β-VAE

Benjamin Levy

December 10, 2020
STAT185

1 Introduction

A β-variational autoencoder (β-VAE) is a family of generative models whose goal is to infer a "disentangled," low-dimensional representation of some high-dimensional data, often images [8]. In general, auto-encoders seek to reconstruct some input $\vec{x} \in \mathbb{R}^d$ while simultaneously learning a latent representation, $\vec{z} \in \mathbb{R}^k$, which is typically chosen to be lower-dimensional than $\vec{x}$. This is accomplished by training two models: an encoder parameterized by ϕ, denoted $g_\phi(\vec{x})$, and a decoder parameterized by θ, denoted $f_\theta(\vec{z})$. The algorithm is, roughly:

1. Given example $\vec{x}$, compute $\vec{z} = g_\phi(\vec{x})$
2. Compute reconstruction $\vec{x}' = f_\theta(\vec{z})$

A limitation of this deterministic autoencoder is that the latent space in which $\vec{z}$ resides is typically quite sparse and/or irregular. This can make it difficult to use the latent representations for inference or other purposes. Variational auto-encoders (VAEs) improve upon this by replacing the decoder f_θ with a stochastic sampling function. Consequently, g_ϕ now outputs a mean $\vec{\mu}$ and a covariance Σ, which parameterize a Gaussian distribution [3]. The algorithm then becomes:

1. Given example $\vec{x}$, compute $(\vec{\mu}, \Sigma) = g_\phi(\vec{x})$
2. Sample latent variable $\vec{z} \sim \mathcal{N}(\vec{\mu}, \Sigma)$
3. Compute reconstruction $\vec{x}' = f_\theta(\vec{z})$

To make this model tractable, we typically use isotropic Gaussians, meaning the covariance matrix Σ is diagonal. By replacing the deterministic algorithm with a stochastic one, the model is forced to learn latent representations that are more robust to Gaussian noise, since if small perturbations led to dramatically different outputs, the model would perform poorly at reconstructing the output. The loss function for the VAE can be expressed as a sum of two terms: (1) reconstruction error and (2) regularization (defined more rigorously later on in the methods section). The reconstruction term pushes the model to faithfully reconstruct the original data, while the regularization term encourages the model to learn a generalizable and efficient latent representation.

While the VAE model solves the problem of irregular latent spaces, there are no guarantees that the latent representations are *meaningful*. This gives rise to the idea of *disentanglement*, which is roughly the notion that the different components of the latent variable $\vec{z}$ should correspond to distinct *generative factors* (i.e. the parameters of the generative process) [2]. For instance, if we are modelling an image dataset consisting of human faces, we might want one dimension to correspond to skin tone, another to correspond to eye shape, a further one to correspond to amount of hair, and so on. Introduced by Burgess et al. in 2018, the β-VAE is a model designed with this goal in mind [4]. An additional hyperparameter, $\beta \in \mathbb{R}$ is added as a coefficient for the regularization term from the VAE loss. When $\beta = 1$, the loss function is identical to the original VAE loss function. When $\beta > 1$, the model is forced to learn a less expressive, and therefore more information-dense, latent encoding for the data. Furthermore, the representations learned by β-VAE

tend to be more disentangled, meaning that it is more possible to recover the original factors that may have generated the data.

2 Methods

2.1 Variational auto-encoders

The goal of a VAE is to learn a distribution p_θ to model our data $\vec{x}$ and its hypothesized latent variables $\vec{z}$. We can arrange related $\vec{x}$ and $\vec{z}$ using Bayes' rule:

$$p_\theta(\vec{x}|\vec{z}) = \frac{p_\theta(\vec{z}|\vec{x})p_\theta(\vec{x})}{p_\theta(\vec{z})}$$

Under this framework, we want to maximize the log-probability of data generated according to the process: (1) sample a point $\vec{z} \sim p_\theta(\vec{z})$, then sample $\vec{x} \sim p_\theta(\vec{x}|\vec{z})$:

$$\ell = \sum_{i=1}^{N} \log p_\theta(\vec{x}_i)$$

Marginalizing over all values of $\vec{x}$,

$$\ell = \sum_{i=1}^{N} \log \int p_\theta(\vec{x}_i|\vec{z})p_\theta(\vec{z})d\vec{z}$$

We now introduce an auxiliary distribution, q_ϕ. allow us to massage this expression into one involving a KL-divergence.

$$\begin{aligned}
\ell &= \sum_{i=1}^{N} \log \int p_\theta(\vec{x}_i|\vec{z}_i)p_\theta(\vec{z}_i)\frac{q_\phi(\vec{z}_i)}{q_\phi(\vec{z}_i)}d\vec{z}_i \\
&= \sum_{i=1}^{N} \log \int \frac{p_\theta(\vec{x}_i, \vec{z}_i)}{q_\phi(\vec{z}_i)}q_\phi(\vec{z}_i)d\vec{z}_i \\
&= \sum_{i=1}^{N} \log \mathbb{E}\left[\frac{p_\theta(\vec{x}_i, \vec{z}_i)}{q_\phi(\vec{z}_i)}\right]
\end{aligned}$$

