IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 2, 2020, accepted October 26, 2020, date of publication October 29, 2020, date of current version November 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034828

Balancing Reconstruction Error and Kullback-Leibler Divergence in Variational Autoencoders

ANDREA ASPERTI AND MATTEO TRENTIN
Department of Informatics, Science, and Engineering (DISI), University of Bologna, 40127 Bologna, Italy
Corresponding author: Andrea Asperti (andrea.asperti@unibo.it)

ABSTRACT Likelihood-based generative frameworks are receiving increasing attention in the deep learning community, mostly on account of their strong probabilistic foundation. Among them, Variational Autoencoders (VAEs) are reputed for their fast and tractable sampling and relatively stable training, but if not properly tuned they may easily produce poor generative performances. The loss function of Variational Autoencoders is the sum of two components, with somehow contrasting effects: the *reconstruction loss*, improving the quality of the resulting images, and the *Kullback-Leibler divergence*, acting as a regularizer of the latent space. Correctly balancing these two components is a delicate issue, and one of the major problems of VAEs. Recent techniques address the problem by allowing the network to *learn* the balancing factor during training, according to a suitable loss function. In this article, we show that learning can be replaced by a simple deterministic computation, expressing the balancing factor in terms of a running average of the reconstruction error over the last minibatches. As a result, we keep a *constant* balance between the two components along training: as reconstruction improves, we proportionally decrease KL-divergence in order to prevent its prevalence, that would forbid further improvements of the quality of reconstructions. Our technique is simple and effective: it clarifies the learning objective for the balancing factor, and it produces faster and more accurate behaviours. On typical datasets such as Cifar10 and CelebA, our technique sensibly outperforms all previous VAE architectures with comparable parameter capacity.

INDEX TERMS Generative models, likelilhood-based frameworks, Kullback-Leibler divergence, two-stage generation, variational autoencoders.

I. INTRODUCTION

Generative models address the challenging task of capturing the probabilistic distribution of high-dimensional data, in order to gain insight in their characteristic manifold, and ultimately paving the way to the possibility of synthesizing new data samples.

The main frameworks of generative models that have been investigated so far are Generative Adversarial Networks (GAN) [13] and Variational Autoencoders (VAE) [17], [21], both of which generated an enormous amount of works, addressing variants, theoretical investigations, or practical applications.

The main feature of Variational Autoencoders is that they offer a strongly principled probabilistic approach to generative modeling. The key insight is the idea of addressing the problem of learning representations as a variational inference problem, coupling the generative model $P(X|z)$ for X given the latent variable z, with an inference model $Q(z|X)$ synthesizing the latent representation of the given data.

The loss function of VAEs is composed of two parts: one is just the log-likelihood of the reconstruction, while the second one is a term aimed to enforce a *known* prior distribution $P(z)$ of the latent space - typically a spherical normal distribution. Technically, this is achieved by minimizing the Kullbach-Leibler distance between $Q(z|X)$ and the prior distribution $P(z)$; as a side effect, this will also improve the similarity of the aggregate inference distribution $Q(z) = \mathbb{E}_X Q(z|Z)$ with the desired prior, that is our final objective.

$$\underbrace{\mathbb{E}_{z\sim Q(z|X)} log(P(X|z))}_{log-likelihood} - \lambda \cdot \underbrace{KL(Q(z|X)||P(z))}_{KL-divergence} \tag{1}$$

Loglikelihood and KL-divergence are typically balanced by a suitable λ-parameter (called β in the terminology of

The associate editor coordinating the review of this manuscript and approving it for publication was Firooz B. Saghezchi.

β-VAE [8], [14]), since they have somewhat contrasting effects: the former will try to improve the quality of the reconstruction, neglecting the shape of the latent space; on the other side, KL-divergence is normalizing and smoothing the latent space, possibly at the cost of some additional "overlapping" between latent variables, eventually resulting in a more noisy encoding [1]. If not properly tuned, KL-divergence can also easily induce a sub-optimal use of network capacity, where only a limited number of latent variables are exploited for generation: this is the so called overpruning/variable-collapse/sparsity phenomenon [7], [20], [26].

Tuning down λ typically reduces the number of collapsed variables and improves the quality of *reconstructed* images. However, this may not result in a better quality of *generated samples*, since we loose control on the shape of the latent space, that becomes harder to be exploited by a random generator.

On the other side, tuning up λ may have beneficial effects on the *disentanglement* of the latent representation [11], [19], but typically produces a larger variance loss [4] for reconstructed data, finally resulting in more blurried images.

Several techniques have been considered for the correct calibration of γ, comprising an annealed optimization schedule [6] or a policy enforcing minimum KL contribution from subsets of latent units [16]. Most of these schemes require hand-tuning and, quoting [26], they easily risk to "take away the principled regularization scheme that is built into VAE."

An interesting alternative that has been recently introduced in [9] consists in *learning* the correct value for the balancing parameter during training, that also allows its automatic calibration along the training process. The parameter is called γ, in this context, and it is considered as a normalizing factor for the reconstruction loss.

Measuring the trend of the loss function and of the learned lambda parameter during training, it becomes evident that the parameter is proportional to the reconstruction error, with the result that the relevance of the KL-component inside the whole loss function becomes independent from the current error.

Considering the shape of the loss function, it is easy to give a theoretical justification for this behavior. As a consequence, there is no need for learning, that can be replaced by a simple deterministic computation, eventually resulting in a faster and more accurate behaviour.

The structure of the article is the following. In Section II, we give a quick introduction to Variational Autoencoders, with particular emphasis on generative issues (Section II-A). In Section III, we discuss our approach to the problem of balancing reconstruction error and Kullback-Leibler divergence in the VAE loss function; this is obtained from a simple theoretical investigation of the loss function in [9], and essentially amounts to keeping a *constant* balance between the two components along training. Experimental results are provided in Section IV, relative to standard datasets such as CIFAR-10 (Section IV-A) and CelebA (Section IV-B): up to our knowledge, we get the best generative scores in terms of Frechet Inception Distance ever obtained by means of Variational Autoencoders. In Section V, we try to investigate the reasons why our technique seems to be more effective than previous approaches, by considering the evolution of latent variables along training. Concluding remarks and ideas for future investigations are offered in Section VI.

