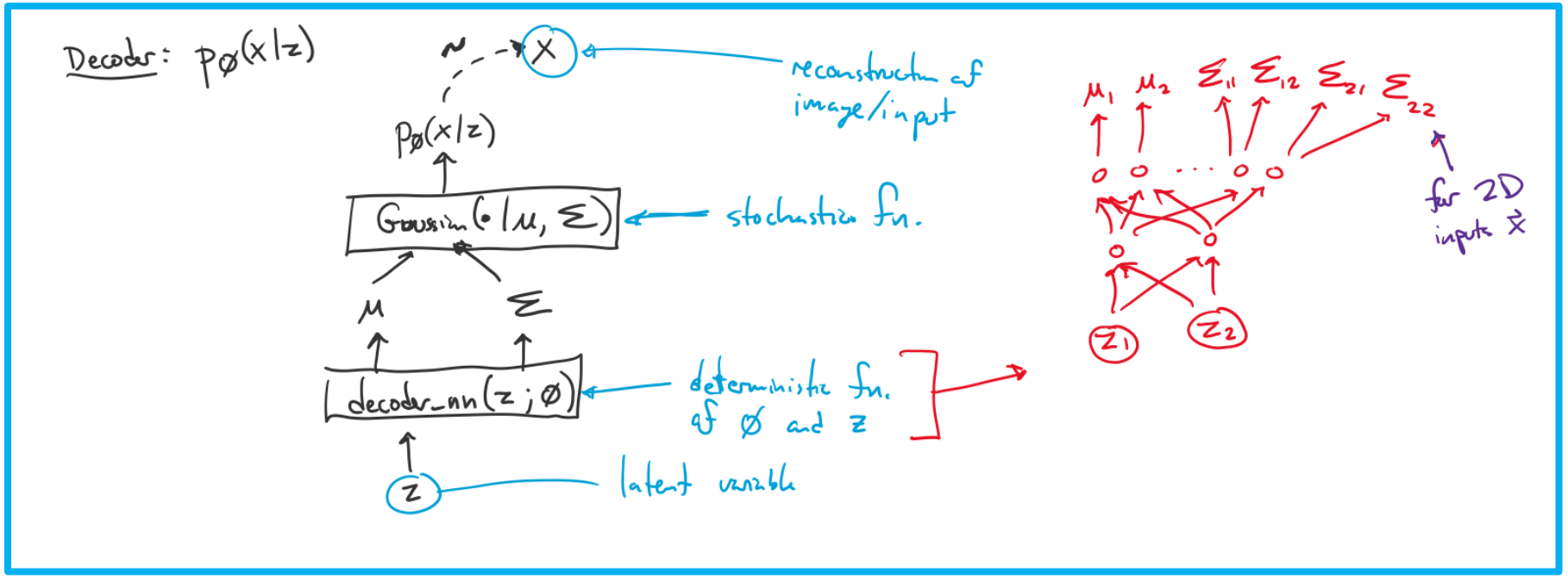
$p_\theta(\mathbf{x}, \mathbf{z})$

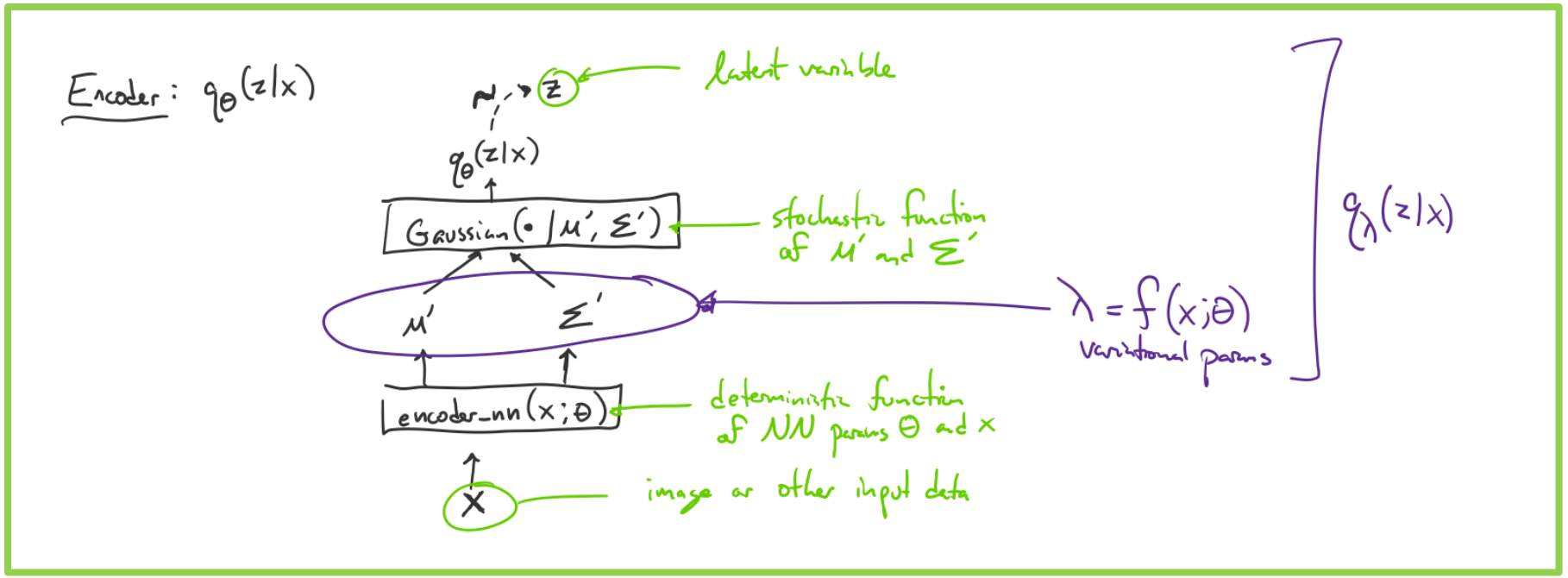


$q_\phi(\mathbf{z} \mid \mathbf{x})$



VAEs: Neural Network View





VAE Reparameterization Trick

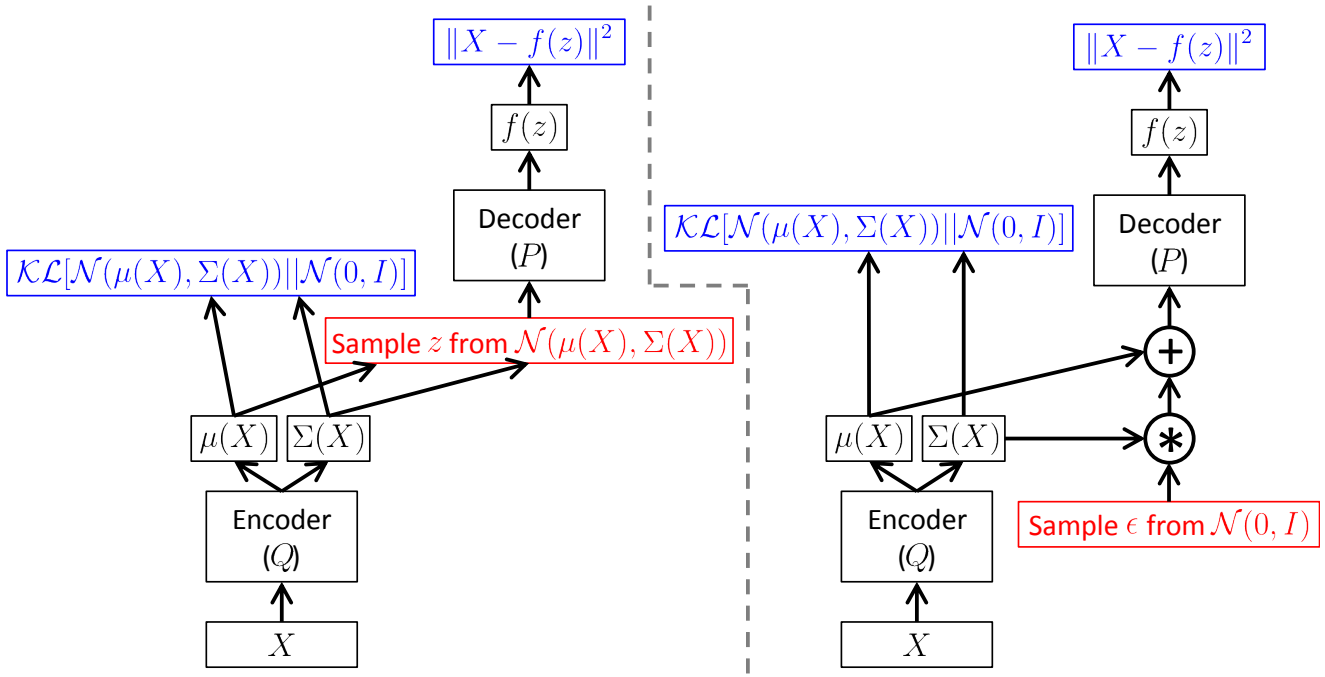


Figure 4: A training-time variational autoencoder implemented as a feed-forward neural network, where $P(X|z)$ is Gaussian. Left is without the "reparameterization trick", and right is with it. Red shows sampling operations that are non-differentiable. Blue shows loss layers. The feedforward behavior of these networks is identical, but backpropagation can be applied only to the right network.

Figure from Doersch (2016)

Background for Variational Autoencoders (VAEs)

To talk about VAEs we need to lay some groundwork:

1. Autoencoders (not the variational kind)
2. KL divergence
3. Variational inference (using KL as an objective)
4. Monte Carlo estimation (for approximating expectations)
5. (Score function trick)
6. Reparameterization trick (for variance reduction)
7. Stochastic Gradient Variational Bayes

Deriving the Variational Autoencoder

1. *Problem:* Cannot sample from an autoencoder.
   Solution: Define a decoder that we can sample from:
   $p_\theta(\mathbf{z}_T) \sim \mathcal{N}(0, \mathbf{I})$
   $p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$ where $\mu_\theta(\mathbf{z})$ and $\Sigma_\theta(\mathbf{z})$ are neural nets.

2. *Problem:* Intractable to maximize data likelihood with this decoder:
   $\hat{\theta} = \text{argmax}_\theta \log p_\theta(\mathbf{x}) = \text{argmax}_\theta \int_\mathbf{z} p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$
   Solution: Maximize a lower bound (ELBO) instead by introducing an encoder distribution:
   $q_\phi(\mathbf{z} \mid \mathbf{x}) \sim \mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$
   $\hat{\theta}, \hat{\phi} = \text{argmax}_{\theta,\phi}$ ELBO(q_ϕ, p_θ) where $\log p_\theta(\mathbf{x}) \geq$ ELBO(q_ϕ, p_θ)

Deriving the Variational Autoencoder

3. *Problem:* Intuitively, we should alternately maximize ϕ to match q_ϕ to the current p_θ as close as possible in KL (variational inference), and then maximize θ to push up the likelihood $\log p_\theta(\mathbf{x})$ as much as possible (learning); but this is slow.
   Solution: Instead, use SGA and maximize θ and ϕ simultaneously:
   $(\theta, \phi) \leftarrow (\theta, \phi) + \gamma \nabla_{\theta,\phi} \text{ELBO}(q_\phi, p_\theta)$

4. *Problem:* Cannot compute the first expectation in $\text{ELBO}(q_\phi, p_\theta)$:
   $\text{ELBO}(q_\phi, p_\theta) = E_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x} \mid \mathbf{z})\big] - \text{KL}\big(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z})\big)$
   Solution: Approximate with Monte Carlo estimation:
   $E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] \approx \frac{1}{S}\sum_{s=1}^{S} \log p_\theta(\mathbf{x} \mid \mathbf{z}^{(s)})$ where $\mathbf{z}^{(s)} \sim q_\phi(\mathbf{z} \mid \mathbf{x})$

Deriving the Variational Autoencoder

5. *Problem:* This approximation has very high variance, a known problem in SGVB (cf. score matching trick):
$\nabla_\phi E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] = E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z} \mid \mathbf{x})]$
Solution: Apply reparameterization trick, then Monte Carlo estimate:
Let $\boldsymbol{\epsilon}^{(s)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and set $\mathbf{z}^{(s)} = f(\boldsymbol{\epsilon}^{(s)}) = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}^{(s)}$.

$$E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] = E_{\boldsymbol{\epsilon} \sim \mathcal{N}}[\log p_\theta(\mathbf{x} \mid \mathbf{z} = f(\boldsymbol{\epsilon}))]$$
$$\approx \frac{1}{S} \sum_{s=1}^{S} \log p_\theta(\mathbf{x} \mid \mathbf{z}^{(s)})$$

Deriving the Variational Autoencoder

6. *Problem:* How do we implement this?
   Solution: It's easy, given a training image $\mathbf{x}^{(i)}$, we take $S = 1$ and do the following:

   Let $\boldsymbol{\epsilon}^{(1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

   Compute enc. MLP: $\boldsymbol{\mu}_\phi(\mathbf{x})$ and $\boldsymbol{\sigma}_\phi$

   Compute latent: $\mathbf{z}^{(1)} = f(\boldsymbol{\epsilon}^{(1)}) = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}^{(1)}$

   Loss: $\ell(\theta, \phi) = \log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}^{(1)}) - \mathrm{KL}\big(q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) \,\|\, p_\theta(\mathbf{z})\big)$

   Backprop: $\nabla_{\theta,\phi}\ell(\theta, \phi)$

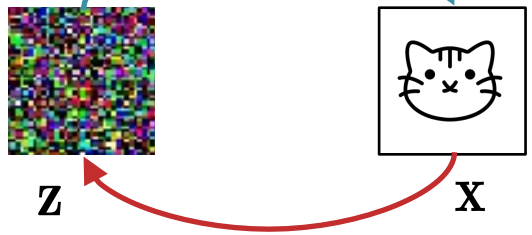
   Gradient step: $\theta, \phi \leftarrow \theta, \phi - \gamma\nabla_{\theta,\phi}\ell(\theta, \phi)$

Variational Autoencoder

VAEs

Model:

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$



MLP

$h = \tanh(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1)$

$\mu_\theta(\mathbf{z}) = \mathbf{W}_2 h + \mathbf{b}_2$

$\Sigma_\theta(\mathbf{z}) = (\mathbf{W}_3 h + \mathbf{b}_3)\mathbf{I}$

$q_\phi(\mathbf{z} \mid \mathbf{x}) \sim \mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$

MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)})$ NOPE!