Using Jensen's inequality, we get the evidence lower bound (ELBO)

$$\begin{aligned}
&\geq \sum_{i=1}^{N} \mathbb{E}_{\vec{z}\sim q_\phi} \log \left[\frac{p_\theta(\vec{x}_i, \vec{z}_i)}{q_\phi(\vec{z}_i)}\right] \\
\ell &= \sum_{i=1}^{N} (\mathbb{E}_{\vec{z}\sim q_\phi}[\log p_\theta(\vec{x}_i, \vec{z}_i)] - \mathbb{E}_{\vec{z}\sim q_\phi}[\log q_\phi(\vec{z})])
\end{aligned}$$

The definition of the KL divergence between q_ϕ and p_θ is:

$$D_{KL}(q_\phi(\vec{z})||p_\theta(\vec{z}|\vec{x})) = \mathbb{E}_{\vec{z}\sim q_\phi}\left[\log \frac{q_\phi(\vec{z})}{p_\theta(\vec{z}|\vec{x})}\right]$$

Since $p_\theta(\vec{z}|\vec{x}) = \frac{p_\theta(\vec{z},\vec{x})}{p_\theta(\vec{x})}$

$$D_{KL}(q_\phi(\vec{z})||p_\theta(\vec{z}|\vec{x})) = -\left(\mathbb{E}_{\vec{z}\sim q_\phi}[\log p_\theta(\vec{z}, \vec{x})] - \mathbb{E}_{\vec{z}\sim q_\phi}[\log q_\phi(\vec{z})]\right) + \log p_\theta(\vec{x})$$

Plugging this back into our log-likelihood,

$$\ell = \sum_{i=1}^{N} \left(\log(p_\theta(\vec{x}_i)) - D_{KL}(q_\phi(\vec{z}_i)||p_\theta(\vec{z}_i|\vec{x}_i)) \right)$$

The first term is maximized when the reconstruction $\vec{x}$ is faithful to the original data, whereas the second term is minimized when p_θ is close to q_ϕ.

2.2 Choice of q_ϕ

In theory, we could choose an arbitrarily expressive family of distributions q_ϕ to model the latent variable $\vec{z}$. In practice, we make the simplifying **mean-field assumption**, which states that the dimensions of q_ϕ are independent:

$$q_\phi(\vec{z}|\vec{x}) = \prod_{j=1}^{d} q_\phi(\vec{z}_j|\vec{x}_j)$$

In our case, we operationalize this by saying that q_ϕ is an isotropic Gaussian (diagonal covariance) [1]. Although this assumption reduces the expressiveness of the variational family q_ϕ, we gain significantly in terms of computational efficiency when it comes time to learn the parameters of the distribution.

At this point, we are in effect trying to learn a unique distribution $q_\phi(\vec{x}|\vec{z})$ for each $\vec{x}$. If we have N observations and a latent space of dimension d, then if the latent distribution is Gaussian, we would need to learn a mean and variance for each dimension of each latent variable, or $2 \times N \times d$. In other words, $N \gg p$ (where p is the number of parameters in the model), which both dramatically increases the variance of the parameter estimates (they are highly dependent on the particular data sample at hand) and increases the cost of fitting the model. To get around this, we perform *amortised* variational inference: instead of directly estimating μ_i and Σ_i for each $i = 1, \ldots, N$, we assume that each μ_i, Σ_i can be approximated by a function $g_\phi(\vec{x})$, typically a neural network [5]. We refer to this function interchangeably as the **encoder** or **inference network**. This is reresented by the rectangle in figure 1 and is denoted g_ϕ here. Because the weights ϕ are shared across all observations, we say that the cost of learning ϕ is amortised across the entire dataset (hence, "amortised variational inference"). Symmetrically, the conditional posterior distribution $p_\theta(\vec{x}|\vec{z})$ is implemented as a neural network that is typically the mirror image of the encoder, which is called the **decoder** or **generative network** and is denoted f_θ here.

2.3 The reparameterization trick

A further advantage of amortisation via the encoder and decoder neural networks is that it allows us to convert this problem from one of sampling from a posterior distribution – which can require the use of computationally intensive or mathematically complex methods such as expectation-maximization – to a simple non-convex optimization problem. That is, all the terms in the ELBO loss function can be expressed in terms of ϕ and θ, meaning that if we can differentiate through all the operations involved, then this problem can be solved with gradient descent. To revisit, the encoder and decoder form a single algorithm:

1. Given $\vec{x}$, compute $\vec{\mu}, \Sigma = g_\phi(\vec{x})$
2. Sample $\vec{z} \sim \mathcal{N}(\vec{\mu}, \Sigma)$
3. Compute reconstruction $\hat{x} = f_\theta(\vec{z})$

The one operation that does not allow for easy differentiation is the sampling step (2). In order to get around this, we use the "reparameterization trick". Recall that a d-dimensional isotropic Gaussian random variable can be converted into a sample from the d-dimensional multivariate normal distribution:

$$\vec{\epsilon} = (\vec{z} - \vec{\mu}) \odot \frac{1}{\vec{\sigma}} + \vec{\mu}$$

