Imperial College London

Variational Autoencoders

Deep Learning Course

Kevin Webster, Pierre Harvey Richemond

Course outline

1. Introduction to deep learning
2. Neural networks optimisation
3. Convolutional neural networks
4. Introduction to reinforcement learning - part 1
5. Introduction to reinforcement learning - part 2
6. Sequence models
7. Generative adversarial networks (GANs)
8. Variational autoencoders (VAEs)
9. Normalising flows
10. Theories of deep learning

What are autoencoders ?

”Autoencoding” is a data compression algorithm where the compression and decompression functions are data-specific, lossy, and learned automatically from examples rather than engineered by a human. Additionally, in almost all contexts where the term ”autoencoder” is used, the compression and decompression functions are implemented with neural networks.

1) Autoencoders are data-specific, which means that they will only be able to compress data similar to what they have been trained on.

2) Autoencoders are lossy, which means that the decompressed outputs will be degraded compared to the original inputs.

3) Autoencoders are learned automatically from data examples, which is a useful property: it means that it is easy to train specialized instances of the algorithm that will perform well on a specific type of input.

What are autoencoders ?

In machine learning language :
Autoencoders attempt to reconstruct input data (that is, learn the identity function) in a *lossy* and *unsupervised* fashion.
In that sense, they are an unsupervised dimensionality reduction method, like for instance, principal component analysis (PCA).

What are autoencoders ? (by Francois Chollet)

To build an autoencoder, you need three things : an encoding function, a decoding function, and a distance function between the amount of information loss between the compressed representation of your data and the decompressed representation (i.e. a ”loss” function). The encoder and decoder will be chosen as parametric functions - typically neural networks
[Hinton and Salakhutdinov, 2006] - and to be differentiable with respect to the distance function, so the parameters of the encoding/decoding functions can be optimize to minimize the reconstruction loss, using Stochastic Gradient Descent.

Two views on variational autoencoders

There are two ways to introduce the variational autoencoder:

- The deep neural network view : the generic autoencoder architecture, plus regularization by a specific Kullback-Leibler divergence term.
- The (variational) Bayesian inference [Blei et al., 2016] view : the loss function is a lower bound on the model evidence.

These are mathematically equivalent, and both shed light to different phenomena. This is because deep learning provides robust optimization in the big data regime, whereas variational Bayes methods recast statistical inference problems, precisely, as optimization. Both views are presented in the two contemporary, foundational papers [Kingma and Welling, 2013] and [Jimenez Rezende et al., 2014], and synthesized in Durk Kingma's reference Ph.D. thesis [Kingma, 2017].

Deterministic Encoder-Decoder - Illustrated

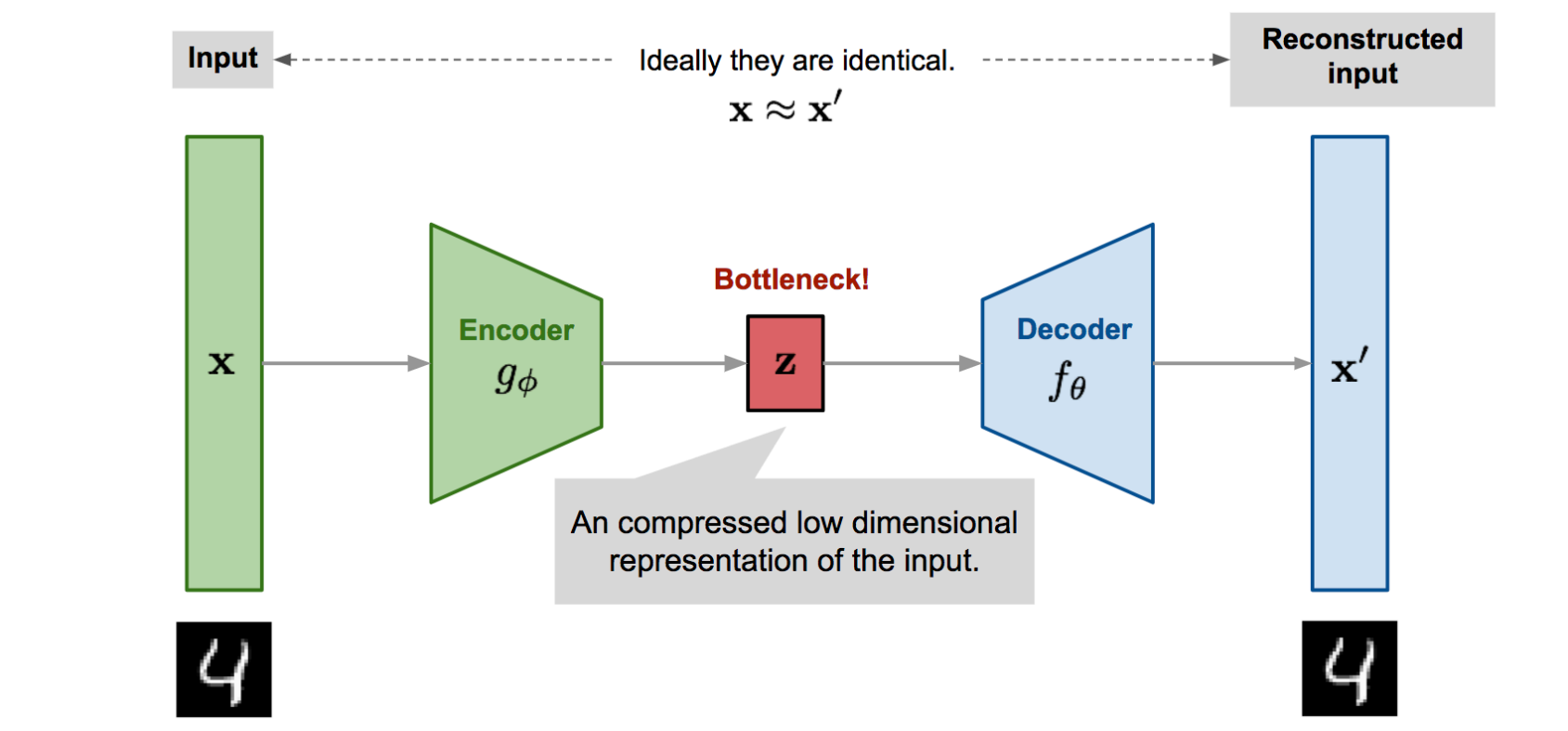

Figure 1: Illustration of an autoencoder model architecture. The **bottleneck** layer learns a latent representation, or code.