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$$

Learn: $\theta, \phi = \arg\min_{\theta,\phi} \mathrm{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z} \mid \mathbf{x}))$

where $p_\theta(\mathbf{z} \mid \mathbf{x})$ is true posterior of $p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})$

Why VAEs?

- **Autoencoders:**
  - learn a low dimensional representation of the input, but hard to work with as a generative model
  - one of the key limitations of autoencoders is that we have no way of **sampling** from them!
- **Variational autoencoders (VAEs)**
  - by contrast learn a continuous latent space that is **easy to sample from**!
  - can **generate** new data (e.g. images) by sampling from the learned generative model

VAE RESULTS

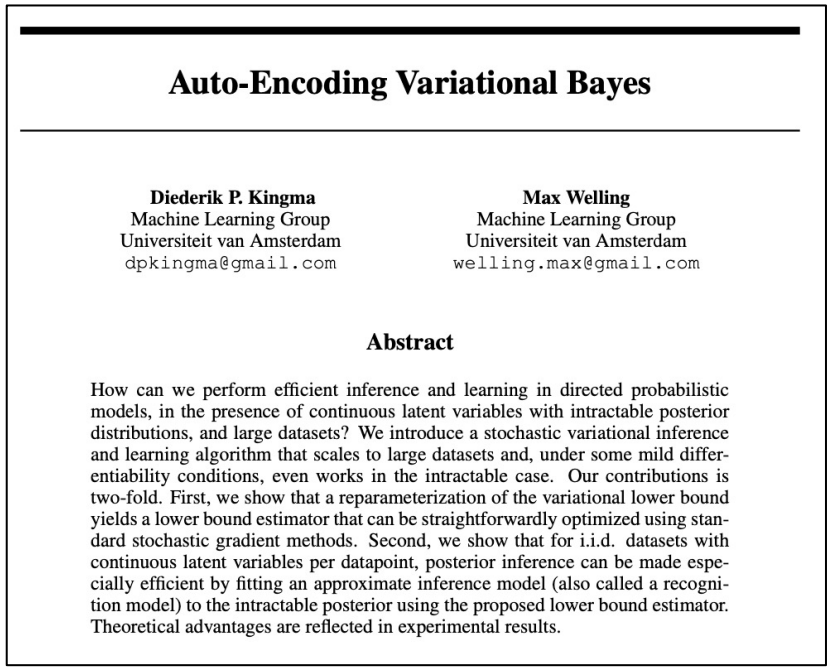
VAEs for Image Generation

Kingma & Welling (2014)

- introduced VAEs
- applied to image generation

Model

- $p_\phi(\mathbf{z}) \sim N(\mathbf{z}; 0, \mathbf{I})$
- $p_\phi(\mathbf{x} \mid \mathbf{z})$ is a multivariate Gaussian with mean and variance computed by an MLP, fully connected neural network with a single hidden layer with parameters ϕ
- $q_\theta(\mathbf{z} \mid \mathbf{x})$ is a multivariate Gaussian with diagonal covariance structure and with mean and variance computed by an MLP with parameters θ



Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

Figure from Kingma & Welling (2014)

VAEs for Image Generation

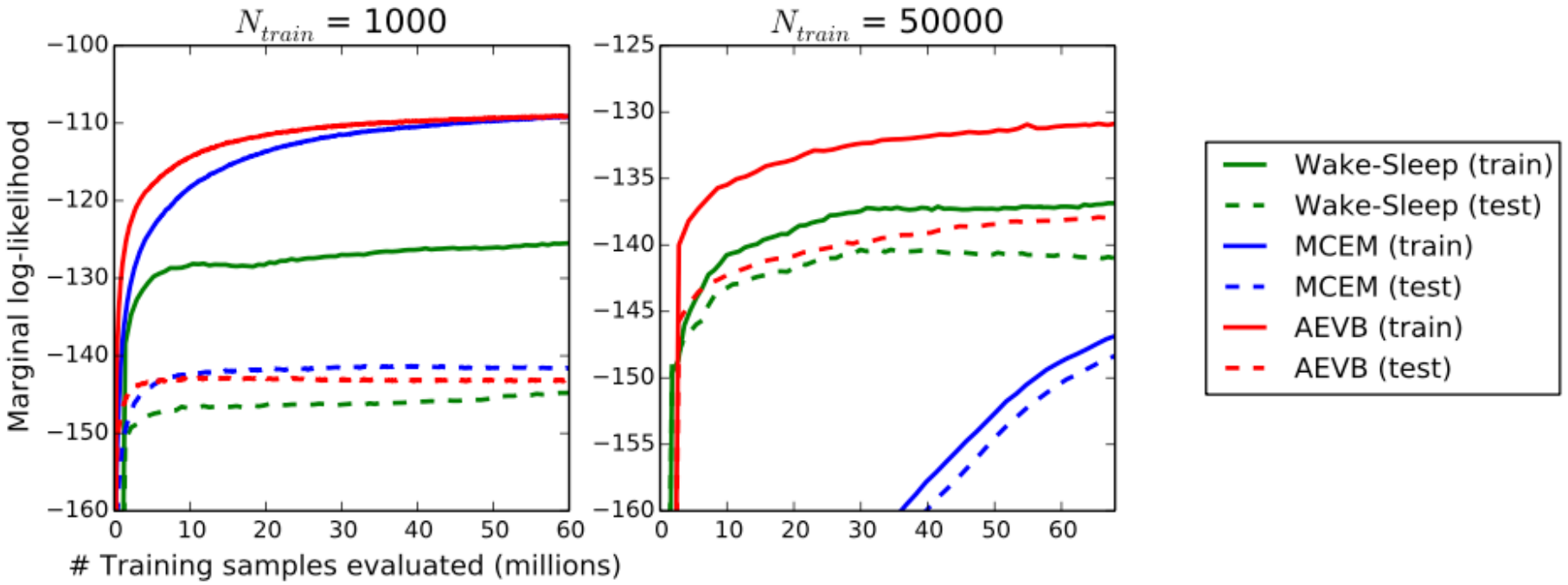
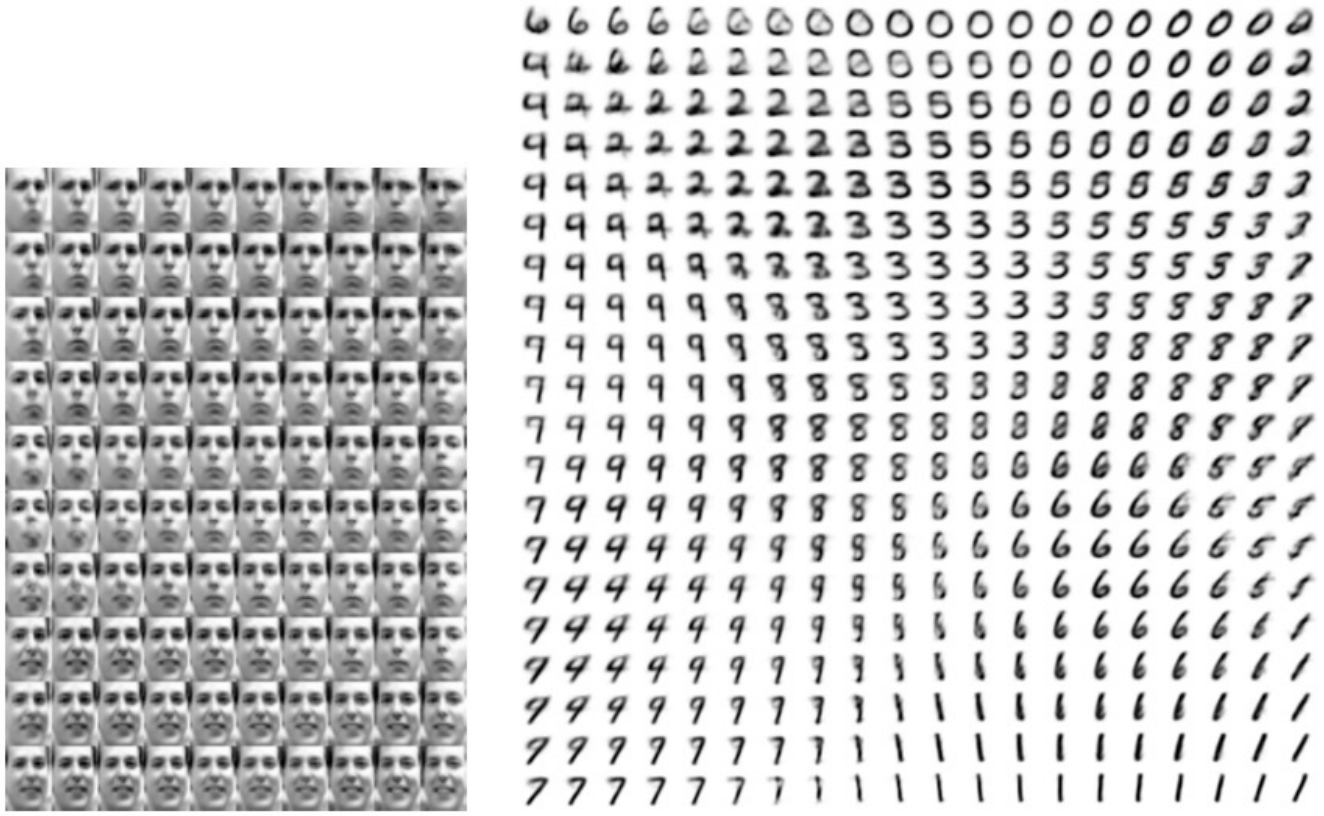