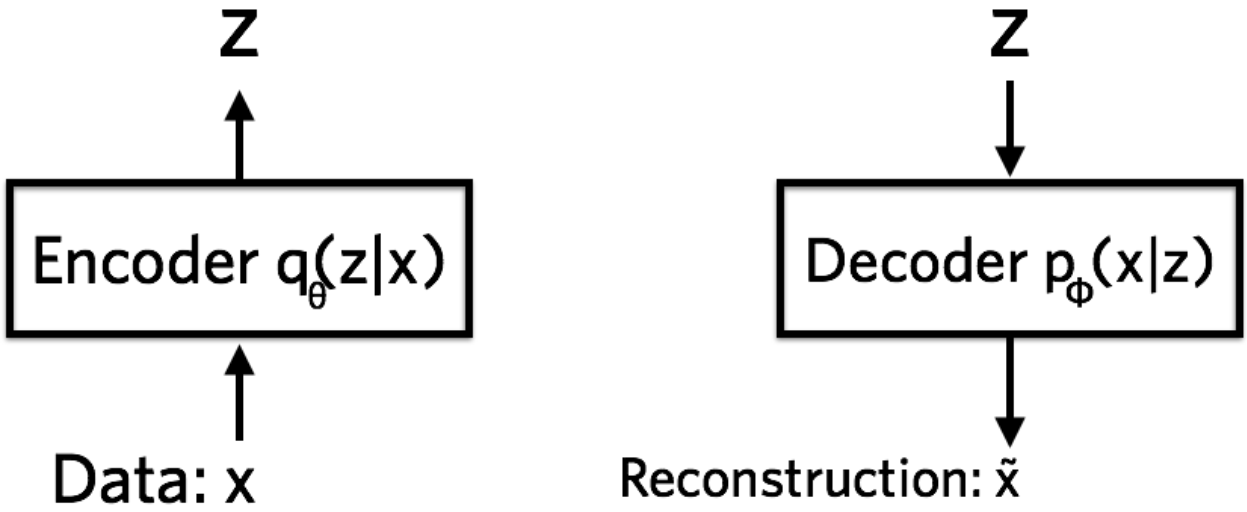

Figure 1: Simplified schematic of the encoder and decoder, where the left and right rectangles labelled $q(z|x)$ and $p(x|z)$ refer to the inferential and generative networks, respectively. First, $\vec{z}$ is generated from $\vec{x}$ using the encoder and then $\vec{x}$ is reconstructed using $\vec{z}$ [5]. Note: the notation in this figure is opposite to what we use in this paper: θ and ϕ have been flipped such that θ parameterizes q and ϕ parameterizes p. Otherwise, the setup is the same to what we have been discussing.

Where $\vec{\epsilon}$ is a standard normal random variable (not to be confused with $\vec{z}$, which is our latent variable with mean μ and covariance Σ) and $\vec{\sigma}$ is the d-dimensional vector of standard deviations for the components of $\vec{z}$. $\odot$ represents element-wise multiplication. Using simple algebra, we can rearrange this:

$$\vec{z} = (\vec{\epsilon} \odot \vec{\sigma}) + \vec{\mu}$$

This means that sampling $\vec{z} \sim \mathcal{N}(\vec{\mu}, \Sigma)$ is equivalent to sampling $\vec{\epsilon} \sim \mathcal{N}(0, I)$ and then transforming it into $\vec{z}$ as above. The gradient operation is well-defined for sampling from the standard normal, so we can make the entire VAE algorithm differentiable by adding this reparameterization step:

1. Given $\vec{x}$, compute $\vec{\mu}, \Sigma = g_\phi(\vec{x})$. Let $\vec{\sigma} = \sqrt{\text{diag}(\Sigma)}$
2. Sample $\vec{\epsilon} \sim \mathcal{N}(0, I)$
3. Let $\vec{z} = \vec{\epsilon} \odot \vec{\sigma} + \vec{\mu}$
4. Compute reconstruction $\hat{x} = f_\theta(\vec{z})$

2.4 β-VAE

In the original paper, the authors provide a derivation of the β-VAE loss using constrained optimization [4]. Given data $\vec{x}$ distributed according to the data distribution $\mathcal{D}$, the goal is to jointly learn generative parameters θ and latent parameters ϕ for the distributions p_θ and q_ϕ, respectively. We want to find θ and ϕ such that the posterior log-probability $p_\theta(\vec{x}|\vec{z})$ where $\vec{z} \sim q_\phi$ is maximized:

$$\theta^*, \phi^* = \underset{\theta,\phi}{\text{argmax}} \mathbb{E}_{\vec{x}\sim\mathcal{D}} \left[\mathbb{E}_{\vec{z}\sim q_\phi} \left[\log p_\theta \left(\vec{x}|\vec{z} \right) \right] \right]$$

Subject to constraint

$$D_{KL} \left(q_\phi(\vec{z}|\vec{x}) || p(\vec{z}) \right) < \delta$$

Where δ is a small positive constant (since KL divergence is nonnegative). The constraint is imposed to narrow the "information bottleneck" by encouraging the latent posterior $q_\phi(\vec{z}|\vec{x})$ to be close to a simple prior $p(\vec{z})$, which in this case is a standard gaussian, $\mathcal{N}(0, I)$. The idea is that we want to make q_ϕ less expressive, which in turn forces it to be as efficient as possible in which latent factors are represented. Hopefully, the efficiency is maximized when the latent factors line up with the original factors that generated the data.