Encoder-Decoder framework

- $q(z|x)$ is the encoder - its input is a datapoint x, its output is a hidden representation z, and it is parameterized (by ϕ, becoming $q_\phi(z|x)$). The encoder 'encodes' the data into a latent (hidden) representation space z, with many times less dimensions.
- $p(x|z)$ is the decoder (usually $p_\theta(x|z)$). Crucially, we will look to make the latent space of z probabilistic, because we want a stochastic decoder/generator.
- For this, we will require the language of Bayesian inference : VAEs can be seen as *deep latent Gaussian models*.
- Note that in deep learning we *minimize the (reconstruction) loss*, but in variational inference we *maximize the evidence lower bound, the ELBO*. Avoid the confusion !

Stochastic encoder-decoder framework

Key point here, once the deterministic autoencoder is understood : instead of letting our neural network learn an arbitrary function, we are learning the parameters of a probability distribution modeling our data.
If we **sample** points from this distribution, we can generate new input data samples; a VAE is a ”generative model”.

Bayesian inference 101

Let us consider some dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ consisting of N i.i.d. samples of some continuous or discrete variable $\mathbf{x}$.

We assume that the data are generated by some random process, involving an unobserved, latent continuous random variable $\mathbf{z}$. The process consists of two steps:

1. a value $\mathbf{z}^{(i)}$ is generated from some prior distribution $p_{\boldsymbol{\theta}^*}(\mathbf{z})$;
2. a value $\mathbf{x}^{(i)}$ is generated from some conditional distribution $p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$.

We assume that the prior $p_{\boldsymbol{\theta}^*}(\mathbf{z})$ and likelihood $p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})$ come from parametric families of distributions $p_{\boldsymbol{\theta}}(\mathbf{z})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$.

Bayesian inference terms & intractabilities

- We generally assume *intractability* : the case where the marginal likelihood[1] $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, d\mathbf{z}$ is an intractable integral with no analytic solution or efficient estimator, and where
- The generative model is $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ (easily tractable), and
- The true posterior density $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) / p_{\boldsymbol{\theta}}(\mathbf{x})$ (Bayes) is intractable when $p_{\boldsymbol{\theta}}(\mathbf{x})$ is.
- We introduce a parameterized variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ that we will fit to the intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

[1] Taken as a function of θ, this is also the *model evidence*.

The associated Graphical Model

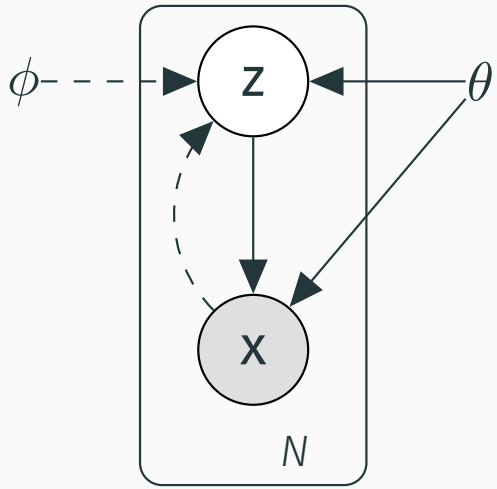

Figure 2: The type of directed graphical model under consideration. Solid lines denote the generative model $p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. The variational parameters $\boldsymbol{\phi}$ are learned jointly with the generative model parameters $\boldsymbol{\theta}$. Taken from [Kingma and Welling, 2013].

Convolutional VAE architecture (generic z)

