IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 May 2022, accepted 15 June 2022, date of publication 23 June 2022, date of current version 29 June 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3185654*

# Enhancing Variational Generation Through Self-Decomposition

**ANDREA ASPERTI, LAURA BUGO, AND DANIELE FILIPPINI**
Department of Informatics: Science and Engineering (DISI), University of Bologna, 40126 Bologna, Italy
Corresponding author: Andrea Asperti (andrea.asperti@unibo.it)

**ABSTRACT** In this article we introduce the notion of Split Variational Autoencoder (SVAE), whose output $\hat{x}$ is obtained as a weighted sum $\sigma \odot \hat{x_1} + (1 - \sigma) \odot \hat{x_2}$ of two generated images $\hat{x_1}, \hat{x_2}$, and $\sigma$ is a *learned* compositional map. The composing images $\hat{x_1}, \hat{x_2}$, as well as the $\sigma$-map are automatically synthesized by the model. The network is trained as a usual Variational Autoencoder with a negative loglikelihood loss between training and reconstructed images. No additional loss is required for $\hat{x_1}, \hat{x_2}$ or $\sigma$, neither any form of human tuning. The decomposition is nondeterministic, but follows two main schemes, that we may roughly categorize as either "syntactic" or "semantic." In the first case, the map tends to exploit the strong correlation between adjacent pixels, splitting the image in two complementary high frequency sub-images. In the second case, the map typically focuses on the contours of objects, splitting the image in interesting variations of its content, with more marked and distinctive features. In this case, according to empirical observations, the Fréchet Inception Distance (FID) of $\hat{x_1}$ and $\hat{x_2}$ is usually lower (hence better) than that of $\hat{x}$, that clearly suffers from being the average of the former. In a sense, a SVAE forces the Variational Autoencoder to *make choices*, in contrast with its intrinsic tendency to *average* between alternatives with the aim to minimize the reconstruction loss towards a specific sample. According to the FID metric, our technique, tested on typical datasets such as Mnist, Cifar10 and CelebA, allows us to outperform all previous purely variational architectures (not relying on normalization flows).

**INDEX TERMS** Deep learning, generative modeling, multi-layer neural networks, representation learning, unsupervised learning, variational autoencoder.

## I. INTRODUCTION

Generative modeling (see e.g. [27] for an introduction) is one of the most fascinating problems in Artificial Intelligence, with many relevant applications in different areas comprising computer vision, natural language processing, medicine or reinforcement learning. The goal is not only to be able to sample new realistic examples starting from a given set of data, but to gain insight in the data manifold, and the way a neural network is able to extract and exploit the characteristic features of data. In case of high-dimensional data, the generative problem can only be addressed by means of Deep Neural Networks, and this topic led to tremendous research in many different directions, particularly nourishing the recent field of unsupervised representation learning.

Among the different kind of generative models which have been investigated, Variational Autoencoders (VAEs) [18], [26] have always exerted a particular fascination [29], [34], [35], mostly due to their strong theoretical foundations, that will be briefly recalled in Section II. Unfortunately, results remained below expectations, and the generative quality of VAEs is systematically outperformed by different generative techniques like e.g. Generative Adversarial Networks.

A particularly annoying problem is that VAEs produce images with a characteristic blurriness, very hard to be removed with traditional techniques [20], [28]. The source of the problem is not easy to identify, but it is likely due to *averaging*, implicitly underlying the VAE frameworks and, more generally, any autoencoder approach. As observed in [11]. In presence of multimodal output, a loglikelihood objective typically results in averaging and hence blurriness. A GAN does not have this problem, since its goal is to fool the Discriminator, not to reconstruct a given input. Since Variational Autoencoders are intrinsically multimodal, both due to dimensionality reduction, and to the sampling process during training, a certain amount of blurriness is unfortunately expected.

The associate editor coordinating the review of this manuscript and approving it for publication was Vivek Kumar Sehgal.

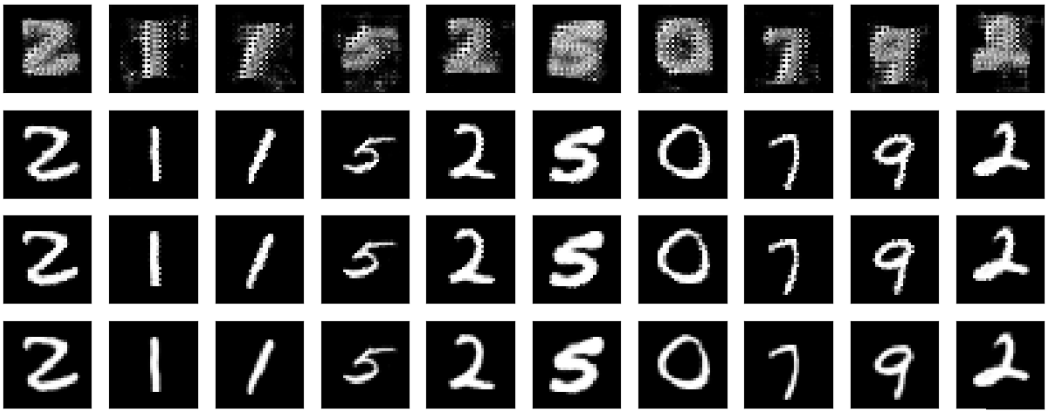

**FIGURE 1.** Example of "syntactic" decomposition for Mnist. In the first line we have the $\sigma$ map, then, in order, $\hat{x_1}$, $\hat{x_2}$, and finally $\hat{x} = \sigma \odot \hat{x_1} + (1-\sigma) \odot \hat{x_2}$ (similarly for the other analogous pictures). In this case, the image is decomposed in complementary subimages at high-frequency. This usually helps to decorrelate adjacent pixels in the latent encoding.

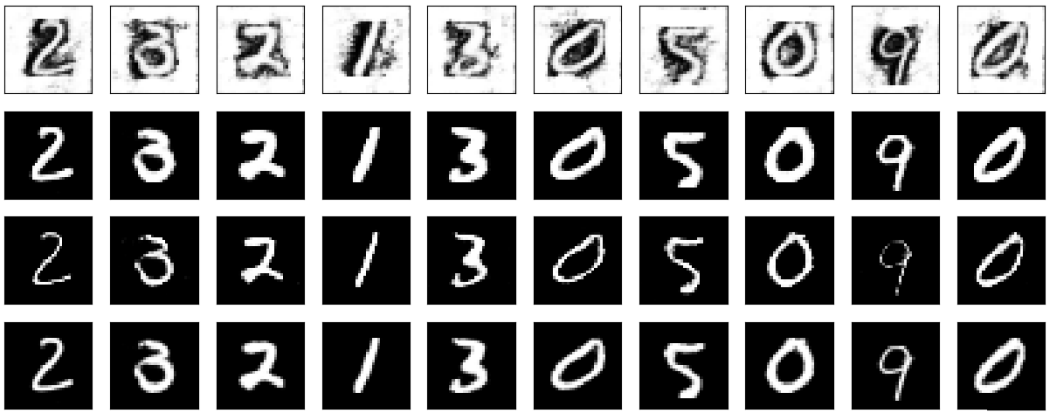

**FIGURE 2.** Example of "semantic" decomposition for Mnist. Digits are usually decomposed in a "fat" and a "thin" version following the contours of objects. The compound image $\hat{x} = \sigma \odot \hat{x_1} + (1-\sigma) \odot \hat{x_2}$ is particularly neat and mushy.