We can rewrite this constrained optimization problem as a Lagrangian subject to the KKT condition, with Lagrange multiplier β and Lagrangian objective $\mathcal{F}$:

$$\begin{aligned}
\mathcal{F}(\theta, \phi, \beta; \vec{x}, \vec{z}) &= \mathbb{E}_{\vec{z}\sim q_\phi}\left[\log p_\theta(\vec{x}|\vec{z})\right] - \beta\left(D_{KL}(q_\phi(\vec{z}|\vec{x})||p(\vec{z})) - \delta\right) \\
&= \mathbb{E}_{\vec{z}\sim q_\phi}\left[\log p_\theta(\vec{x}|\vec{z})\right] - \beta\left(D_{KL}(q_\phi(\vec{z}|\vec{x})||p(\vec{z}))\right) + \beta\delta \\
&\geq \mathbb{E}_{\vec{z}\sim q_\phi}\left[\log p_\theta(\vec{x}|\vec{z})\right] - \beta\left(D_{KL}(q_\phi(\vec{z}|\vec{x})||p(\vec{z}))\right)
\end{aligned}$$

Immediately, we can recognize that this is nearly identical to the vanilla VAE loss, except for the hyperparameter β. When $\beta = 1$, this is equivalent to the vanilla VAE loss. When $\beta = 0$, the model becomes similar to a non-variational auto-encoder, since the only objective is to faithfully reconstruct the data. As β increases, the bottleneck in the latent space becomes narrower, promoting a sparser latent representation (i.e. a latent space that more closely resembles a standard Gaussian).

2.4.1 Measuring disentanglement

The authors of the original β-VAE paper go a step further and propose a "disentanglement metric" to quantify how well the model's latent representations align with the hypothesized generative factors for the data [4].

Formally, we assume the data $\vec{x}$ are drawn from a dataset $\mathcal{D} = \{X, V, W\}$, where X is the data matrix, and V, W are matrices of hypothesized *generative factors* for each data point. For instance, a factor for a dataset of faces might be *skin tone* or *glasses*. An image is created using a simulator: $\vec{x} \sim \mathbf{Sim}(\vec{v}, \vec{w})$. Then,

1. Uniformly sample factor y from the set of K possible generative factors
2. Repeat L times:
   (a) Sample two sets of generative latent representations, $\vec{v}_{1,l}$ and $\vec{v}_{2,l}$ such that both $\vec{v}_{1,l}$ and $\vec{v}_{2,l}$ have the same value for factor y and are free to differ in all other generative factors
   (b) Simulate an image $x_{1,l\sim\mathbf{Sim}(v_{1,l})}$ and compute the latent representation $\vec{z}_{1,l} \sim q(\vec{z}|\vec{x}_{1,l})$ using the encoder $g_\phi(\vec{x}_{1,l})$. Repeat for $v_{1,l}$
   (c) Compute a difference vector $z^l_{\text{diff}} = |z_{1,l} - z_{2,l}|_1$
3. Given all z^l_{diff} computed as above, calculate the average difference vector $\bar{z}_{\text{diff}} = \frac{1}{L}\sum_{l=1}^{L} z^l_{\text{diff}}$
4. Use $\bar{z}_{\text{diff}}$ as the input for a linear classifier (i.e. logistic regression), where the goal is to predict the factor y that was kept constant between all pairs of latent factors

This algorithm can be seen in figure 2

2.5 Implementation

Depending on the type and complexity of the data, various architectures for the inference and generative networks are used. In the simplest case, each is a feed-forward neural network, which is common for either numeric data or small-sized image data (e.g. MNIST, which is $28 \times 28 \times 1$). For larger image data, such as the CelebA dataset ($64 \times 64 \times 3$), convolutional neural networks (CNNs) are often used, with the structure of the generative network (often referred to as the "encoder") mirroring the structure of the inference network (often referred to as the "decoder"). The weights of the generative and inference networks are found using

