RV-VAE: Integrating Random Variable Algebra into Variational Autoencoders

Vassilis C. Nicodemou[1,2] Iason Oikonomidis[2] Antonis Argyros[1,2]

[1]Computer Science Department, University of Crete, Heraklion, Greece
[2]Institute of Computer Science, FORTH, Heraklion, Greece

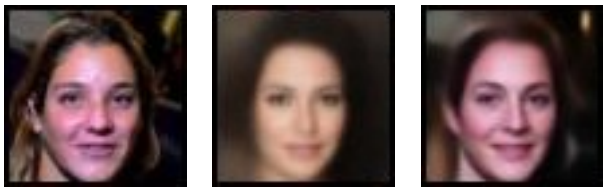
{nikodim, oikonom, argyros}@ics.forth.gr

Abstract

Among deep generative models, variational autoencoders (VAEs) are a central approach in generating new samples from a learned, latent space while effectively reconstructing input data. The original formulation requires a stochastic sampling operation, implemented via the reparameterization trick, to approximate a posterior latent distribution. In this paper, we introduce a novel approach that leverages the full distributions of encoded input to optimize the model over the entire range of the data, instead of discrete samples. We treat the encoded distributions as continuous random variables and use operations defined by the algebra of random variables during decoding. This approach integrates an innate mathematical prior into the model, helping to improve data efficiency and reduce computational load. Experimental results across different datasets and architectures confirm that this modification enhances VAE-based architectures' performance. Specifically, our approach improves the reconstruction error and generative capabilities of several VAE architectures, as measured by the Fréchet Inception Distance (FID) metric, while exhibiting similar or better training convergence behavior. Our method exemplifies the power of combining deep learning with inductive priors, promoting data efficiency and less reliance on brute-force learning. Code available at https://github.com/VassilisCN/RV-VAE.

1. Introduction

Generative properties in deep learning models can be achieved through three notable approaches: generative adversarial networks (GANs)[11], diffusion models[36], and variational autoencoders (VAEs)[20]. Each possesses unique challenges: GANs, though able to generate high-quality images, face training instability and mode collapse issues. Diffusion models, while powerful, are not easily interpretable and are computationally intensive. VAEs stand out with stable training and efficiency but need significant amounts of data for optimal performance. Despite advancements in these methods, refining generative models remains an active research field [38, 32, 37, 44, 42, 31, 17, 10, 47, 14, 25, 12, 29, 13, 7].



(a) Input image (b) Original VAE (c) RV-VAE

Figure 1: The proposed RV modifications of VAE architectures enhances the models' capabilities so that a given input image can be reconstructed by the RV-VAE more satisfactory compared to the original VAE.

We present an enhancement for VAE architectures based on a novel modification to neural network nodes. Traditional nodes typically utilize single scalar activations, especially when approximating probabilistic quantities, like the reparameterization trick in VAEs. We propose to replace these scalar activations with Probability Density Function (PDF)-based activations, represented by expected value and variance. Through the algebra of Random Variables (RVs) [39], we compute the expected values and variances of typical neural operations. In special cases, such as ReLU and Batch Normalization, certain assumptions ensure tractable computations. Employing this RV-aware methodology, we adapt the decoder section of various tested VAE architectures, eliminating the need for the reparameterization trick, a step essential in other methods [20, 7]. This revamped VAE structure taps into visual inductive priors to diminish data needs and enhance data efficiency, all without compromising the convergence rate during training.

In summary, the contributions of this work are: (a) A theoretically justified approach for utilizing continuous distributions in ANNs and specifically in VAEs for incorporating inductive priors, that (b) significantly improves image

reconstruction and (c) achieves similar or improved generative results, while (d) maintaining training convergence rate. This paper emphasizes the potential of combining deep learning with inductive priors, towards more data-efficient deep learning practices. The source code of our method is available at https://github.com/VassilisCN/RV-VAE.

2. Related Work

The related work can be roughly divided into two categories: works on enhancing and improving VAEs, and works on neural networks and autoencoders that use a stochastic or probabilistic approach. Our work fits in both classes with greater emphasis on the former, since it can enhance the performance of most VAE-based architectures.

Variational autoencoders (VAEs), one of the first successful generative deep learning models, were proposed by Kingma and Welling [20]. Another very successful generative model, Generative Adversarial Neural networks (GANs) [11] was proposed almost simultaneously. Since VAEs were introduced, a significant amount of work has been devoted to both theoretical analysis [9] and improving the base architecture. Improvements on VAEs can be achieved through various means, including redesigning the network architecture [37, 44], incorporating stronger priors [42, 31, 17], adding regularization [10, 47], or incorporating adversarial objectives [14, 25, 12, 29, 13].