Figure 3: Comparison of AEVB to the wake-sleep algorithm and Monte Carlo EM, in terms of the estimated marginal likelihood, for a different number of training points. Monte Carlo EM is not an on-line algorithm, and (unlike AEVB and the wake-sleep method) can't be applied efficiently for the full MNIST dataset.

Figure from Kingma & Welling (2014)

VAEs for Image Generation

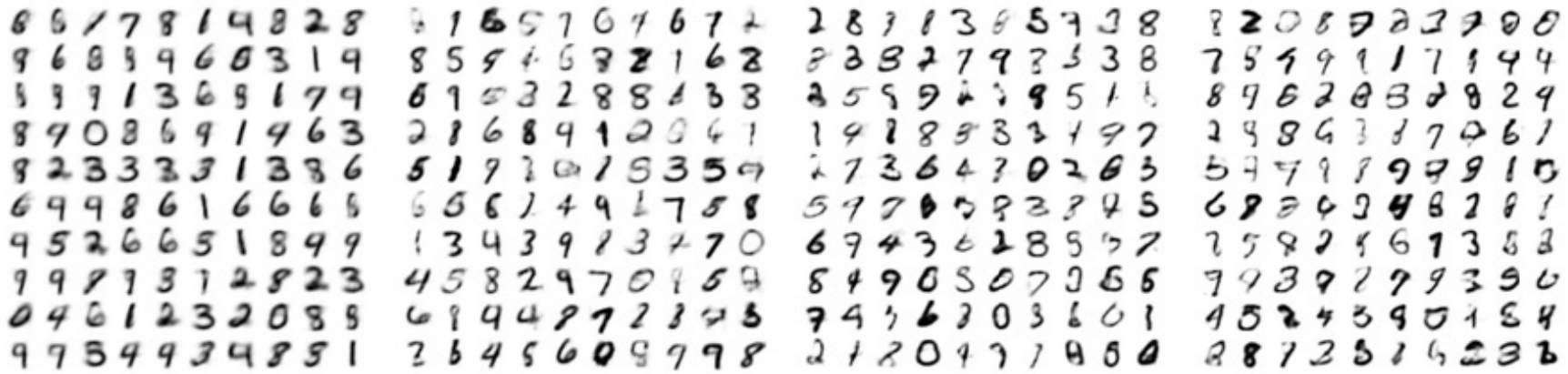


(a) Learned Frey Face manifold (b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Figure from Kingma & Welling (2014)

VAEs for Image Generation



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

Figure from Kingma & Welling (2014)

VAEs for Text Generation

Bowman et al. (2015)

- example of an application of VAEs to discrete data
- built on the sequence-to-sequence framework:
  - input is read in by an LSTM
  - output is generated by an LSTM-LM

Model

- $p_\phi(\mathbf{z}) \sim N(\mathbf{z}; 0, \mathbf{I})$
- $p_\phi(\mathbf{x} \mid \mathbf{z})$ is an LSTM Language Model with parameters ϕ
- $q_\theta(\mathbf{z} \mid \mathbf{x})$ is a multivariate Gaussian with mean and variance computed by an LSTM with parameters θ

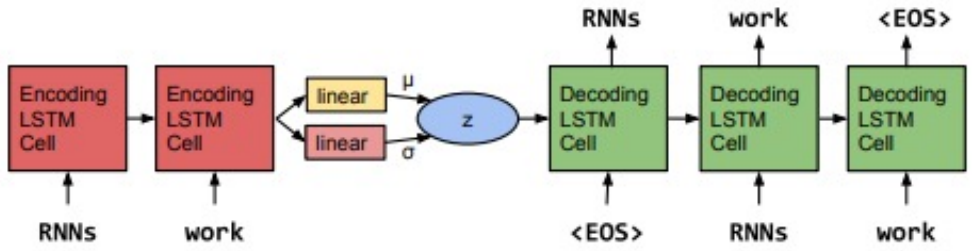


Figure 1: The core structure of our variational autoencoder language model. Words are represented using a learned dictionary of embedding vectors.

Figure from Bowman et al. (2015)

VAEs for Text Generation

INPUT	**we looked out at the setting sun .**	**i went to the kitchen .**	**how are you doing ?**
MEAN	*they were laughing at the same time .*	*i went to the kitchen .*	*what are you doing ?*
SAMP. 1	*ill see you in the early morning .*	*i went to my apartment .*	*“ are you sure ?*
SAMP. 2	*i looked up at the blue sky .*	*i looked around the room .*	*what are you doing ?*
SAMP. 3	*it was down on the dance floor .*	*i turned back to the table .*	*what are you doing ?*

Table 7: Three sentences which were used as inputs to the VAE, presented with greedy decodes from the mean of the posterior distribution, and from three samples from that distribution.

“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

Table 8: Paths between pairs of random points in VAE space: Note that intermediate sentences are grammatical, and that topic and syntactic structure are usually locally consistent.