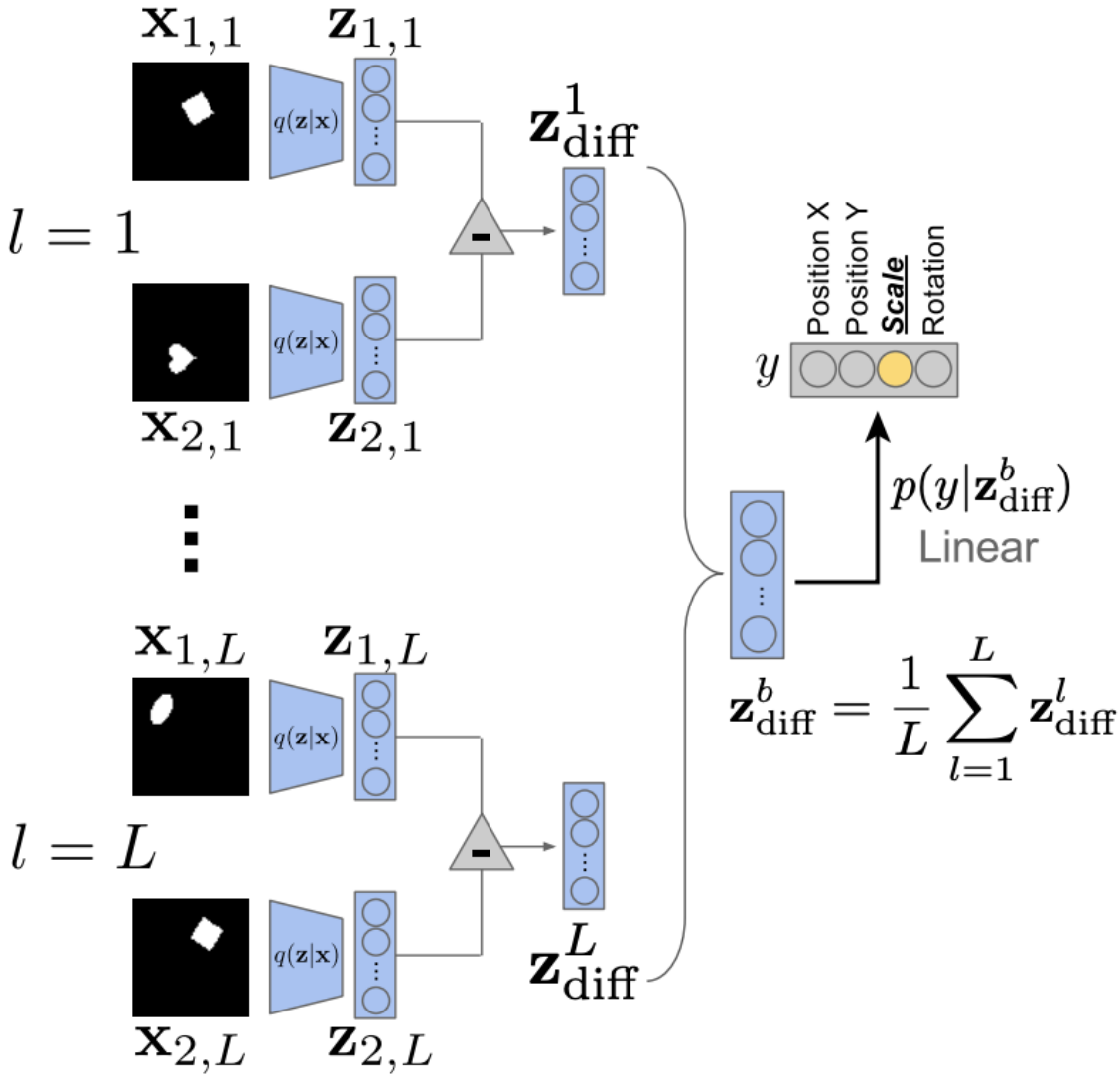

Figure 2: A schematic of the proposed disentanglement metric from [4]. Each vector $\vec{z}^l_{\text{diff}}$ in the process is generated from pairs of latent representations $z_{1,l}, z_{2,l}$ with the exact same value for a chose generative factor y. All of the L pairs in batch b may differ in terms of the value of the generative factor, but the choice of y is constant between pairs. The linear classifier, $p(y|\vec{z}^b_{\text{diff}})$ is a multivariate classification algorithm whose goal is to predict which generative factor was used to generate the average difference vector $\vec{z}^b_{\text{diff}}$ (in the text, we refer to this average difference vector as $\bar{z}_{\text{diff}}$).