II. VARIATIONAL AUTOENCODERS

In a generative setting, we are interested to express the probability of a data point X through marginalization over a vector of latent variables:

$$P(X) = \int P(X|z)P(z)dz \approx \mathbb{E}_{z\sim P(z)}P(X|z) \tag{2}$$

For most values of z, $P(X|z)$ is likely to be close to zero, contributing in a negligible way in the estimation of $P(X)$, and hence making this kind of sampling in the latent space practically unfeasible. The variational approach exploits sampling from an auxiliary "inference" distribution $Q(z|X)$, hopefully producing values for z more likely to effectively contribute to the (re)generation of X. The relation between $P(X)$ and $\mathbb{E}_{z\sim Q(z|X)}P(X|z)$ is given by the following equation, where KL denotes the Kullback-Leibler divergence:

$$\begin{aligned} &log(P(X)) - KL(Q(z|X)||P(z|X)) \\ &\qquad = \mathbb{E}_{z\sim Q(z|X)}log(P(X|z) - KL(Q(z|X)||P(z)) \end{aligned} \tag{3}$$

KL-divergence is always positive, so the term on the right provides a lower bound to the loglikelihood $P(X)$, known as Evidence Lower Bound (ELBO).

If $Q(z|X)$ is a reasonable approximation of $P(z|X)$, the quantity $KL(Q(z)||P(z|X))$ is small; in this case the loglikelihood $P(X)$ is close to the Evidence Lower Bound: the learning objective of VAEs is the maximization of the ELBO.

In traditional implementations, we additionally assume that $Q(z|X)$ is normally distributed around an encoding function $\mu_\theta(X)$, with variance $\sigma^2_\theta(X)$; similarly $P(X|z)$ is normally distributed around a decoder function $d_\theta(z)$. The functions μ_θ, σ^2_θ and d_θ are approximated by deep neural networks. Knowing the variance of latent variables allows sampling during training.

Provided the model for the decoder function $d_\theta(z)$ is sufficiently expressive, the shape of the prior distribution $P(z)$ for latent variables can be arbitrary, and for simplicity we may assume it is a normal distribution $P(z) = G(0, 1)$. The term $KL(Q(z|X)||P(z)$ is hence the KL-divergence between two Gaussian distributions $G(\mu_\theta(X), \sigma^2_\theta(X))$ and $G(1, 0)$ which can be computed in closed form:

$$\begin{aligned} &KL(G(\mu_\theta(X), \sigma_\theta(X)), G(0, 1)) \\ &\qquad = \frac{1}{2}(\mu_\theta(X)^2 + \sigma^2_\theta(X) - log(\sigma^2_\theta(X)) - 1) \end{aligned} \tag{4}$$

As for the term $\mathbb{E}_{z\sim Q(z|X)}log(P(X|z)$, under the Gaussian assumption, the logarithm of $P(X|z)$ is just the quadratic distance between X and its reconstruction $d_\theta(z)$; the λ parameter balancing reconstruction error and KL-divergence can be understood in terms of the variance of this Gaussian [10].

The problem of integrating sampling with backpropagation, is solved by the well known reparametrization trick [17], [21].

A. GENERATION OF NEW SAMPLES

The whole point of VAEs is to force the generator to produce a marginal distribution[1] $Q(z) = \mathbb{E}_X Q(z|X)$ close to the prior $P(z)$. If we average the Kullback-Leibler regularizer $KL(Q(z|X)||P(z))$ on all input data, and expand KL-divergence in terms of entropy, we get:

$$\begin{aligned}
&\mathbb{E}_X KL(Q(z|X)||P(z)) \\
&\quad = -\mathbb{E}_X \mathcal{H}(Q(z|X)) + \mathbb{E}_X \mathcal{H}(Q(z|X), P(z)) \\
&\quad = -\mathbb{E}_X \mathcal{H}(Q(z|X)) + \mathbb{E}_X \mathbb{E}_{z\sim Q(z|X)} log P(z) \\
&\quad = -\mathbb{E}_X \mathcal{H}(Q(z|X)) + \mathbb{E}_{z\sim Q(z)} log P(z) \\
&\quad = -\underbrace{\mathbb{E}_X \mathcal{H}(Q(z|X))}_{\substack{\text{Avg. Entropy} \\ \text{of } Q(z|X)}} + \underbrace{\mathcal{H}(Q(z), P(z))}_{\substack{\text{Cross-entropy} \\ \text{of } Q(X) \text{ vs } P(z)}}
\end{aligned} \tag{5}$$

The cross-entropy between two distributions is minimal when they coincide, so we are pushing $Q(z)$ towards $P(z)$. At the same time, we try to augment the entropy of each $Q(z|X)$; under the assumption that $Q(z|X)$ is Gaussian, this amounts to enlarge the variance, further improving the coverage of the latent space, essential for generative sampling (at the cost of more overlapping, and hence more confusion between the encoding of different datapoints).

Since our prior distribution is a Gaussian, we expect $Q(z)$ to be normally distributed too, so the mean μ should be 0 and the variance σ^2 should be 1. If $Q(z|X) = N(\mu(X), \sigma^2(X))$, we may look at $Q(z) = \mathbb{E}_X Q(z|X)$ as a Gaussian Mixture Model (GMM). Then, we expect

$$\mathbb{E}_X \mu(X) = 0 \tag{6}$$

and especially, assuming the previous equation (see [2] for details),

$$\sigma^2 = \mathbb{E}_X \mu(X)^2 + \mathbb{E}_X \sigma^2(X) = 1 \tag{7}$$

This rule, that we call *variance law*, provides a simple sanity check to test if the regularization effect of the KL-divergence is properly working.

