SELF-REFLECTIVE VARIATIONAL AUTOENCODER: APPENDIX

Anonymous authors
Paper under double-blind review

S.I EXPLANATORY FIGURES

Figure S1 shows a high-level overview of a Deep Latent Gaussian Model (DLGMs) (Rezende et al., 2014). One key-feature of this architecture is the transformational layers that are utilized by both the posterior and the prior layers. Other variations of deep VAE architectures may consider conditional prior distributions $p(\boldsymbol{z}_l \mid \boldsymbol{z}_{l-1})$ and/or the posterior distributions $q(\boldsymbol{z}_l \mid \boldsymbol{z}_{l-1}, \boldsymbol{x})$, but they omit the joint transformational layers.

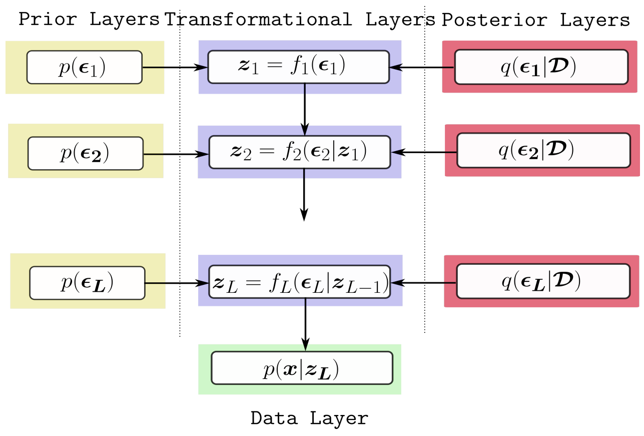

Prior Layers
Transformational Layers
Posterior Layers
$p(\boldsymbol{\epsilon}_1)$
$\boldsymbol{z}_1 = f_1(\boldsymbol{\epsilon}_1)$
$q(\boldsymbol{\epsilon_1}|\boldsymbol{\mathcal{D}})$
$p(\boldsymbol{\epsilon_2})$
$\boldsymbol{z}_2 = f_2(\boldsymbol{\epsilon}_2|\boldsymbol{z}_1)$
$q(\boldsymbol{\epsilon_2}|\boldsymbol{\mathcal{D}})$
$p(\boldsymbol{\epsilon_L})$
$\boldsymbol{z}_L = f_L(\boldsymbol{\epsilon}_L|\boldsymbol{z}_{L-1})$
$q(\boldsymbol{\epsilon_L}|\boldsymbol{\mathcal{D}})$
$p(\boldsymbol{x}|\boldsymbol{z_L})$
Data Layer

Figure S1: Independent VAE.

Figure S2 shows a high-level overview of both the generative and the inference part of the SeRe-VAE. Some key differences of our model from existing works are i) the joint transformational layers are enforced to be bijective, in contrast to Figure S1, ii) also in contrast to Figure S1, there is statistical feedback to the next layer of both the generative and inference network, iii) the latent variables of all layers directly interact with the data distribution $p(\boldsymbol{x}|\boldsymbol{z})$.

Figure S3 offers a detailed illustration of the generative procedure of the SeRe-Vae described in section 3.1. The nodes represent random variables and the edges the conditional dependencies induced by the generative procedure. The colors match the random variables with the layers in Figure S2 which are responsible for their generation.

Figure S4 shows the computational graph of an amortized Gaussian prior layer, as described in section 3.4.1 and according to the reparametrization trick (Kingma & Welling, 2014).