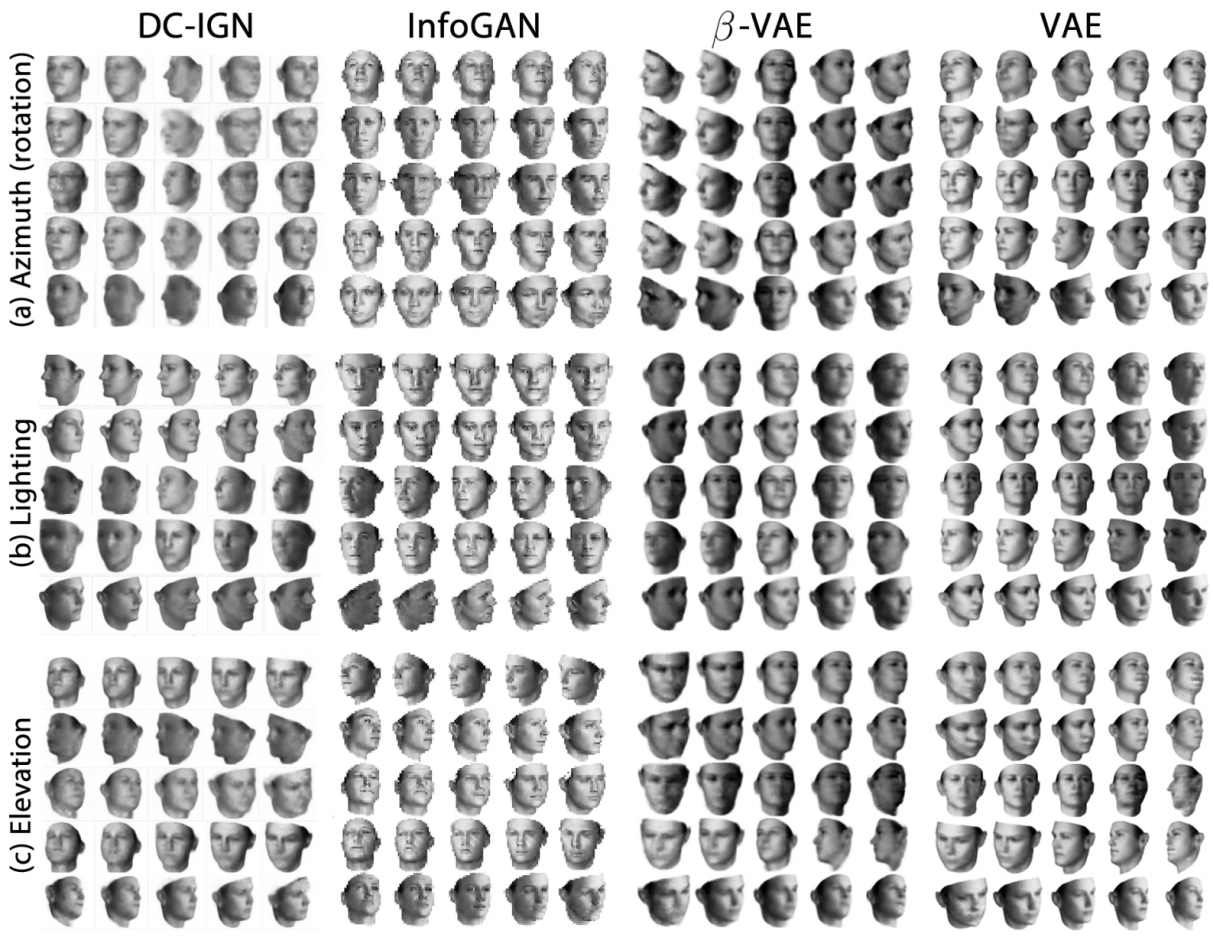

Figure 3: Qualitative comparison of three dimensionality reduction techniques with β-VAE (β-VAE) on the 3D faces dataset. Although all models are relatively good at disentangling (b) lighting and (c) elevation, β-VAE far outperforms the other three in terms of disentangling azimuth.

well-known gradient descent algorithms (e.g. the authors use Adam, RMSProp, and Adagrad for different experiments).

3 Discussion

3.1 Strengths

The authors show that β-VAE is able to outperform other comparable deep methods of nonlinear dimensionality reduction in terms of learning a latent space that aligns with interpretable factors. They compare β-VAE to three other methods qualitatively: DC-IGN, InfoGAN, and a vanilla VAE (see fig. 3).

On its own, β-VAE can lead to some interesting results. For instance, the authors find that the model can learn effective latent representations for (a) skin colour, (b) age/gender, and (c) image saturation on the CelebA dataset (fig. 4).

3.2 Disentanglement and information bottleneck

Burgess et al. [7] take an information-theoretic approach to understanding how β-VAE improves disentanglement in the latent representations $\vec{z}$. Both the prior $p(\vec{z})$ and the posterior $q_\phi(\vec{z}|\vec{x})$ are factorized, as

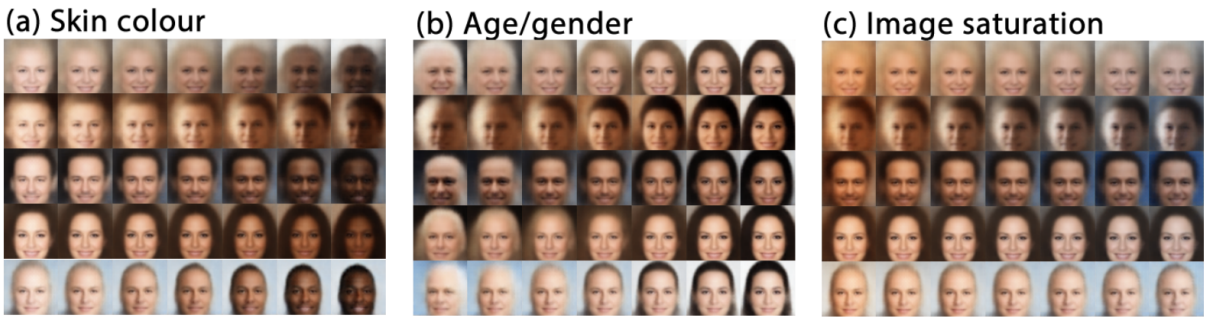

Figure 4: β-VAE latent factors for the CelebA dataset. Each panel shows a traversal of one of the latent dimensions with all others fixed. The model is able to learn clear disentangled representations for (a) skin colour, (b) age/gender, and (c) image saturation.

per the mean-field assumption, so we can conceptualize $q(\vec{z}|\vec{x})$ as a set of noisy channels, each transmitting some information about the input $\vec{x}_i$. The KL-divergence term in the loss, $D_{KL}(q_\phi(\vec{z}|\vec{x})||p(\vec{z}))$ can be seen as a bound on the information contained in q_ϕ. D_{KL} will be 0 only if $\vec{\mu}_i = 0, \Sigma_i = I$, meaning that q_ϕ is simply white noise and contains no useful information about $\vec{x}_i$. As D_{KL} increases, q_ϕ can deviate further from $\mathcal{N}(0, I)$ and contain more information that is particular to $\vec{x}_i$. As can be seen in figure 3.2, when β is increased, the sparsity of the latent encoding is greater, since only two out of the six possible latent dimensions differ from the prior, while the reconstructions are indistinguishable from the vanilla VAE reconstructions.

3.3 Dimensionality reduction perspective

We can also analyze β-VAE more explicitly from the perspective of dimensionality reduction. Just like a vanilla VAE, β-VAE is a nonlinear dimensionality reduction technique. More specifically, because the relationship between the latent probability space $q_\phi(\vec{z}|\vec{x})$ and the data space $p_\theta(\vec{x}|\vec{z})$ is parameterised by neural networks, there is no easily interpretable geometric similarity between the two spaces. This separates (β-)VAE from simple linear dimensionality reduction techniques, such as PCA and CCA, but also from nonlinear manifold or kernel techniques such as LLE, ISOMAP, MDS, and kernel PCA.