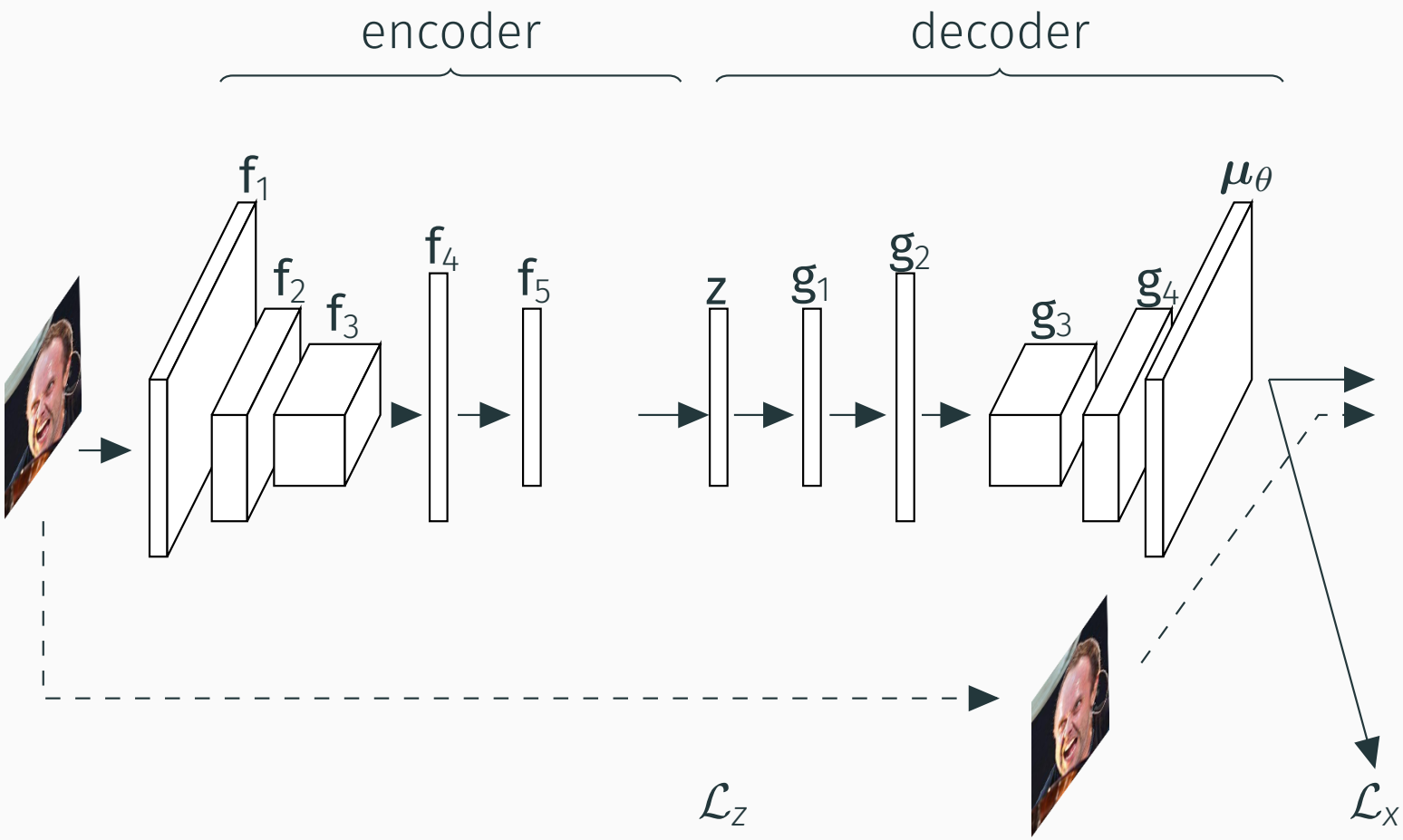

Figure 3: Typical architecture for a convolutional VAE. Note the use of *deconvolutional* layers for g_3, g_4. Adapted from [Lamb et al., 2016].

Interpretation

The variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$ (the encoder) maps a datapoint $\mathbf{x}^{(i)}$ to a distribution over latent variables $\mathbf{z}$ from which the datapoint could have been generated. The function $g_{\boldsymbol{\phi}}(.)$ is chosen such that it maps a datapoint $\mathbf{x}^{(i)}$ and a random noise vector $\boldsymbol{\epsilon}^{(l)}$ to a sample from the approximate posterior for that datapoint: $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})$ where $\mathbf{z}^{(i,l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$. Subsequently, the sample $\mathbf{z}^{(i,l)}$ is then input to function $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})$, which equals the probability density (or mass) of datapoint $\mathbf{x}^{(i)}$ under the generative model, given $\mathbf{z}^{(i,l)}$. This term is a negative *reconstruction error* in auto-encoder parlance.

ELBO derivation - 1

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$, which can each be rewritten :

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{1}$$

The first RHS term is the KL divergence of the approximate from the true posterior. Since this KL-divergence is non-negative, the second RHS term $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ is called the (variational) *lower bound* on the marginal likelihood of datapoint i, and can be written as :

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] \tag{2}$$

ELBO derivation - 2

This ELBO can also be written as :

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right] - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) \quad (3)$$

We want to maximize therefore differentiate (gradient ascent !) the surrogate objective, the lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$ w.r.t. both the variational parameters $\boldsymbol{\phi}$ and generative parameters $\boldsymbol{\theta}$. We easily get $\nabla_\theta \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \nabla_\theta \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$. However, the gradient of the lower bound w.r.t. $\boldsymbol{\phi}$ is a bit problematic.

From importance sampling to variational inference

Simple, yet easily one of the most important derivations in machine learning...

$$\log p(x) = \log \int p(x|z)p(z)dz$$

$$\log p(x) = \log \int p(x|z)\frac{p(z)}{q(z)}q(z)dz \qquad \text{(proposal distribution)}$$

$$\log p(x) \geq \int \log \left(p(x|z)\frac{p(z)}{q(z)}\right)q(z)dz \qquad \text{(Jensen's inequality)}$$

$$= \int q(z) \log p(x|z)dz - \int q(z) \log \frac{q(z)}{p(z)}dz$$

gives the ELBO $\forall q, p$:

$$\boxed{\log p(x) \geq \mathbb{E}_{q(z)}\left[\log p(x|z)\right] - D_{KL}\left[q(z)||p(z)\right]}$$

Now to estimate its gradients...

$\nabla_\phi \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x})$: Reparameterization trick

Under mild assumptions, for a chosen approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ we can reparameterize the random variable $\widetilde{\mathbf{z}} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ using a differentiable transformation $g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$ of an auxiliary noise variable $\boldsymbol{\epsilon}$:

$$\widetilde{\mathbf{z}} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{4}$$

We can now form Monte-Carlo estimates of expectations of some function $f(\mathbf{z})$ w.r.t. $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as follows:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[f(\mathbf{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}^{(i)}))\right] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})) \tag{5}$$

$$\text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}). \tag{6}$$

$\nabla_\phi \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x})$: Reparameterization trick - 2

We apply this technique to the variational lower bound (eq. (2)), yielding our generic Stochastic Gradient Variational Bayes (SGVB) estimator $\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$:

$$\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)} | \mathbf{x}^{(i)})$$
$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}). \tag{7}$$

The reparameterization trick is a simple change of variables. Let $\mathbf{z}$ be a continuous random variable, and $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ be some conditional distribution. It is then *often possible to express the random variable* $\mathbf{z}$ *as a deterministic variable* $\mathbf{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$, *where* $\boldsymbol{\epsilon}$ *is an auxiliary variable with independent marginal* $p(\boldsymbol{\epsilon})$, and $g_{\boldsymbol{\phi}}(.)$ is some vector-valued function parameterized by $\boldsymbol{\phi}$.

$\nabla_\phi \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x})$: Reparameterization trick - 3

Often, the KL-divergence $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}))$ of eq. (3) can be integrated analytically, such that only the expected reconstruction error $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$ requires estimation by sampling. The KL-divergence term can then be interpreted as regularizing $\boldsymbol{\phi}$, encouraging the approximate posterior to be close to the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$. This yields a second version of the SGVB estimator $\widetilde{\mathcal{L}}^B(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$, corresponding to eq. (3), which typically has less variance than the generic estimator:

$$\widetilde{\mathcal{L}}^B(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \underbrace{-D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{KL regularizer (often analytical)}} + \underbrace{\frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}))}_{\text{reconstruction error (sampled)}}$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}). \tag{8}$$

Regularizer : interpretation

- The KL regularizer term means 'keep the representations z of each digit sufficiently diverse'. If we didn't include it, the encoder could learn to cheat and give each datapoint a representation in a different region of Euclidean space. We could actually get rid of this term entirely, but it does help in learning well-formed latent spaces, and reducing overfitting to the training data.
- Given multiple datapoints from a dataset $\mathbf{X}$ with N datapoints, we can construct an estimator of the marginal likelihood lower bound of the full dataset, based on minibatches:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \simeq \widetilde{\mathcal{L}}^{M}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^{M}) = \frac{N}{M} \sum_{i=1}^{M} \widetilde{\mathcal{L}}^{B}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \tag{9}$$

The VAE : Gaussians as prior and variational approximation

- Let the prior over the latent variables be the centered isotropic multivariate Gaussian $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. In this case, the prior lacks parameters, and is completely factorizable : it is the Gaussian hypersphere (in dimension d).
- A single encoder neural network performs posterior inference over all of the datapoints in the dataset, and is also picked as a Gaussian :

$$(\mu, \log \sigma) = EncoderNeuralNet_{\phi}(x)$$
$$q_{\phi}(z|x) = \mathcal{N}(z; \mu, diag(\sigma))$$

- The VAE learns stochastic mappings between observed x-space (complex distribution) and latent z space with spherical distribution !