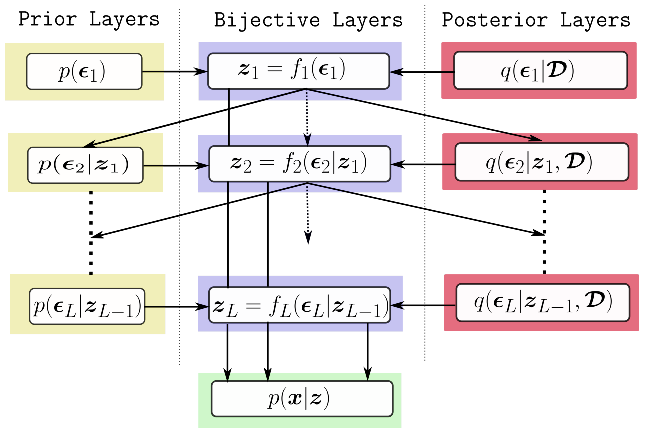

Prior Layers
Bijective Layers
Posterior Layers
$p(\epsilon_1)$
$z_1 = f_1(\epsilon_1)$
$q(\epsilon_1|\mathcal{D})$
$p(\epsilon_2|z_1)$
$z_2 = f_2(\epsilon_2|z_1)$
$q(\epsilon_2|z_1, \mathcal{D})$
$p(\epsilon_L|z_{L-1})$
$z_L = f_L(\epsilon_L|z_{L-1})$
$q(\epsilon_L|z_{L-1}, \mathcal{D})$
$p(x|z)$

Figure S2: Self-Reflective VAE.

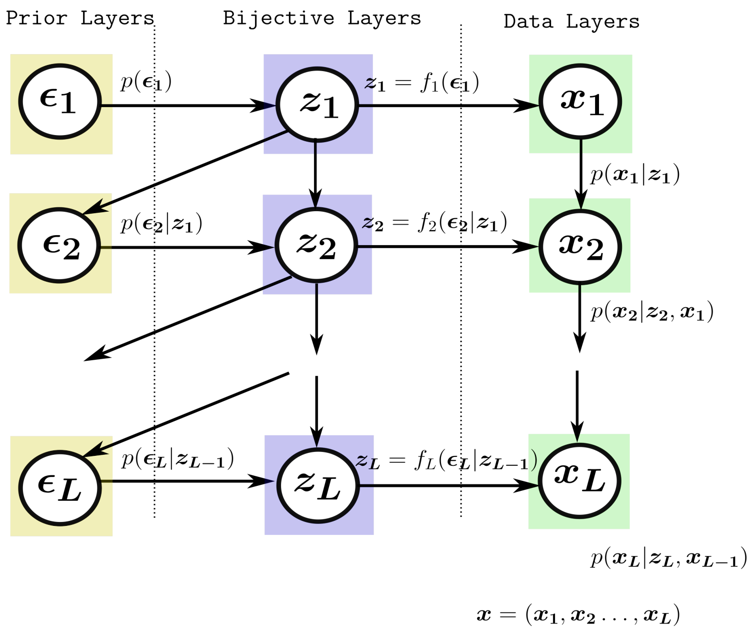


Figure S3: Self-Reflective Generative Model.

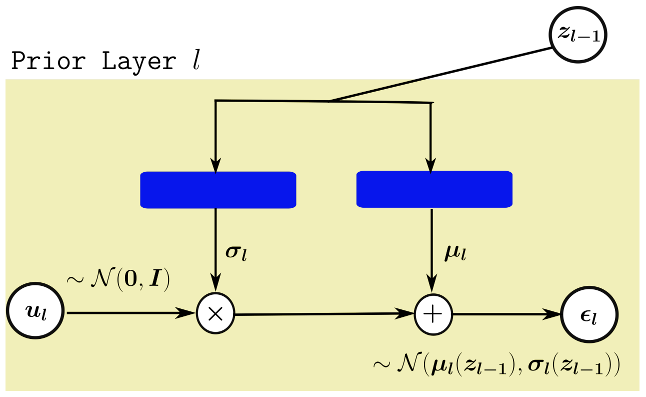


Figure S4: Amortized Gaussian Prior Layer.

S.II Implementation Details

Our experiments use `Tensorflow Probability` (TFP, 2018). All models were optimized using Adam (Kingma & Ba, 2015) on a 4xGeForce GTX 2080Ti.

S.II.A Training a Self-Reflective Probabilistic Model

In this section, we describe two modifications of the ELBO objective in equation 2. We found empirically that they both facilitate the training of the proposed model.

