

EE-608: Deep Learning for Natural Language Processing: Variational Auto-Encoders and NVIB

James Henderson



DLNLP, Lecture 7

Outline

Overview of Variational Auto-Encoders (VAE)

Variational Auto-Encoders with a Latent Vector Space
Extensions to VAEs

Variational Auto-Encoders for Transformers
Attention-based Representation are Mixture Distributions
A Variational Bayesian Framework for Attention Layers
Nonparametric Variational Auto-Encoder for Transformers

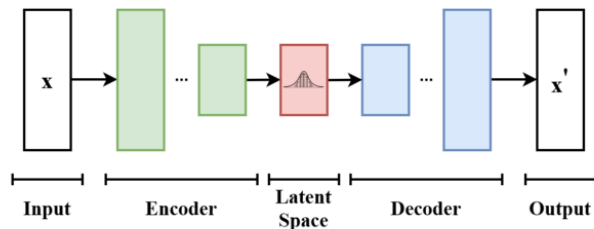
Outline

Overview of Variational Auto-Encoders (VAE)

Variational Auto-Encoders with a Latent Vector Space
Extensions to VAEs

Variational Auto-Encoders for Transformers
Attention-based Representation are Mixture Distributions
A Variational Bayesian Framework for Attention Layers
Nonparametric Variational Auto-Encoder for Transformers

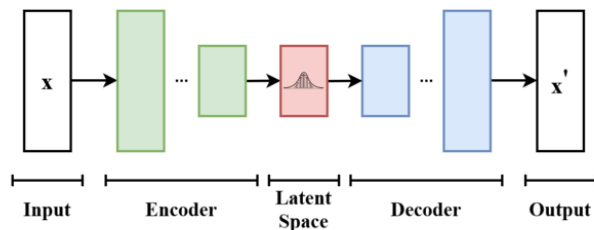
A Bayesian Approach to Auto-Encoders



- ▶ Auto-encoders learn to compress the observations into a smaller latent representation
- ▶ Thus learning a representation where underlying correlations are captured

But what does “smaller” mean? Can’t we encode an arbitrary amount of information in continuous values?

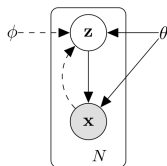
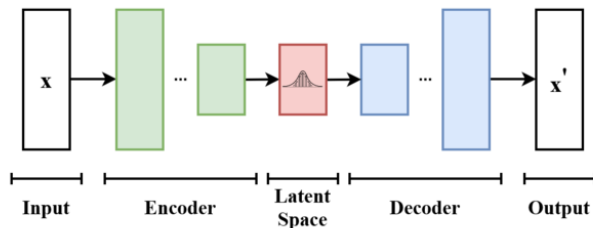
A Bayesian Approach to Auto-encoders



Adding noise to the latent representation controls the amount of information that can pass through it.

- ▶ **Information Bottleneck:** minimise the amount of information that passes through the latent representation
- ▶ **Reconstruction Loss:** ensure that enough information is passed to reconstruct the input

A Variational Bayesian Approach to Auto-Encoders

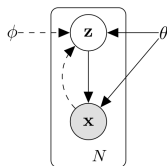
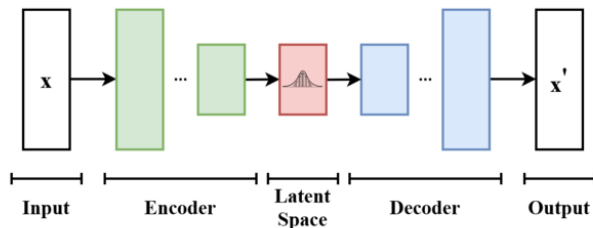


But Bayesian inference in deep models is intractable!

- **Variational Information Bottleneck:** train a deep variational approximation to the intractable parts of Bayesian inference

[[Auto-Encoding Variational Bayes](#), Kingma and Welling, 2014]

A Variational Bayesian Approach to Auto-Encoders



But how can we backprop through a stochastic model?

- **Reparameterisation Trick:** move the noise to an input so that everything we need to backprop through is deterministic

[[Auto-Encoding Variational Bayes](#), Kingma and Welling, 2014]

Taxonomy of Generative Models

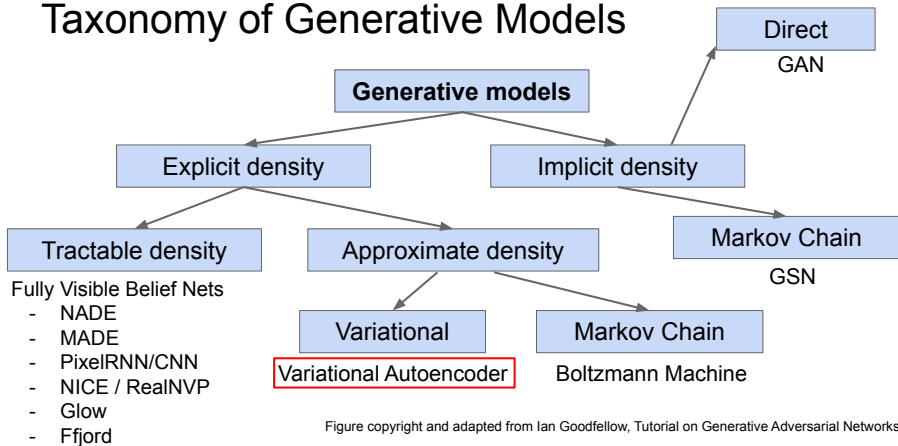


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

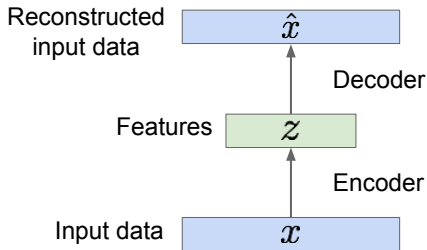
Outline

Overview of Variational Auto-Encoders (VAE)

Variational Auto-Encoders with a Latent Vector Space
Extensions to VAEs

Variational Auto-Encoders for Transformers
Attention-based Representation are Mixture Distributions
A Variational Bayesian Framework for Attention Layers
Nonparametric Variational Auto-Encoder for Transformers

Some background first: Autoencoders



Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Features capture factors of variation in training data.

But we can't generate new images from an autoencoder because we don't know the space of z .

How do we make autoencoder a **generative model**?

Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

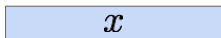
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\{x^{(i)}\}_{i=1}^N$ is generated from the distribution of unobserved (latent) representation $\mathbf{z}$

Sample from
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$



Sample from
true prior

$$z^{(i)} \sim p_{\theta^*}(z)$$

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

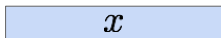
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\{x^{(i)}\}_{i=1}^N$ is generated from the distribution of unobserved (latent) representation $\mathbf{z}$

Sample from
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$



Sample from
true prior

$$z^{(i)} \sim p_{\theta^*}(z)$$

Intuition (remember from autoencoders!):
x is an image, **z** is latent factors used to
generate **x**: attributes, orientation, etc.

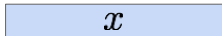
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

We want to estimate the true parameters θ^* of this generative model given training data $\mathbf{x}$.

Sample from
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$



Sample from
true prior

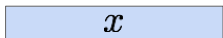
$$z^{(i)} \sim p_{\theta^*}(z)$$

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

We want to estimate the true parameters θ^* of this generative model given training data x .

Sample from
true conditional
 $p_{\theta^*}(x | z^{(i)})$



Sample from
true prior
 $z^{(i)} \sim p_{\theta^*}(z)$

How should we represent this model?

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

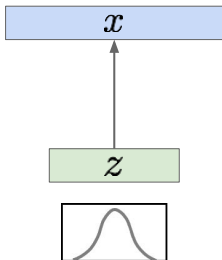
Variational Autoencoders

Sample from
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$

Sample from
true prior

$$z^{(i)} \sim p_{\theta^*}(z)$$



We want to estimate the true parameters θ^* of this generative model given training data x .

How should we represent this model?