The fact that the two first moments of the marginal inference distribution are 0 and 1, does not imply that it should look like a Normal. The possible mismatching between $Q(z)$ and the expected prior $P(z)$ is indeed a problematic aspect of VAEs that, as observed in several works [2], [15], [23] could compromise the whole generative framework. To fix this, some works extend the VAE objective by encouraging the aggregated posterior to match $P(z)$ [24] or by exploiting more complex priors [5], [16], [25].

In [9] (that is the current state of the art), a second VAE is trained to learn an accurate approximation of $Q(z)$; samples from a Normal distribution are first used to generate samples of $Q(z)$, that are then fed to the actual generator of data points. Similarly, in [12], the authors try to give an ex-post estimation of $Q(z)$, e.g. imposing a distribution with a sufficient complexity (they consider a combination of 10 Gaussians, reflecting the ten categories of MNIST and Cifar10).

III. THE BALANCING PROBLEM

As we already observed, the problem of correctly balancing reconstruction error and KL-divergence in the loss function has been the object of several investigations. Most of the approaches were based on empirical evaluation, and often required manual hand-tuning of the relevant parameters. A more theoretical approach has been recently pursued in [9]

The generative loss (GL), to be summed with the KL-divergence, is defined by the following expression (directly borrowed from the public code[2]):

$$GL = \frac{\text{mse}}{2\gamma^2} + log\gamma + \frac{log 2\pi}{2} \tag{8}$$

where mse is the mean square error on the minibatch under consideration and γ is a parameter of the model, learned during training. The previous loss is derived in [9] by a complex analysis of the VAE objective function behavior, assuming the decoder has a gaussian error with variance γ^2, and investigating the case of arbitrarily small but explicitly nonzero values of γ^2.

Since γ has no additional constraints, we can explicitly minimize it in equation 8. The derivative GL' of GL is

$$GL' = -\frac{\text{mse}}{\gamma^3} + \frac{1}{\gamma} \tag{9}$$

having a zero for:

$$\gamma^2 = \text{mse} \tag{10}$$

corresponding to a minimum for equation 8.

This suggests a very simple deterministic policy for *computing* γ instead of *learning* it: just use the current estimation of the mean square error. This can be easily computed as a discounted combination of the mse relative to the current minibatch with the previous approximation: in our implementation, we just take the minimum between these two values, in order to have a monotically decreasing value for γ (we work with minibatches of size 100, that is sufficiently large to provide a reasonable approximation of the real mse). Updating is done at every minibatch of samples.

Compared with the original approach in [9], the resulting technique is both faster and more accurate.

An additional contribution of our approach is to bring some light on the effect of the balancing technique in [9]. Neglecting constant addends, that have no role in the loss function, the total loss function for the VAE is simply:

$$GL = \frac{\text{mse}}{2\gamma^2} + KL \tag{11}$$

So, computing gamma according to the previous estimation of mse has essentially the effect of keeping a *constant*

[1] called by some authors *aggregate posterior distribution* [18].

[2] https://github.com/daib13/TwoStageVAE.

balance between reconstruction error and KL-divergence during the whole training: as mse is decreasing, we normalize it in order to prevent a prevalence of the KL-component, that would forbid further improvements of the quality of reconstructions.

Our approach shares some analogies with [22], where a Generalized ELBO with Constrained Optimization (GECO) is introduced following a similar thinking; however, while they also use a running average of the mse, they target a "desired mse", and more generally a desired performance. They then tune the γ parameter by backpropagating directly through the mse, learning the optimal value of the coefficient as a function of a "tolerance hyper-parameter"; this hyper-parameter is explicitly defined by the user to specify the required performance according to a particular constraint.

While both approaches address the issue of balancing the VAE's objective, our model does so by directly adjusting the "weight" of the reconstruction loss, with no additional hyper-parameters; moreover, as shown in Equation 11, the model is not optimized according to a desired performance, instead balancing the loss function by simply computing γ as the mse changes.

IV. EMPIRICAL EVALUATION

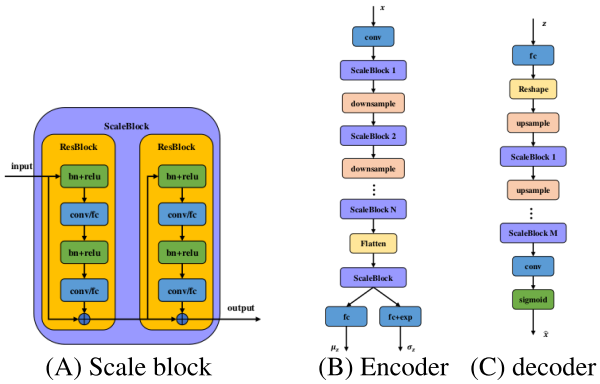
We compared our proposed Two Stage VAE with computed γ against the original model with learned γ using the same network architectures. In particular, we worked with many different variants of the so called ResNet version, schematically described in Figure 1 (pictures are borrowed from [9]). In all our experiments, we used a batch size of 100, and adopted Adam with default TensorFlow's hyperparameters as optimizer. Other hyperparameters, as well as additional architectural details will be described below, where we discuss the cases of Cifar and CelebA separately. The main results are summarized in Table 2 for Cifar and Table 4 for CelebA.

In general, in all our experiments, we observed a high sensibility of Fid scores to the learning rate, and to the deployment of auxiliary regularization techniques. As we shall discuss in Section V, modifying these training configurations may easily result in a different number of inactive[3] latent variables at the end of training. Having both too few or too many active variables may eventually compromise generative sampling, for opposite reasons: few active variables usually compromise reconstruction quality, but an excessive number of active variables makes controlling the shape of the latent space sensbibly harder.

