Overview of Variational Autoencoder-based Generative Models

Maximilian Schik

***Abstract*—This work tries to give the reader a general understanding of how a Variational Autoencoder learns its latent features. It also explains how more advanced frameworks build upon the Variational Autoencoder framework and improve the disentanglement of latent features.**

I. Introduction

A generative model deals with the modeling of probability distributions from raw data [5]. After training, this model can be used for generating new data instances. Having accurate models of random processes is advantageous for many fields. It could, for example, be used to enhance a dataset with more samples or for detecting outliers or anomalies. It is also useful for the generation of new data. When generating samples, those with high probability are desired to be more likely than those with lower probability [5]. At the same time, at least a low amount of control about the samples being generated is beneficial. When generating 3d-models of trees for a video game [5], it might be convinient to choose the "style" of tree (conifer or deciduous) or the color of the leaves. Learning an efficient representation of such higher dimensional features is one goal of generative models. Those features should also be disentangled. When adjusting the hair style of a generated face, the gender should not change.

While those models are trained autoassociativ, which means that it tries to map an input to itself, only the decoder part of the models is used as a generator. This is done by using noise sampled from a probability distribution and using it as an input for the decoder. The architecture of an *Autoencoder* (***AE***) was first proposed in [1]. There it is defined as an autoassociative model trained by mean square error. Its intended use was to extract features from input data for use in different machine learning algorithms as a way to deal with the low performing hardware used at that time [1]. Today it is used for many more applications, like noise reduction or anomaly detection. AEs are not really capable of generating meaningful data, but they are the basis for the other models discussed here.

An enhanced version of the AE is the *Vartiational Autoencoder* (***VAE***) as proposed in [7]. It has a similar architecture to an AE, but the VAE differs in the loss function and the way its latent variables are learned. The *Beta Variational Autoencoder* (β-***VAE***) is a further improvement of the VAE proposed by [6]. It introduces a regularization of the latent space and improves the disentanglement of the latent features. !*Beta Total Correlation Variational Autoencoders* (β-***TCVAE***) [4] are another improvement made by decomposing the ELBO used by VAEs and β-VAEs and then analyzing it by using concepts taken from Information Theory. It even further improves the disentanglement of β-VAE and learns an even more efficient representation of inputs in its latent space.

In the next three sections each model will be discussed and the way it learns to represent the given data will be explained. It will also be discussed how each model improves over the previous. After that we will compare the models with each other. The article will end with a conclusion about the featured models.

II. Autoencoder

A. Method

AEs are the simplest of the discussed models. They consist of two parts: the encoder and the decoder. They will be treated as mathematical functions with $f : X \mapsto Z$ as the encoder and $g : Z \mapsto X$ as the decoder. X will be called the data space and Z the latent space. f and g will be learned from data. When implementing an AE, neural networks are used as function approximaters for f and g.

The objective of AEs is the minimization of the loss function:

$$L(\theta_1, \theta_2) := \frac{1}{n} \sum_{i=1}^{n} (x_i - g_{\theta_2}(f_{\theta_1}(x_i)))^2,$$

where θ_1 and θ_2 are the parameters of f and g, n is the number of datapoints and x_i is the ith datapoint. This loss function is also called *mean square error* (***MSE***). During training, f will learn an encoding of $x \in X$ in the latent space $f(x) \in Z$. The decoder will learn to decode a latent vector $z \in Z$ to a vector in the data space $g(z) \in X$. So, a perfect AE would map an input to itself, meaning $x = g(f(x))$. The MSE calculates the difference between the reconstruction $g(f(x))$ and the actual input x.

To prevent either f or g from just learning the identity function $id(x) = x$, an artificial information bottleneck is introduced by making Z lower dimensional than X. Therefore, f must learn a representation (encoding) of x in its latent space. This is also called representation learning [2].