Choose prior $p(z)$ to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

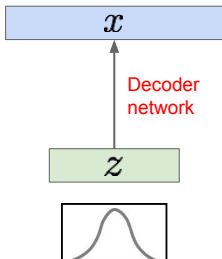


Sample from
true conditional

$$p_{\theta^*}(x | z^{(i)})$$

Sample from
true prior

$$z^{(i)} \sim p_{\theta^*}(z)$$



We want to estimate the true parameters θ^* of this generative model given training data x .

How should we represent this model?

Choose prior $p(z)$ to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Conditional $p(x|z)$ is complex (generates image) $\Rightarrow$ represent with neural network

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

We want to estimate the true parameters θ^* of this generative model given training data x .

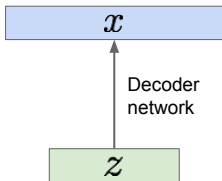
How to train the model?

Sample from
true conditional

$$p_{\theta^*}(x | z^{(i)})$$

Sample from
true prior

$$z^{(i)} \sim p_{\theta^*}(z)$$



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

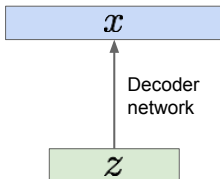
Variational Autoencoders

Sample from
true conditional

$$p_{\theta^*}(x | z^{(i)})$$

Sample from
true prior

$$z^{(i)} \sim p_{\theta^*}(z)$$



We want to estimate the true parameters θ^* of this generative model given training data x .

How to train the model?

Learn model parameters to maximize likelihood of training data

$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$$

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

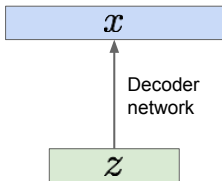
Variational Autoencoders

Sample from
true conditional

$$p_{\theta^*}(x | z^{(i)})$$

Sample from
true prior

$$z^{(i)} \sim p_{\theta^*}(z)$$



We want to estimate the true parameters θ^* of this generative model given training data x .

How to train the model?

Learn model parameters to maximize likelihood of training data

$$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$$

Q: What is the problem with this?

Intractable!

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Intractability

Data likelihood: $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Intractability

Data likelihood: $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$

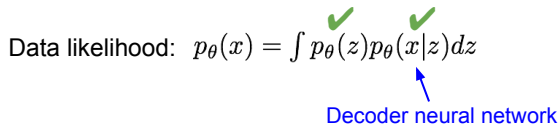
 Simple Gaussian prior

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Intractability

Data likelihood: $p_{\theta}(x) = \int p_{\theta}(z) p_{\theta}(x|z) dz$

Decoder neural network



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Intractability

Data likelihood: $p_{\theta}(x) = \int p_{\theta}(z) p_{\theta}(x|z) dz$



 Intractable to compute $p(x|z)$ for every z !

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Intractability

Data likelihood: $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$

 Intractable to compute $p(x|z)$ for every z !

$$\log p(x) \approx \log \frac{1}{k} \sum_{i=1}^k p(x|z^{(i)}), \text{ where } z^{(i)} \sim p(z)$$

Monte Carlo estimation is too high variance

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Intractability

Data likelihood: $p_{\theta}(x) = \int p_{\theta}(z) p_{\theta}(x|z) dz$

Posterior density: $p_{\theta}(z|x) = p_{\theta}(x|z)p_{\theta}(z)/p_{\theta}(x)$

Intractable data likelihood

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Intractability

Data likelihood: $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$

Posterior density also intractable: $p_{\theta}(z|x) = p_{\theta}(x|z)p_{\theta}(z)/p_{\theta}(x)$

Solution: In addition to modeling $p_{\theta}(x|z)$, learn $q_{\phi}(z|x)$ that approximates the true posterior $p_{\theta}(z|x)$.

Will see that the approximate posterior allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize.

Variational inference is to approximate the unknown posterior distribution from only the observed data x

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z$$

Variational Autoencoders

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} [\log p_{\theta}(x^{(i)})] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z)$$



Taking expectation wrt. z
(using encoder network) will
come in handy later

Variational Autoencoders

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\ &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule})\end{aligned}$$

Variational Autoencoders

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\ &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\ &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant})\end{aligned}$$

Variational Autoencoders

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z) q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)}) q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\&= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms})\end{aligned}$$

Variational Autoencoders

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z) q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)}) q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\&= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\&= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z)) + D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))\end{aligned}$$



The expectation wrt. z (using encoder network) let us write nice KL terms

Variational Autoencoders

$$\begin{aligned}
 \log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z) q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)}) q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\
 &= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\
 &= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z)) + D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))
 \end{aligned}$$

↑
Decoder network gives $p_{\theta}(x|z)$, can compute estimate of this term through sampling (need some trick to differentiate through sampling).

↑
This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

↑
 $p_{\theta}(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always ≥ 0 .

Variational Autoencoders

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z) q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)}) q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z)) + D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))$$

↑
We want to
maximize the
data
likelihood

↑
Decoder network gives $p_{\theta}(x|z)$, can
compute estimate of this term through
sampling.

↑
This KL term (between
Gaussians for encoder and z
prior) has nice closed-form
solution!

↑
 $p_{\theta}(z|x)$ intractable (saw
earlier), can't compute this KL
term :(But we know KL
divergence always ≥ 0 .

Variational Autoencoders

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z) q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)}) q_{\phi}(z | x^{(i)})} \right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad (\text{Logarithms})$$

$$= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}$$

Tractable lower bound which we can take
gradient of and optimize! ($p_{\theta}(x|z)$ differentiable,
KL term differentiable)

↑
We want to
maximize the
data
likelihood

Variational Autoencoders

$$\begin{aligned}
 \log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\
 &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z) q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)}) q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\
 &= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\
 &= \underbrace{\mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))}_{\geq 0}
 \end{aligned}$$

Decoder: reconstruct the input data

Encoder: make approximate posterior distribution close to prior

Tractable lower bound which we can take gradient of and optimize! ($p_{\theta}(x|z)$ differentiable, KL term differentiable)

Variational Autoencoders

Putting it all together: maximizing the
likelihood lower bound

$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right]}_{\mathcal{L}(x^{(i)}, \theta, \phi)} - \underbrace{D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\text{KL Divergence}}$$

Let's look at computing the KL divergence between the estimated posterior and the prior given some data

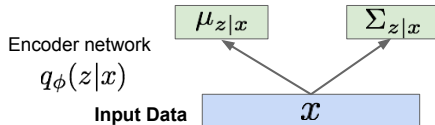
Input Data

$\mathcal{X}$

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right]}_{\mathcal{L}(x^{(i)}, \theta, \phi)} - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))$$



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right]}_{\mathcal{L}(x^{(i)}, \theta, \phi)} - \underbrace{D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\text{Make approximate posterior distribution close to prior}}$$

$$D_{KL}(\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) || \mathcal{N}(0, I))$$

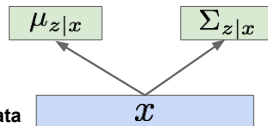
Have analytical solution

Make approximate posterior distribution close to prior

Encoder network

$$q_\phi(z|x)$$

Input Data



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

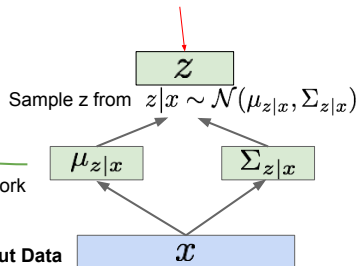
$$\underbrace{\mathbb{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior

Encoder network
 $q_\phi(z|x)$

Input Data

Not part of the computation graph!