Figure from Bowman et al. (2015)

VQ-VAE

- Vector Quantized VAE (VQ-VAE) learns a continuous codebook, but the encoder outputs discrete codes
- Decoder takes a code and generates a sample conditioned on it

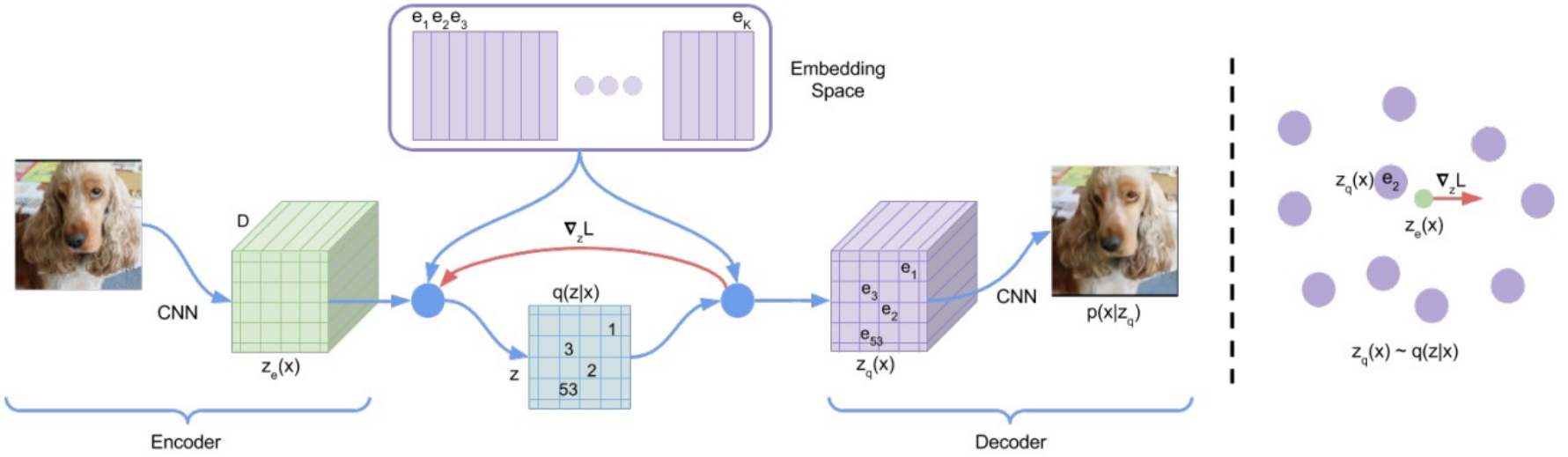


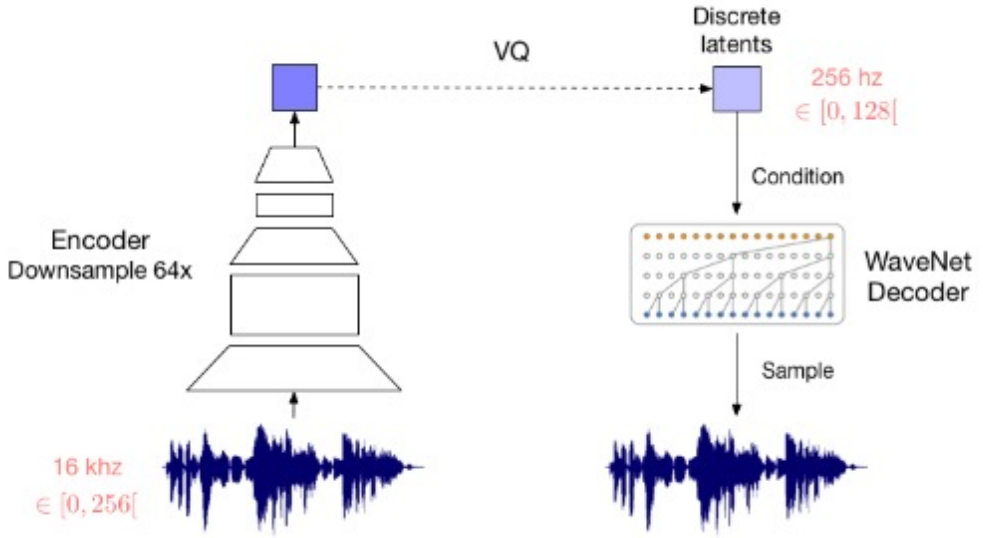
Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point e_2. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

Figure from van den Oord et al. (2018)

VQ-VAE

- Vector Quantized VAE (VQ-VAE) learns a continuous codebook, but the encoder outputs discrete codes
- Decoder takes a code and generates a sample conditioned on it

Example: Generating Audio

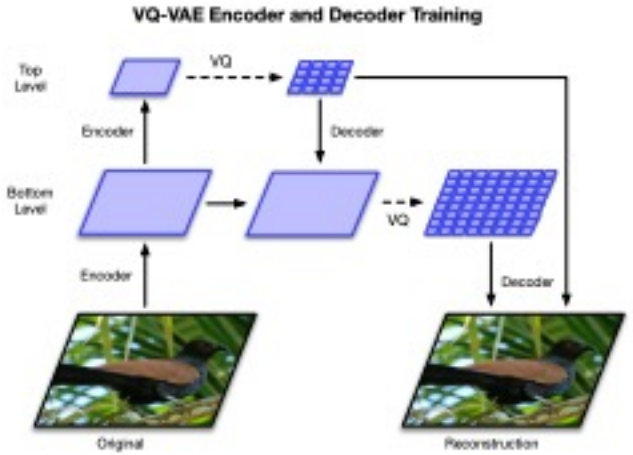


https://avdnoord.github.io/homepage/vqvae

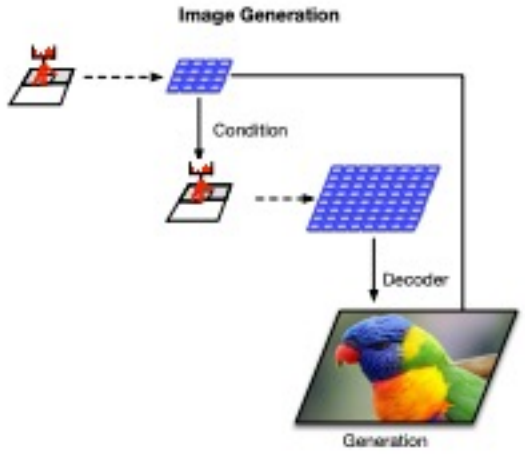
Figure from van den Oord et al. (2018)

VQ-VAE

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity



(a) Overview of the architecture of our hierarchical VQ-VAE. The encoders and decoders consist of deep neural networks. The input to the model is a 256×256 image that is compressed to quantized latent maps of size 64×64 and 32×32 for the *bottom* and *top* levels, respectively. The decoder reconstructs the image from the two latent maps.



(b) Multi-stage image generation. The top-level PixelCNN prior is conditioned on the class label, the bottom level PixelCNN is conditioned on the class label as well as the first level code. Thanks to the feed-forward decoder, the mapping between latents to pixels is fast. (The example image with a parrot is generated with this model).