The tools of probability theory have been heavily used in the field of neural networks, mostly for facilitating the theoretical analysis of their behavior. Mean-field theory has been applied to the analysis of networks, either on single layers [35] or on multiple layers [26, 6], closely resembling modern deep architectures. Furthermore, Bayesian Neural Networks (BNNs) embed probabilistic modeling directly into neural architectures [24]. By placing priors on the weights and biases [3] and with the usage of variational inference [41], BNNs introduce a principled uncertainty estimation into deep learning, making them particularly advantageous for tasks where understanding uncertainty is crucial. On the practical front, mainstream Deep Learning frameworks such as TensorFlow [1] and PyTorch [28] include libraries [8, 2, 34, 43] that facilitate the development and integration of stochastic operations in neural networks. Recently, Kim [19] presented work on the VAE architecture that employs an inference model to enhance the encoding of the data, aiming to reduce the posterior approximation error of inference in VAEs. While both approaches aim to improve the performance of VAEs, Kim's work focuses on a more accurate modeling of the computation of latent space values. In contrast, our work focuses on a theoretically justified way of utilizing the encoded latent space.

Within this research area, a particular direction that is closely related to ours is that of Probabilistic Circuits. Poon and Domingos [30] and Shen *et al*. [33] have explored the possibility of developing deep probabilistic models, where the propagation of the PDF throughout the network nodes is constrained to use specific operations, similar to our approach. Vergari *et al*. [45] compiled a comprehensive list of operations that can be used for the layers of the network toward this end. Additionally, Jaini *et al*. [16] and Cohen *et al*. [5] advocate for the use of tensor decompositions to bring probabilistic models closer to modern deep neural networks. Compared to traditional deep neural networks, probabilistic circuits have some limitations such as their high computational cost and the lack of diversity of their generated samples.

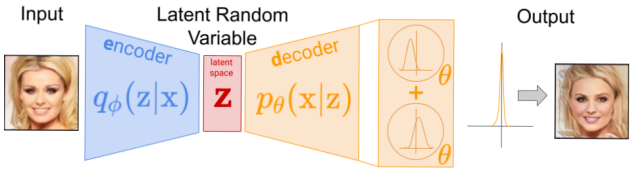


Figure 2: The proposed VAE formulation avoids the need for stochastic sampling from the latent space variables **z**, by directly forwarding the encoded distributions q_ϕ to the decoder. This is achieved by treating the latent space as an instance from a family of distributions and employing random variable operations inside the decoder. The final output is also a distribution and, by minimizing its variance, we effectively constrain it to become a constant (image). Following standard VAE notation (as *e.g*. in Kingma and Welling [20]), $q_\phi(z|x)$ and $p_\theta(x|z)$ denote the encoder and decoder part of the network respectively, and vectors ϕ, θ and x denote respectively the parameters of the encoder, decoder, and the input.

The proposed work can be categorized as a general improvement for VAE architectures, applicable to any VAE architecture that employs sampling in the latent space. The modification is applied only to the decoder part of the architecture, where we substitute the traditional components with our proposed Random Variable-aware ones. To the best of our knowledge, no similar approaches have been proposed so far in the relevant literature.

3. Random Variable Modules

All data that are processed by ANNs such as images, video sequences, audio, and text, are samples of (possibly implicit) underlying distributions. Even though the distribution domains can be infinite, the networks we employ always operate on specific, constant instances belonging to these domains, that is, samples of the distributions. This is the assumption on which we base the design of the operations that comprise an ANN, such as fully connected layers, convolutional layers, activation functions, etc.

A simple, effective and powerful way to represent a potentially infinite range of such instances is to resort to a stochastic representation of the input, a representation that must also be propagated in the same way through the network. If we want the network to process a tensor of non-constant values, such as Random Variables (RVs), we must redefine the network's operations to treat the tensors as such. This can be achieved through the algebra of random variables [39]. Representing network inputs and activations as RVs is very general and powerful, however in practice it quickly leads to intractable computations. In order to come up with a practical solution that can compete the extremely efficient modern neural networks, some simplifying assumptions need to be made. We adopt those assumptions as they are validated empirically[1].

An arbitrary probability distribution over the real numbers is fully determined by the infinite series of its moments. In this work we choose to approximate it using only the first two moments, namely the expected value $\mathbb{E}[X]$ and variance $\text{var}[X]$. Throughout the computations performed over the layers, we keep this representation by calculating the new expected value and variance. Another simplifying assumption is the handling of correlations between the involved random variables. Similarly to Batch Normalization [15], for computational efficiency, we choose to ignore all correlations between the inputs to a network layer, and only estimate the variance of each Random Variable.