Variational Autoencoders

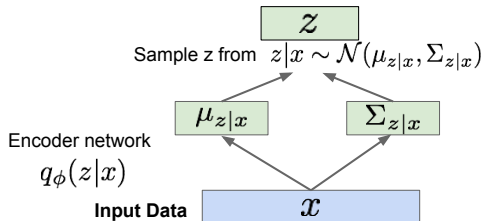
Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbb{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Reparameterization trick to make sampling differentiable:

Sample $\epsilon \sim \mathcal{N}(0, I)$

$$z = \mu_{z|x} + \epsilon \sigma_{z|x}$$



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbb{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

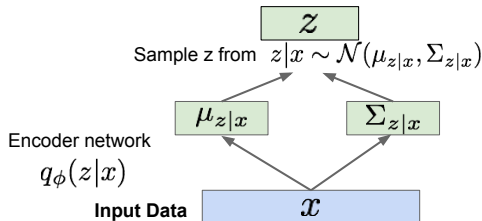
Reparameterization trick to make sampling differentiable:

Sample $\epsilon \sim \mathcal{N}(0, I)$

$$z = \mu_{z|x} + \epsilon \sigma_{z|x}$$

Input to the graph

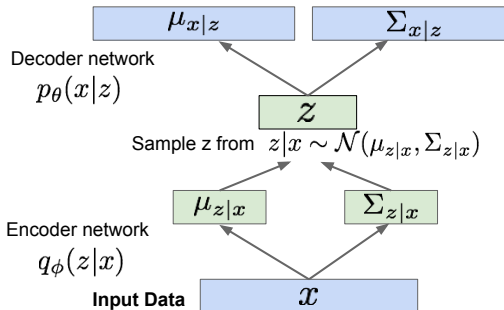
Part of computation graph



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbb{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

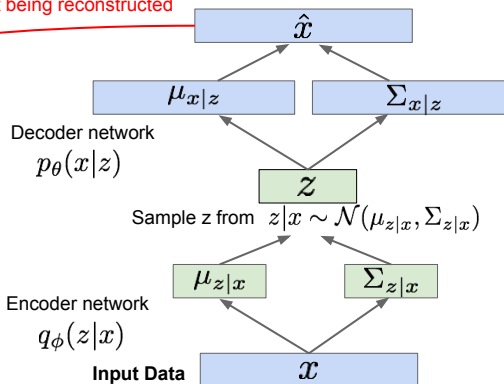


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

$$\underbrace{\mathbb{E}_z \left[\log p_\theta(x^{(i)} | z) \right]}_{\mathcal{L}(x^{(i)}, \theta, \phi)} \leftarrow D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))$$

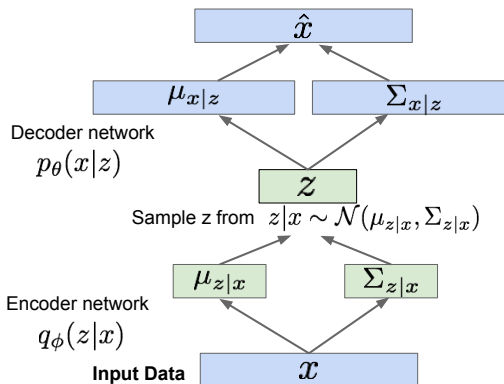


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

For every minibatch of input data: compute this forward pass, and then backprop!



Variational Autoencoders: Generating Data!

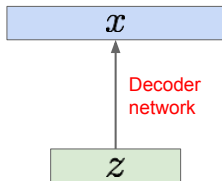
Our assumption about data generation process

Sample from true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$

Sample from true prior

$$z^{(i)} \sim p_{\theta^*}(z)$$



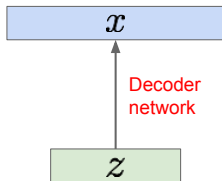
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Generating Data!

Our assumption about data generation process

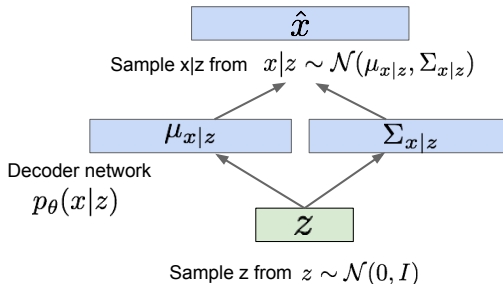
Sample from true conditional
 $p_{\theta^*}(x | z^{(i)})$

Sample from true prior
 $z^{(i)} \sim p_{\theta^*}(z)$



Now given a trained VAE:

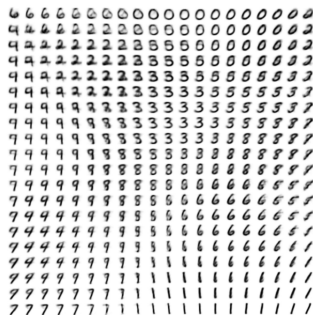
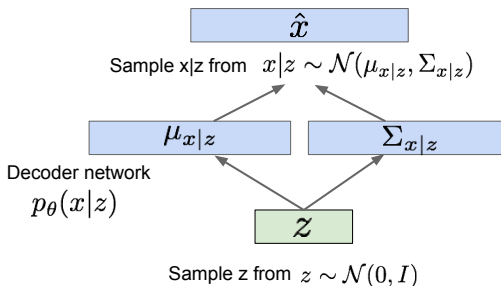
use decoder network & sample z from prior!



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Generating Data!

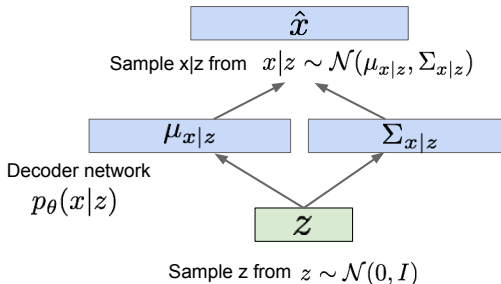
Use decoder network. Now sample z from prior!



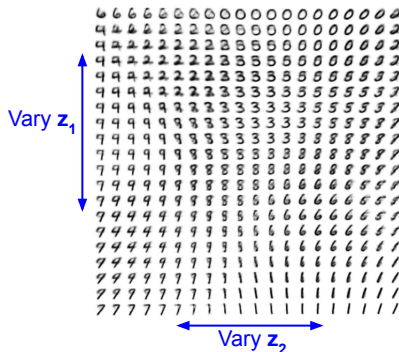
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!



Data manifold for 2-d z



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Generating Data!

Diagonal prior on $\mathbf{z}$
=> independent
latent variables

Different
dimensions of $\mathbf{z}$
encode
interpretable factors
of variation

Degree of smile

Vary z_1



Head pose

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Generating Data!

Diagonal prior on $\mathbf{z}$
=> independent
latent variables

Different
dimensions of $\mathbf{z}$
encode
interpretable factors
of variation

Degree of smile

Vary z_1

Also good feature representation that
can be computed using $q_\phi(\mathbf{z}|\mathbf{x})$!



Head pose

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

Probabilistic spin to traditional autoencoders => allows generating data

Defines an intractable density => derive and optimize a (variational) lower bound

Pros:

- Principled approach to generative models
- Interpretable latent space.
- Allows inference of $q(z|x)$, can be useful feature representation for other tasks

Cons:

- Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

Active areas of research:

- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian, e.g., Gaussian Mixture Models (GMMs), Categorical Distributions.
- Learning disentangled representations.

Backprop from Decoder to Encoder

The decoder and encoder are deep neural networks, so we know how to backprop through those.

- ▶ In between the encoder and decoder, we sample from the posterior distribution $q_{\phi}(z|x)$, so we need to backprop through this sampling step.
- ▶ In general, it is hard to get a good estimate of the derivative of a sampling step.
- ▶ But because $q_{\phi}(z|x)$ is a Gaussian, there is a simple solution.

Reparameterisation Trick: rewrite the sampling noise as an input ϵ , so that everything we need to backprop through is deterministic.

Outline

Overview of Variational Auto-Encoders (VAE)

Variational Auto-Encoders with a Latent Vector Space
Extensions to VAEs

Variational Auto-Encoders for Transformers

Attention-based Representation are Mixture Distributions

A Variational Bayesian Framework for Attention Layers

Nonparametric Variational Auto-Encoder for Transformers

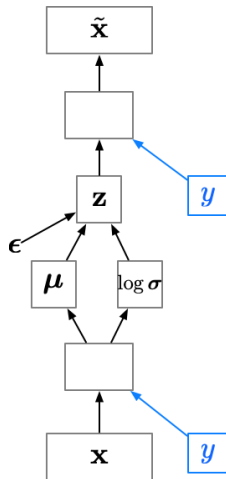
Conditional VAEs

What if we want to generate something which is appropriate for a given context?