Starting from the averaging assumption [2], it is natural to try to address blurriness by offering to the Variational Autoencoder the possibility to create multiple images, and then synthesize a result as a (learned) weighted combination of them. This is precisely what our Split Variational Autoencoder (SVAE) is supposed to do: the generator returns two images $\hat{x_1}, \hat{x_2}$ and a probability map $\sigma$ with the same spatial dimension of the images, and synthesize a resulting image $\hat{x} = \sigma \odot \hat{x_1} + (1-\sigma) \odot \hat{x_2}$ where $\odot$ is point-wise multiplication (broadcasted over channels). The Autoencoder is trained by minimizing the reconstruction loss between $x$ and $\hat{x}$, together with the traditional regularization component over latent variables. No additional loss is imposed over $\hat{x_1}$, $\hat{x_2}$ or $\sigma$.

The resulting decomposition is non deterministic, mostly depending on the network architecture and the dimension of the latent space. However, it seems to follow two main schemes, that we call "syntactic" (see Figures 1, 3), and "semantic" (see Figures 2, 4). In this Figures, the top line is the $\sigma$ map, the second line is $\hat{x_1}$, the third line is $\hat{x_2}$, and in the last line we have $\hat{x} = \sigma \odot \hat{x_1} + (1-\sigma) \odot \hat{x_2}$. Let us also remark that all images in the pictures have been *generated*, not reconstructed.

In the first case, the map takes advantage of the strong correlation between adjacent pixels splitting the image in two complementary high frequency sub-images. Each image has more freedom in filling the ignored parts, easing the generative task.

In the second, even more interesting case, the map focuses on the *contours* of objects, splitting the image in interesting variations around them, typically resulting in more marked and distinctive features. In this case, the Fréchet Inception Distance (FID) of $\hat{x_1}$ and $\hat{x_2}$ may also be lower (hence better) than that of $\hat{x}$, that apparently suffers from being the average of the former.

An interesting aspect of SVAEs, and possibly one of reasons behind their effectiveness, is that they allow to work with a number of latent variables sensibly higher than usual, hence implicitly addressing the variable collapse phenomenon [1], [6], [25], [31], [36]. This seems to be an indication that

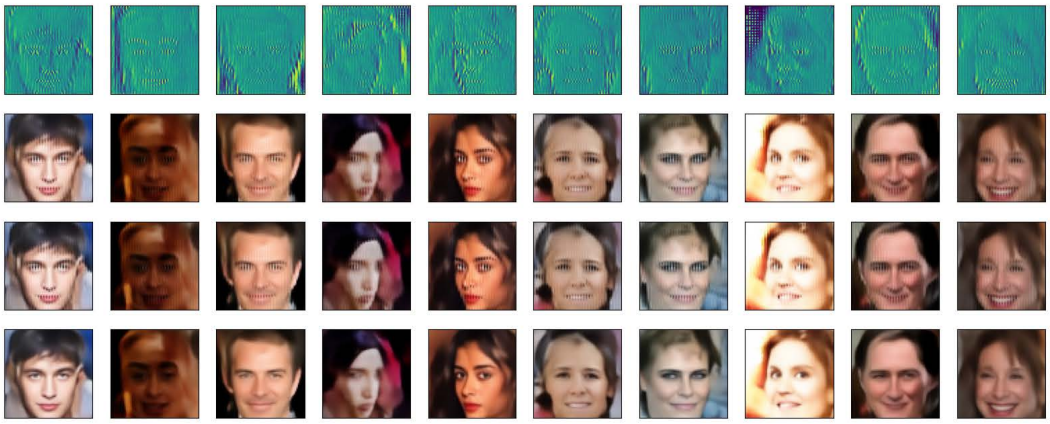

**FIGURE 3.** Example of "syntactic" decomposition for CelebA. In this case, the FID score for $\hat{x_1}$ and $\hat{x_2}$ is usually bad. Still, the decomposition helps to get a stable and robust training, typically resulting in good generative results for the compound image $\hat{x} = \sigma \odot \hat{x_1} + (1 - \sigma) \odot \hat{x_2}$.

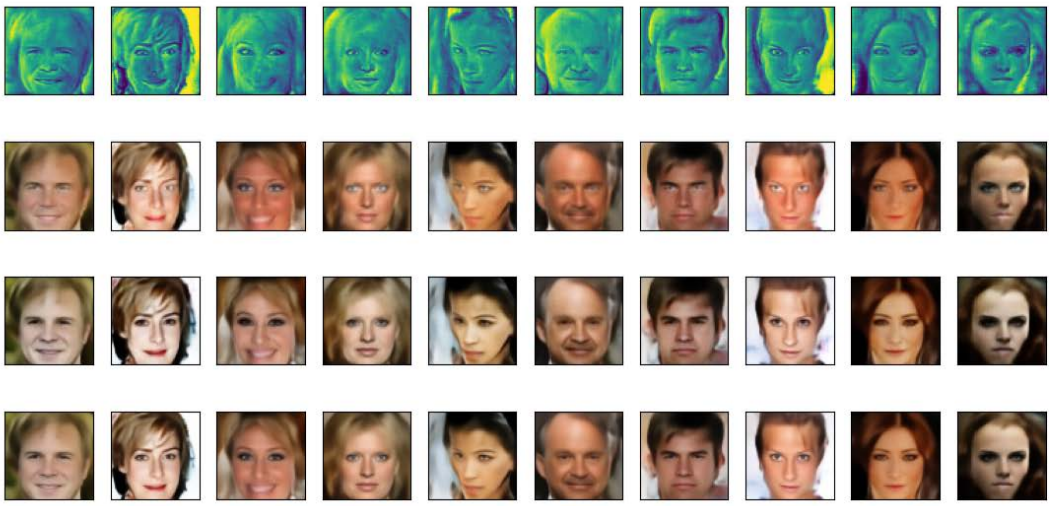

**FIGURE 4.** Example of "semantic" decomposition for CelebA. This is the most interesting case. The map focuses on contours of objects, emphasizing them in opposite directions. This is frequently rewarding in terms of FID score for $\hat{x_1}$ and $\hat{x_2}$, that are usually better than that of $\hat{x}$.

self-splitting is indeed a convenient way to induce the model to synthesize a large number of uncorrelated latent features.

We tested SVAE on typical datasets such as Mnist, Cifar10 and CelebA, and in all cases we observed substantial improvements w.r.t. the "vanilla" approach. Excluding models that make use of sophisticated techniques like normalizing flows [16], [23], [32] (typically requiring thousands of latent variables, and practically hindering a fruitful exploration of the latent space), SVAE outperforms all previous variational architectures.