S.II.A.1 KL Annealing

We applied deterministic warm-up as suggested in Sønderby et al. (2016a); Rezende & Viola (2018); Bowman et al. (2016); Sønderby et al. (2016b). This technique introduces a scheduled regularization coefficient β for the KL-divergence. Formally, the regularized ELBO objective becomes:

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_q[\log p(\boldsymbol{x} \mid \boldsymbol{z})] - \beta D_{KL}(q(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z})), 0 \leq \beta \leq 1, \tag{S1}$$

where β linearly increases from 0 to 1 for a number of epochs at the beginning of the training.

S.II.A.2 minimizing Ensemble Reconstruction Losses

In order to train the residual data layers presented in section 3.4.2, we modify the conditional likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ in equation 2, so that for each layer l, the average conditional likelihood of the two estimation levels 1 is used. Formally, we replace $p(\boldsymbol{x}_l; \boldsymbol{\gamma}_l(\boldsymbol{z}_l, \boldsymbol{x}_{l-1}))$ with:

$$\hat{p}(\boldsymbol{x}_l|\boldsymbol{z}_l, \boldsymbol{x}_{l-1}) = 0.5 \times p(\boldsymbol{x}_l; \boldsymbol{\gamma}_l^1(\boldsymbol{z}_l)) + 0.5 \times (\boldsymbol{\gamma}_l^1(\boldsymbol{z}_l) + \delta\boldsymbol{\gamma}_l(\boldsymbol{x}_{l-1}, \boldsymbol{z}_l)). \tag{S2}$$

This change helps prevent overfitting triggered by the computational block at the first estimation level. We apply this change only for the warm-up training epochs mentioned in the previous subsection (for which $\beta < 1$). For inference, we use the most refined parameters $\boldsymbol{\gamma}_l$ of the last level of the residual data layers.

S.III Experimental Details of the MLP-SeRe VAE for binarized MNIST

S.III.A Learning Plots

In this experiment, we trained the model for 2000 epochs. For the first 1024 epochs (Figure S5) the regularized, ensemble loss function as described in Section S.II.A was minimized. For the second half, the negative ELBO of equation 2 (Figure S6) was minimized.

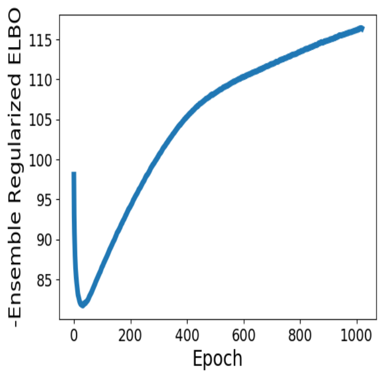


Figure S5: Loss function during warm-up.

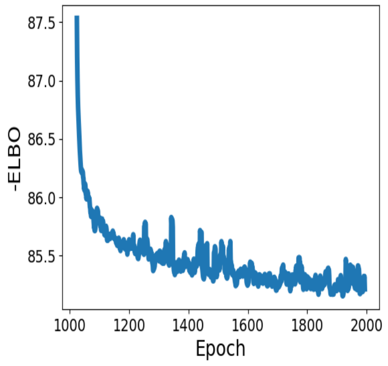


Figure S6: Loss function after warm-up.

In Figure S7, we plot the conditional loglikelihoods obtained when the first estimations (red line) of the parameters $\gamma^1_{1:L}$ of the data distribution is used and when they are rectified by the residual functions $\delta\gamma_{1:L}$ to give the final estimation $\gamma_{1:L}$ (blue line). As we see, the residual functions $\delta\gamma_{1:L}$ significantly improve the reconstruction loss. For the first 1024 epochs, the average of these plots is maximized, while for the second half of the training only the conditional likelihood computed from $\gamma_{1:L}$ is maximized.

In Figure S8, we plot the KL divergence of each stochastic layer. As we see, all the stochastic layers remain active (they do not collapse to the prior).