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

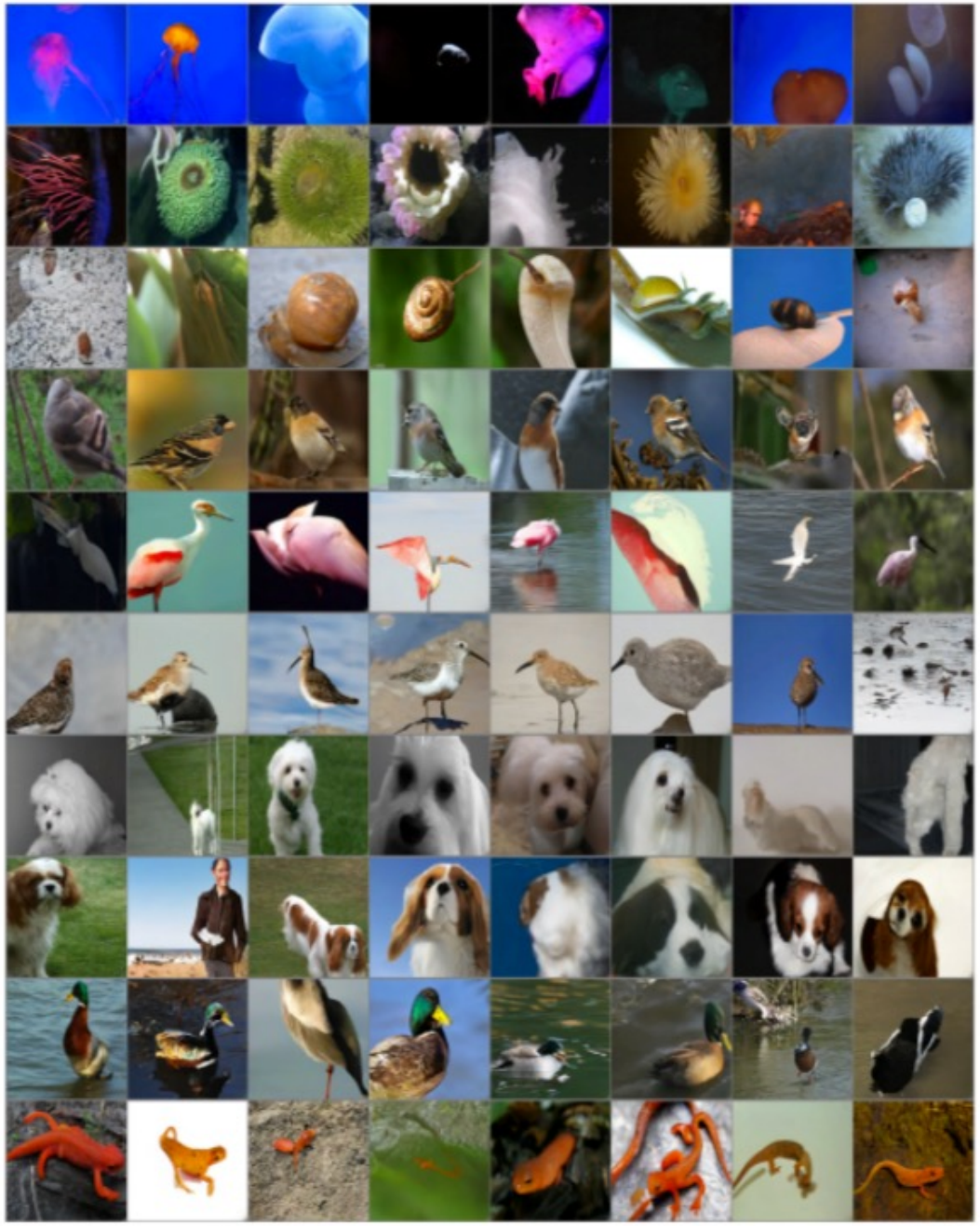


Figure 4: Class conditional random samples. Classes from the top row are: 108 sea anemone, 109 brain coral, 114 slug, 11 goldfinch, 130 flamingo, 141 redshank, 154 Pekinese, 157 papillon, 97 drake, and 28 spotted salamander.

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)

Connections between VAE and Diffusion

Variational Autoencoder

VAE The VAE learns via stochastic gradient variational Bayes (SGVB):

- true posterior: $p_\theta(\mathbf{z} \mid \mathbf{x})$ (intractable)
- variational approx: $q_\phi(\mathbf{z} \mid \mathbf{x})$ (encoder)
- model: $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{z}) p_{\phi)(\mathbf{z})}$
  where $p_\theta(\mathbf{x} \mid \mathbf{z})$ (decoder) and $p_\theta(\mathbf{z})$ (Gaussian)
- learning samples an example i:

$$\widetilde{\mathcal{L}}^B(\theta, \phi; \mathbf{x}^{(i)}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) \| p_\theta(\mathbf{z})) + \log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i)})$$
$$[\theta, \phi] \leftarrow [\theta, \phi] - \nabla_{[\theta,\phi]} \widetilde{\mathcal{L}}^B(\theta, \phi; \mathbf{x}^{(i)})$$

Denoising Diffusion Probabilistic Model

DDPM The DDPM learns in a similar fashion:

- true posterior: $p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \mathbf{x}_0)$ (intractable)
- variational approx: $q_\phi(\mathbf{x}_{0:T})$ (forward process)
- model: $p_\theta(\mathbf{x}_{0:T})$ (reverse process)
- learning only learns θ (i.e. no inference):

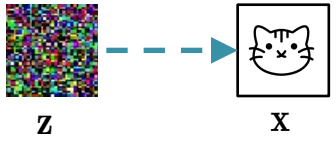
$$\ell_t(\theta) \leftarrow \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2$$
$$\theta \leftarrow \theta - \nabla_\theta \ell_t(\theta)$$

GAN → VAE → Diffusion (in 5 minutes)

GANs

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{x} \mid \mathbf{z}) =$ point mass on $G_\theta(\mathbf{z})$

Model:



MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^N \log p(\mathbf{x}^{(i)})$ NOPE!

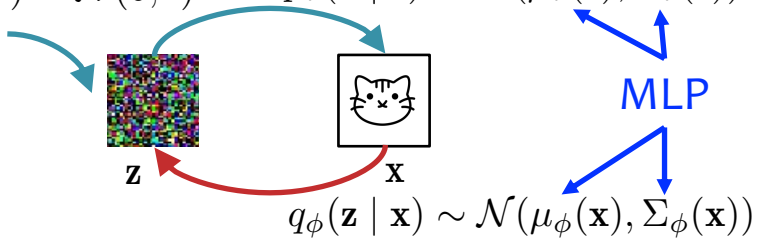
$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$

Learn: minimax game

VAEs

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z}))$

Model:



$h = \tanh(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1)$
$\mu_\theta(\mathbf{z}) = \mathbf{W}_2 h + \mathbf{b}_2$
$\Sigma_\theta(\mathbf{z}) = (\mathbf{W}_3 h + \mathbf{b}_3)\mathbf{I}$

MLE?: $\hat{\theta} = \arg\max_\theta \sum_{i=1}^N \log p(\mathbf{x}^{(i)})$ NOPE!

$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$

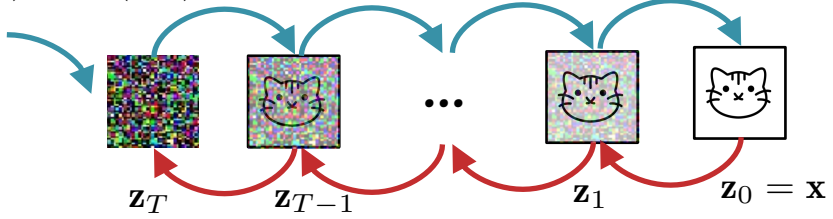
Learn: $\theta, \phi = \arg\min_{\theta,\phi} \mathrm{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z} \mid \mathbf{x}))$

where $p_\theta(\mathbf{z} \mid \mathbf{x})$ is true posterior of $p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})$

Diffusion

$p_\theta(\mathbf{z}) \sim \mathcal{N}(0, \mathbf{I})$ $p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t) \sim \mathcal{N}(\mu_\theta(\mathbf{z}_t), \Sigma_\theta(\mathbf{z}_t))$

Model:



$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \sim \mathcal{N}(\mu_\phi(\mathbf{z}_{t-1}), \Sigma_\phi(\mathbf{z}_{t-1}))$ $q(\mathbf{z}_0) =$ data dist.