The code relative to this work can be accessed on Github in the following public repository: https://github.com/asperti/Split-VAE Pretrained weights for the models discussed in the article are available at the following page: https://www.cs.unibo.it/~asperti/SVAE.html.

### A. STRUCTURE OF THE ARTICLE

In Section II we briefly recall the theory behind Variational Autoencoder, show their encoder-decoder architecture (Section II-A), and discuss some aspects related to the dimension of the latent space and the so called variable-collapse phenomenon (Section II-B). Section III introduces the notion of Split Variational Autoencoder and provides a detailed description of the ResNet-like architecture used in our experiments. In Section IV we outline our experimental setting, discussing the metrics and datasets used for the benchmarks. Quantitative results are given in Section V along with a critical discussion. Ablation investigations are debated in Section VI. In the Conclusions VII, we summarize the content of the article and outline research directions for future developments.

## II. BACKGROUND

There exist in the literature several good introductions to Variational Autoencoders (VAEs) [4], [9], [19], so in this section we provide a quite short introduction to the topic, mostly with the purpose to fix notation and terminology.

In a latent variable approach, the probability distribution $p(x)$ of a data point $x$ is expressed through marginalization over a vector $z$ of *latent variables*:

$$p(x) = \int_z p(x|z)p(z)dz = \mathbb{E}_{p(z)}[p(x|z)] \tag{1}$$

where $z$ is the latent encoding of $x$ distributed with a known distribution $p(z)$ named *prior distribution*. If we can learn a good approximation of $p(x|z)$ from the data, we can use it to generate new samples via ancestral sampling:

- sample $z \sim p(z)$.
- generate $x \sim p(x|z)$.

Supposing to have a parametric family of probability distributions $\{p_\theta(x|z)\}$ (e.g. modelled by a neural network), the goal is to find $\theta^*$ that optimize the loglikelihood over all $x \in \mathbb{D}$ (MLE):

$$\begin{aligned}\theta^* &= \arg\max_\theta \mathbb{E}_{\mathbb{D}}[\log p_\theta(x)] \\ &= \arg\max_\theta \mathbb{E}_{\mathbb{D}}\Big[\log \int_z p_\theta(x|z)p(z)dz\Big]\end{aligned} \tag{2}$$

Addressing directly the previous optimization problem is usually computationally infeasible. For this reason, VAEs exploit another probability distribution $q_\phi(z|x)$ named *inference* (or encoder) distribution, expressing the relation between a data point $x$ and its associated latent representation $z$. Hopefully, $q_\phi(z|x)$ should approximate $p_\theta(z|x)$, so that their Kullback-Leibler divegence

$$D_{KL}(q_\phi(z|x)||p_\theta(z|x)) = \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(z|x)]$$

should be small. Further expanding the previous equation, we get:

$$\begin{aligned}&D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \\ &= \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(z|x)] \\ &= \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z|x) - \log p_\theta(x|z) - \log p_\theta(z) + \log p_\theta(x)] \\ &= D_{KL}(q_\phi(z|x)||p(z)) - \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + \log p_\theta(x)\end{aligned}$$

Hence,

$$\begin{aligned}&\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \\ &\qquad = \log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x))\end{aligned} \tag{3}$$

Recalling that $D_{KL}$ is always positive, we get

$$\underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))}_{\text{ELBO}} \leq \log p_\theta(x)$$

stating that the left hand side is a lower bound for the loglikelihood of $p_\theta(x)$, known as Evidence Lower Bound (ELBO).

Since ELBO is more tractable than MLE, it is used as the cost function for training of neural networks. Optimizing ELBO we are jointly improving the loglikelihood of $p_\theta(x)$, and implicitly minimizing the distance between $q_\phi(z|x)$ and $p_\theta(z|x)$.

The ELBO has a form similar to an autoencoder: the inference distribution $q_\phi(z|x)$ encodes the input $x$ to its latent representation $z$, and $p_\theta(x|z)$ decodes $z$ back to $x$.

For generative sampling, we just exploit the decoder, sampling latent variables according to the prior distribution $p(z)$ (that must be known).

### A. VANILLA VAE AND ITS TRAINING

In the vanilla VAE, we assume $q_\phi(z|x)$ to be a Gaussian (spherical) distribution $G(\mu_\phi(x), \sigma^2_\phi(x))$, so that learning $q_\phi(z|x)$ amounts to learning its two first moments. It is important to know the variance $\sigma^2_\phi(x)$ since during training we need to sample according to $q_\phi(z|x)$.

Similarly, we also assume $p_\theta(x|z)$ to have Gaussian distribution centered around a decoder function $\mu_\theta(z)$. The functions $\mu_\phi(x)$, $\sigma^2_\phi(x)$ and $\mu_\theta(z)$ are modelled by deep neural networks.

Supposing that the model approximating the decoder function $\mu_\theta(z)$ is sufficiently expressive, the shape of the prior distribution $p(z)$ does not really matter, and it is traditionally assumed to be a normal distribution $p(z) = G(0, I)$.

Under these assumptions, the term $D_{KL}(q_\phi(z|x)||p(z))$ is the KL-divergence between two Gaussian distributions $G(\mu_\phi(x), \sigma^2_\phi(x))$ and $G(0, I)$ that has the following closed form expression:

$$\begin{aligned}&D_{KL}(G(\mu_\phi(x), \sigma_\phi(x)), G(0, I)) \\ &\qquad = \frac{1}{2}\sum_{i=1}^{k} \mu_\phi(x)_i^2 + \sigma^2_\phi(x)_i - log(\sigma^2_\phi(x)_i) - 1\end{aligned} \tag{4}$$

where $k$ is the dimension of the latent space.

Coming to the reconstruction loss $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$, under the Gaussian assumption, the logarithm of $p_\theta(x|z)$ is proportional to the quadratic distance between $x$ and its reconstruction $\mu_\theta(z)$; the variance of this Gaussian distribution can be understood as a parameter balancing the relative importance between reconstruction error and KL-divergence [5], [9].

The problem of integrating sampling with backpropagation during training is addressed by the so called *reparametrization trick* [18], [26]: sampling is performed using a standard distribution outside of the backpropagation flow and this value is rescaled with $\mu_\phi(x)$ and $\sigma_\phi(x)$.

It is important to stress that sampling at training time has no relation with ancestral sampling for generation (we do not have $x$ at generation time!). The purpose of sampling at training time is to provide estimates for the main moment of $q(z|x)$, that are then subject to KL-regularization. In turn, the final goal of this regularization is to bring the marginal inference distribution $q(z) = \mathbb{E}_{x\in\mathbb{D}}\, q(z|x)$ close to the prior $p(z)$.