The rules provided by the algebra for symbolic manipulation are applied in two cases: (a) between two RVs, and (b) between an RV and a constant. Since all commonly used ANN operations are essentially instances of one of these cases, we can adapt and use them to derive all the cases we are interested in. In summary, for a RV X that can be sufficiently described by its mean and variance, and an ANN operation $op(\cdot)$, in order to compute $Y = op(X)$, it is sufficient for our representation to calculate $\mathbb{E}[op(X)]$ and $\text{var}[op(X)]$. It is important to note that, the number of network parameters remains constant, even as the number of operations and network activations increases. In the following sections, we elaborate on the calculation of expected value and variance for the most common operations of an ANN's modules.

3.1. Linear operations

General linear operations between the inputs of a neuron are commonly used to implement fully connected layers and convolution operations. Such a linear operation is defined as:

$$\boldsymbol{y} = \boldsymbol{x}\boldsymbol{A}^T + \boldsymbol{b}, \tag{1}$$

where $\boldsymbol{x}$ is the input vector of RVs, $\boldsymbol{A}$ the matrix of learnable weights, and $\boldsymbol{b}$ the learnable bias vector. The expected value for the output vector of RVs $\boldsymbol{y}$ is given by:

$$\mathbb{E}[\boldsymbol{y}] = \mathbb{E}[\boldsymbol{x}\boldsymbol{A}^T + \boldsymbol{b}] = \mathbb{E}[\boldsymbol{x}]\boldsymbol{A}^T + \boldsymbol{b}, \tag{2}$$

and its variance is given by:

$$\text{var}[\boldsymbol{y}] = \text{var}[\boldsymbol{x}\boldsymbol{A}^T + \boldsymbol{b}] = \text{var}[\boldsymbol{x}](\boldsymbol{A} \odot \boldsymbol{A}), \tag{3}$$

where $\odot$ denotes the Hadamard (element-wise) product.

As already mentioned, convolution operations are similarly treated. A derivation for this case can be found in the supplementary material.

3.2. ReLU activation function

For the case of the ReLU activation function, the output vector $\boldsymbol{y}$ is defined as $\boldsymbol{y} = max(\boldsymbol{x}, 0)$. Thus, in this case, the calculation of the expected value and variance is not straightforward. To calculate the expected value and variance of a RV that is distributed according to the output of the ReLU function, we make the assumption that each input RV follows a normal distribution. This hypothesis is grounded on the empirical observation that after some ANN linear operations, data tend to become approximately normally distributed. This is evidenced in the cases of uniformly and normally distributed data, where the summation of such data is approximately normally distributed[2] as well as for the mixed cases of normal plus normal and normal plus uniform [46]. This applies to our cases as well, as most of the defined operations perform linear operations over their input. It is also further supported by our own experiments.

Using the law of total expectation, we can define the expected value of $max(X, c)$ for a normally distributed RV X and a constant c, as follows:

$$\begin{aligned}\mathbb{E}[max(X, c)] = \mathbb{E}[X|X > c]P(X > c) \\ + \mathbb{E}[c|X \leq c]P(X \leq c),\end{aligned} \tag{4}$$

where $P(\cdot)$ denotes the probability function for the provided event. From Eq.(4), for the case $c = 0$ of ReLU, $\mathbb{E}[0|X \leq 0]P[X \leq 0] = 0$. Since X is assumed to follow a normal distribution, $P(X > c) = 1 - \Phi(a)$ where $\Phi(\cdot)$ is the standard normal cumulative distribution function (CDF) of the normally distributed RV X, and $a = \frac{c-\mu}{\sigma}$ with μ the mean of the normal distribution which is also the expected value, and σ its standard deviation where $\sigma^2 = \text{var}[X]$. The term $\mathbb{E}[X|X > c]$ can be calculated based on the one-sided truncated normal distribution, as:

$$\mathbb{E}[X|X > c] = \mu + \frac{\sigma\phi(a)}{1 - \Phi(a)}, \tag{5}$$

where $\phi(\cdot)$ is the standard PDF of the normally distributed RV X. By injecting Eq.(5) in Eq.(4) we obtain the final

[1] More details on this are documented in the supplementary material.