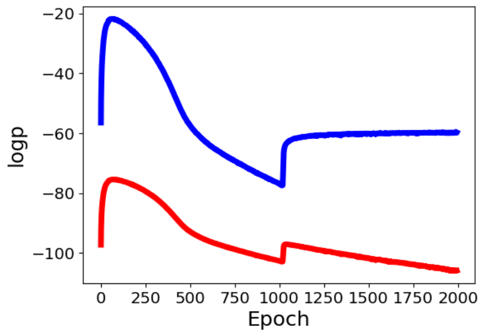


Figure S7: Conditional loglikelihood with :
Red: $\gamma^{\mathbf{1}}_{1:L}$, Blue: $\gamma^{\mathbf{2}}_{1:L}$

S.III.B ARCHITECTURAL HYPERPARAMETERS OF THE MLP-SERE VAE FOR BINARIZED MNIST

This section provides detailed description of the training parameters and architectural hyperparameters for the experiments in Section 4.1.1 in the main paper.

Specifically, in Table S1, we provide the parameters of the training procedure. A constant learning rate and a small amount of weight regularization was used. We did not observe overfitting. Moreover, batch normalization layers were added at the input of each layer in the hierarchy.

Table S1: Training Hyperparameters of the MLP-SeRe VAE's for binarized MNIST

Parameter	Value
batch size	`256`
warm up epochs	`1024`
warm up schedule	`linear`
epochs	`2000`
learning rate	`1e-3`
batch normalization	`Yes`
kernel/bias regularization	ℓ_2, $\lambda = 1e-5$
kernel/bias initializer	`glorot normal`

The model consists of 10 layers of 10 latent variables each. This experiment uses exclusively MultiLayer Perceptrons (MLP) as building blocks of the prior, posterior and bijective layers and of the final data distribution in the decoder. The hyperparameters for each component in the hierarchy are given in Table S2. The evidence encoders of each layer are decoupled: they receive the raw binary image as input and not the output of the encoders at the upper (top-down inference) or lower (bidirectional inference Kingma et al. (2016)) layer in the hierarchy. We also use *latent encoders*, for all but the first inference layer in the hierarchy. These components process the latent codes provided by the bijective layer at the upper layer, before it is passed to the posterior. In contrast to the