### B. THE DIMENSION OF THE LATENT SPACE

A critical aspect of VAEs is the dimension of the latent space. Typically, having many latent variables reduces the compression loss and improves reconstruction. However, this

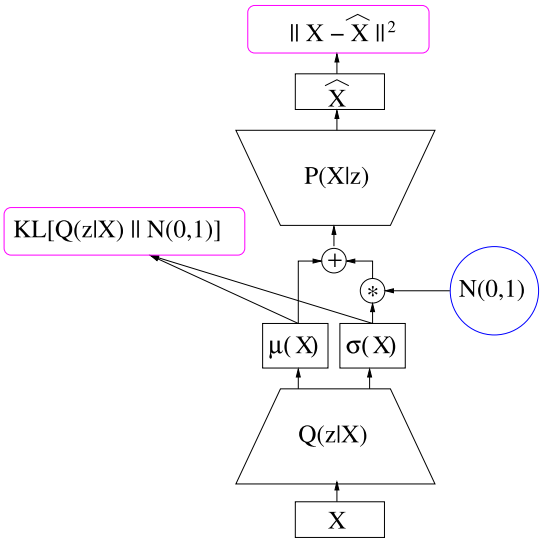


**FIGURE 5.** VAE architecture.

may not result in an improvement of the generative model: more variables we have and the harder is to ensure their independence and force them to assume the desired prior distribution. We may try to tame them by strengthening the KL-regularization component in the loss function, in the spirit of a $\beta$-VAE [7], [13], but this typically results in the *collapse* of the less informative variables, that get completely ignored by the decoder [1], [6], [25], [31], [36]. A collapsed variable $z$ has a very characteristic behaviour: since it is ignored by the decoder, it is free to minimize KL-regularization, with a mean value $\mu_z(x) = 0$ and a variance $\sigma_z^2(x) = 1$ for any $x$.

A more expressive architecture may typically result in a better exploitation of latent variables, allowing to work with a larger number of them. The dimension of the latent space also reflects the complexity of the data manifold: for instance, with non-hierarchical architectures, it is customary to work with 16 variables for Mnist, 128 variables for Cifar10 and 64 variables for CelebA. As we shall see, with a SVAE we can sensibly enlarge these numbers.

## III. SPLIT-VAE

The general notion of VAE does not impose any specification on the architecture of the encoder and the decoder, and many different variants have been investigated in the literature: dense, convolutional, with residuality, with autoregressive flows, hierarchical, and so on (see e.g. [4], [34] for a discussion).

A Split-VAE (SVAE) is just another architectural variant: we do not touch the theory or the loss function. In a SVAE the output $\hat{x}$ is computed as a weighted sum

$$\hat{x} = \sigma \odot \hat{x_1} + (1 - \sigma) \odot \hat{x_2}$$

of two generated images $\hat{x_1}, \hat{x_2}$, and $\sigma$ is a *learned* compositional map. Typically, to turn a vanilla VAE into a Split-VAE it is enough to change the number of channels of the last layer of the network: in case of a grayscale image, passing from 1 to 3, and in case of a color image passing from 3 to $1 + 3 + 3 = 7$ (the $\sigma$ map and two color images $\hat{x_1}, \hat{x_2}$). The compound image $\hat{x}$ can be computed internally of externally to the network, as part of the loss function. Since the increased number of channels only concerns the very last layer, the total number of parameters remains comparable to the vanilla version.

The philosophy underlying a SVAE has been already discussed in the introduction: we merely create an opportunity to be exploited by the model. From a more practical perspective, it can be be understood as a way to induce *diversification* in the features learned by the network, via a simple but highly effective self-attention mechanism [33]. From this point of view, it is not too far from techniques like squeeze and excitation [14] or feature-wise linear modulation [24]; the main difference is that we operate on the visible level, and along spatial dimensions. This allows us, among other things, to provide intelligible visualizations of the splitting learned by the network.

One of main characteristic of Split-VAEs is that they allow to work with a sensibly larger number of latent variables: 32 for Mnist, 200 for Cifar10 and 150 for CelebA. This testifies the diversification of latent features, and partially explains the improved generative quality.

### A. ENCODER-DECODER ARCHITECTURE

For the implementation of the encoder and the decoder we adopted a ResNet-like architecture derived from [8] that we already used in previous works [4], [5]; this allows us to do a fair comparison of the split-technique with previous approaches, without additional biases. The network architecture is schematically described in Figure 6.

The encoder is a fully convolutional model where the input is progressively downsampled for a configurable number of times, jointly doubling the number of channels. Before downsampling, the input is processed by a so called *Scale Block*, that is just a sequence of *Residual Blocks*. A Residual Block is an alternated sequence of BatchNormalization and spatial preserving Convolutional layers, intertwined with residual connections. The number of Scale Blocks at each scale of the image pyramid, the number of Residual Blocks inside each Scale Block, and the number of convolutions inside each Residual Block are user configurable hyperparameters.

In the encoder, after the last Scale Block, a global average level extracts spatial agnostic features. These are first passed through a so called *Dense Block* (similar to a Residual Block but with dense layers instead of convolutions), and finally used to synthesize mean and variance for latent variables. The decoder first maps the internal encoding $z$ to a small map of dimension $4 \times 4 \times$ *base_dim* via a dense layer suitably reshaped. This is then up-sampled to the final expected dimension, inserting a configurable number of Scale Blocks at each scale.

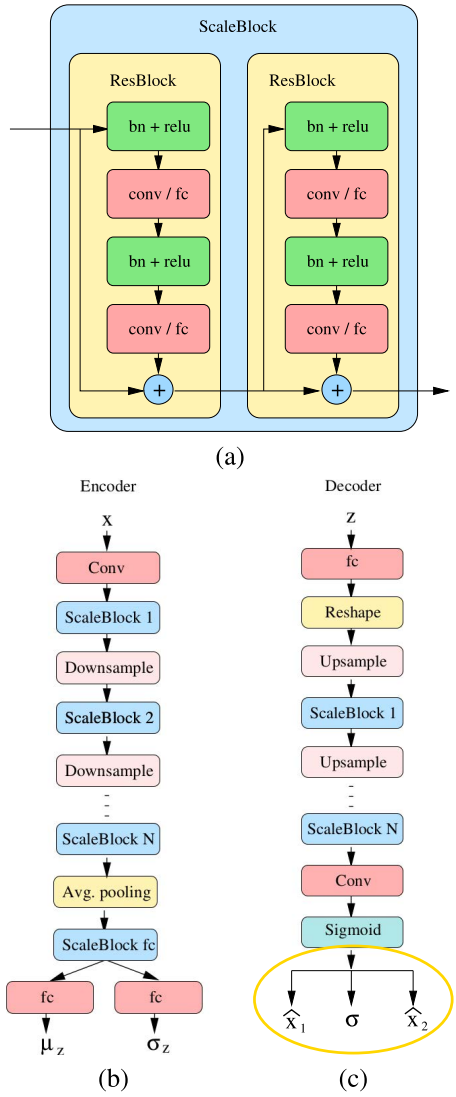


**FIGURE 6.** **(a) Scale Block: a Scale Block is a sequence of Residual Blocks intertwined with residual connections. A Residual Block alternates BatchNormalization layers, non-linear units and convolutions. (b) Encoder: the input is progressively downsampled via convolutions, preceded by Scale Blocks. At the final scale, a global average pooling layer extract features that are further processed via dense layers to compute mean and variance for latent variables. (c) Decoder: the decoder is essentially symmetric. A SVAE only differs in the final layer (circled in the picture): instead of directly producing $\hat{x}$, it produces two images $\hat{x_1}$ and $\hat{x_2}$ and a compositional map $\sigma$, defining $\hat{x} = \sigma \odot \hat{x_1} + (1 - \sigma) \odot \hat{x_2}$.**

## IV. EXPERIMENTAL SETTING

We compare the performance of SVAE with state-of-the-art variational autoencoders comprising Two-stage models [4], [5], [8], and Regularized Autoencoders [10]. We are not considering models relying on normalizing flows, such as [16], [23], [32]: these models typically require *thousands* of latent variables making them of relatively little interest from the point of view of representation learning.