[2] Specifically, it is distributed according to the Irwin-Hall distribution.

form of the expected value as follows:

$$\mathbb{E}[max(X,0)] = (1-\Phi(a))\left(\mu + \frac{\sigma\phi(a)}{1-\Phi(a)}\right). \quad (6)$$

For calculating the variance, we use the law of total variance in a similar manner:

$$\begin{aligned}\text{var}[max(X,c)] &= \text{var}[X|X>c]P(X>c) \\ &+\mathbb{E}[X|X>c]^2(1-P(X>c))P(X>c).\end{aligned} \quad (7)$$

In this case, the one-sided truncated normal distribution gives us the term:

$$\text{var}[X|X>c] = \sigma^2\left(1 + \frac{a\phi(a)}{1-\Phi(a)} - \left(\frac{\phi(a)}{1-\Phi(a)}\right)^2\right). \quad (8)$$

Using the equations (8) and (5) in Eq.(7) we obtain the final expression of the variance:

$$\begin{aligned}\text{var}[max(X,0)] &= (1-\Phi(a)) \\ &\left(\sigma^2\left(1 + \frac{a\phi(a)}{1-\Phi(a)} - \left(\frac{\phi(a)}{1-\Phi(a)}\right)^2\right)\right. \\ &\left. + \left(\mu + \frac{\sigma\phi(a)}{1-\Phi(a)}\right)^2\Phi(a)\right).\end{aligned} \quad (9)$$

3.3. Batch normalization

As described by Ioffe and Szegedy [15], the batch normalization operation can be broken down into the following two steps: (1) data normalization and (2) data scaling and shifting. The authors state that data normalization is performed for each feature dimension of the data. After normalizing the data they add a linear operation that scales and shifts the data given the learnable parameters γ and β, respectively. Since the second step is essentially a linear operation, it can be handled as in Sec. 3.1. On the contrary, the first step needs further analysis, as follows.

We follow the same reasoning we stated in Sec. 3.2 that all our data become normally distributed after an ANN linear operation. This holds since a batch normalization layer is used only after linear ones. As already stated in Sec. 3.2, all data become normally distributed after some ANN operations. Since each network activation is an RV that follows a distribution, the normalization in the feature dimension can be described as an equally weighted mixture of these distributions in that dimension. Consequently, we want to calculate the mean and the variance of an equally weighted mixture of normal distributions. Toward this end, for a mixture of n component distributions with $X_1, \dots X_n$ RVs with known expected values and variances, the total mean is defined as:

$$\mathbb{E}[X] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[X_i]. \quad (10)$$

The total variance can be calculated as:

$$\begin{aligned}\text{var}[X] &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ &= \left(\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[{X_i}^2]\right) - \mathbb{E}[X]^2 \\ &= \left(\frac{1}{n}\sum_{i=1}^{n}(\text{var}[X_i] + \mathbb{E}[X_i]^2)\right) - \mathbb{E}[X]^2 .\end{aligned} \quad (11)$$

4. Random Variable Variatonal Autoencoders

The building blocks described, are capable of handling RVs, and can be used to replace stochastic procedures in ANNs. A suitable group of architectures is VAEs, which feature a stochastic process using scalar value samples. The stochastic usage of latent encoded data distributions at the bottleneck layer that forwards samples of these distributions to the decoder, makes them suitable to be used with RV modules. By design, the encoded latent space in most VAE architectures is normally distributed, and consequently is able to use RVs of appropriate distributions to represent it. Therefore, we can adjust VAE architectures to incorporate RV modules.

4.1. Relation to the original VAE formulation

Our goal in this work is to avoid imposing a stochastic estimation in the lower bound. To better understand this proposed contribution in the VAE approach, we first present here the variational lower bound as defined in the original VAE formulation by Kingma and Welling [20]. The evidence lower bound $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$ is defined for a single input data point $\mathbf{x}^{(i)}$ as:

$$\begin{aligned}\mathcal{L}(\theta,\phi;\mathbf{x}^{(i)}) &= -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) \\ &+ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})].\end{aligned} \quad (12)$$

The formulation follows the standard VAE notation of [20]. $q_\phi(z|x)$ and $p_\theta(x|z)$ denote the encoder and decoder part of the network respectively, and vectors ϕ, θ and x denote the parameters of the encoder, decoder, and the input.

The formulation's goal is to differentiate and optimize this lower bound w.r.t. ϕ, the variational parameters, and θ, the generative parameters. In order to estimate the second term, the authors suggest forming Monte Carlo estimates of expectation, where the lower bound is estimated as:

$$\begin{aligned}\mathcal{L}(\theta,\phi;\mathbf{x}^{(i)}) &\simeq \tilde{\mathcal{L}}(\theta,\phi;\mathbf{x}^{(i)}) \\ &= -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) \\ &+ \frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})),\end{aligned} \quad (13)$$

where L is the number of samples drawn. The authors state that $L = 1$ is sufficient since the training procedure is performed in batches of satisfactory size.

In this work, instead of adopting a stochastic approach in the estimation of the lower bound that also depends on the batch size, we employ the whole distribution $q_\phi(\mathbf{z}|\mathbf{x})$, according to which the RV $\mathbf{z}$ is distributed, as seen and in Fig. 2. Therefore, by implementing differentiable RV operations during the generative process, the method we propose is closely related to the original lower bound formulation of Eq.(12), using this term directly during training, instead of the commonly used approximation Eq.(13).