- ▶ Condition the encoder and decoder on the context c ,
 $p_{\theta}(x|z, c) \ q_{\phi}(z|x, c)$
- ▶ Use a deep neural network to learn a representation of the context (if needed)
- ▶ Everything else stays the same

Class-Conditional VAE

- So far, we haven't used the labels y . A **class-conditional VAE** provides the labels to both the encoder and the decoder.
- Since the latent code z no longer has to model the image category, it can focus on modeling the stylistic features.
- If we're lucky, this lets us **disentangle** style and content. (Note: disentanglement is still a dark art.)
- See Kingma et al., "Semi-supervised learning with deep generative models."



Beta VAEs

- ▶ In practice, the two loss terms of a VEA are weighted by a hyperparameter β

$$L(\theta, \phi) = E_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x|z) - \beta D_{KL}(q_{\phi}(z|x) || p_{\theta}(z))$$

- ▶ Reducing β below 1 can help prevent “mode collapse”, where only the prior is learned
- ▶ Increasing β greater than 1 is used to force very compressed representations, which give more “disentangled” latent representations.

[[beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework](#). Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, Alexander Lerchner. 2017]

Vector-Space VAEs in NLP

Most work applying VAEs to language has focussed on generating text, particularly with conditional VAEs, or explainability.

- ▶ [Generating Sentences from a Continuous Space](#). Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, Samy Bengio. 2015
- ▶ [Neural Variational Inference for Text Processing](#). Yishu Miao, Lei Yu, Phil Blunsom. 2015
- ▶ [Educating Text Autoencoders: Latent Representation Guidance via Denoising](#). Tianxiao Shen, Jonas Mueller, Regina Barzilay, Tommi Jaakkola. 2019
- ▶ [A transformer-based variational autoencoder for sentence generation](#). Danyang Liu, Gongshen Liu. 2019
- ▶ [Variational Transformers for Diverse Response Generation](#). Zhaojiang Lin, Genta Indra Winata, Peng Xu, Zihan Liu, Pascale Fung. 2020
- ▶ [Disentangling generative factors in natural language with discrete variational autoencoders](#). Giangiacomo Mercatali and André Freitas. 2021.

Summary of Variational Auto-Encoders

- ▶ Auto-encoders are an unsupervised approach to representation learning
- ▶ Deep Variational Bayesian approaches bring information and probability theory to representation learning
- ▶ Variational Auto-Encoders (VAE) are a variational Bayesian approach to auto-encoders
- ▶ VAEs have been widely applied, and are the basis of many approaches to disentanglement

Also see:

[An Introduction to Variational Autoencoders](#),

Diederik P Kingma, 2019

Outline

Overview of Variational Auto-Encoders (VAE)

Variational Auto-Encoders with a Latent Vector Space
Extensions to VAEs

Variational Auto-Encoders for Transformers

Attention-based Representation are Mixture Distributions

A Variational Bayesian Framework for Attention Layers

Nonparametric Variational Auto-Encoder for Transformers

Variational Auto-Encoders with an Attention-Based Latent Space

A **vector space is not an adequate latent representation** for language; we need an attention-based latent space.

- ▶ Texts vary widely in length, so a fixed dimensionality is often either too big or too small
- ▶ Attention-based models have a variable number of vectors in their latent spaces, depending on the number of tokens in the text
- ▶ Empirically, attention-based representations are much much better than fixed-dimensional representation

How do we define **distributions over attention-based latent representations**?

Outline

Overview of Variational Auto-Encoders (VAE)

Variational Auto-Encoders with a Latent Vector Space

Extensions to VAEs

Variational Auto-Encoders for Transformers

Attention-based Representation are Mixture Distributions

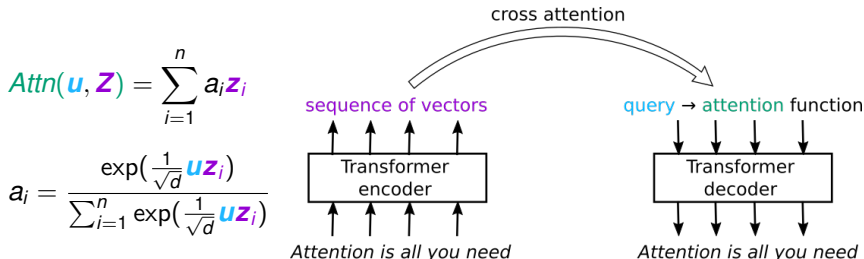
A Variational Bayesian Framework for Attention Layers

Nonparametric Variational Auto-Encoder for Transformers

Attention-based Transformer Embeddings

A Transformer encoder passes information to a Transformer decoder using **attention**.

- ▶ Encoder embeds the input in a **sequence of vectors**
- ▶ Decoder issues a **query** to access the embedding
- ▶ Attention returns an **average** of the **vectors** near the **query**

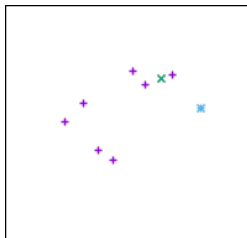


The Latent Space of Transformers

- ▶ Attention function is **permutation invariant** in the vectors

$$\text{Attn}(\mathbf{u}, \mathbf{Z}) = \sum_{i=1}^n a_i \mathbf{z}_i$$

$$a_i = \frac{\exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}{\sum_{i=1}^n \exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}$$

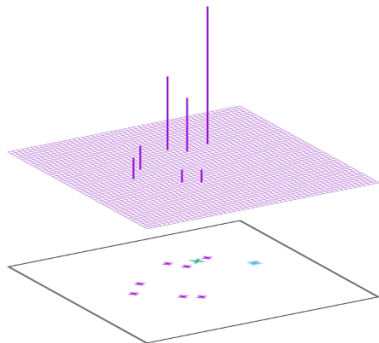
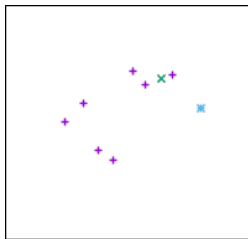


The Latent Space of Transformers

- ▶ Attention function is **permutation invariant** in the vectors
- ▶ Attention imposes a **normalised weighting** over vectors

$$\text{Attn}(\mathbf{u}, \mathbf{Z}) = \sum_{i=1}^n a_i \mathbf{z}_i$$

$$a_i = \frac{\exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}{\sum_{i=1}^n \exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}$$

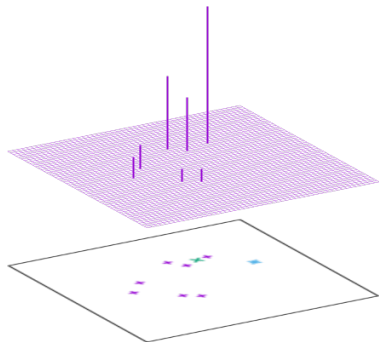
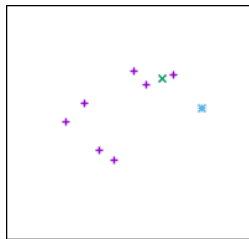


The Latent Space of Transformers

- ▶ Attention function is **permutation invariant** in the vectors
- ▶ Attention imposes a **normalised weighting** over vectors
- ▶ Attention supports a **variable number** of vectors

$$\text{Attn}(\mathbf{u}, \mathbf{Z}) = \sum_{i=1}^n a_i \mathbf{z}_i$$

$$a_i = \frac{\exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}{\sum_{i=1}^n \exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}$$



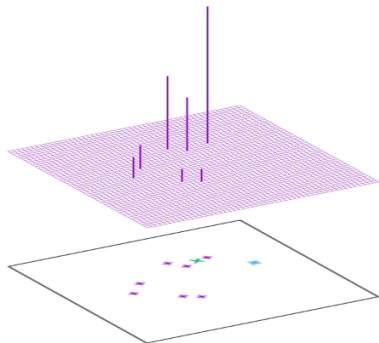
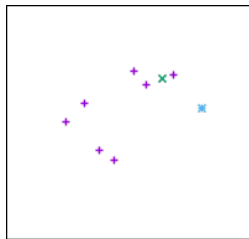
The Latent Space of Transformers