MLE?: NOPE!

Learn: $\theta = \arg\min_\theta \mathrm{KL}(q(\mathbf{z}_{1:T} \mid \mathbf{x}) \,\|\, p_\theta(\mathbf{z}_{1:T} \mid \mathbf{x}))$

where $p_\theta(\mathbf{z}_{1:T} \mid \mathbf{x})$ is true posterior of $p(\mathbf{x} \mid \mathbf{z}_1) \cdots p(\mathbf{z}_{T-1} \mid \mathbf{z}_T) p(\mathbf{z}_T)$

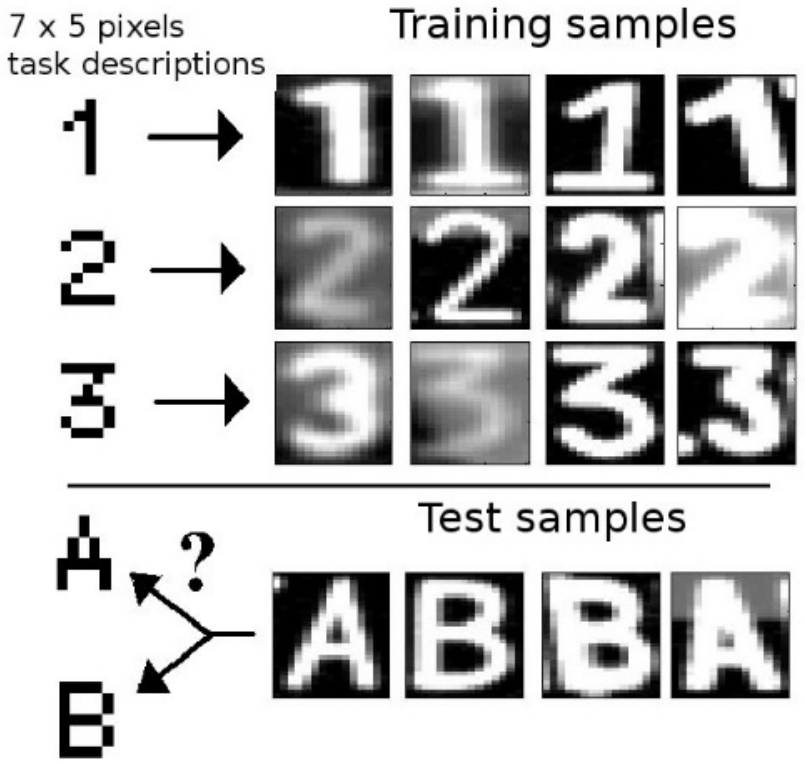
ZERO-SHOT AND FEW-SHOT LEARNING

Zero-shot Learning

- ***Definition***: in **zero-shot learning** we assume that training data does not contain any examples of the labels that appear in the test data

$z \rightarrow$	1	2	3	4	5
x_t	y_t^1	y_t^2	y_t^3	y_t^4	y_t^5
1	1	0	0	-	-
2	0	1	0	-	-
3	0	0	1	-	-
A	-	-	-	1	0
B	-	-	-	0	1

training data test data



Question:
How can we hope to learn in a setting where have no labeled examples?

Answer:

Figure from https://cdn.aaai.org/AAAI/2008/AAAI08-103.pdf

Few-shot Learning

- ***Definition***: in **few-shot learning** we assume that training data contains a handful (maybe two, three, or four) examples of each label

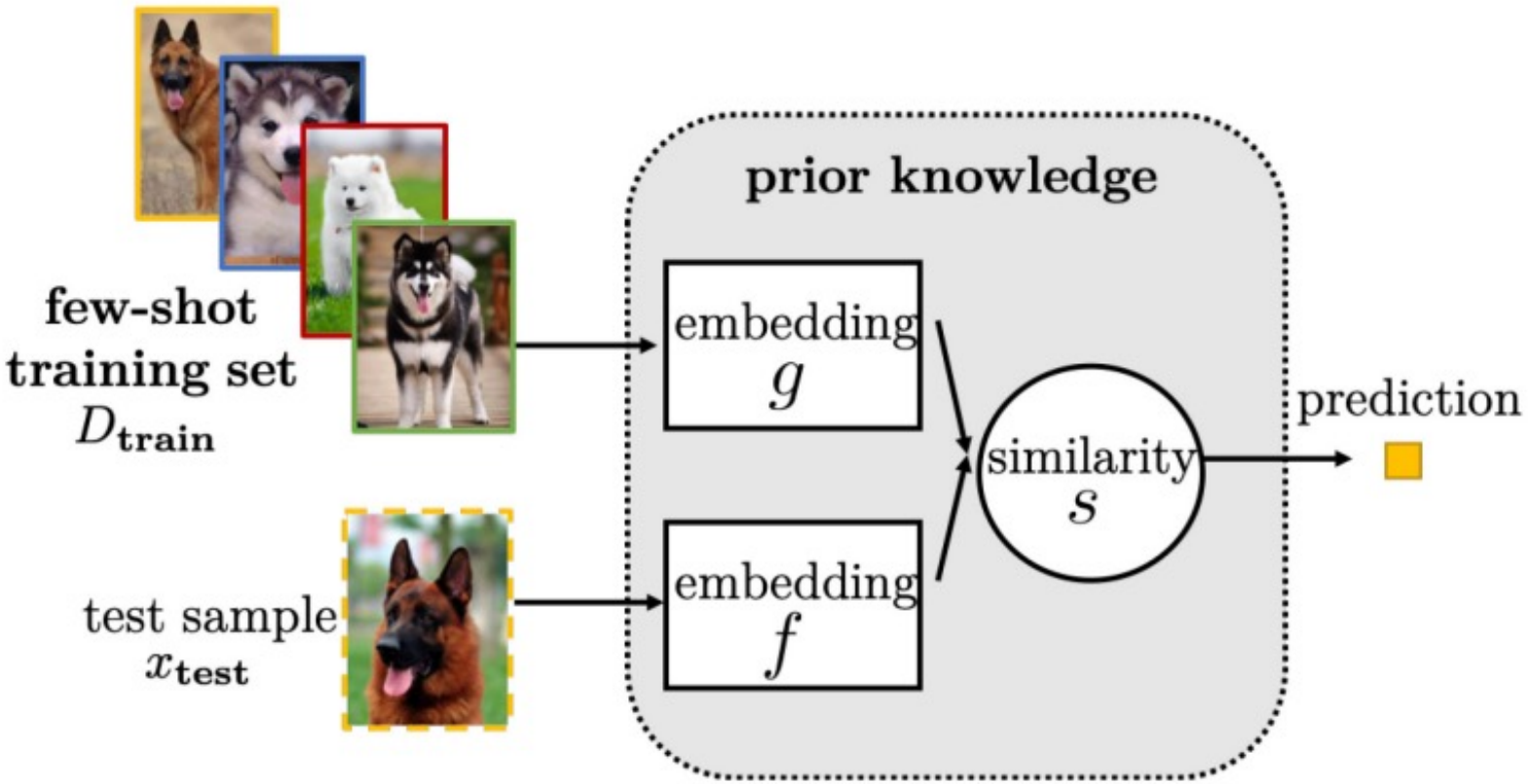


Figure from https://dl.acm.org/doi/10.1145/3386252

Few-shot Learning

- ***Definition***: in **few-shot learning** we assume that training data contains a handful (maybe two, three, or four) examples of each label