The code is available on GitHub.[4] Checkpoints for Cifar10 and CelebA are available at the project's page.[5]

A. CIFAR10

For Cifar10, we got relatively good results with the basic ResNet architecture with 3 Scale Blocks, a single Resblock



(A) Scale block (B) Encoder (C) decoder

FIGURE 1. "Resnet" architecture. (A) Scale block: a sequence of residual blocks. We mostly worked with a single residual block; two or more blocks makes the architecture sensibly heavier and slower to train, with no remarkable improvement. (B) Encoder: the input is first transformed by a convolutional layer into and then passed to a chain of Scale blocks; after each Scale block, input is downsampled with a a convolutional layer with stride 2 channels are doubled. After *N* Scale blocks, the feature map is flattened to a vector. and then fed to another Scale Block composed by fully connected layers of dimension 512. The output of this Scale Block is used to produce mean and variances of the *k* latent variables. Following [9], $N = 3$ and $k = 64$ for CIFAR-10. For CelebA, we tested many different configurations. (C) Decoder: the latent representation *z* is first passed through a fully connected layer, reshaped to 2D, and then passed through a sequence of deconvolutions halving the number of channels at the same.

for every Scaleblock, and 64 latent variables. We trained our model for 700 epochs on the first VAE and 1400 epochs on the second VAE; the initial learning rate was 0.0001, halving it every 200 epochs on the first VAE and every 100 epochs on the second VAE. Details about the evolution of reconstruction and generative error during training are provided in Figure 2 and Table 1.

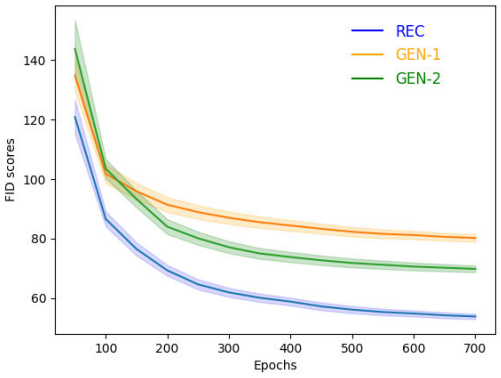


FIGURE 2. Evolution during 700 epochs of training on the CIFAR-10 dataset of the FID scores for reconstructed images (blue), first-stage generated images (orange), and second-stage generated images. The number of epochs refer to the first VAE, and it is doubled for the second VAE. The filled region around the line corresponds to the standard deviation from the expected value. Mean and variances have been estimated over 10 different trainings.

The data refer to ten different but "uniform" trainings ending with the same number of active latent variables, (17 in this case). Few pathological trainings resulting in less or higher sparsity (and worse FID scores) have been removed from the statistic.

[3] for the purposes of this work, we consider a variable *inactive* when $\mathbb{E}_X \sigma^2(X) \geq .8$

[4] https://github.com/asperti/BalancingVAE.git

[5] http://www.cs.unibo.it/~asperti/balancingVAE

TABLE 1. Evolution during training on the CIFAR-10 dataset of several different metrics. REC, GEN-1 and GEN-2 are FID scores relative to reconstructed images (REC), images generated by the first stage VAE (GEN-1) and images generated using the additional second stage (GEN-2); mse is the mean square error, and the variance low has been defined in Section II-A.

epochs	REC	GEN-1	GEN-2	mse	variance law
50	120.9 ± 5.8	134.9 ± 5.9	143.8 ± 9.7	$.0089 \pm .0002$	$1.012 \pm .012$
100	86.6 ± 2.5	101.8 ± 3.1	103.5 ± 3.2	$.0071 \pm .0001$	$.969 \pm .010$
150	76.5 ± 2.3	95.9 ± 2.8	93.3 ± 2.8	$.0062 \pm .0001$	$.939 \pm .011$
200	69.3 ± 2.1	91.4 ± 2.5	84.0 ± 2.5	$.0056 \pm .0001$	$.918 \pm .012$
250	64.6 ± 2.0	88.9 ± 2.3	80.1 ± 2.2	$.0052 \pm .0001$	$.922 \pm .013$
300	61.9 ± 1.8	87.0 ± 2.1	77.1 ± 2.0	$.0049 \pm .0001$	$.925 \pm .012$
350	60.1 ± 1.7	85.5 ± 1.9	75.0 ± 1.8	$.0047 \pm .0001$	$.928 \pm .010$
400	58.8 ± 1.5	84.4 ± 1.8	73.8 ± 1.7	$.0046 \pm .0001$	$.930 \pm .008$
450	57.2 ± 1.3	83.3 ± 1.7	72.7 ± 1.6	$.0045 \pm .0001$	$.937 \pm .007$
500	56.1 ± 1.2	82.3 ± 1.6	71.8 ± 1.5	$.0044 \pm .0001$	$.942 \pm .007$
550	55.3 ± 1.1	81.7 ± 1.5	71.2 ± 1.4	$.0043 \pm .0001$	$.946 \pm .007$
600	54.8 ± 1.1	81.2 ± 1.4	70.6 ± 1.3	$.0043 \pm .0001$	$.950 \pm .006$
650	54.2 ± 1.0	80.7 ± 1.4	70.2 ± 1.2	$.0042 \pm .0001$	$.954 \pm .005$
700	53.8 ± 0.9	80.2 ± 1.3	69.8 ± 1.1	$.0041 \pm .0001$	$.957 \pm .005$

TABLE 2. CIFAR-10: summary of results; see Table 1 for the definition of the metrics.

model	epochs	REC	GEN-1	GEN-2
RAE-l2 [12] (128 vars)	100	$32.24\pm?$	$80.8\pm?$	$74.2\pm?$
2S-VAE, learned γ [9]	1000		76.7 ± 0.8	72.9 ± 0.9
2S-VAE, learned γ, replicated	1000	54.1 ± 0.9	76.8 ± 1.2	73.1 ± 1.2
2S-VAE, computed γ	700	53.8 ± 0.9	80.2 ± 1.3	$\mathbf{6}9.8 \pm 1.1$

In Table 2), we compare our approach with the original version with learned γ [9]. Since some people had problems in replicating the results in [9] (see the discussion on OpenReview[6]), we repeated the experiment (also in order to compute the reconstruction FID). Using the learning configuration suggested by the authors, namely 1000 epochs for the first VAE, 2000 epochs for the second one, initial learning rate equal to 0.0001, halved every 300 and 600 epochs for the two stages, respectively, we obtained results essentially in line with those declared in [9].