- ▶ Attention function is **permutation invariant** in the vectors
- ▶ Attention imposes a **normalised weighting** over vectors
- ▶ Attention supports a **variable number** of vectors

Like a **nonparametric** space of **mixture distributions**

$$\text{Attn}(\mathbf{u}, \mathbf{Z}) = \sum_{i=1}^n a_i \mathbf{z}_i$$

$$a_i = \frac{\exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}{\sum_{i=1}^n \exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}$$



The Latent Space of Transformers

Transformer embeddings are:

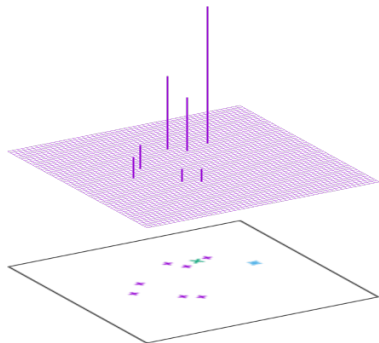
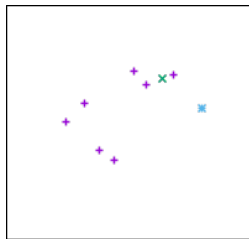
nonparametric mixtures of impulse distributions

Attention function is:

Bayesian query denoising

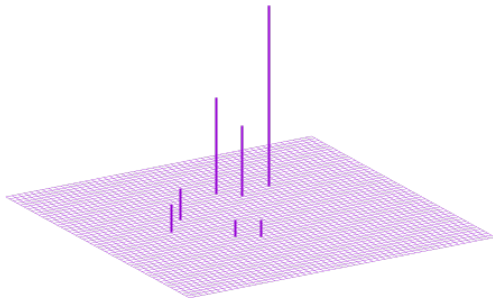
$$\text{Attn}(\mathbf{u}, \mathbf{Z}) = \sum_{i=1}^n a_i \mathbf{z}_i$$

$$a_i = \frac{\exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}{\sum_{i=1}^n \exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}$$

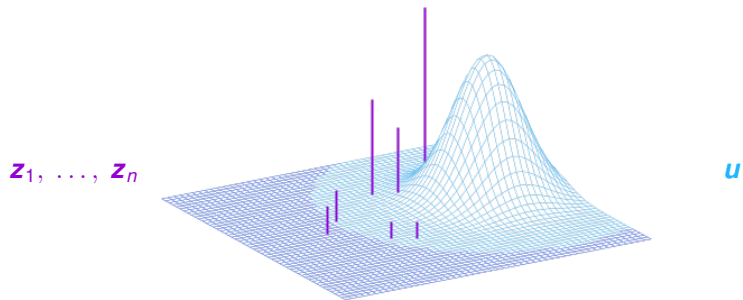


Query Denoising Attention Function

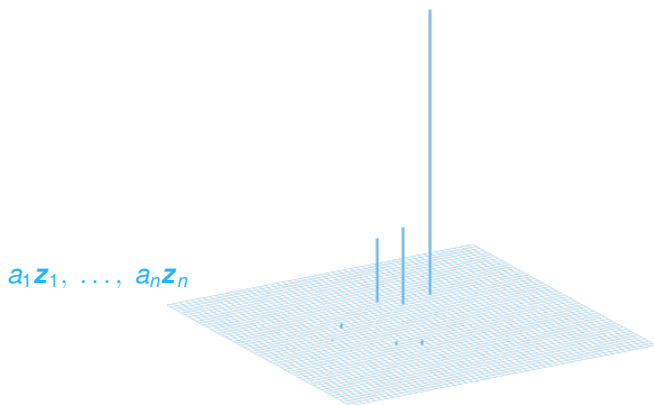
$\mathbf{z}_1, \dots, \mathbf{z}_n$



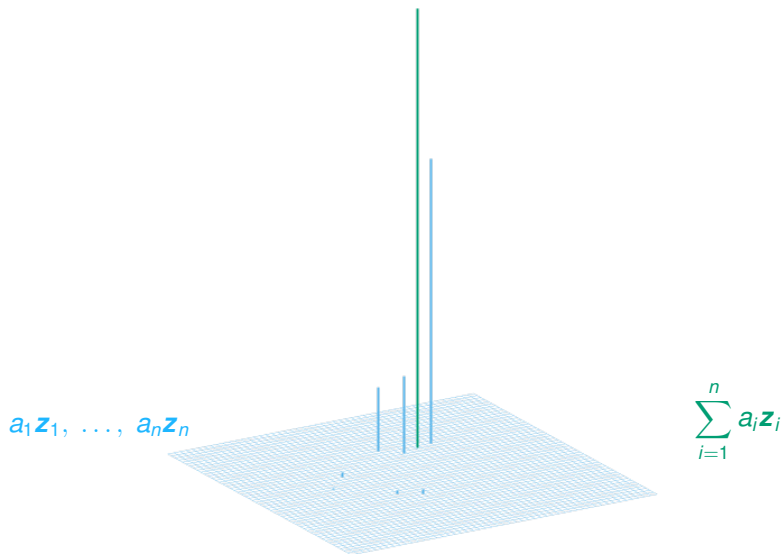
Query Denoising Attention Function



Query Denoising Attention Function



Query Denoising Attention Function



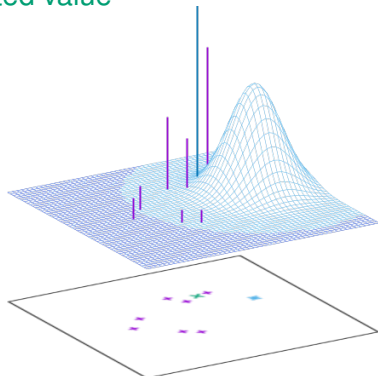
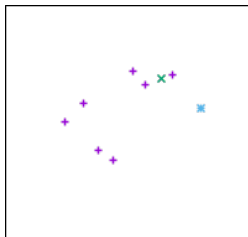
Denoising Attention

The **attention function** is **query denoising** using a mixture of impulses.

- ▶ Attention takes a **sequence of vectors** and a **query vector** and returns an **attention vector**
- ▶ Denoising takes a **prior distribution** and a **noisy observation** and returns its **expected value**

$$\text{Attn}(\mathbf{u}, \mathbf{Z}) = \sum_{i=1}^n a_i \mathbf{z}_i$$

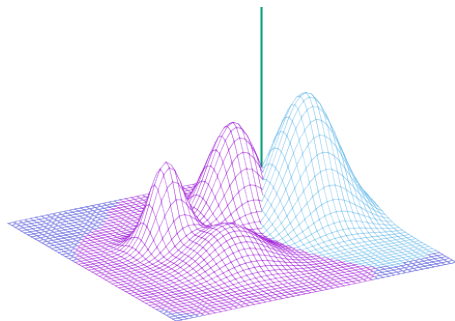
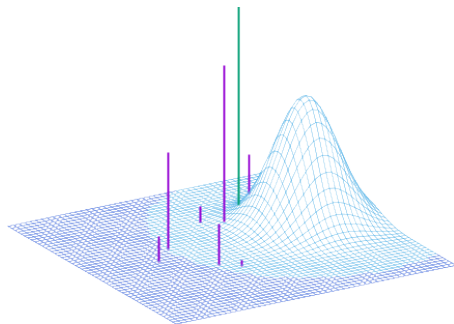
$$a_i = \frac{\exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}{\sum_{i=1}^n \exp(\frac{1}{\sqrt{d}} \mathbf{u} \mathbf{z}_i)}$$



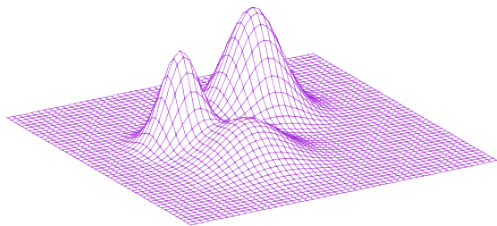
Denoising Attention

Query denoising is a **generalisation** of attention.