B. Latent Space

For an efficient encoding, the latent space must have specific qualities: *a)* latent vectors close to each other in Z should produce samples close to each other in X (**continuity**) and *b)* every vector $z \in Z$ should produce meaningful data (**completeness**). Using MNIST digits as an example, two fours that look similar to each other should produce $z_1, z_2 \in Z$ that are close to each other. At the same time, a random sample $z \in Z$ should produce a meaningful digit.

May $Z = \mathbb{R}^2$ be taken as an example. When encoding a digit $x \in X$ with $f(x) = z$, it is desired that the features of x be represented by z. For example, the value of the digit ($n \in \{0, 1, ..., 9\}$) could be one encoded feature, and the angle of the number another encoded feature. For an efficient encoding, each feature would be encoded in an own dimension, so they could be changed independently of each other.

C. Disadvantages of Autoencoders

AEs are not used for generative purposes. The lack of structure in their latent space is the reason for that. The training of an AE doesn not necessarily produce a continuous and complete latent space. This leads to close inputs not being close in latent space, and sampled inputs not resulting in any meaningful outputs.

III. Variational Autoencoder

An enhancement that builds up upon the traditional AE is the Variational Autoencoder. It is a directed probabilistic graphical model [7]. It improves upon normal AEs by regulating the latent space and consequently motivates the model to learn a more efficient representation of inputs in the latent space.

A. Method

The assumption a VAE makes is that the data points $x \in X$ are produced by a random process where first, a z is sampled from a $p(z)$, and then a x is sampled from $p(x|z)$. Now, when defining the encoder from a probabilistic point of view, we want it to learn the distribution $p(z|x)$. Similarly, the decoder should learn the distribution $p(x|z)$. Sampling $z \sim p(z|x)$ would be equivalent to encoding an x to z. The same applies for the decoder and $p(x|z)$.

May $q(z|x)$ be an approximation of $p(z|x)$. The *Kullback-Leibler divergence* $\mathcal{D}(q \parallel p) \geq 0$ is a metric for comparing two probability distributions. $\mathcal{D}(q \parallel p) = 0$ would be equivalent to $q = p$. To learn an accurate approximation $p(z|x)$, the KL-divergence

$$\mathcal{D}\left(q(z|x) \parallel p(z|x)\right) = \mathbb{E}_{z\sim q}\left[\log q(z|x) - \log p(z|x)\right] \quad (1)$$

has to be minimized. Minimizing (1) would result in a better approximation $q(z|x)$ of $p(z|x)$, and therefore, a better encoding of x in Z. Rearranging with Bayes rule results in

$$= \mathbb{E}_{z\sim q}\left[\log q(z|x) - \log p(x, z)\right] + \log p(x) \quad (2)$$

where the first term is the *Evidence Lower Bound* (***ELBO***)

$$ELBO = \mathbb{E}_{z\sim q}\left[\log q(z|x) - \log p(x, z)\right] \quad (3)$$

with $p(x, z) = p(x|z)p(z)$. The ELBO is a lower bound for the log probability $\log p(x) \geq ELBO$ [8]. Plugging this into (2) allows to rewrite the equation as

$$\log p(x) = ELBO - \mathcal{D}\left(q(z|x) \parallel p(z|x)\right)$$

showing that minimizing $\mathcal{D}\left(q(z|x) \parallel p(z|x)\right)$ is equivalent to maximizing the ELBO, following from $\log p(x) \geq ELBO$ and $\mathcal{D}(q \parallel p) \geq 0$.

Using this knowledge (3) can be rearranged:

$$\begin{aligned} ELBO &= \mathbb{E}_{z\sim q}\left[\log p(x|z) + \log p(z) - \log q(z|x)\right] \\ &= \mathbb{E}_{z\sim q}\left[\log p(x|z)\right] - \mathcal{D}(q(z|x) \parallel p(z)). \end{aligned} \quad (4)$$

The first term in (4) performs the same role as MSE for the traditional AE. It tries to maximize the probability of a latent vector $z \in Z$ sampled from $q(z|x)$ to be decoded into a $\tilde{x}$ being similar to x. As $p(x|z)$ will be modeled by the decoder $g : Z \mapsto X$, the negative MSE can be used instead:

$$ELBO = \mathbb{E}_{z\sim q}\left[-||x - g(z)||^2\right] - \mathcal{D}(q(z|x) \parallel p(z)).$$