4.2. Architecture modifications

The modifications needed to make any VAE-based architecture appropriate for handling RVs are mainly on the decoder of the architecture. Since the encoder outputs the latent distribution (in the form of means and variances), in an unmodified VAE architecture we would forward it through the decoder via sampling. In our case, we do not need to perform any sampling operation, therefore we operate on the expected values and variances resulting from the encoder, and defer the conversion of distributions to values until the end of the pipeline. Any VAE-based architecture that decodes the encoded distribution can take advantage of the proposed modification. Every layer/module of the decoder must be replaced with the appropriate RV module described in Sec. 3. Specifically, the decoder of a VAE consists in most cases of linear modules (such as fully connected layers and convolutional/transposed convolutional layers Sec. 3.1), ReLU activation functions (Sec. 3.2) and batch normalization layers (Sec. 3.3).

4.3. Loss adjustment

Since every module outputs RVs (represented by their means and variances), by doing the above modifications in the decoder, the final output of the network will also be a RV. This is not useful in our test cases, but can be accommodated with the following adjustment in the loss function. Provided that the goal of a VAE is to output a result of a constant form (e.g. an image), we can make the modified VAE architecture to achieve this by enforcing a constraint on the output RVs of the final layer. A constant can be expressed as a RV in a form of a Dirac delta function, with expected value the constant itself and variance equal to zero. The network's output can be viewed as a constant value if we retain only the mean value and enforce the variance to be close to zero. This can be achieved by minimizing the following term:

$$L_c(\boldsymbol{y}) = \frac{\lambda}{n}\sum_{i=1}^{n}\text{var}[\boldsymbol{y}_i]^2, \tag{14}$$

where $\boldsymbol{y}$ is the final output vector of n RVs of a VAE and λ an experimentally determined constant scale factor that acts as a balancing weight between the terms. The final loss is therefore defined as the sum of the original variational lower bound and the new term L_c:

$$L(\boldsymbol{x}) = \mathcal{L}(\theta, \phi; \boldsymbol{x}) + L_c(\boldsymbol{y}). \tag{15}$$

5. Experiments

The conducted experiments aim to verify and assess the following goals of our approach: (1) the ability and ease to adopt RV-awareness by different VAE-based architectures, (2) the improvement in image reconstruction , (3) the improvement of generative capabilities, while (4) maintaining satisfactory or even improved training convergence time.

To tackle these goals we used four different VAE-based architectures, i.e., the original VAE [20], β-TCVAE [4], Soft-Intro-VAE [7] and DC-VAE [27]. The first two are very popular VAE approaches, whereas the latter two are recent state-of-the-art approaches. These methods were trained and tested on several datasets, each using the respective implementation by the authors wherever possible. The VAE and β-TCVAE architecture implementations were used from [40]. Our evaluation employed the following datasets: MNIST [22], CIFAR-10 [21], CelebA [23] and CelebA-HQ [18] resized to 128×128.

5.1. Constructing RV-aware VAE architectures

Using the modifications of Sec. 4.2 we created an RV-based version of all the above architectures. As mentioned, the modifications were on the decoder and the loss function. In practice, in all architectures we omitted the reparameterization trick, and sent the encoded distributions directly into the respective decoders.

The loss adjustment described in Sec. 4.3 was employed by all architectures and was added to their originally defined losses. Specifically, for every output RV tensor $\mathcal{Y}$, the $\mathbb{E}[\mathcal{Y}]$ was responsible for minimizing the reconstruction error, while var$[\mathcal{Y}]$ was used for the added loss L_c described in Eq. (14), with $\lambda = 50$ for all the experiments. All architectures were trained using the proposed hyper-parameters in their respective manuscripts and provided code and trained for 150 epochs.

5.2. Implementation details

We implemented the modules described in Sec. 3 using PyTorch [28]. We used implementations (also in PyTorch), of all the architectures we experimented with, and replaced the appropriate network layers with their RV-enabled respective ones. In practice, we found that our modified VAE architecture implementation is about 3 times slower. However, it only required approximately 30% more FLOPS than the unmodified version. Therefore, we believe that suitable optimizations, such as implementing demanding modules like ReLU and Batch Normalization in C++ API, can significantly increase the overall speed of our implementation. More specifically, we expect the speed to improve to around