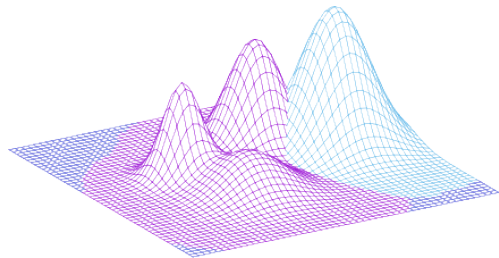
- ▶ The **prior distribution** can be **any mixture distribution**
- ▶ Gaussian mixtures make denoising easy to compute



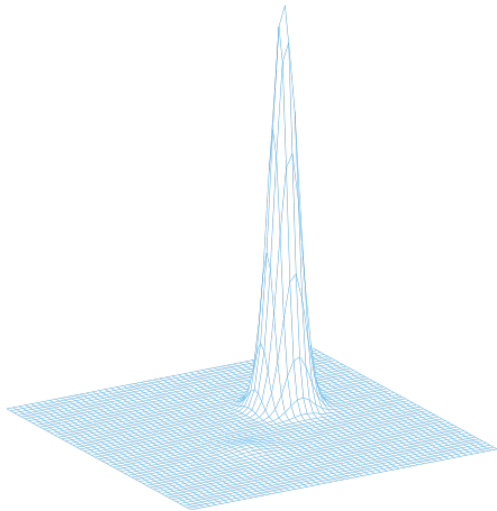
Denoising Attention



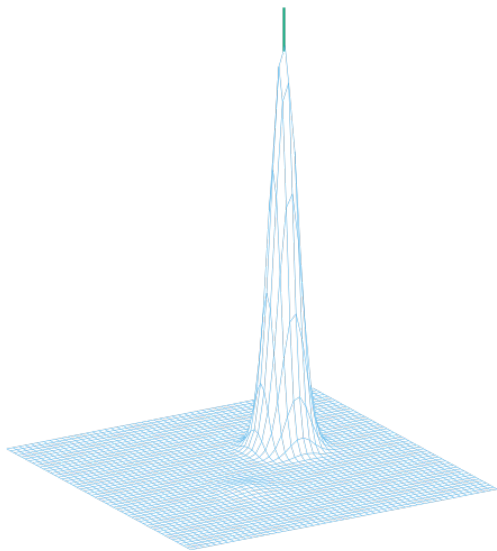
Denoising Attention



Denoising Attention



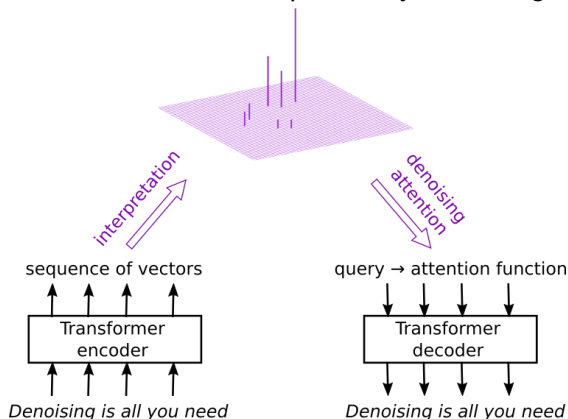
Denoising Attention



Attention-based Representation generalise to Mixture Distributions

Every attention-based representation has an **equivalent mixture distribution**.

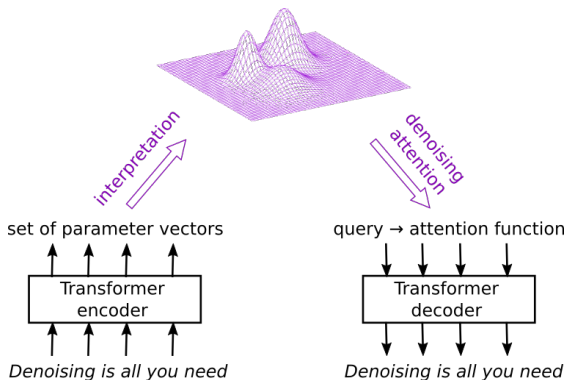
- ▶ Sets-of-vectors can be replaced by mixtures of impulse distributions
- ▶ Attention function can be replaced by denoising attention



Attention-based Representation generalise to Mixture Distributions

Mixture distribution are more general than set-of-vector representations.

- ▶ Sets-of-vectors can parameterise any mixture distribution
- ▶ Denoising attention function applies to **any mixture distribution**



Outline

Overview of Variational Auto-Encoders (VAE)

Variational Auto-Encoders with a Latent Vector Space

Extensions to VAEs

Variational Auto-Encoders for Transformers

Attention-based Representation are Mixture Distributions

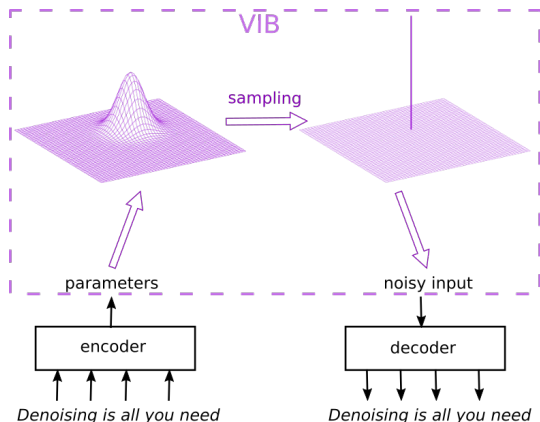
A Variational Bayesian Framework for Attention Layers

Nonparametric Variational Auto-Encoder for Transformers

Variational Information Bottleneck

Variational Auto-Encoders use a Variational Information Bottleneck (VIB) to **regularise** their latent representation.

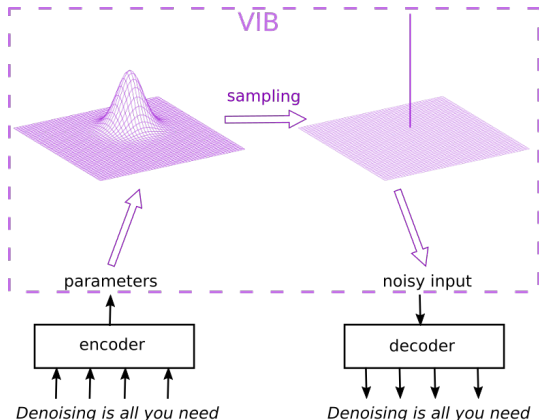
- ▶ Latent space is a vector space (previously)
- ▶ Encoder outputs a distribution over vectors



Variational Information Bottleneck

VIB only allows required information to pass from encoder to decoder.

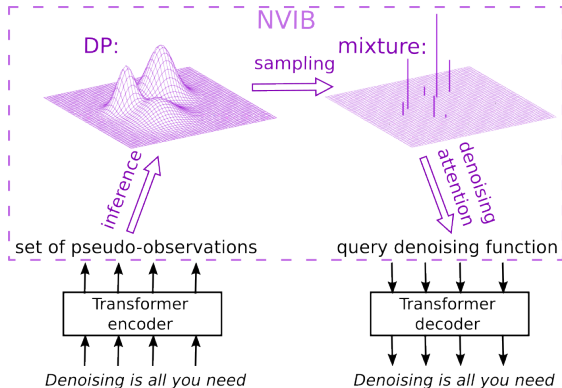
- ▶ Sampling noise controls the information that passes
- ▶ KL divergence with a prior regularises the noise level



Variational Information Bottleneck for Transformers

We can define a VIB for mixture distributions (NVIB) with **Bayesian nonparametrics**

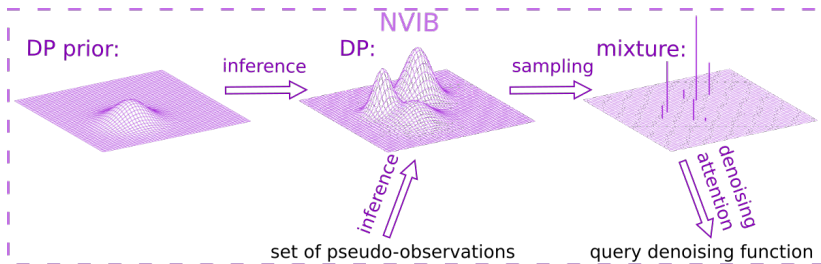
- ▶ Defines distributions over mixture distributions for the posterior and prior
- ▶ Defines sampling, means, and inference of the posterior