The VAE - Gaussian character

- It is enough to encode only latent $(\mu_i)_{i \leq d}$ and $(\sigma_i^2)_{i \leq d}$ - therefore g is an affine function, and the reparameterization trick enables us to backprop through the random number generator $p(\epsilon) \sim \mathcal{N}(0, I)$, learning the VAE end-to-end.
- Crucially, the KL term, in the Gaussian prior and posterior approximation case, is known analytically :

$$-D_{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z})\right] = \frac{1}{2}\sum_{j=1}^{d}\left(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2\right)$$

Convolutional VAE architecture - Gaussian z

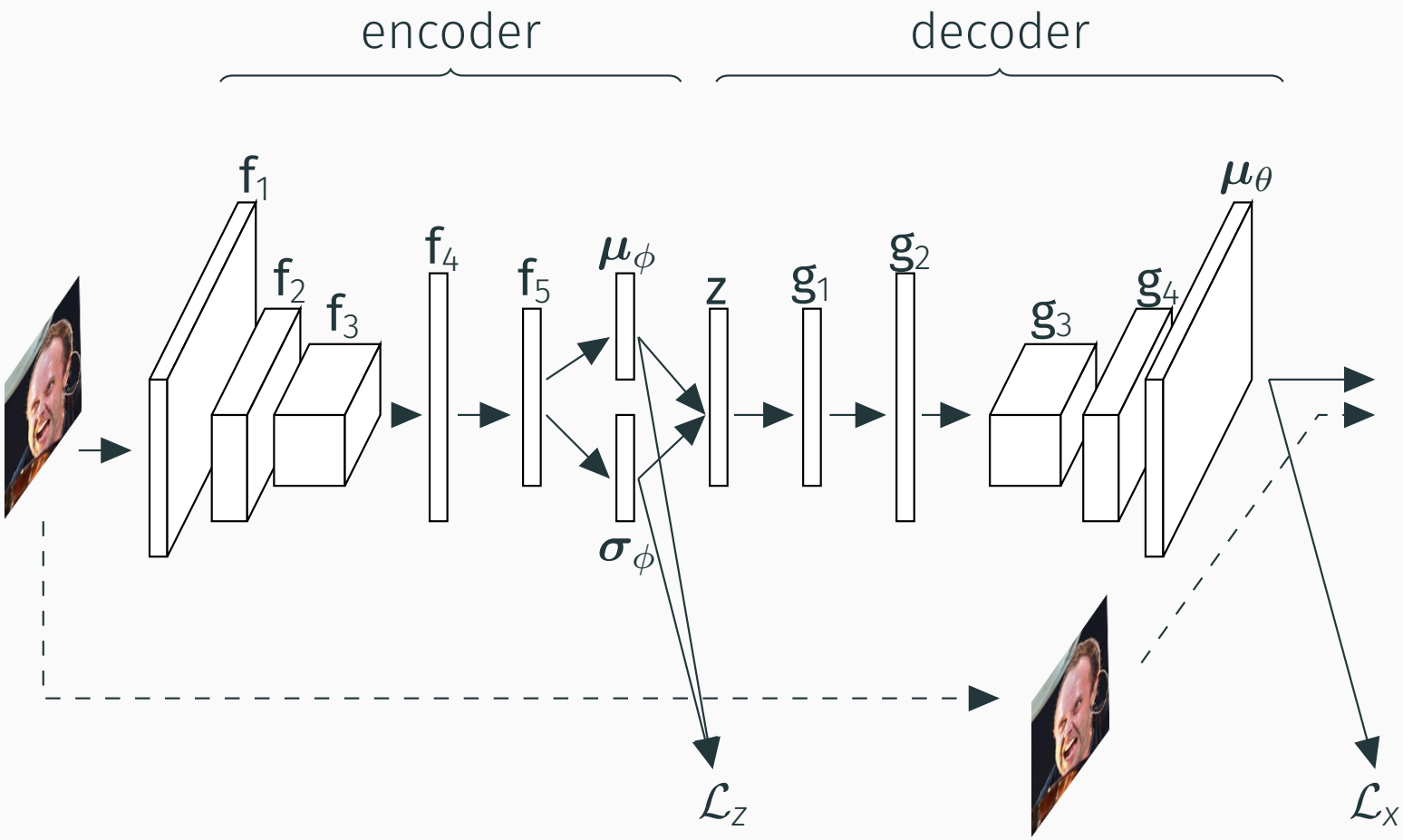

Figure 4: Typical architecture for a convolutional VAE. Note the use of *deconvolutional* layers for g_3, g_4. Adapted from [Lamb et al., 2016].