As the real distribution of $p(z)$ is not known, it will be approximated by a normal distribution $p(z) \sim \mathcal{N}(0, 1)$. $q(z|x)$ will be modeled by $\mathcal{N}(\mu(x), \Sigma(x))$, where $\mu(x)$ and $\Sigma(x)$ are two functions mapping an input $x \in X$ to a mean and a covariance in Z. These functions will be implemented as function approximators in the shape of neural networks and will be learned from data.

Plugging in the approximations gives the objective that the model has to maximize:

$$\mathbb{E}_{z\sim q}\left[-||x - g(z)||^2\right] - \mathcal{D}(\mathcal{N}(\mu(x), \Sigma(x)) \parallel \mathcal{N}(0, 1)).$$

Because z is still a random variable that gets sampled from a distribution and sampling is not continuous, gradient descent is not applicable to this objective. This can be taken care of by moving the sampling to the input using the *reparameterization trick*: A random variable ϵ will be sampled from a normal distribution $\mathcal{N}(0, 1)$ and transformed by $h(x, \epsilon) = \mu(x) + \sqrt{\Sigma(x)} \cdot \epsilon$.

Following this the objective can be rewritten as a loss function that the model has to minimize:

$$\begin{gathered} \mathcal{L}(\theta_1, \theta_2, \theta_3) = \\ ||x - g_{\theta_3}(h(x, \epsilon))||^2 + \mathcal{D}(\mathcal{N}(\mu_{\theta_1}(x), \Sigma_{\theta_2}(x)) \parallel \mathcal{N}(0, 1)) \end{gathered}$$

with $\epsilon \sim \mathcal{N}(0, 1)$ and $x \in X$. θ_1, θ_2 and θ_3 are the parameters or weights of the functions μ, Σ and g. The Kullback-Leibler divergence of two normal distributions is trivial to compute.

B. How does a Variational Autoencoder improve upon traditional AEs?

The VAE improves upon the traditional AE primarily in two points: By encoding the input not in a single vector but into a probability distribution, an area of the latent space gets reserved for latent vectors that produce similar x when decoded by the decoder. This addresses the problem of the continuous latent space. At the same time, by having a random aspect in the sampling of z, the decoder learns to deal with the "blurriness" of the latent vector. This helps with the completeness of the latent space, resulting in random samples $z \in Z$ producing more meaningful data. This makes the VAE capable of being used as a generator for meaningful data. This is done by detaching the decoder from the encoder and applying noise sampled from a distribution. When doing this, the encoder is not needed anymore and can be discarded.

C. Problems with Variational Autoencoders

While VAEs allow for the generation of meaningful data, they still lacking structure in the latent space. Features tend to be entangled with each other, showing that the model does not learn the most efficient representation in the latent space. The optimization of the Kullback-Leibler divergence also comes with its drawbacks in the shape of lower reconstruction accuracy.

IV. Beta-Variational Autoencoder

The β-VAE improves the disentanglement of VAEs by adding a single scalar to the objective:

$$\mathbb{E}_{z\sim q}\left[\log p(x|z)\right] - \beta\mathcal{D}(q(z|x) \parallel p(z)).$$

If $\beta = 1$, the model is equivalent to a VAE. Higher values of β enforce stricter regularizations on the latent space and improve the structure in the latent space.