For the sake of completeness, we also compare with the FID scores for the recent RAE-l2 model [12] (variance was not provided by authors). In this case, the comparison is purely indicative, since in [12] they work, in the CIFAR-10 case, with a latent space of dimension 128. This also explains their particularly good reconstruction error, and the few training epochs.

B. CelebA

In the case of CelebA, we had more trouble in replicating the results of [9], although we were working with their own code. This was partly due to a mistake on our side (see Appendix), but this pushed us to an extensive investigation of different architectures, and different hyperparameters settings.

In Table 3 we summarize *some* of the results we obtained, over a large variety of different network configurations. The metrics given in the table refer to the following models:

- Model 1: This is our base model, with 4 scale blocks in the first stage, 64 latent variables, and dense layers with inner dimension 4096 in the second stage.
- Model 2: As Model 1 with l2 regularizer added in upsampling and scale layers in the decoder.
- Model 3: Two resblocks for every scale block, l2 regularizer added in downsampling layers in the encoder.
- Model 4: As Model 1 with 128 latent variables, and 3 scale blocks.

All models have been trained with Adam, with an initial learning rate of 0.0001, halved every 48 epochs in the first stage and every 120 epochs in the second stage.

According to the results in Table 3, we can do a few noteworthy observations:

1) for a given model, the technique computing γ systematically outperforms the version learning it, both in reconstruction and generation on both stages;
2) after the first 40 epochs, FID scores (comprising reconstruction FID) do not seem to improve any further, and can even get worse, in spite of the fact that the mean square error keep decreasing; this is in contrast with the intuitive idea that FID REC score should be proportional to mse;
3) the variance law is far from one, that seems to suggest Kl is too weak, in this case; this justifies the mediocre generative scores of the first stage, and the sensible improvement obtained with the second stage;
4) l2-regularization, as advocated in [12], seems indeed to have some beneficial effect.

Similarly to the case of Cifar10, the correct balance between reconstruction error and KL-divergence seems to be crucial to improve the generative performance. As observed above, in the case of CelebA the KL-divergence seems too weak, as clearly testified by the moments of latent variables

[6]https://openreview.net/forum?id=B1e0 × 3C9tQ

TABLE 3. CelebA: effect of the new balancing technique. Using the metrics described in Table 1, we compare the performance during training of different models, just replacing the learned balancing factor γ of [9], with our variant exploiting an explicitly computed γ. The models investigated are the following: (1) base model, with 4 scale blocks in the first stage, 64 latent variables, and dense layers with inner dimension 4096 in the second stage; (2) as model 1 with additional l2 regularization in upsampling and scale layers of the decoder; (3) two resblocks for every scale block, and additional l2 regularization in downsampling layers of the encoder; (4) as model 1 with 128 latent variables, and 3 scale blocks.

		learned γ					computed γ				
Model	epochs	REC	GEN-1	GEN-2	mse	var.law	REC	GEN-1	GEN-2	mse	var.law
1	40/120	53.8	66.0	59.3	.0059	1.024	45.8	56.9	57.1	.0056	0.805
1	80/210	54.3	65.9	59.8	.0049	0.803	46.0	61.1	58.1	.0047	0.688
1	120/300	54.9	66.5	60.4	.0044	0.775	48.1	63.8	59.9	.0043	0.687
2	40/120	48.5	58.2	54.5	.0059	0.985	41.5	58.7	53.7	.0058	1.024
2	80/210	48.8	60.7	55.5	.0048	0.889	42.0	58.8	55.1	.0048	0.877
2	120/300	49.1	62.8	56.9	.0043	0.880	43.0	60.2	56.2	.0043	0.863
3	40/120	59.4	74.3	63.4	.0050	0.893	56.3	74.0	63.9	.0049	0.637
3	80/210	55.6	72.6	62.2	.0039	0.840	54.4	72.3	61.8	.0038	0.621
3	120/300	55.2	72.0	62.1	.0037	0.785	54.4	71.3	62.0	.0036	0.744
4	40/120	52.8	68.2	60.4	.0072	0.789	48.0	65.0	57.7	.0072	0.742
4	80/210	49.4	67.9	58.5	.0060	0.822	45.3	65.0	53.4	.0059	0.785
4	120/300	49.1	68.0	58.4	.0053	0.844	44.4	65.4	54.0	.0053	0.804

TABLE 4. CelebA: summary of results.

model	epochs	REC	GEN-1	GEN-2
RAE-SN [12]	70	36.0±?	44.7±?	40.9±?
2S-VAE, learned γ [9]	120		60.5 ± 0.6	44.4 ± 0.7
2S-VAE, computed γ with latent space norm.	2*70	2*$\mathbf{3}3.9 \pm 0.8$	2*$\mathbf{4}3.6 \pm 1.3$	42.7 ± 1.0 $\mathbf{3}8.6 \pm 1.0$

expressed by the variance law. Actually, in the loss function of [9], both mse and KL-divergence are computed as *reduced sums*, respectively over pixels and latent variables. Now, passing from CIFAR-10 to CelebA, we multiplied the number of pixels by four, passing from 32×32 to 64×64, but kept a constant number of latent variables. So, in order to keep the same balance we used for CIFAR-10, we should multiply the KL-divergence by a factor 4.