The Variational Autoencoder

Finally we get the loss function by summing the below for each batch datapoint $\mathbf{x}^{(i)}$:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{d} \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)} | \mathbf{z}^{(i,l)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{I}) \tag{10}$$

VAE training : Algorithm

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters

repeat

$\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)

$\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$

$\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator)

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or ADAM)

until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$

Results on MNIST - Latent manifold interpolation

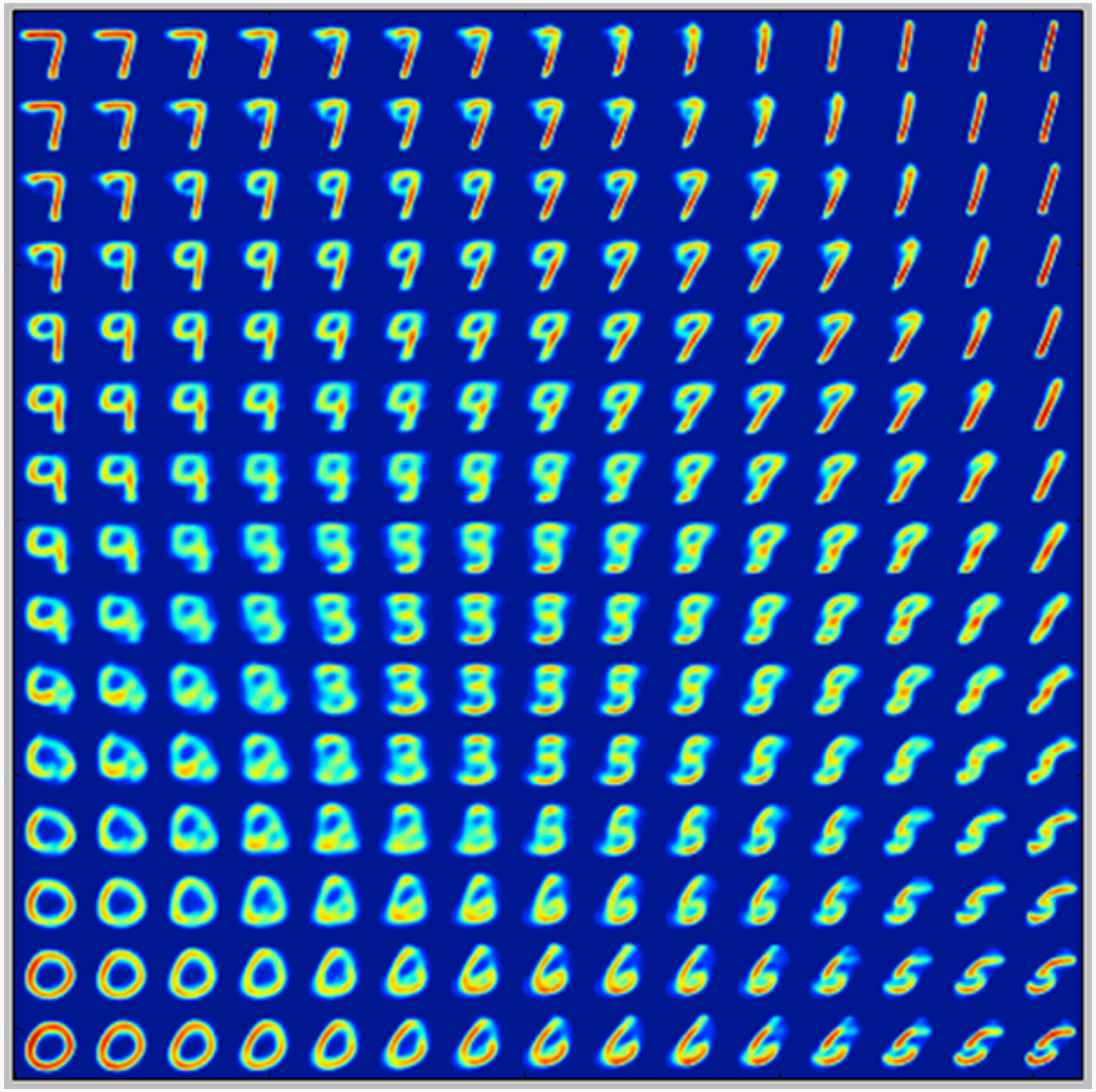

Figure 5: MNIST manifold traversal, for an embedding dimension of 2.