Nonparametric Variational Information Bottleneck

Bayesian nonparametrics provides an extensive theory on mixture distributions

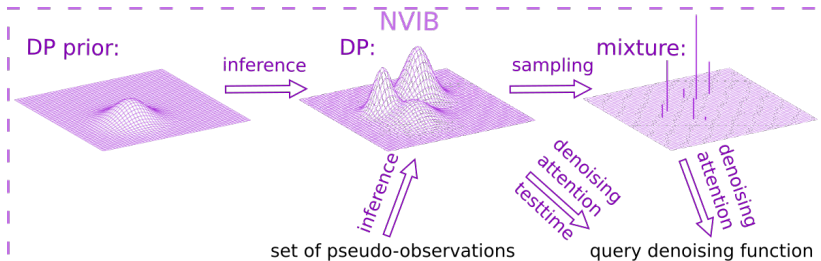
- **Dirichlet processes** specify distributions over mixtures of impulse distributions



Nonparametric Variational Information Bottleneck

Bayesian nonparametrics provides an extensive theory on mixture distributions

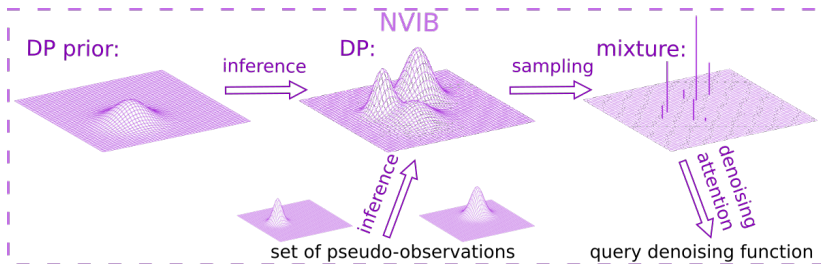
- ▶ Dirichlet processes specify distributions over mixtures of impulse distributions
- ▶ **Mean of the samples** is a continuous mixture distribution



Nonparametric Variational Information Bottleneck

Bayesian nonparametrics provides an extensive theory on mixture distributions

- ▶ Dirichlet processes specify distributions over mixtures of impulse distributions
- ▶ Mean of the samples is a continuous mixture distribution
- ▶ Dirichlet processes are **conjugate priors**



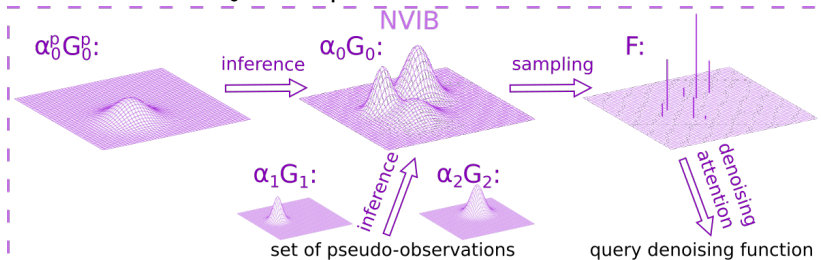
Inference of the Dirichlet Process Posterior

- ▶ One pseudo-observation per encoder token
- ▶ Each is a **Gaussian** G_i and its **pseudo-count** α_i
- ▶ Prior DP is a Gaussian G_0^p and its pseudo-count α_0^p
- ▶ Posterior DP is a mixture G_0 of the Gaussians and a total α_0 of the pseudo-counts

$$F \sim \text{DP}(G_0, \alpha_0)$$

$$\alpha_0 = \alpha_0^p + \sum_{i=1}^n \alpha_i$$

$$G_0 = \frac{\alpha_0^p}{\alpha_0} G_0^p + \sum_{i=1}^n \frac{\alpha_i}{\alpha_0} G_i$$

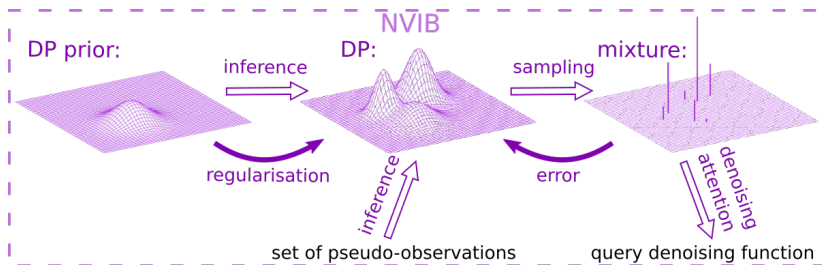


Nonparametric Variational Information Bottleneck

Our **Nonparametric Variational Information Bottleneck**

required proposing some effective approximations:

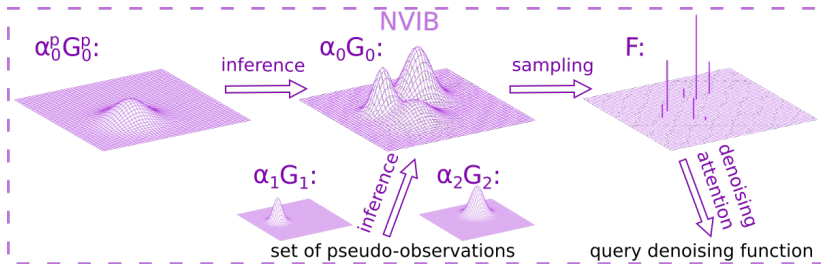
- ▶ the KL divergence between the posterior and prior DPs to regularise the posterior
- ▶ The reparameterisation trick for sampling from DPs with backpropagation through the sampling step



Sparseness of the Dirichlet Process Posterior

NVIB regularises the number of components.

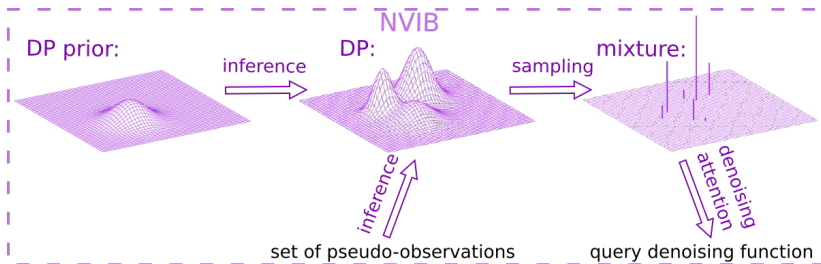
- ▶ Total pseudo-count α_0 determines the effective number of vectors in the sampled mixture, and thus its noise level
- ▶ KL divergence regularises the total pseudo-count to encourage more noise
- ▶ A **zero pseudo-count** removes that component



Nonparametric Variational Information Bottleneck

NVIB only allows required information to pass from encoder to decoder.

- ▶ Dirichlet processes define distributions over mixture distributions
- ▶ Sampled mixture is a noisy version of the base distribution
- ▶ Noise level is regularised by the effective number of vectors and their individual noise level



Outline

Overview of Variational Auto-Encoders (VAE)

Variational Auto-Encoders with a Latent Vector Space

Extensions to VAEs

Variational Auto-Encoders for Transformers

Attention-based Representation are Mixture Distributions

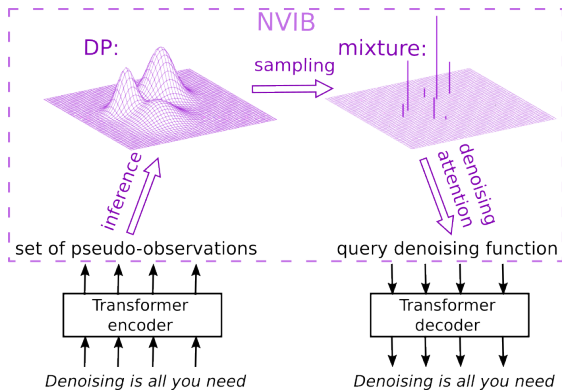
A Variational Bayesian Framework for Attention Layers

Nonparametric Variational Auto-Encoder for Transformers

Nonparametric Variational Auto-Encoder

We can define a VAE for Transformers (NVAE) using NVIB.