For the comparison, we used traditional datasets, such as MNIST, CIFAR-10 [21] and CelebA [22]; the metrics adopted is the usual Frechèt Inception Distance [12] (FID), shortly discussed in section IV-A. All comparative results reported Section V are borrowed from the original publications.

In addition to the FID-score for generated images (GEN field, in Tables), we also provide an ex-post estimation of the probability distribution of the latent-space. This is done through a second VAE in [5], [8], and by fitting a Gaussian Mixture Model (GMM) in [10] (a normalizing autoregressive flow can be used with a similar purpose [17], [23]). Although possibly less expressive, the GMM technique is simple and effective, so we use it in our experiments (this aspect is somewhat orthogonal to the content of this article). The FID-score after resampling in the latent space is reported in the GMM entry, in the following Tables.

For each architecture, we also provide the number of parameters as an indicative measure of its complexity and energetic footprint (according to recent investigations [3], the number of parameters seem to provide a more reliable measure of efficiency than the number of Floating Point Operations).

### A. FRECHÈT INCEPTION DISTANCE

The Frechèt Inception Distance [12] (FID) does not try to assess the "quality" of a single generated sample but merely compares the overall probability *distribution* of generated vs. real images. The dimension of the visible space is typically too large to allow a direct comparison; the main idea behind FID is to use, instead of raw data, their internal representations generated by some third party, agnostic network. In the case of FID, the Inception v3 network [30] trained on Imagenet is used to this purpose. The activations that are traditionally used are those relative to the last pooling layer, resulting in a vector of 2048 features.

Let $a_1$ and $a_2$ be the activations relative to real and generated images, and $\mu_i, i = 1, 2$, $C_i, i = 1, 2$ their empirical mean and covariance matrix, respectively. Then the Frèchet Distance between $a_1$ and $a_2$ is the just the squared Wasserstein distance, namely:

$$FID(a_1, a_2) = ||\mu_1 - \mu_2||^2 + Tr(C_1 + C_2 - 2(C_1 * C_2)^{\frac{1}{2}}) \quad (5)$$

where $Tr$ is the trace of the matrix.

## V. NUMERICAL RESULTS

In this section we give numerical results relative to Mnist, CIFAR-10 and CelebA, discussing the training process and the relevant hyperparameters.

### A. MNIST

In the case of Mnist we worked with a latent space of dimension 32, in contrast with the traditional dimension of 16. We tested several versions, with different balancing $\beta$-factor between reconstruction and KL-regularization: we report results for $\beta = 8$ and $\beta = 3$. Even when starting with the relatively high balancing factor $\beta = 8$, the number of inactive variables at the end of training is low: between

2 and 5. In all our experiments, the $\beta$-factor is progressively reduced along training in order to preserve the initial balance between the two components, as described in [5].

In the case of Mnist, both syntactic and semantic decomposition (see Figures 1 and 2) usually give good results on the compound image, slightly better in the latter case.

Training lasted 300 epochs, using Adam optimizer with a learning rate of $1.0e-3$. Numerical results in terms of FID-scores are given in Table1.

**TABLE 1. MNIST: FID scores for generated images (GEN). The GMM value refers to ex-post estimation of the latent space distribution via a Gaussian Mixture Model in the spirit of [10]. For MNIST, we use a GMM with 20 components.**

| model | params | GEN | GMM |
|---|---|---|---|
| RAE-GP [10] | 22,386,449 | 22.2 | 11.5 |
| RAE-L2 [10] | 22,386,449 | 22.2 | 8.7 |
| RAE-SN [10] | 22,386,449 | 19.7 | 11.7 |
| 2S-VAE, [8] | 9,032,769 | 21.9 | 12.6 |
| 2S-VAE, [5] | 9,032,769 | 21.8 | 11.8 |
| SVAE ($\beta = 8$) | 12,793,667 | 20.4 | 7.3($20c.$) 7.2($100c.$) |
| SVAE ($\beta = 3$) | 12,793,667 | 41.1 | 7.2($20c.$) **4.8**($100c.$) |

In [10], they observed that enlarging the number of components beyond 10 was not beneficial. However, in our case, presumably due to the larger number of latent variables, we found convenient to use a larger mix. In Table1 we compare the cases of 20 and 100 components. To have an acceptable generative score before GMM-resampling we need to work with a high $\beta$ factor, like e.g. $\beta = 8$; however, resampling compensates the need of regularization, and we obtain the best results with $\beta = 3$ and 100 components.

See Figure 7 for examples of generated Mnist-like digits.

**FIGURE 7. Examples of generated images for MNIST.**

**TABLE 2. FID score for each one of the CIFAR-10 categories versus the whole dataset. Observe the high values, in comparison with the extremely low score of the whole dataset vs. itself (mix).**

| label | fid | label | fid | label | fid | label | fid |
|---|---|---|---|---|---|---|---|
| plane | 80.7 | car | 98.3 | bird | 65.0 | cat | 60.8 |
| deer | 66.3 | dog | 74.3 6 | frog | 100.2 | horse | 84.3 |
| ship | 97.7 | truck | 107.5 | | mix | 5.3 | |

**TABLE 3. CIFAR-10: summary of results.**

| model | params | GEN | GMM |
|---|---|---|---|
| RAE-GP [10] | 30,510,339 | 83.0 | 74.2 |
| RAE-L2 [10] | 30,510,339 | 80.8 | 74.2 |
| RAE-SN [10] | 30,510,339 | 84.2 | 74.2 |
| 2S-VAE [8] | 27,766,275 | 76.7 | 72.9 |
| 2S-VAE, [5] | 27,766,275 | 80.2 | 69.8 |
| SVAE | 13,061,143 | 81.9 | **69.5** |

### B. CIFAR10

CIFAR-10 confirms its somehow pathological nature. The complexity of the dataset can be readily appreciated by looking at the pictures in Figure 9 where we compare the mean images for CIFAR-10 and CelebA: the former is completely gray, while the latter is a relatively well defined ''average'' face (also observe, by the way, the strong bias of the CelebA dataset towards feminine, frontal, young, smiling faces).