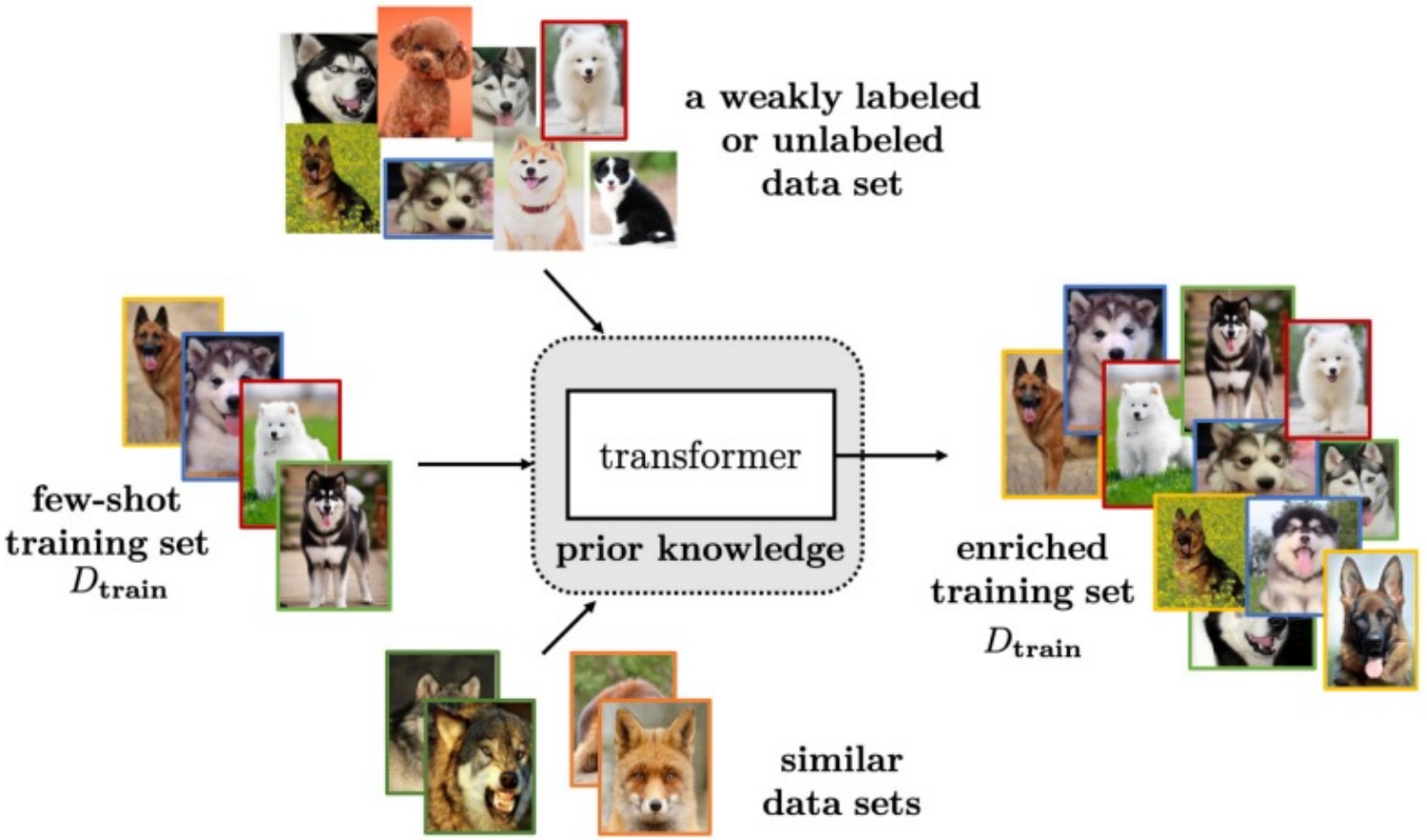


Figure from https://dl.acm.org/doi/10.1145/3386252

PROMPTING

Prompting

- Language models are trained to maximize the likelihood of sequences in the training data
- Most condition on the previous tokens to generate the next tokens

Definition: An *autoregressive language model* defines a probability distribution over sequences $\mathbf{x}_{1:T}$ of the form:

$$p(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

- Key idea behind prompting: provide a prefix string to the such that its likely completion is the answer you want

text completion

Prompting Examples

GPT-3

Prompt (i.e. context)

```
The City
BY C. P. CAVAFY
TRANSLATED BY EDMUND KEELEY
[Poem text omitted]


SOME TREES
John Ashbery
[Poem text omitted]


Shadows on the Way
Wallace Stevens
```

Output (i.e. generated poem)

```
I must have shadows on the way
If I am to walk I must have
Each step taken slowly and alone
To have it ready made

And I must think in lines of grey
To have dim thoughts to be my guide
Must look on blue and green
And never let my eye forget
That color is my friend
And purple must surround me too

The yellow of the sun is no more
Intrusive than the bluish snow
That falls on all of us.  I must have
Grey thoughts and blue thoughts walk with me
If I am to go away at all.
```

Conditional Sampling:

To draw a sample, feed in a prompt (i.e. context) and then sample from the model conditioned on that prompt

Examples from GPT-3 paper

What else can a large LM (attempt to) do?

Using the idea of **prompts**, we can apply LMs to a **variety of different problems** in natural language processing.

In the **zero-shot setting**, we simply feed the context to the model and observe how it completes the sequence. (i.e. there is no additional training)

Answer fact-based questions:

Context →	Organisms require energy in order to do what?
Correct Answer →	mature and develop.
Incorrect Answer →	rest soundly.
Incorrect Answer →	absorb light.
Incorrect Answer →	take in nutrients.

Complete sentences logically:

Context →	My body cast a shadow over the grass because
Correct Answer →	the sun was rising.
Incorrect Answer →	the grass was cut.

Complete analogies:

Context →	lull is to trust as
Correct Answer →	cajole is to compliance
Incorrect Answer →	balk is to fortitude
Incorrect Answer →	betray is to loyalty
Incorrect Answer →	hinder is to destination
Incorrect Answer →	soothe is to passion

Reading comprehension:

Context →	anli 1: anli 1: Fulton James MacGregor MSP is a Scottish politician who is a Scottish National Party (SNP) Member of Scottish Parliament for the constituency of Coatbridge and Chryston. MacGregor is currently Parliamentary Liaison Officer to Shona Robison, Cabinet Secretary for Health & Sport. He also serves on the Justice and Education & Skills committees in the Scottish Parliament. Question: Fulton James MacGregor is a Scottish politican who is a Liaison officer to Shona Robison who he swears is his best friend. True, False, or Neither?
Correct Answer →	Neither
Incorrect Answer →	True
Incorrect Answer →	False

Examples from GPT-3