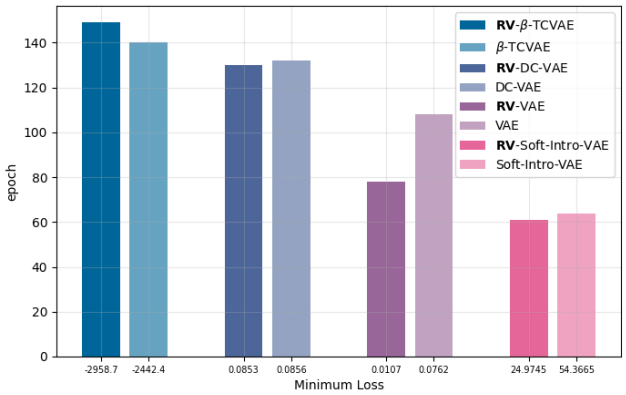


Figure 3: The epoch of training (as bar height) that each architecture reached its minimum validation loss value on the CIFAR-10 dataset.

70% of the unmodified version, as judged by the number of additional FLOPS required by our approach. Overall, our experiments demonstrate that the benefits of our approach in terms of improved reconstruction error and generative capabilities justify the additional computational cost.

5.3. Training speed convergence comparison

With the elimination of sampling in the latent space during training, an RV-based VAE network does not rely on thoroughly sampling the training data space, since the whole data distribution is being forwarded to the decoder. This can lead to faster convergence. This is documented in Fig. 3 which illustrates the time (in epochs) needed for each VAE architecture to reach its minimum validation loss during training. In all cases, the RV-aware architectures reach their minimum loss in a similar or earlier epoch than their original counterparts. Moreover, the minimum loss of RV-based networks is always, by far, lower than the loss of the original architectures, despite the fact that the loss of RV-aware VAEs includes the additional, non-negative term, L_c.

5.4. Image reconstruction

To demonstrate the reconstruction capabilities of the RV-aware VAEs, we conducted several experiments comparing the original architectures with our RV-aware modified ones on the employed datasets. Table 1 shows the Mean Square Error (MSE) for all test sets of datasets between the original images and their reconstructed ones. In all cases, our proposed RV modifications enhance the reconstruction performance of all the reported architectures, even by a large margin in some cases. To further illustrate those results in a qualitative context, we also provide some reconstructed test samples in Figs. 4 to 6.

5.5. Image generation

The proposed RV modifications are also beneficial due to their generative properties. To illustrate this, we report in Table 2 the Fréchet Inception Distance (FID) based on $50,000$ generated samples. For generating new samples, we follow the same procedure as in the original VAEs by sampling the mean from a Gaussian distribution and fixing the variance to $\text{var}[X] = 1$. For all cases, we observe lower FID in the modified RV-aware networks. We can also see in Figs. 7 to 9 some qualitative results of RV-aware generated samples compared to the samples generated by the unmodified networks. Moreover, to show the continuity of the latent space, in Fig. 10, we present generated images that are created by interpolating between two latent space samples.

5.6. Transferability (from RV-VAE to regular VAE)

Despite the proposed changes, the trainable parameters of the resulting, RV-aware networks remain the same. Therefore, after training, it is conceivable to consider the same network weights transferred to a non-RV counterpart. This should be expected to operate without any changes since the Expected Values of the involved quantities behave linearly: For scalar parameters a and b, a relation $Y = aX + b$ between two RVs X and Y implies that the Expected Value of their samples is similarly related, $\mathbb{E}[y \sim Y] = a\mathbb{E}[x \sim X] + b$. Therefore, we can transfer the learned parameters of an RV-VAE network to a regular VAE and keep its functionality.

To provide evidence of this, in Fig. 11 we present some examples of reconstructed images. Specifically, we can observe that the third row which presents the reconstructions from a non-RV network that had its parameters transferred from a RV-aware one, has similar results to the last row which has the reconstructions of the original RV-aware network. Moreover, as stated previously these results are significantly better than the ordinary trained non-RV network, second row. This is also justified by the MSE between the RV-aware reconstruction and the reconstruction of transferred RV parameters to a non-RV network to be 1.85×10^{-6}.

Apart from the theoretical feasibility and the experimental validation, it is also useful to address the reason to follow such an approach. Training an RV-aware network is demonstrably beneficial as already presented. Furthermore, evaluating using a *non*-RV network is computationally faster since the computation of the variances in each layer is no longer necessary. Essentially, the proposed approach acts as a regularization technique that enables better/more accurate results.