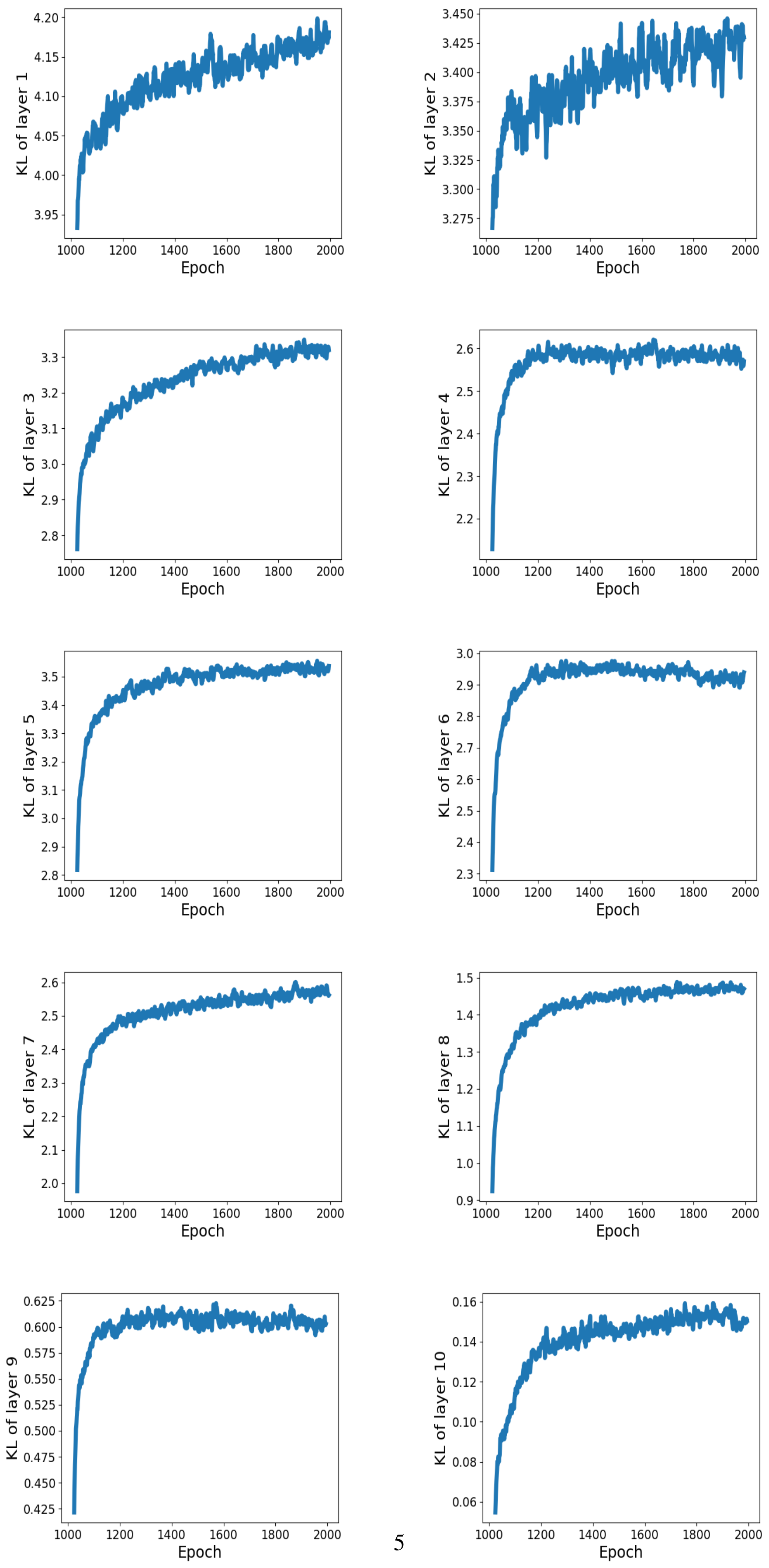


Figure S8: KL per layer.

parametrization adopted in Equations (13) and (14) of Kingma et al. (2016) for the mean and variance, which restricts the scale in $(0, 1)$ to ensure training stability, we use the following alternative that was found to be both more flexible and stable:

$$\sigma^2 = softplus(elu(\Sigma_{out})), \tag{S3}$$

where Σ_{out} is the network responsible for learning the scale of the distribution. According to equation S3, large positive entries are left unaffected, while negative outputs of Σ_{out} are first suppressed by the elu activation, and then mapped to a small positive value through the softplus transformation. A small offset is added to the small positive entries by the softplus to discriminate them by the negative outputs. A similar parametrization is used for the scale of the diagonal plus unit-rank affine transformations (to ensure positivity of the diagonal part and hence invertibility of the resulting bijective function).

Table S2: Architectural Hyperparameters of the MLP-SeRe VAE's layers for binarized MNIST

Component	Parameter	Value
Evidence Encoder	`# hidden layers`	`2`
	`hidden dimension`	`256`
	`feature size`	`20`
	`activation`	`Relu`
	`output activation`	`None`
Latent Encoder	`# hidden layers`	`2`
	`hidden dimension`	`256`
	`feature size`	`20`
	`activation`	`Relu`
	`output activation`	`None`
Posterior Layer (diagonal Gaussian) 2 identical networks (for loc and scale_diag) `MultivariateNormalDiag` in TFP (2018)	`# hidden layers`	`2`
	`hidden dimension`	`256`
	`activation`	`Relu`
	`output activation`	`None`
	`hidden feature size`	`3`
Bijective Layer (diagonal plus unit-rank affine) 3 identical networks (for shift, scale_diag, and scale_perturb_factor) `Affine` in TFP (2018)	`# hidden layers`	`2`
	`hidden dimension`	`20`
	`activation`	`Relu`
	`output activation`	`Relu`
Prior Layer (diagonal Gaussian) 2 identical networks (for loc and scale_diag) `MultivariateNormalDiag` in TFP (2018)	`# hidden layers`	`2`
	`hidden dimension`	`500`
	`activation`	`Relu`
	`output activation`	`None`
Decoder logit-based parametrization of `Bernoulli` in TFP (2018)	`# hidden layers`	`2`
	`hidden dimension`	`128`
	`activation`	`Relu`
	`output activation`	`None`
	`hidden feature size`	`10`