Results on Celeb-A

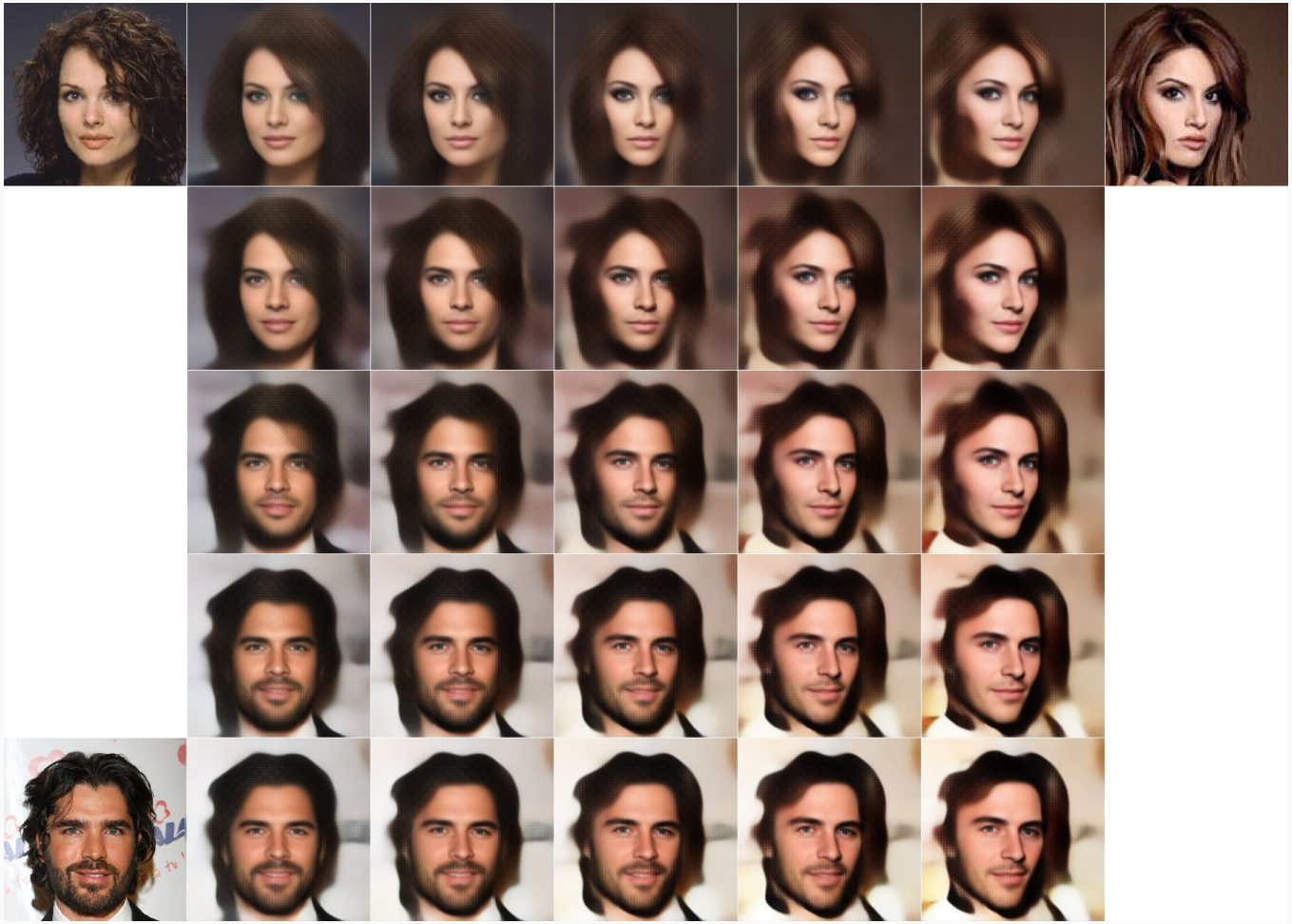

Figure 6: *J-diagram* interpolating, in latent space, between reconstructions of three Celeb-A face datapoints. This simple arithmetic is useful in word embeddings where $king + woman - man \simeq queen$. Taken from [White, 2016].

Results on Celeb-A - Smile vector

Figure 7: Interpolations varying the strength of a *smile vector* computed by doing latent space averaging of labeled pictures. Taken from [White, 2016], which also contains several useful sampling techniques for improving the appearance of generations.

Why are results blurry ?

”VAE blurriness can be countered by making the model less constrained, by incorporating non-factorized decoders, priors and/or approximate posteriors. Inverse Autoregressive Flows are solution for the latter, and in combination with an autoregressive prior, samples can be as crisp as any log-likelihood based model. An obvious advantage for synthesis in VAEs, compared to autoregressive models, is that synthesis is parallelizable and therefore much faster in terms of wall-clock time.”

Tighter bounds with IWAE (2015)

- Idea = change the log-likelihood lower bound to make it tighter, see [Burda et al., 2015].
- To this end, use several samples (from importance weighting)
- The recognition network generates multiple approximate posterior samples, and their weights are averaged.

Tighter bounds with IWAE - 2

$$L(x) = \mathbb{E}_{z \sim q(z|x)} \Big[\log \frac{p(x,z)}{q(z|x)} \Big]$$

is the standard ELBO objective.
This gets replaced by

$$L_k(x) = \mathbb{E}_{z_1, \dots, z_k \sim q(z|x)} \Big[\log \frac{1}{k} \sum_{i=1}^{k} \frac{p(x,z_i)}{q(z_i|x)} \Big]$$

for k a chosen hyperparameter ($k = 1$ recovers the standard VAE).
The $(z_i)_{i \leq k}$ are sampled independently. By Jensen's inequality, this is still a lower bound on the marginal log-likelihood (exercise !).

Tighter bounds with IWAE - 3

Using more samples improves the tightness of the bound :

Theorem

For all k, the lower bounds (L_k) satisfy

$$\log p(x) \geq L_{k+1} \geq L_k.$$

Moreover, if $p(x,z)/q(z|x)$ is bounded, then $L_k \underset{k\to\infty}{\to} \log p(x)$.

The training procedure remains very similar, but uses a slightly different estimator for gradients (see paper for details).

Disentanglement with β-VAE (2017)

- Idea [Higgins et al., 2017]: give the latent variables interpretability and semantic meaning by modifying the loss function, in order to automatically learn a disentangled representation of input data.
- They propose a simple modification to the VAE loss.
- To this end, introduce a scalar hyperparameter β (adjustable).
- We will use it in order to define 'good features' to learn as 'features that are useful for compressing the data.'

Disentanglement with β-VAE - Loss function

The standard ELBO

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

is simply turned into

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right] - \beta \cdot D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}))$$

β can be interpreted as a Lagrangian/Karush-Kuhn-Tucker multiplier, or more simply, a regularization strength. 'With $\beta > 1$ (strong pressure on the latent bottleneck), the model is pushed to learn a more efficient latent representation of the data, which is disentangled if the data contains at least some underlying factors of variation that are independent : β-VAE aligns latent dimensions with components that make different contributions to reconstruction.'