Finally, learning seems to proceed quite fast in the case of CelebA, that suggests to work with a lower initial learning rate: 0.00005. We also kept l2 regularization on downsampling and upsampling layers.

With these simple expedients, we were already able to improve on generative scores in [9], (see Table 4), but not with respect to [12].

Analyzing the moments of the distribution of latent variables generated during the second stage, we observed that the actual variance was sensibly below the expected unitary variance (around.85). The simplest solution consists in normalizing the generated latent variables, to meet the expected variance. This point is a bit outside the scope of this contribution, and we refer the interested reader to [4] for further details.

This final precaution caused a sudden burst in the FID score for generated images, permitting to obtain, to the best of our knowledge, the best generative scores ever produced for CelebA with a *variational* approach (for models with comparable parameter capacity).

In Figure 3 we provide examples of randomly generated faces. Note the particularly sharp quality of the images, so unusual for variational approaches.

V. DISCUSSION

The reason why the balancing policy between reconstruction error and KL-regularization addressed in [9] and revisited in this article is so effective seems to rely on its laziness in the choice of the latent representation.

A Variational Autoencoder computes, for each latent variable z and each sample X, an expected value $\mu_z(X)$ and a variance $\sigma_z^2(X)$ around it. During training, the variance $\sigma_z^2(X)$ usually drops very fast to values close to 0, reflecting the fact that the network is highly confident in its choice of $\mu_z(X)$. The KL-component in the loss function can be understood as a mechanism aimed to reduce this confidence, by forcing a not negligible variance. By effect of the KL-regularization, some latent variables may be even neglected by the VAE, inducing *sparsity* in the resulting encoding [3]. The "collapsed" variables have, for any X, a value of $\mu_z(X)$ close to 0 and a mean variance $\sigma_z^2(X)$ close 1. So, typically, at a relatively early stage of training, the mean variance $\mathbb{E}_X\sigma_z^2(X)$ of each latent variable z gets either close to 0, if the variable is exploited, of close to 1 if the variable is neglected (see Figure 4).

Traditional balancing policies addressed in the literature start with a low value for the KL-regularization, increasing it during training. The general idea is to start privileging the quality of reconstruction, and then try to induce a better coverage of the latent space. Unfortunately, this reshaping ex post of the latent space looks hard to achieve, in practice.

The balancing property discussed in this article does the opposite: it starts attributing a relatively high importance to KL-divergence, to balance the high initial reconstruction error, progressively reducing its relevance in a way proportional to the improvement of the reconstruction. In this way,

FIGURE 3. Examples of generated faces. The resulting images do not show the blurred appearance so typical of variational approaches, sensibly improving their perceptive quality.

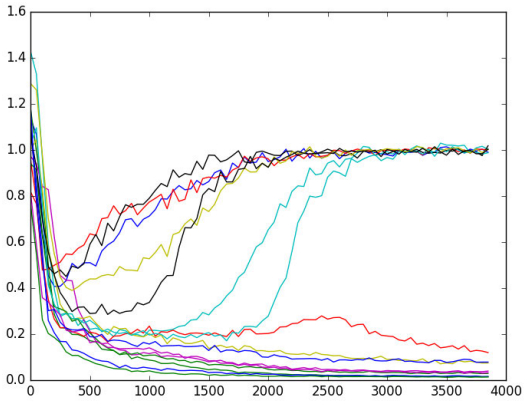


FIGURE 4. Typical evolution of the mean variance $\mathbb{E}_X \sigma_z^2(X)$ of latent variables during training in a variational autoencoder. Relevant variables have a variance close to 0, while inactive variables have a variance going to 1. The picture was borrowed from [3] and is relative to the first epoch of training for a dense VAE over the MNIST data set.

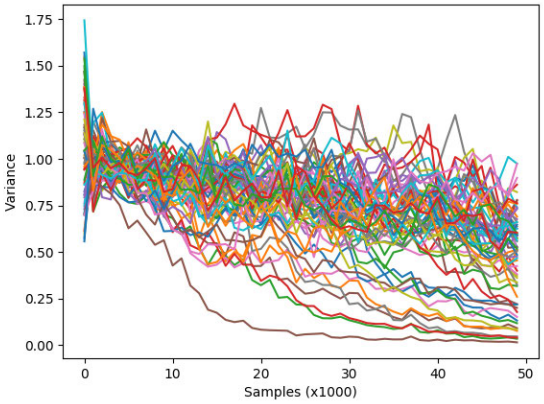


FIGURE 5. Evolution of the mean variance of the 64 latent variables during the first epoch of training on Cifar10. Due to the "lazy" balancing technique, even after a full epoch, the destiny of most latent variables is still uncertain: they could collapse or be exploited for reconstruction.

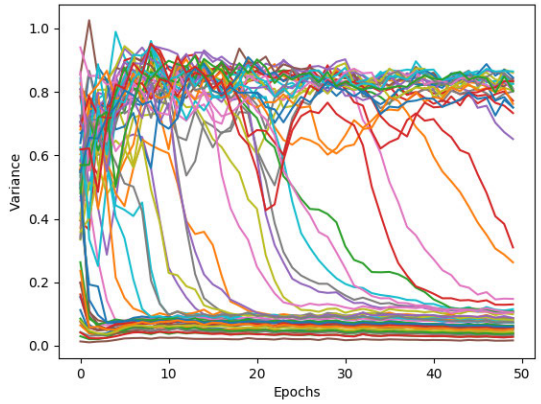


FIGURE 6. Evolution of the mean variance of the 64 latent variables first 50 epochs of training on Cifar10. One by one, latent variables are retrieved from the limbo (variance around 0.8), and put to work by the autoencoder.