S.IV Experimental Details of the ResNet-SeRe VAE for binarized MNIST

This section provides detailed description of the training parameters, Table S4 and architectural hyperparameters, Table S3 for the experiments in Section 4.1.2 in the main paper. For the ResNet encoders and decoder, we use ResNet blocks, with batch normalization layers between them, that follow the design rule suggested in He et al. (2016): if the feature map size is halved, the number of filters is doubled, and reversely if the feature map size is doubled the number of filters is halved, so as to preserve the time complexity per layer.

In Figure S9a, we plot some samples from the generative network of the SeRe-VAE. In Figure S9c, we plot the reconstructed images from the latent codes of the images in Figure S9b.

Table S3: Architectural Hyperparameters of the ResNet-SeRe VAE's layers for binarized MNIST

Component	Parameter	Value
Evidence Encoder	`initial # filters`	16
	`# ResNet blocks`	2
	`ResNet blocks' scale`	$[\downarrow 2, \downarrow 2]$
	`feature size`	64
	`kernel size`	3
	`activation`	Relu
	`output activation`	None
Latent Encoder	`# hidden layers`	2
	`hidden dimension`	256
	`feature size`	20
	`activation`	Relu
	`output activation`	None
Posterior Layer (diagonal Gaussian) 2 identical networks (for loc and scale_diag) `MultivariateNormalDiag` in TFP (2018)	`# hidden layers`	2
	`hidden dimension`	256
	`activation`	Relu
	`output activation`	None
	`hidden feature size`	3
Bijective Layer (diagonal plus unit-rank affine) 3 identical networks (for shift, scale_diag, and scale_perturb_factor) `Affine` in TFP (2018)	`# hidden layers`	2
	`hidden dimension`	20
	`activation`	Relu
	`output activation`	Relu
Prior Layer (diagonal Gaussian) 2 identical networks (for loc and scale_diag) `MultivariateNormalDiag` in TFP (2018)	`# hidden layers`	2
	`hidden dimension`	256
	`activation`	Relu
	`output activation`	None
Decoder logit-based parametrization of `Bernoulli` in TFP (2018)	`initial # filters`	32
	`# ResNet blocks'`	2
	`ResNet blocks' scale`	$[\uparrow 2, \uparrow 2]$
	`kernel size`	3
	`activation`	Relu
	`output activation`	None
	`hidden feature size`	3

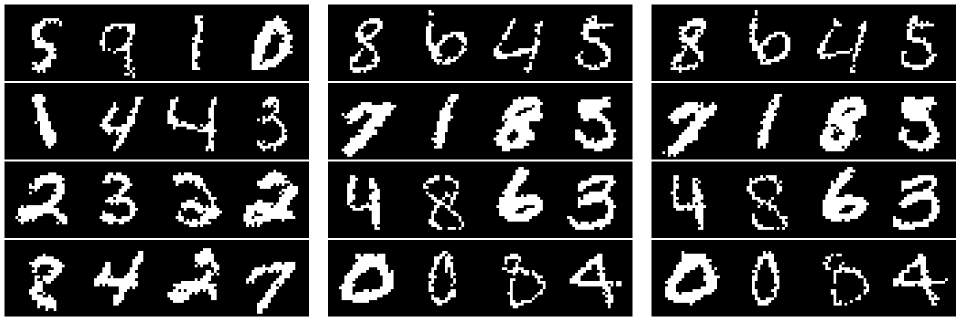
S.V SELF-REFLECTIVE NORMALIZING FLOWS

S.V.A DEFINITION AND BASICS

Normalizing flows (Tabak & Vanden-Eijnden, 2010; Tabak & Turner, 2013) are models for learning probability distributions based on iterative transformations of samples $\boldsymbol{u}$ drawn from a simple base distribution. Specifically, let $\boldsymbol{x} \in \mathbb{R}^D$ with $\boldsymbol{x} \sim p(\boldsymbol{x}; \boldsymbol{\gamma})$ the distribution of interest. A chain of T