Method	MSE ↓ on MNIST		MSE ↓ on CIFAR-10		MSE ↓ on CelebA		MSE ↓ on CelebA-HQ	
	Original	RV-VAE (ours)	Original	RV-VAE (ours)	Original	RV-VAE (ours)	Original	RV-VAE (ours)
VAE [20]	0.0081	**0.0005**	0.0763	**0.0107**	0.0478	**0.0192**	-	-
β-TCVAE [4]	0.0021	**0.0004**	0.0412	**0.0157**	0.0412	**0.0130**	-	-
DC-VAE [27]	-	-	0.1245	**0.1139**	-	-	-	-
Soft-Intro-VAE [7]	0.0194	**0.0129**	0.0211	**0.0155**	-	-	0.0247	**0.0151**

Table 1: Image reconstruction results for all datasets.

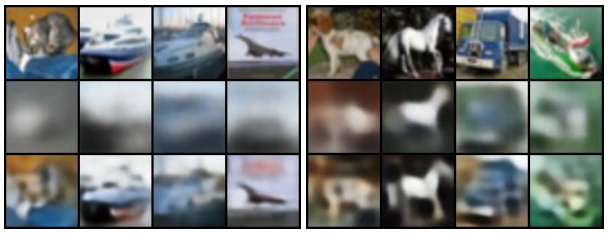
Figure 4: Reconstructions of CelebA-HQ images (1st row) by Soft-Intro-VAE (2nd row) and RV-Soft-Intro-VAE (3rd row).

(a) VAE (b) β-TCVAE

Figure 5: 1st rows: CelebA images; 2nd, 3rd rows: reconstructions by original VAEs & their RV-aware versions.

Method	FID ↓ CIFAR-10		FID ↓ CelebA-HQ	
	Orig.	RV-VAE (ours)	Orig.	RV-VAE (ours)
DC-VAE [27]*	26.78	**23.44**	-	-
Soft-Intro [7]*	5.31	**5.26**	2.85	**2.82**

Table 2: Comparison of FID scores for CIFAR-10 and CelebA-HQ datasets. *FIDs calculated by the implementations provided by the authors.



(a) VAE (b) β-TCVAE

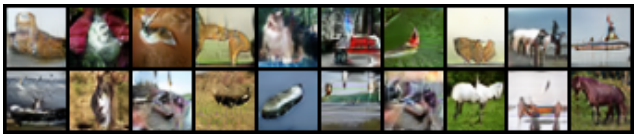
(c) DC-VAE (d) Soft-Intro-VAE

Figure 6: 1st rows: CIFAR-10 images; 2nd, 3rd rows: reconstructions by original VAEs & their RV-aware versions.

6. Summary

In this work, we have introduced an approach to incorporate continuous distributions into VAE architectures, which improves latent space utilization, using the algebra of random variables to treat decoder node activations as distributions. This modification can be readily applied to most VAE architectures by simply replacing decoder layers with RV-aware ones, followed by retraining. This novel prior de-

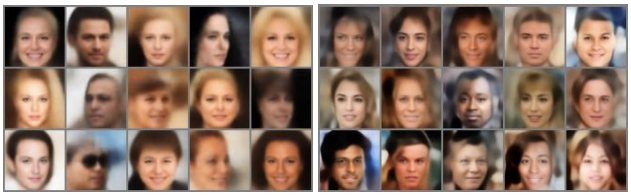
Figure 7: Generated samples on CelebA-HQ using original Soft-Intro-VAE (1st row) and our RV-Soft-Intro-VAE (2nd row).



(a) DC-VAE

(b) Soft-Intro-VAE

Figure 8: Generated samples on CIFAR-10 using original VAEs (1st rows) and their RV-aware versions (2nd rows).



(a) VAE (b) RV-VAE

Figure 9: Generated samples on CelebA using (a) original VAE and (b) RV-VAE.

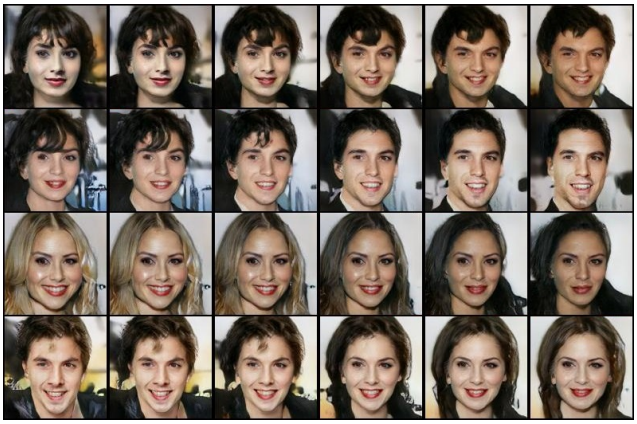


Figure 10: Interpolated generations between two CelebA-HQ samples from RV-Soft-Intro-VAE.