Another interesting indicator of the complexity of CIFAR-10 is given by the iFID score between each category and the full dataset (we derive 10000 image per category by flipping images in the training set, and compare them with the test-set). The score is extremely high, in spite of the fact that the ''texture'' is apparently quite similar. The FID score ''magically'' drops to 5.3 for a random mix.

Coming to SVAE, in the case of CIFAR-10, splitting produces results of the kind described in Figure 8. The FID scores for $\hat{x_1}$ and $\hat{x_2}$ are usually lower to that of $\hat{x}$ testifying that the network is attempting a ''semantic'' decomposition; however, our networks failed, apart exceptions, to generate recognizable contours. In spite of this problem, numerical results, reported in Table 3 are quite good.

These results have been obtained exploiting a latent space of dimension 200 (in contrast with the traditional dimension of 128) and a balancing factor $\beta = 3$ between reconstruction and KL-regularization. Training lasted 110 epochs (fast!), using Adam optimizer with an initial learning rate of $1.0e-3$. For ex-post re-estimation of the distribution of the latent space we used a GMM with 100 components. Examples of generated CIFAR-10-like images are given in Figure 10.

### C. CelebA

The splitting technique for CelebA works particularly well, automatically producing remarkable ''semantical'' maps similar to drawings (see Figures 4 and 13). The quality and precision in the design of details is impressive and largely unexpected.

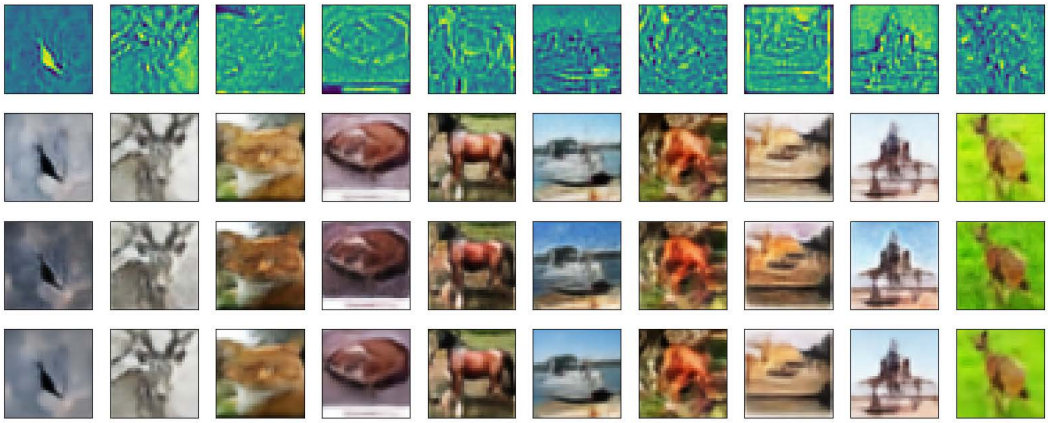

**FIGURE 8.** Example of reconstructed images in CIFAR-10. The network clearly struggles to derive a "sense" out of the pictures, but fails. Without the underlying images, contours would be hardly recognizable. Enlarging the resolution does not seem to help.

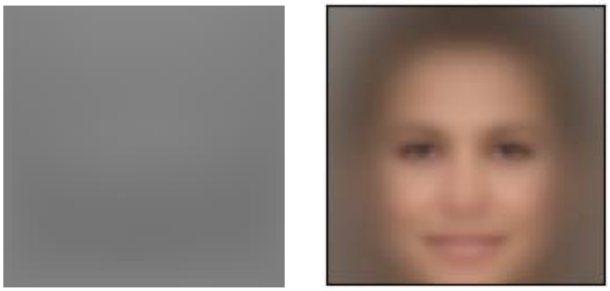


**FIGURE 9.** Comparison of the mean image for CIFAR-10 (left) and CelebA (right). Cifar-10 is a sensibly more complex and randomic dataset.

**TABLE 4.** CelebA: FID scores. In the case of SVAE, the three GEN values respectively refer to $\hat{x}$, $\hat{x_1}$, $\hat{x_2}$. As before, GMM is the FID result after ex-post estimation of the latent space by means of a Gaussian mixture model (100 components). In the case of SVAE the best result is obtained by taking a random mix of images from $\hat{x_1}$ and $\hat{x_2}$.

| model | params | GEN | | | GMM |
|---|---|---|---|---|---|
| RAE-GP [10] | 30,510,339 | 116.3 | | | 45.3 |
| RAE-L2 [10] | 30,510,339 | 51.1 | | | 47.9 |
| RAE-SN [10] | 30,510,339 | 44.7 | | | 40.9 |
| 2S-VAE, [8] | 27,766,275 | 60.5 | | | 44.4 |
| 2S-VAE, [5] | 27,766,275 | 43.6 | | | 38.6 |
| SVAE | 28,989,363 | 54.8 | 49.0 | 45.7 | **35.1** |

Comparative values are reported in Table 4. In this case we provide fid scores for $\hat{x}$, $\hat{x_1}$, $\hat{x_2}$. We worked with a latent space of dimension 150, in contrast with more traditional dimensions like 64 or 128. The initial $\beta$-factor was 3.

Training lasted 160 epochs, using Adam optimizer with an initial learning rate of $1.0e-3$, but already after 40/50 epochs we get excellent FID scores for $\hat{x_1}$ and $\hat{x_2}$. A typical evolution of FID scores during training is shown in Figure 11; a more thorough investigation is given in the next section.

Additional examples of generated CelebA-like images and splitting masks are given in Figures 12 and 13, respectively.

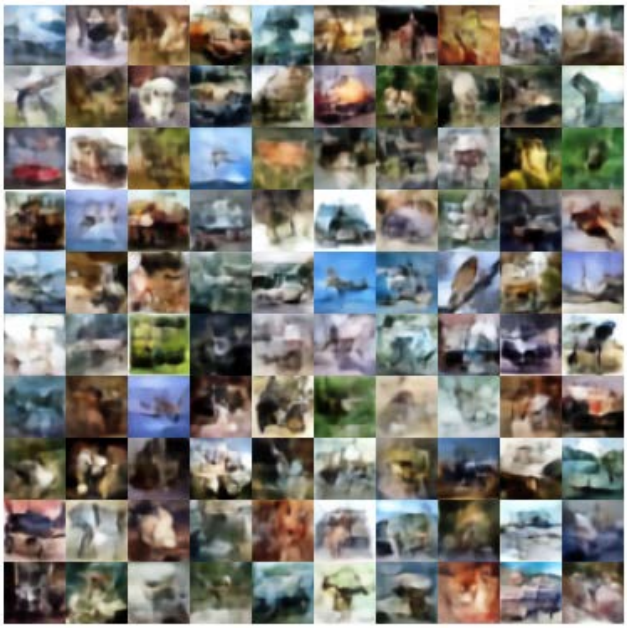

**FIGURE 10.** Examples of generated images for Cifar-10.

## VI. ABLATION

The splitting technique is simple and non-invasive. There are however a few additional modifications suggested and induced by splitting - most notably the increased number of latent variables - and one could naturally wonder if this is not the actual source of the observed improvements in FID scores.