TABLE 5. Effect of the learning rate on sparsity and different metrics. A high learning rate reduces sparsity and improves on reconstruction. However, this does not result in a better generative score. With a low rate, too many variables remains inactive.

l.r.	inact.	REC	GEN-1	GEN-2	mse
.00020	13	53.0	80.6	74.5	.0039
.00015	15	53.3	79.9	71.8	.0040
.00010	17	53.8	80.2	68.8	.0041
.00005	19	58.2	83.2	75.8	.0047

the relative importance between the two components of the loss function remains constant during training.

The practical effect is that latent variables are kept for a long time in a sort of limbo from which, one at a time, they are retrieved and put to work by the autoencoder, as soon as it realizes how they can contribute to the reconstruction.

The previous behaviour is evident by looking at the evolution of the mean variance $\mathbb{E}_X \sigma_z^2(X)$ of latent variables during training (not to be confused with the variance of the mean values $\mu_z(X)$, that according to the variance law should approximately be the complement to 1 of the former).

In Figure 5 we see the evolution of the variance of the 64 latent variables during the first epoch of training on the Cifar10 data set: even after a full epoch, the "status" of most latent variables is still uncertain.

During the next 50 epochs, in a very slow process, some of the "dormient" latent variables are woken up by the autoencoder, causing their mean variance to move towards 0: see Figure 6.

With the progress of training, less and less variables change their status, until the process finally stabilizes.

It would be nice to think, as hinted to in [9], that the number of active latent variables at the end of training corresponds to the *actual dimensionality of the data manifold*. Unfortunately, this number still depends on too many external factors to justify such a claim. For instance, a mere modification of the learning rate is sensibly affecting the sparsity of the resulting latent space, as shown in Table 5 where we compare, for different initial learning rates (l.r.), the final number of inactive variables, FID scores, and mean square error.

Specifically, a high learning rate appears to be in conflict with the lazy way we would like latent variables to be chosen for activation; this typically results in less sparsity, that is not always beneficial for generative purposes. The annoying point is that with respect to the dimensionality of the latent space with the best generative FID, activating more variables can result in a lower reconstruction error, that should not be the case if we correctly identified the datafold dimensionality.

So, while the balancing strategy discussed in this article (similarly to the one in [9]) is eventually beneficial, still could take advantage of some tuning.

VI. CONCLUSION

In this article, we stressed the importance of keeping a constant balance between reconstruction error and Kullback-Leibler divergence during training of Variational Autoencoders. We did so by normalizing the reconstruction error by an estimation of its current value, computed as a running average over minibatches. We developed the technique by an investigation of the loss function used in [9], where the balancing parameter was instead learned during training. Our technique seems to outperform all previous *variational* approaches, permitting us to obtain - over traditional datasets such as CIFAR-10 and CelebA - unprecedented FID scores for this class of generative models with comparable model capacity.

In spite of its relevance, the politics of keeping a constant balance does not seem to entirely solve the balancing issue, that still seems to depend from many additional factors, such as the network architecture, the complexity and resolution of the dataset, or training parameters such as the learning rate.

Also, the regularization effect of the KL-component must be better understood, since it frequently fails to induce the expected distribution of latent variables, possibly requiring and justifying ex-post adjustments.

Most of the ideas and results contained in this article are to be credited to the first author. The second author mainly contributed on the experimental side.

APPENDIX
IMPACT OF THE RESIZING MODE ON THE MODEL PERFORMANCE

There is a significant discrepancy between our observations in Table 3 and the results claimed in [9].

Investigating this phenomenon, we inspected the elements of the dataset with worse reconstruction errors and remarked a particularly bad quality of some of the images, resulting from the resizing of the face crop of dimension 128×128 to the canonical dimension 64×64 expected from the neural network. The resizing function used in the source code of [9] available at was the deprecated `imresize` function of the scipy library.[7] Following the suggestion in the documentation, we replaced the call to imresize with a call to

[7]scipy imresize: https://docs.scipy.org/doc/scipy-1.2.1/reference/generated/scipy.misc.imresize.html

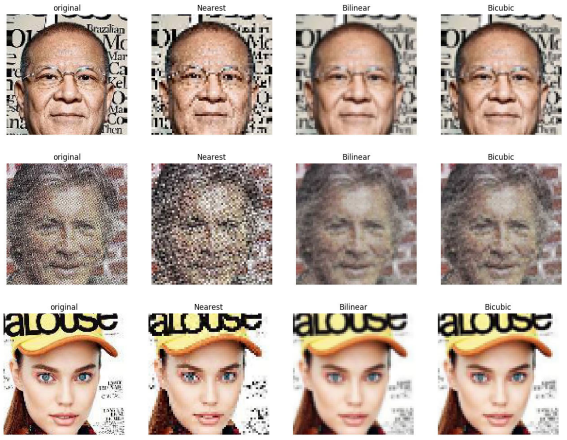


FIGURE 7. Effect of resizing mode on a few CelebA samples. Nearest neighbours produces bad staircase effects; bilinear, that is the common choice, is particularly smooth, suiting well to VAEs; bicubic is sligtly sharper. According to our experience, resizing the dataset with bilinear or bicubic interpolation makes little difference in terms of generative FID.

PILLOW: `numpy.array(Image.fromarray(arr).resize())` Unfortunately, and surprisingly, the default resizing mode of PILLOW is Nearest Neighbours that, as described in Figure 7, introduces annoying jaggies that sensibly deteriorate the quality of images. This probably also explains the anomalous behaviour of FID REC with respect to mean squared error. The Variational Autoencoder fails to reconstruct images with high frequency jaggies, while keep improving on smoother images. This can be experimentally confirmed by the fact that while the minimum mse keeps decreasing during training, the maximum, after a while, stabilizes. So, in spite of the fact that the average mse decreases, the overall distribution of reconstructed images may remain far from the distribution of real images, and possibly get even more distant.

Resizing images with the traditional bilinear interpolation produces a substantial improvement, but not sufficient to obtain the generative scores claimed in [9].