However, VAE-type models are still similar to other dimensionality reduction techniques in the sense that the goal is to embed the data in a latent space where the geometry is more comprehensible to a human or simple classifier. Further, β-VAE's augmentation over vanilla VAE is to make the axes of this lower-dimensional space align with human-interpretable generative factors. This contrasts with a method like PCA, where the goal is to find a lower-dimensional space that maximizes variance along each axis, with no regard to the relationship between these low-dimensional axes and the factors that generated the data.

3.4 Weaknesses

Despite the promise of learning a disentangled latent representation of complex image or other high dimensional data, the β-VAE is not without significant weakensses. Most glaring is the arbitrariness of the choice of β. The authors state that if even a small amount of labelled generative factors are available for the data, β can be chosen through a hyperparameter sweep (e.g. a gridsearch or some other hyperparameter optimization algorith). If such labels are not available, then β can be chosen through "visual inspection of what effect the traversal of each single latent unit z_m has on the generated images $(\vec{x}|\vec{z})$ in pixel space" [4]. In other words, β is chosen through trial-and-error. In most cases, few if any labels are available, meaning that the visual inspection technique, along with some "rules of thumb" will be all that is available to the analyst.

Furthermore, there are few guarantees that this technique will work outside of the toy datasets employed in the paper. The major assumption underlying the technique is that the *generative factors exist*. That is,

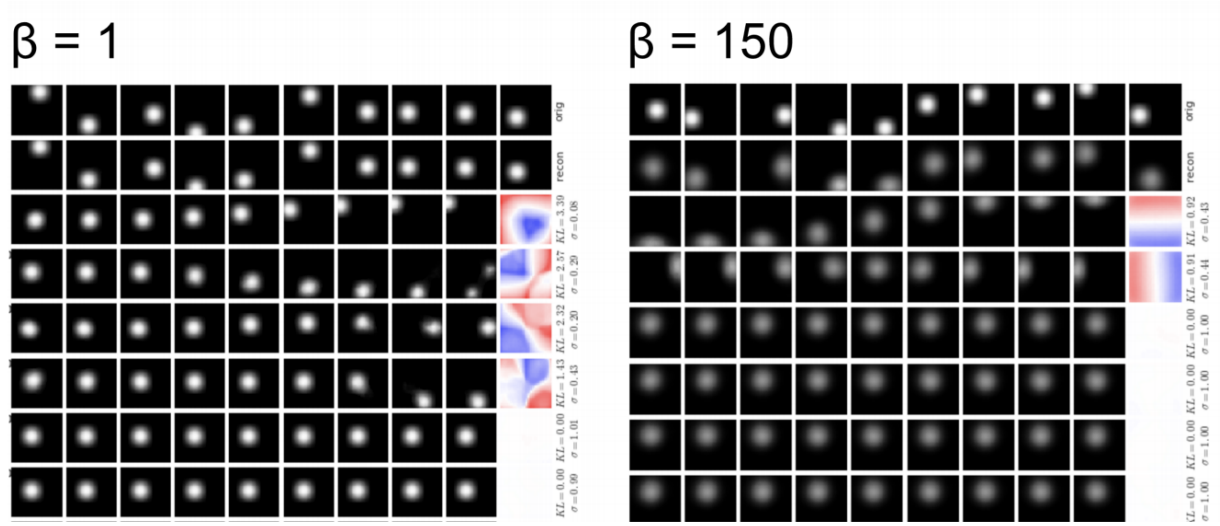

Figure 5: Comparison of vanilla VAE ($\beta = 1$) with strongly disentangled β-VAE ($\beta = 150$). The dataset is a set of Gaussian blobs located in various locations in the cartesian plane (top row of both panels). The second row of each panel is the reconstructions of the original images. The remaining rows are traversals of one of the latent dimensions. The latent dimensions are arranged from bottom to top in order of highest to lowest D_{KL} from the prior standard Gaussian.

we have no reason to believe that there genuinely is a data-generating process that creates each image from a random choice of a small set of discrete or continuous generative factors. This is crucial, since if there is no set of generative factors to be recovered, then no choice of β or careful optimization procedure will allow these phantom factors to be found. As well, even if the factors exist, there is no guarantee that they are in any way human-interpretable. I.e. we may be able to recover some kind of generative factors, but we might not know them when we see them.

4 Experiment

In this section we explore the results of a Python implementation of β-VAE on a toy EMNIST (extension to MNIST) dataset of handwritten digits and characters. The code can be found here: https://github.com/benlevyx/beta-vae.

4.1 Model architecture and dataset

We use the EMNIST dataset, which is an image dataset similar to MNIST, except that it also includes handwritten letters [6]. Each digit is 28×28 pixels with a single grayscale channel. The dataset includes handwritten digits from 0 to 9 and letters from a to z, including lowercase and uppercase. For simplicity, we take a subset of the data comprising the letters A through D (uppercase and lowercase).

The model encoder and decoder both consist of feed-forward neural networks. The 2D images are flattened to a vector of 784, which are then fed into the network in batches of 256 for computational efficiency. Since the goal of this experiment is to be able to plot the low-dimensional representations of the data for different levels of β, we use a latent dimension $d = 2$. We train models with $\beta = 0.0, 1.0, 5.0$ for 15 epochs each using the Adam optimizer with learning rate 10^{-3}.