To clarify the point we compared the behaviours of *precisely* the same architectures just changing the final layer of the decoder.

We focused the attention on the most interesting cases of CIFAR10 and CelebA, comparing the evolution of the fid score for $\hat{x}, \hat{x_1}, \hat{x_2}$ for 5 different trainings for each dataset.

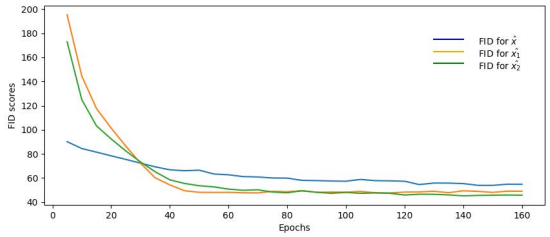


**FIGURE 11.** Evolution of the FID score for $\hat{x}$, $\hat{x_1}$ and $\hat{x_2}$ during training. The score for $\hat{x_1}$ and $\hat{x_2}$ (and for GMM too) is already quite good after 50-60 epochs of training.

**FIGURE 12.** Examples of generated images. In the case of CelebA, the quality of samples generated with a variational approach should be judged on those details with the highest variability: hairs, background, accessories. Note also the wide differentiation in pose, illumination, colors, age and expressions.

All scores have been computed after resampling in the latent space according to a GMM with 100 components. Results are given in Figures 14 and 15. The improvement for $\hat{x}$ in the case of the split-network is marginal, but the Fid score for $\hat{x_1}$ and $\hat{x_2}$ is *significantly* smaller, and frequently even smaller than the reconstruction FID, that is traditionally supposed to be a lower bound for this metric on generated samples.

Our understanding of the phenomenon is that splitting allows the network to decompose each image towards higher-density regions in the given neighbourhood, hence creating more realistic variants of the usual "average" result.

On the other side, it is also interesting to remark the high sensibility of the FID score to apparently minor modifications of generated images (a phenomenon already pointed out in [2]).

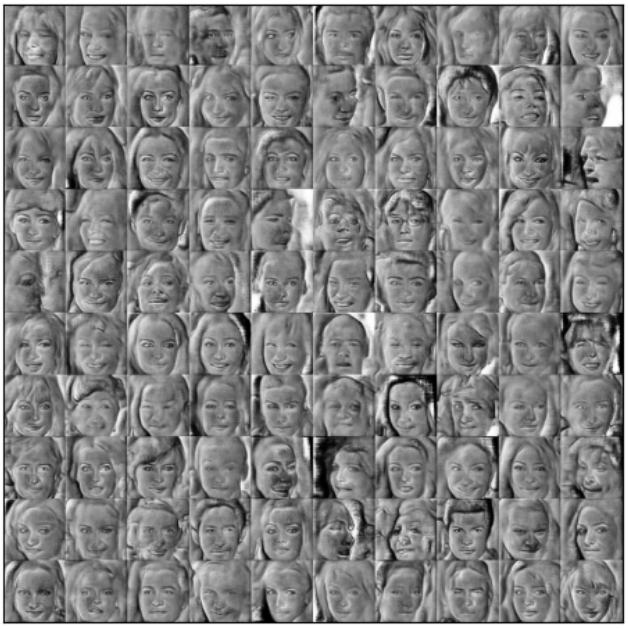

**FIGURE 13.** Example of generated boolean maps. The quality and precision of contours is both unexpected and remarkable.

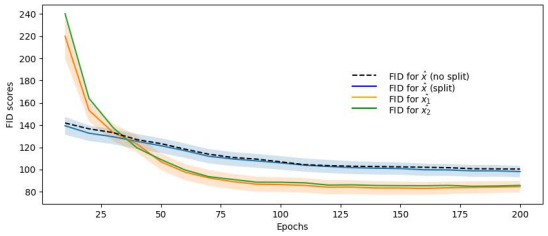


**FIGURE 14.** CIFAR10: Comparative evolution of the FID score for $\hat{x}$ in vanilla version (dotted line) versus $\hat{x}$ $\hat{x_1}$ and $\hat{x_2}$ for the split-version. FID scores have been computed after resampling in the latent space with a GMM with 100 components. Results are relative to an average over 5 distinct trainings, along the first 200 epochs. Only (smoothed) standard deviations for $\hat{x}$ (split version) and $\hat{x_1}$ are depicted, for readability reasons; the standard deviation for the other FID scores is similar.

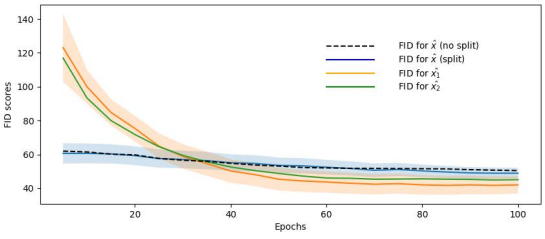


**FIGURE 15.** CelebA: Comparative evolution of the FID score for $\hat{x}$ in vanilla version (dotted line) versus $\hat{x}$ $\hat{x_1}$ and $\hat{x_2}$ for the split-version. FID scores have been computed after resampling in the latent space with a GMM with 100 components. Results are relative to an average over 5 distinct trainings, along the first 100 epochs. Similarly to Figure 14, only standard deviations for $\hat{x}$ (split version) and $\hat{x_1}$ are shown.

## VII. CONCLUSION

In this article, we introduced the notion of Split Variational AutoEncoder (SVAE). In a SVAE the output $\hat{x}$ is computed as

a weighted sum $\sigma \odot \hat{x_1} + (1-\sigma) \odot \hat{x_2}$ where $\hat{x_1}, \hat{x_2}$ are two distinct generated images, and $\sigma$ is a learned compositional map. A Split VAE is trained as a normal VAE: no additional loss is added over the split images $\hat{x_1}$ and $\hat{x_2}$. Splitting is meant to offer to the network a way to generate variants of the expected result, in the attempt of overcoming the averaging problem inherent to the adoption of a loglikelihood loss function. At the same time, the network may specialize its generative capabilities towards more oriented and specific subsets of the data manifold, possibly learning additional and differentiated features. As a side result, even with a relatively high balancing factor for KL-regularization, the variable collapse phenomenon is less constraining, and the possibility of exploiting a larger number of latent variables improve the quality and diversity of generated samples. This has been experimentally confirmed on traditional benchmarks such as Mnist, Cifar10 and CelebA. The SVAE architecture systematically improves over its vanilla counterpart, and outperforms state-of-the-art loglikelihood-based generative models such as Two-Stage architectures or Regularized autoencoders. We intentionally avoided to test the architecture on high-resolution datasets such as CelebA-HQ [15], mostly for ethical and ecological reasons: they are too demanding in terms of computational resources. We think that there are a lot of interesting problems to be investigated and solved even on relatively cheap datasets, so there is no actual need to move to high-resolution domains.

As for future developments of this work, a particularly interesting research direction seems to be the possibility to add *control* over the splitting operation, possibly segmenting the input image in other interesting and meaningful components, and specializing subnets for their respective processing.