A. How does beta-VAE learn to disentangle the features in the latent space?

For $\beta > 1$, the Kullback-Leibler Divergence is weighted more in the overall objective, resulting in higher pressure on the model to minimize $\mathcal{D}(q(z|x) \parallel p(z))$. This means that the model gets pressured to learn $q(z|x)$ close to $p(z)$. Because we approximate $q(z|x)$ with $\mathcal{N}(\mu(x), \Sigma(x))$ and $p(z)$ with $\mathcal{N}(0, 1)$, this means that $\mu(x)$ will get closer to 0, and $\Sigma(x)$ will get closer to 1. Therefore, the encodings in the latent space will get squeezed together. A consequence of that is the overlapping of the distributions $\mathcal{N}(\mu(x), \Sigma(x))$. This would result in a higher reconstruction error $\mathbb{E}_{z\sim q}\left[\log p(x|z)\right]$, as the sampling of z $q(z|x_1)$, due to its random nature, could result in a z that has a much higher probability under a different x_2. Therefore the decoder would map it with higher probability to x_2 rather than to x_1, even though it was sampled from $q(z|x_1)$ [3]. To reduce the reconstruction error again, the model has to encode $x_1, x_2 \in X$, that are close to each other in data space, close to each other in latent space, therefore reducing the amount of reconstruction error produced by the overlapping. Therefore, inputs that share features or are similiar to each other will be encoded close to each other. Inputs that differ only in one feature should be very close to each other, while inputs that share almost no features will be further apart, therefore separating and disentangling the features in the latent space [3].

B. Improvements of Beta-Variational Autoencoders over Variational Autoencoders

Experiments show β-VAEs performing better at disentangling latent features than normal VAEs. This comes from the more efficient use of the latent space motivated by the higher penality for the Kullback-Leibler Divergence introduced by β, be it at a lower reconstruction accuracy.

V. Beta-Total Correlation Variational Autoencoder

A further refinement of the VAE framework has been introduced by [4] in the shape of the Beta Total Correlation Autoencoder. It improves the disentanglement even further by decomposing the ELBO and scaling the resulting terms.

A. Decomposition and Analysis of the ELBO

The Kullback-Leibler Divergence in the ELBO as seen in (4) can expanded to

$$\begin{aligned}\mathcal{D}(q(z|x) \parallel p(z)) = \mathbb{E}_{z\sim q}\left[\log q(z|x) - \log p(z)\right. \\ \left. + \log q(z) - \log q(z) + \log \prod_j q(z_j) - \log \prod_j q(z_j)\right],\end{aligned}$$

which can be rearranged to

$$\begin{aligned}&= \mathcal{D}(q(z,x) \parallel q(z)p(x)) \\ &\quad + \mathcal{D}(q(z) \parallel \prod_j q(z_j)) \\ &\quad + \sum_j \mathcal{D}(q(z_j) \parallel p(z_j))\end{aligned}$$

$$= I[z;x] + \mathcal{D}(q(z) \parallel \prod_j q(z_j)) + \sum_j \mathcal{D}(q(z_j) \parallel p(z_j))$$

where the first term will be called the *index-code mutual information* (**MI**), the second term is known as the *total correlation* (**TC**), and the third term will be called the *dimension wise Kullback-Leibler Divergence* [4].

The mutual information $I\left[z;x\right]$ is defined as

$$I[z;x] = H(z) - H(z|x) = \mathcal{D}(q(z,x) \parallel q(z)p(x)),$$

where $H(x)$ is the *entropy* of a random variable. The entropy is a measurement of uncertainty about the random variable. So, the mutual information $I[z;x]$ is the amount the uncertainty about z gets reduced by when given an observation about x. If the mutual information is zero, the variables are completely independent of each other.

The total correlation is a generalization of the mutual information for more than one variable. It measures the dependency among a set of random variables, in this case $z_1 \dots z_n$. If the total correlation would be zero, it would mean that all z_j are independent of each other and are therefore completely disentangled. As shown in [4], this term is the main reason why β-VAE is succesful at disentangling the latent features.

The dimension-wise Kullback-Leibler Divergence mainly acts as a restriction on the individual latent dimensions to keep them from deviating too far from the approximated normal distribution for $p(z) \sim \mathcal{N}(0, 1)$.

An improved objective can be obtained by adding weights to the components:

$$\begin{aligned}ELBO = \mathbb{E}_{z\sim q}\left[\log p(x|z)\right] - \alpha I[z;x] \\ -\beta\mathcal{D}(q(z) \parallel \prod_j q(z_j)) \\ -\gamma \sum_j \mathcal{D}(q(z_j) \parallel p(z_j)).\end{aligned}$$

Experiments have shown that $\alpha = \gamma = 1$ and $\beta > 1$ gave the best results [4].