REFERENCES

[1] A. Alexander Alemi, B. Poole, I. Fischer, V. Joshua Dillon, A. Rif Saurous, and A. K. Murphy, "Fixing a broken ELBO," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018, pp. 159–168.

[2] A. Asperti, "About generative aspects of variational autoencoders," in *Proc. Int. Conf. Mach. Learn., Optim., Data Sci.*, Siena, Italy, Sep. 2019, pp. 71–82.

[3] A. Asperti, "Sparsity in variational autoencoders," in *Proc. 1st Int. Conf. Adv. Signal Process. Artif. Intell. (ASPAI)*, Barcelona, Spain, Mar. 2019, pp. 1–9.

[4] A. Asperti, "Variance loss in variational autoencoders," in *Proc. Int. Conf. Mach. Learn., Optim., Data Sci.* Berlin, Germany: Springer, Jul. 2020, pp. 1–12.

[5] M. Bauer and A. Mnih, "Resampled priors for variational autoencoders," in *Proc. 22nd Int. Conf. Artif. Intell. Statist. (AISTATS)*, in Proceedings of Machine Learning Research, vol. 89, K. Chaudhuri and M. Sugiyama, Eds. Naha, Japan: PMLR, Apr. 2019, pp. 66–75. [Online]. Available: http://proceedings.mlr.press/v89/bauer19a.html

[6] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *CoRR*, vol. abs/1511.06349, pp. 1–12, Nov. 2015.

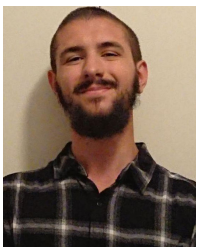
[7] Y. Burda, B. Roger Grosse, and R. Salakhutdinov, "Importance weighted autoencoders," *CoRR*, abs/1509.00519, pp. 1–14, Sep. 2015.

[8] P. Christopher Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-VAE," 2018, *arXiv:1804.03599*. [Online]. Available: https://arxiv.org/abs/1804.03599
[9] B. Dai and P. David Wipf, "Diagnosing and enhancing VAE models," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–44.
[10] C. Doersch, "Tutorial on variational autoencoders," *CoRR*, vol. abs/1606.05908, pp. 1–23, Jun. 2016.
[11] B. Esmaeili, H. Wu, S. Jain, A. Bozkurt, N. Siddharth, B. Paige, D. H. Brooks, J. Dy, and J.-W. Meent, "Structured disentangled representations," in *Proc. 22nd Int. Conf. Artif. Intell. Statist. (AISTATS)*, K. Chaudhuri and M. Sugiyama, Eds., Naha, Japan, vol. 89, 2019, pp. 2525–2534.
[12] P. Ghosh, S. M. M. Sajjadi, A. Vergari, J. M. Black, and A. Schölkopf, "From variational to deterministic autoencoders," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–25.
[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst. 27th, Annu. Conf. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Dec. 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets
[14] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. ICLR*, 2017.
[15] D. Matthew Hoffman and J. Matthew Johnson, "Elbo surgery: Yet another way to carve up the variational evidence lower bound," in *Proc. Workshop Adv. Approx. Bayesian Inference (NIPS)*, vol. 1, 2016, p. 2.
[16] D. Kingma, T. Salimans, R. Josefowicz, X. Chen, I. Sutskever, and M. Welling, "Improving variational autoencoders with inverse autoregressive flow," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 4736–4744.
[17] P. Diederik Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, Banff, AB, Canada, Apr. 2014, pp. 1–14.
[18] A. Makhzani, J. Shlens, N. Jaitly, and J. Ian Goodfellow, "Adversarial autoencoders," *CoRR*, vol. abs/1511.05644, pp. 1–16, Nov. 2015.
[19] E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh, "Disentangling disentanglement in variational autoencoders," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, K. Chaudhuri and R. Salakhutdinov, Ed., Long Beach, CA, USA, vol. 97, 2019, pp 4402–4412.
[20] A. Razavi, A. V. D. Oord, B. Poole, and O. Vinyals, "Preventing posterior collapse with delta-VAEs," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–24.
[21] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. 31th Int. Conf. Mach. Learn. (ICML)*, Beijing, China, vol. 32, Jun. 2014, pp. 1278–1286.
[22] D. J. Rezende and F. Viola, "Taming VAEs," *CoRR*, abs/1810.00597, pp. 1–21, Oct. 2018.
[23] M. Rosca, B. Lakshminarayanan, and S. Mohamed, "Distribution matching in variational inference," 2018, *arXiv:1802.06847*. [Online]. Available: https://arxiv.org/abs/1802.06847
[24] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf, "Wasserstein auto-encoders," *CoRR*, vol. abs/1711.01558, pp. 1–20, Nov. 2017.
[25] M. J. Tomczak and M. Welling, "VAE with a VampPrior," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, Canary Islands, Spain, Apr. 2018, pp. 1214–1223.
[26] S. Yeung, A. Kannan, Y. Dauphin, and L. Fei-Fei, "Tackling over-pruning in variational autoencoders," *CoRR*, vol. abs/1706.03643, pp. 1–11, Apr. 2017.

ANDREA ASPERTI was born in Bergamo, Italy, in 1961. He received the Ph.D. degree in computer science from the University of Pisa, in 1989.

He has been the Head of the Department of Computer Science from 2005 to 2007. He was responsible for several national and international projects. He is currently a Full Professor with the University of Bologna, where he teaches courses in machine learning and deep learning. He has authored three books and published a number of scientific publications in international peer-reviewed conferences and journals. His recent research interests include deep learning and deep reinforcement learning.

Dr. Asperti has acted as a member of the Advisory Committee of the World Wide Web Consortium from 2000 to 2007.



MATTEO TRENTIN was born in Bentivoglio, Italy, in 1997. He received the B.S. degree in computer science from the University of Bologna in 2019, where he is currently pursuing the master's degree.

His research interests include artificial intelligence and deep learning.