4.2 Effect of β on latent space

In figure 6, we can see the effect of β on the latent space. The starkest difference is between $\beta = 0$ and the cases where $\beta > 0$. In the former, we see that the latent space is extended along a line such that each point is

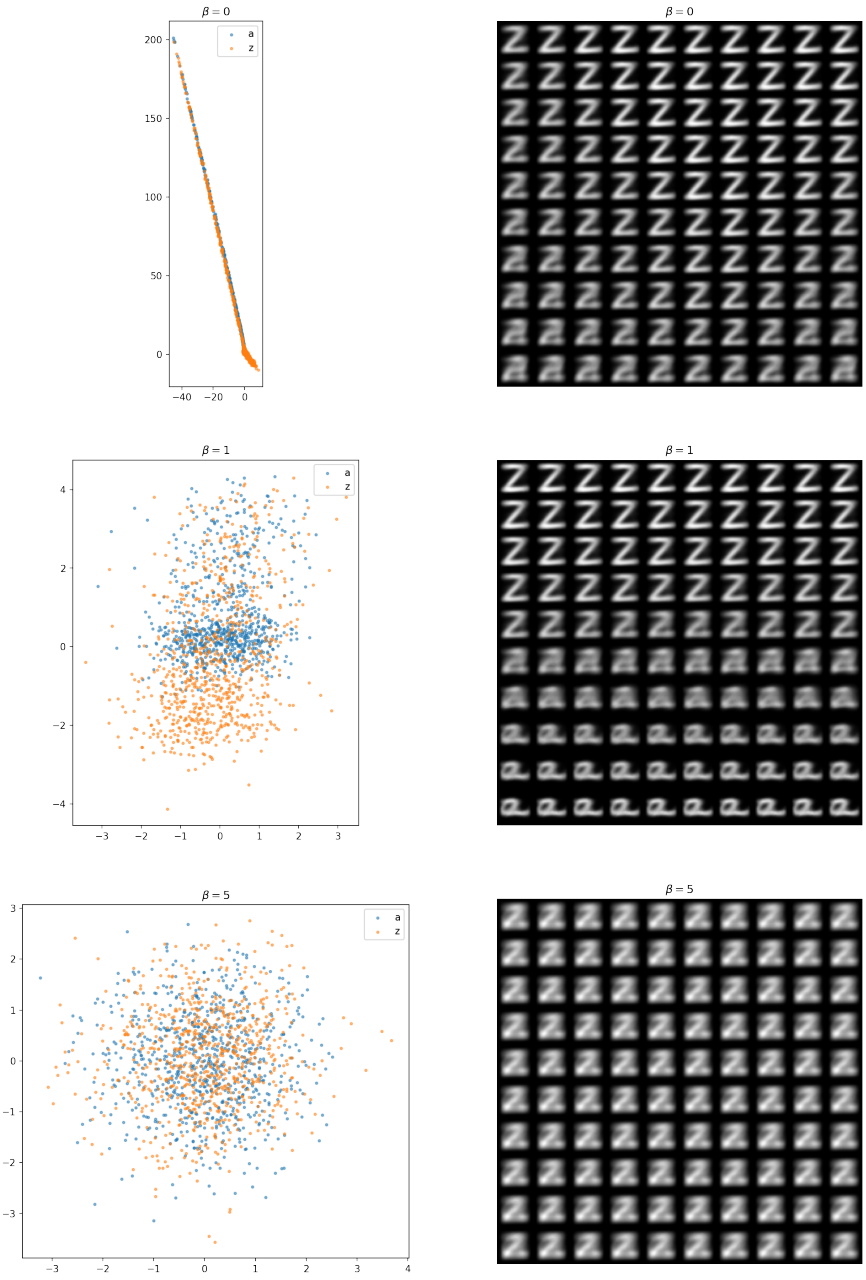

Figure 6: Results from experiment with different levels of β. Each row represents a model trained with a different level of β. The left column plots the 2-dimensional representation of the evaluation data, colour-coded according to whether the point is an 'A' or 'Z'. The right column shows a traversal of the latent space in the fixed range $(-4, 4)$ for both latent dimensions.

essentially assigned its own location in that latent space. This resembles the latent space for a non-Bayesian autoencoder, since the model is only incentivized to reconstruct the data faithfully and is not encouraged to learn a smooth latent space.

As β increases, we see that the latent space becomes closer to a standard Gaussian distribution. This is a double-edged sword: althouh the latent space is smoother and perhaps generalizes better, we completely lose the ability to discriminate between the two classes.

Of course, this is an extreme case meant to illustrate the effect of strong and weak regularization on the geometry of the latent space. In the real world, we would implement a β-VAE with a larger bottleneck dimension (2 is quite extreme), which would also allow for more fine-tuning of β.

References

[1] David Blei. *Variatonal Inference*. Lecture notes for COS597c. 2011. URL: https://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives. ArXiv e-prints". In: *arXiv preprint arXiv:1206.5538* (2012).

[3] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[4] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: (2016).

[5] https://jaan.io/what-is-variational-autoencoder-vae-tutorial/. *Tutorial - What is a variational autoencoder?* 2016.

[6] Gregory Cohen et al. "EMNIST: Extending MNIST to handwritten letters". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2017, pp. 2921–2926.

[7] Christopher P Burgess et al. "Understanding disentangling in β-VAE". In: *arXiv preprint arXiv:1804.03599* (2018).

[8] Lilian Weng. "From Autoencoder to Beta-VAE". In: *lilianweng.github.io/lil-log* (2018). URL: http://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html.