VI. Comparison

blablabla images blablabla disentanglement blablabla β-TCVAE is the best because of regularization blablabla i have to generate the plots blablabla

AE VAE β-VAE β-TCVAE

Fig. 2: Comparison between the discussed models.

VII. Conclusion

References

[1] Dana H. Ballard. Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, page 279–284. AAAI Press, 1987.

[2] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[3] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae, 2018.

[4] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders, 2019.

[5] Carl Doersch. Tutorial on variational autoencoders, 2016.

[6] I. Higgins, Loïc Matthey, A. Pal, C. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[8] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

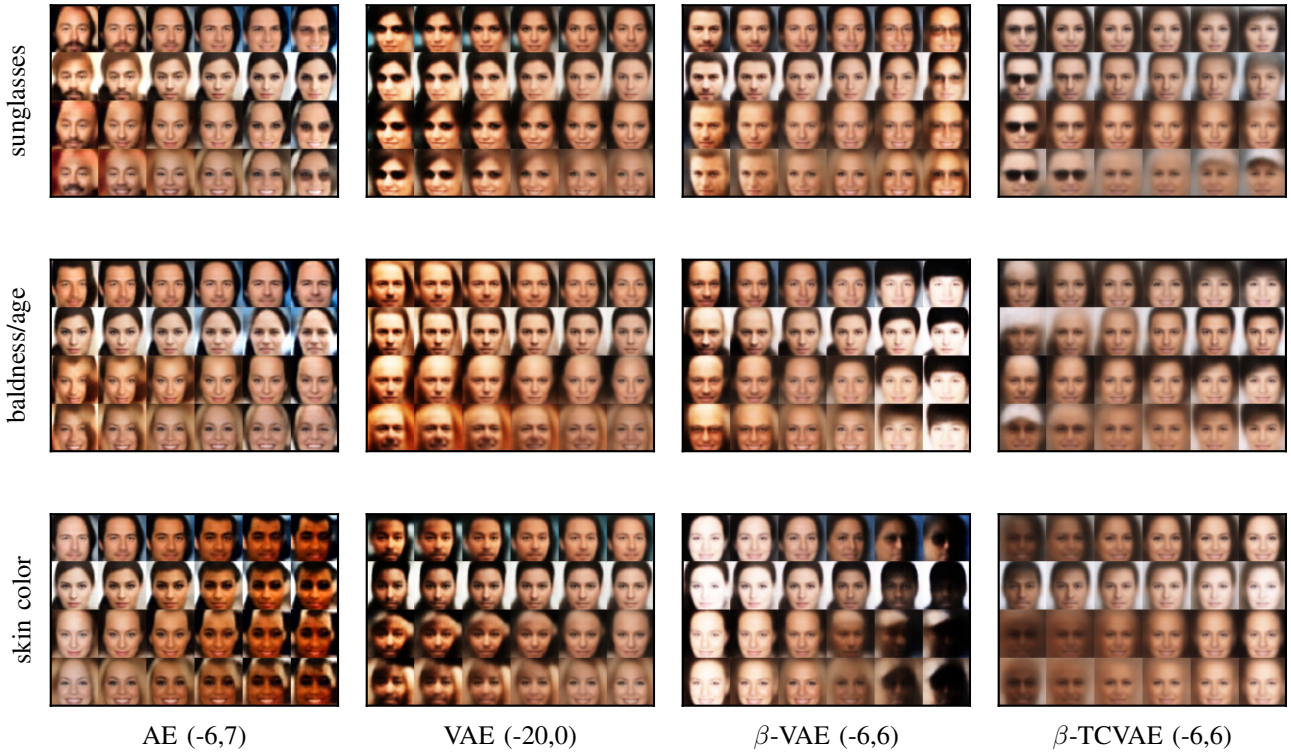

Fig. 1: Comparison between the discussed models.