**Code** The code relative to this work is available on Github in the following repository: https://github.com/asperti/SplitVAE Pretrained weights for the models discussed in the article are available at the following page: https://www.cs.unibo.it/~asperti/SVAE.html.

**Conflict of Interest** On behalf of all authors, the corresponding author states that there is no conflict of interest.

## REFERENCES


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

[2] A. Asperti, "Variance loss in variational autoencoders," in *Machine Learning, Optimization, and Data Science* (Lecture Notes in Computer Science). Siena, Italy: Springer, Sep. 2020.

[3] A. Asperti, D. Evangelista, and M. Marzolla, "Dissecting flops along input dimensions for GreenAI cost estimations," in *Proc. 7th Int. Conf. Mach. Learn., Optim. Data Sci.* Grasmere, U.K.: Springer, 2022, pp. 86–100.

[4] A. Asperti, D. Evangelista, and E. Loli Piccolomini, "A survey on variational autoencoders from a green AI perspective," *Social Netw. Comput. Sci.*, vol. 2, no. 4, p. 301, Jul. 2021.

[5] A. Asperti and M. Trentin, "Balancing reconstruction error and Kullback–Leibler divergence in variational autoencoders," *IEEE Access*, vol. 8, pp. 199440–199448, 2020.

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

[7] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in $\beta$-VAE," *CoRR*, vol. abs/1804.03599, pp. 1–11, Apr. 2018.

[8] B. Dai and D. Wipf, "Diagnosing and enhancing VAE models," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–44.

[9] C. Doersch, "Tutorial on variational autoencoders," *CoRR*, vol. abs/1606.05908, pp. 1–23, Jun. 2016.

[10] P. Ghosh, M. S. M. Sajjadi, A. Vergari, M. Black, and B. Schölkopf, "From variational to deterministic autoencoders," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–25.

[11] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, pp. 1–57, Dec. 2017.

[12] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, vol. 30, Long Beach, CA, USA, Dec. 2017, pp. 6629–6640.

[13] 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. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–22.

[14] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, Aug. 2016.

[15] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–26.

[16] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible $1 \times 1$ convolutions," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, vol. 31, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Montreal, QC, Canada, Dec. 2018, pp. 10236–10245.

[17] D. P. Kingma, T. Salimans, R. Józefowicz, X. Chen, I. Sutskever, and M. Welling, "Improving variational autoencoders with inverse autoregressive flow," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, vol. 29, Barcelona, Spain, Dec. 2016, pp. 4736–4744.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, Banff, AB, Canada, Apr. 2014, pp. 1–14.

[19] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019.

[20] J. Koh, J. Lee, and S. Yoon, "Single-image deblurring with neural networks: A comparative survey," *Comput. Vis. Image Understand.*, vol. 203, Feb. 2021, Art. no. 103134.

[21] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://scholar.google.it/scholar?q=learning+multiple+layers+of+features+from+tiny+images&hl=it&as_sdt=0&as_vis=1&oi=scholart

[22] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.

[23] R. Morrow and W.-C. Chiu, "Variational autoencoders with normalizing flow decoders," *CoRR*, vol. abs/2004.05617, Apr. 2020. [Online]. Available: https://arxiv.org/abs/2004.05617

[24] E. Perez, F. Strub, H. D. Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proc. 32nd AAAI Conf. Artif. Intell. (AAAI), 30th Innov. Appl. Artif. Intell. (IAAI), 8th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI)*, S. A. McIlraith and K. Q. Weinberger, Eds. New Orleans, LA, USA: AAAI Press, Feb. 2018, pp. 3942–3951.

[25] 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.

[26] 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)*, vol. 32, Beijing, China, Jun. 2014, pp. 1278–1286.

[27] L. Ruthotto and E. Haber, "An introduction to deep generative modeling," *CoRR*, vol. abs/2103.05180, 2021. [Online]. Available: https://arxiv.org/abs/2103.05180

[28] S. Sahu, M. K. Lenka, and P. K. Sa, "Blind deblurring using deep learning: A survey," 2019, *arXiv:1907.10128*. [Online]. Available: https://arxiv.org/abs/1907.10128, doi: 10.48550/arxiv.1907.10128.
[29] A. Spindler, J. E. Geach, and M. J. Smith, "AstroVaDEr: Astronomical variational deep embedder for unsupervised morphologicalclassification of galaxies and synthetic image generation," *Monthly Notices Roy. Astronomical Soc.*, vol. 502, no. 1, pp. 985–1007, Nov. 2020, doi: 10.1093/mnras/staa3670.
[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Las Vegas, NV, USA: IEEE Computer Society, Jun. 2016, pp. 2818–2826.
[31] B. Trippe and R. Turner, "Overpruning in variational Bayesian neural networks," 2018, *arXiv:1801.06230*. [Online]. Available: https://arxiv.org/abs/1801.06230, doi: 10.48550/arxiv.1801.06230.
[32] A. Vahdat and J. Kautz, "NVAE: A deep hierarchical variational autoencoder," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, H. Larochelle, M. A. Ranzato, R. Hadsell, M.-F. Balcan, and H.-T. Lin, Eds., Dec. 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/e3b21256183cf7c2c7a66be163579d37-Abstract.html
[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, vol. 30, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.
[34] R. Wei, C. Garcia, A. El-Sayed, V. Peterson, and A. Mahmood, "Variations in variational autoencoders—A comparative evaluation," *IEEE Access*, vol. 8, pp. 153651–153670, 2020.
[35] R. Wei and A. Mahmood, "Recent advances in variational autoencoders with representation learning for biomedical informatics: A survey," *IEEE Access*, vol. 9, pp. 4939–4956, 2021.
[36] S. Yeung, A. Kannan, Y. Dauphin, and L. Fei-Fei, "Tackling over-pruning in variational autoencoders," *CoRR*, vol. abs/1706.03643, pp. 1–11, Jun. 2017.

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

He was the Head of the Department of Computer Science, from 2005 to 2007. He is currently a Full Professor with the University of Bologna, where he teaches courses of machine learning and deep learning. From 2000 to 2007, he acted as a member of the Advisory Committee of the World Wide Web Consortium. He was responsible for several national and international projects. He is the author of three books, and a number of scientific publications in international peer reviewed conferences and journals. His research interests include deep learning, generative modeling, and deep reinforcement learning.

**LAURA BUGO** was born in Bologna, Italy, in 1995. She received the B.S. degree in computer science from the University of Bologna, in 2019, where she is currently pursuing the master's degree. Her research interest includes artificial intelligence applied to improvement of well-being for most vulnerable people. She is also an Athlete of judo kata, she has won a bronze medal at European Kata Championships, in 2021, three silver medals at European Kata Championships U24, and a silver medal at World Judo Kata Grand Slam U35.

**DANIELE FILIPPINI** was born in Ostiglia, Italy, in 1996. He received the B.S. degree in information science for management from the University of Bologna, in 2019, where he is currently pursuing the master's degree in computer science. His research interests include artificial intelligence and deep learning.