Disentanglement with β-VAE - Results

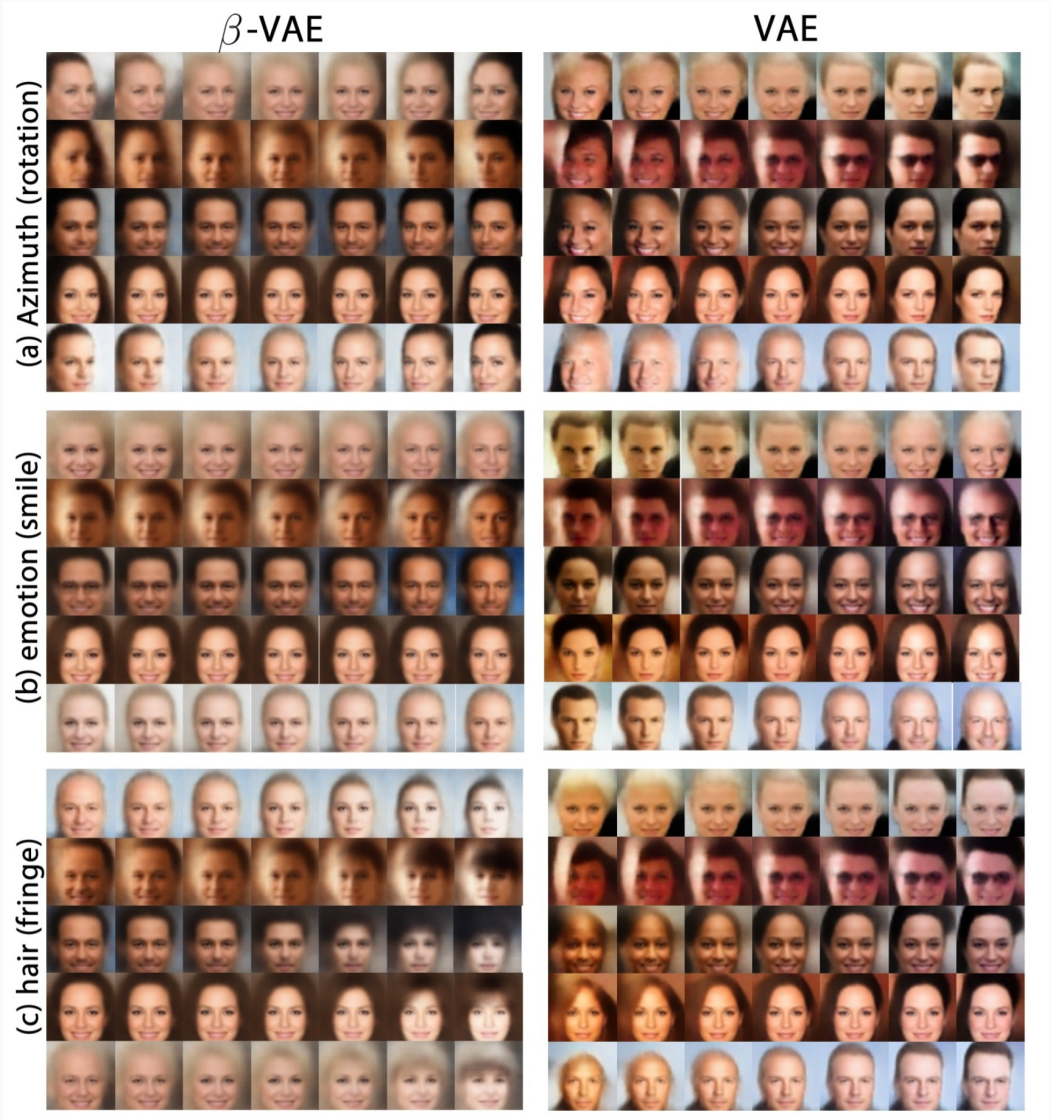

Figure 8: Comparison of β-VAE ($\beta = 250$), and a standard VAE on the Celeb-A faces dataset. The β-VAE spontaneously learns disentanglement factors.

Disentanglement with β-VAE

- This works because the variational Gaussian approximation is already *fully factored* (orthogonalized) - think of the KL as extra bits of information required to code the approximate vs the true prior. Part of minimizing this is aligning the marginals, and part of this is aligning the dependence structure.
- Problem with β-VAE : no guidance to find the right β value(s).
- Connections with information theory & convex analysis
- See [Zhao et al., 2017] for a solution to the problem of 'variational autoencoders tend to ignore the latent variables when combined with a decoding distribution that is too flexible'.
- See also VQ-VAE [van den Oord et al., 2017] for another, coding theory-oriented solution to this problem.

An application : Grammar-VAE for molecular synthesis

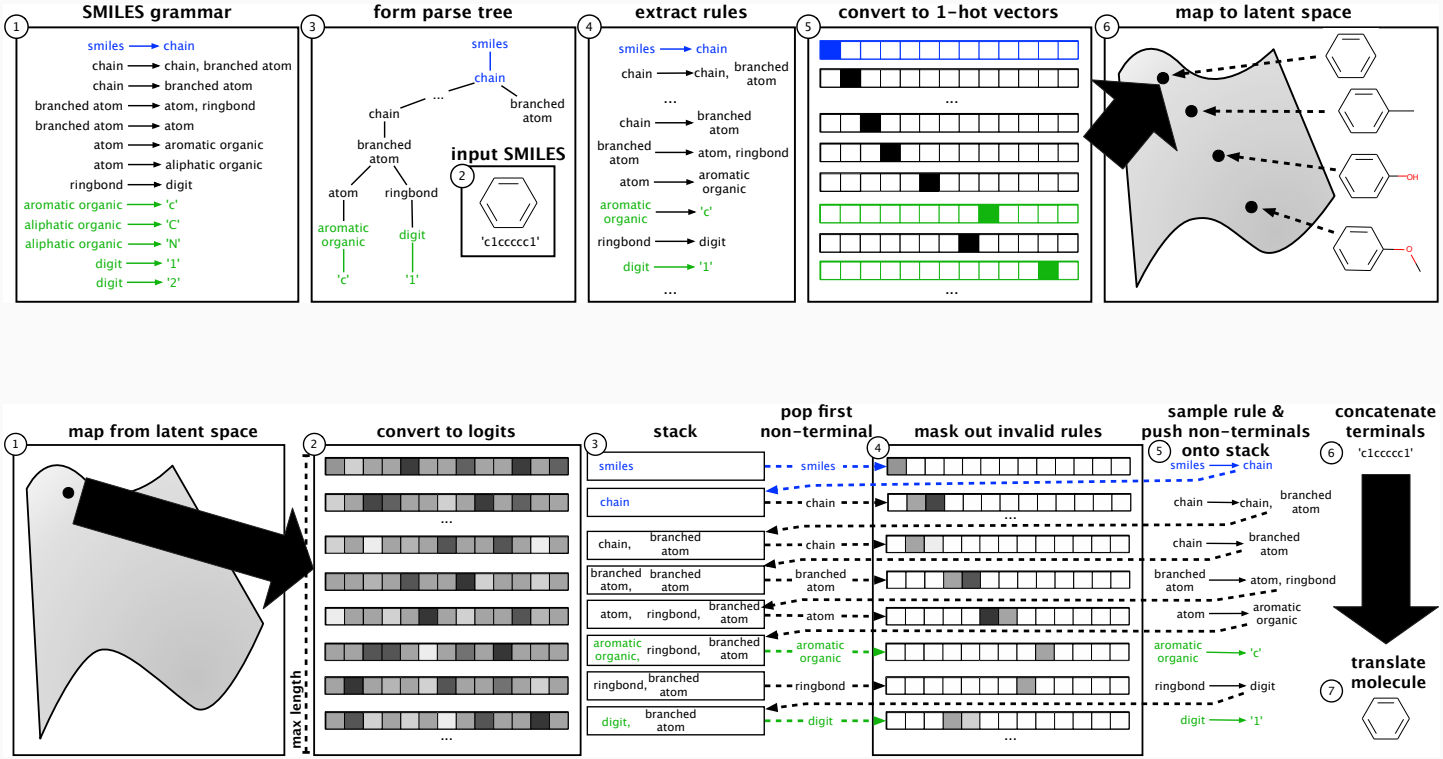

Figure 9: Encoder and decoder architecture for Grammar-VAE. Sampling a random molecule in latent space won't necessarily yield meaningful (chemically correct) results. The VAE in ADDREF is significantly modified, so as to generate only grammatically correct expressions.