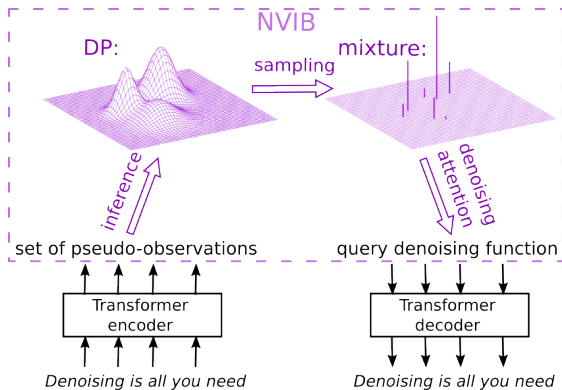
- ▶ **Transformer encoder** outputs pseudo-observations
- ▶ **NVIB** regularises the encoder-decoder interface
- ▶ **Transformer decoder** uses denoising attention



Nonparametric Variational Auto-Encoder

NVAE trained as an auto-encoder.

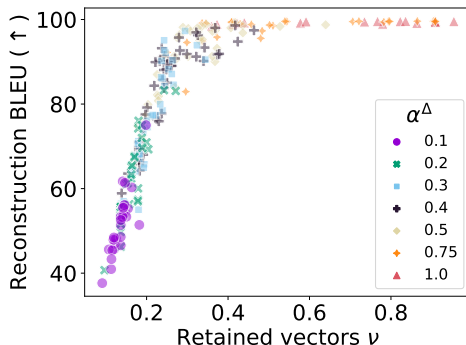
- ▶ Loss for cross-entropy of reconstructing the input sentence
- ▶ Loss for KL regulariser between the posterior and prior
- ▶ Hyperparameters to balance the **reconstruction** loss, and the two KL terms for the **Gaussians** and **pseudo-counts**



Inducing Representations of Text

We trained NVAE models on text data.

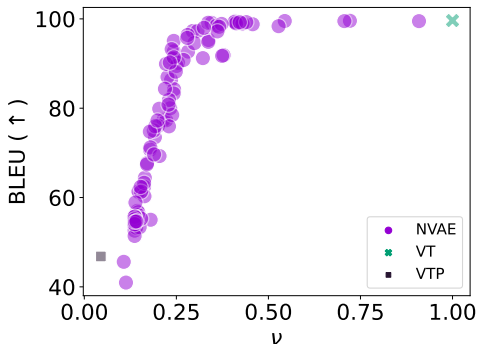
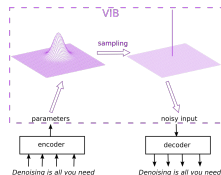
- ▶ Trained to reconstruct Wikipedia sentences
- ▶ Hyperparameter adjusts the sparsity rate
- ▶ Up to 2/3 of vectors can be dropped before reconstruction is degraded



Inducing Compressed Representations

Baselines don't drop the right number of vectors.

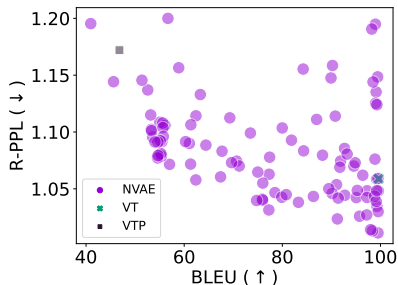
- ▶ **Variational Transformer Pooled** only keeps one vector
- ▶ **Variational Transformer** keeps all the vectors
- ▶ **NVAE** can adjust the number of vectors



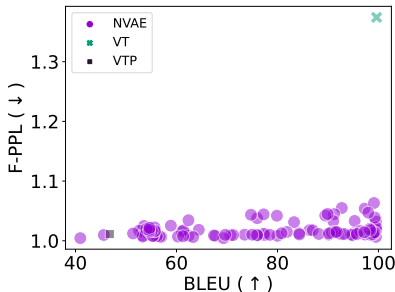
Inducing a Smooth Representation Space

Compressed representations generate text distributions better.

- ▶ Keeping all vectors
 - ▶ reconstructs text well
 - ▶ covers the space of texts well
 - ▶ many representations generate **garbage**
- ▶ NVIB compressed representations
 - ▶ reconstruct text well
 - ▶ cover the space of texts well
 - ▶ all representations generate good text



generation coverage vs reconstruction

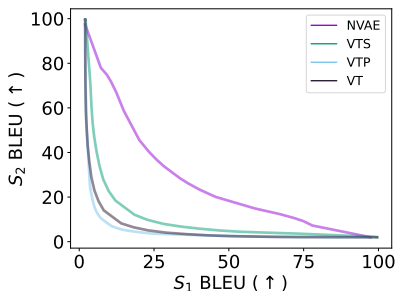


generation quality vs reconstruction

Inducing a Smooth Representation Space

Compressed representation space has smoother interpolation.

- ▶ Keeping all vectors does not interpolate well
 - ▶ decodes to the same sentence over a large region
 - ▶ generates unrelated text in between sentences
- ▶ NVIB induces a latent space with good interpolation
 - ▶ smoothly transitions between sentences
 - ▶ generates related sentences in between



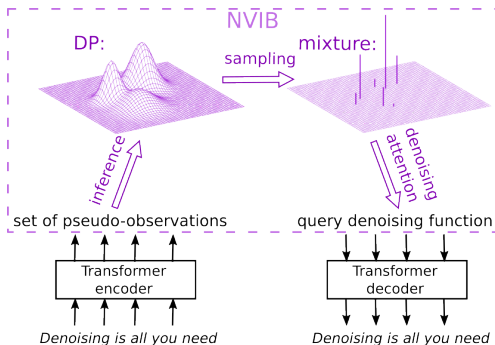
interpolation between sentences S₁ and S₂

Inducing Representations of Text

NVAE induces a better representation space.

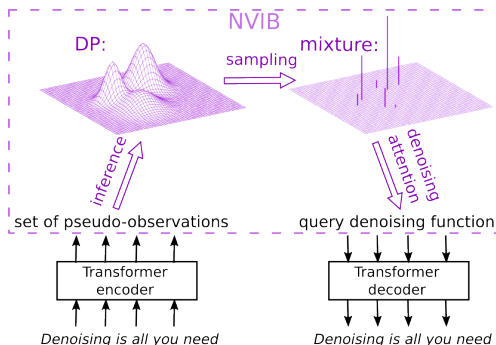
- ▶ Has an appropriate number of vectors
- ▶ Generates text distributions better
- ▶ Has smoother interpolation

A better approximation to nonparametric variational Bayesian models gives a better induced representation space.



Summary of NVIB

- ▶ NVIB combines the effectiveness of **Transformers** with the regulariser of **Variational AutoEncoders**.
- ▶ NVIB has a good **inductive bias** for natural language.

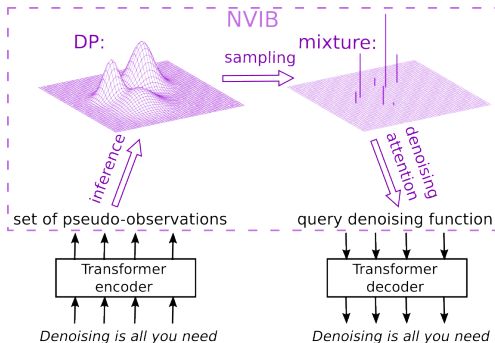


A VAE for Transformers with Nonparametric Variational Information Bottleneck. James Henderson and Fabio Fehr. ICLR 2023.

Why do Transformers work so well?

Transformers are **nonparametric variational Bayesian** models.

- ▶ Their latent representations are **nonparametric mixture distributions**
- ▶ Their encoders approximate **nonparametric variational Bayesian inference**
- ▶ Their decoders approximate **nonparametric Bayesian generative models**



Why do Transformers work so well?

Hypothesis: Transformers work well ***because*** human thoughts are Dirichlet processes.

- ▶ Transformers are models of Dirichlet processes.
- ▶ Transformers are astoundingly good models of language.
- ▶ Language is (approximately) isomorphic to thought.