Figure 11: 1st row: input images; 2nd row: the reconstructions of input images from an ordinary trained non-RV network; 3rd row: reconstructions from a non-RV network with RV-trained parameters; 4th row: the reconstructions from the RV-aware network.

parts from the traditional sampling-based method, enhancing both reconstruction quality and generative result fidelity without hindering convergence rate. Future work aims at expanding the proposed mathematical framework with additional RV-aware ANN layers. We will also explore the viability of this approach in network types other than VAEs.

Acknowledgments

This research work was supported by the Hellenic Foundation for Research and Innovation (HFRI) under the HFRI Ph.D. Fellowship grant (Fellowship Number: 803), by HFRI under the "1st Call for HFRI Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment", project I.C.Humans, number 91. The authors would like to gratefully acknowledge support for this research from the VMware University Research Fund (VMURF). We also acknowledge the assistance of Dr. Lefteris Koumakis who generously provided access to a high-performance computer during the initial stages of development.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016. 2

[2] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019. 2

[3] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015. 2

[4] Tian Qi Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 2018. 5, 7

[5] Nadav Cohen, Or Sharir, Yoav Levine, Ronen Tamari, David Yakira, and Amnon Shashua. Analysis and design of convolutional networks via hierarchical tensor decompositions. *arXiv preprint arXiv:1705.02302*, 2017. 2

[6] J Cook. The mean-field theory of a q-state neural network model. *Journal of Physics A: Mathematical and General*, 22(12):2057, 1989. 2

[7] Tal Daniel and Aviv Tamar. Soft-introvae: Analyzing and improving the introspective variational autoencoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4391–4400, June 2021. 1, 5, 7

[8] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017. 2

[9] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey. *arXiv preprint arXiv:2101.00734*, 2021. 2

[10] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019. 1, 2

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1, 2

[12] Tian Han, Erik Nijkamp, Linqi Zhou, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. Joint training of variational auto-encoder and latent energy-based model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7978–7987, 2020. 1, 2

[13] Ari Heljakka, Arno Solin, and Juho Kannala. Towards photographic image manipulation with balanced growing of generative autoencoders. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3120–3129, 2020. 1, 2

[14] Huaibo Huang, Zhihang Li, Ran He, Zhenan Sun, and Tieniu Tan. Introvae: Introspective variational autoencoders for photographic image synthesis. In *Advances in Neural Information Processing Systems*, 2018. 1, 2

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 3, 4

[16] Priyank Jaini, Pascal Poupart, and Yaoliang Yu. Deep homogeneous mixture models: representation, separation, and approximation. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[17] Dimitris Kalatzis, David Eklund, Georgios Arvanitidis, and Søren Hauberg. Variational autoencoders with riemannian brownian motion priors. *arXiv preprint arXiv:2002.05227*, 2020. 1, 2

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018. 5

[19] Minyoung Kim. Gaussian process modeling of approximate inference errors for variational autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2022. 2

[20] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014. 1, 2, 4, 5, 7

[21] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. ... *Science Department, University of Toronto, Tech.* ..., 2009. 5

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[23] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 5

[24] David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992. 2

[25] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 1, 2

[26] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018. 2

[27] Gaurav Parmar, Dacheng Li, Kwonjoon Lee, and Zhuowen Tu. Dual contradistinctive generative autoencoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 823–832, June 2021. 5, 7

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2, 5

[29] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14104–14113, 2020. 1, 2

[30] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011. 2

[31] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems*, pages 14866–14876, 2019. 1, 2

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 1

[33] Yujia Shen, Arthur Choi, and Adnan Darwiche. Tractable operations for arithmetic circuits of probabilistic models. *Advances in Neural Information Processing Systems*, 29, 2016. 2

[34] N. Siddharth, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah D. Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. Learning disentangled representations with semi-supervised deep generative models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5927–5937. Curran Associates, Inc., 2017. 2

[35] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020. 2

[36] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 1

[37] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Advances in neural information processing systems*, 29:3738–3746, 2016. 1, 2

[38] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 1

[39] Melvin Dali Springer. The algebra of random variables. Technical report, 1979. 1, 3

[40] A.K Subramanian. Pytorch-vae, 2020. 5

[41] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019. 2

[42] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018. 1, 2

[43] Dustin Tran, Alp Kucukelbir, Adji B Dieng, Maja Rudolph, Dawen Liang, and David M Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016. 2

[44] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020. 1, 2

[45] Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. *Advances in Neural Information Processing Systems*, 34:13189–13201, 2021. 2

[46] Robin Willink. *Measurement uncertainty and probability*. Cambridge University Press, 2013. 3

[47] Hongteng Xu, Dixin Luo, Ricardo Henao, Svati Shah, and Lawrence Carin. Learning autoencoders with relational regularization. In *International Conference on Machine Learning*, pages 10576–10586. PMLR, 2020. 1, 2