Grammar VAE for molecular representation - Results

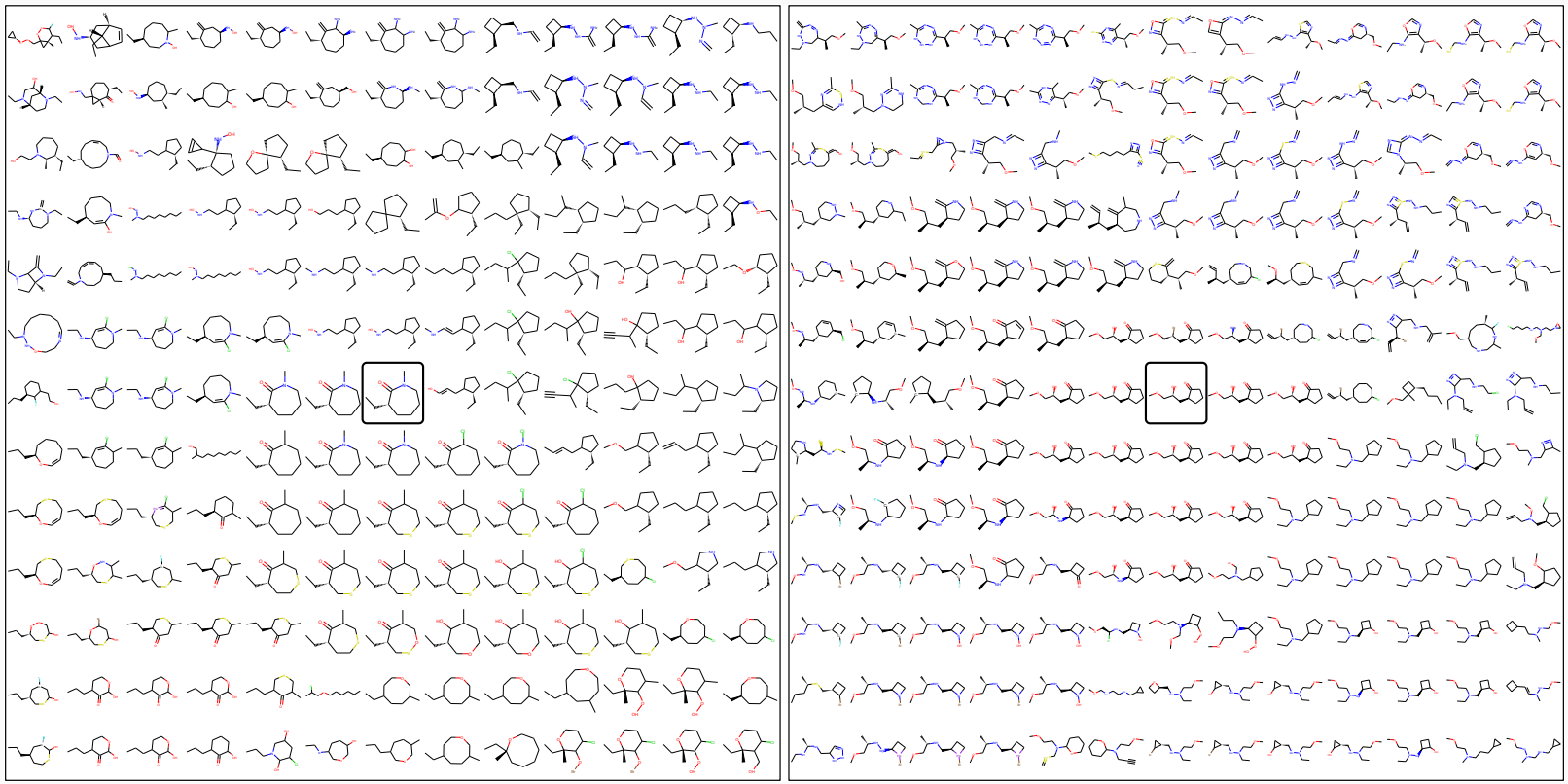

Figure 10: Molecules generated via latent space traversal in 56 dimensions. The central molecule (boxed) is the starting point.

VAEs as generative models : Pros and cons

(h/t David Duvenaud)

Pros :

- Flexible generative model
- End-to-end gradient training
- Measurable objective via the ELBO (lower bound : model is at least this good)
- Fast test-time inference

Cons :

- Suboptimal variational factors
- Limited approximation to true posterior
- Blurry
- Can have high-variance gradients

GitHub code repositories

TensorFlow

VAE InfoVAE IWAE VQ-VAE

Keras

VAE Grammar-VAE

Pytorch

VAE β-VAE InfoVAE VQ-VAE

References i


Blei, D., Mohamed, S., and Ranganath, R. (2016).
Variational Inference: Foundations and Modern Methods.
NIPS 2016 tutorial.


Burda, Y., Grosse, R., and Salakhutdinov, R. (2015).
Importance Weighted Autoencoders.
ArXiv e-prints.


Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017).
beta-VAE : Learning Basic Visual Concepts with a Constrained Variational Framework.
ICLR 2017.

References ii


Hinton, G. E. and Salakhutdinov, R. R. (2006).
Reducing the Dimensionality of Data with Neural Networks.
Science.


Jimenez Rezende, D., Mohamed, S., and Wierstra, D. (2014).
Stochastic Backpropagation and Approximate Inference in Deep Generative Models.
ArXiv e-prints.


Kingma, D. P. (2017).
Variational inference and deep learning : a new synthesis.
Ph.D. thesis.


Kingma, D. P. and Welling, M. (2013).
Auto-Encoding Variational Bayes.
ArXiv e-prints.

References iii

- Lamb, A., Dumoulin, V., and Courville, A. (2016).
  Discriminative Regularization for Generative Models.
  ArXiv e-prints.
- van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017).
  Neural Discrete Representation Learning.
  ArXiv e-prints.
- White, T. (2016).
  Sampling Generative Networks.
  ArXiv e-prints.
- Zhao, S., Song, J., and Ermon, S. (2017).
  InfoVAE: Information Maximizing Variational Autoencoders.
  ArXiv e-prints.