Table S4: Training Hyperparameters of the ResNet-SeRe VAE's for binarized MNIST

Parameter	Value
batch size	`128`
warm up epochs	`256`
warm up schedule	`linear`
epochs	`1000`
learning rate	`1e-3`
batch normalization	`Yes`
kernel/bias regularization	`No`
kernel/bias initializer	`glorot normal`



(a) Generated MNIST digits (b) MNIST digits (c) Reconstructed MNIST digits

Figure S9: Qualitative Performance of the ResNet SeRe-VAE.

invertible transformations g^t parameterized by $\boldsymbol{\gamma}^t$ is applied on a sample $\boldsymbol{u}^0 \in \mathbb{R}^D$, drawn from a base distribution $\pi(\boldsymbol{u}; \boldsymbol{\gamma}^0)$ parameterized by $\boldsymbol{\gamma}^0$, such that:

$$\boldsymbol{u}^0 \sim \pi(\boldsymbol{u}; \boldsymbol{\gamma}^0),\ \boldsymbol{u}^t = g^t(\boldsymbol{u}^{t-1}; \boldsymbol{\gamma}^t)\ \forall t = 1, \dots, T, \text{with } \boldsymbol{x} \equiv \boldsymbol{u}^T, \tag{S4}$$

and $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}^0, \boldsymbol{\gamma}^1, \dots, \boldsymbol{\gamma}^T\}$. In the case of invertible and differentiable transformations g^t and differentiable $(g^t)^{-1}$, the change of variables formula (Rudin, 2006; Bogachev, 2007) provides a closed form for $p(\boldsymbol{x})$.

Normalizing flows were popularized for density estimation and variational inference by Dinh et al. (2015) and Rezende & Mohamed (2015), respectively. An extensive review on normalizing flows is provided in Papamakarios et al. (2019).

S.V.B Hierarchical Latent Variable Normalizing Flows

In order to capture high-dimensional dependencies, normalizing flows typically require a long sequence of transformations g^t and a large hidden dimension, two factors that introduce scalability issues. This fact motivates our design of *variational normalizing flows*. The latent variables $\boldsymbol{z}$ in this case can be incorporated in the flow in two ways: i) conditioning the base distribution by determining its parameters so that $\boldsymbol{\gamma}^0 \triangleq \boldsymbol{\gamma}^0(\boldsymbol{z}; \boldsymbol{c}^0_\gamma)$, and ii) conditioning the bijective transformations so that $\boldsymbol{u}^t = g^t(\boldsymbol{u}^{t-1}; \boldsymbol{\gamma}^t(\boldsymbol{z}))$. In the case of a Masked Autoregressive Flow (Papamakarios et al., 2017) or an Inverse Autoregressive Flow (Kingma et al., 2016), the latter amounts to designing *conditional* MADE layers (Germain et al., 2015) that account for a mask offset so that the additional inputs $\boldsymbol{z}$ are not masked out, see Section S.V.C. For the construction of the base distribution, amortized distributional layers are used which receive as input the latent codes of the $l-$th layer $\boldsymbol{z}_l$.

S.V.C Conditional Masked AutoEncoder

In this section, we describe the construction of *conditional Masked Autoencoders* (MADE layers) used as building blocks for the variational normalizing flow. We use notation identical to those used in Germain et al. (2015).

Let C be the dimension of the conditioning inputs. C acts as a *mask offset* in the construction of the masked autoregressive encoder, as we explain below. We assign unique numbers $1, 2, \dots, C + D$ to the inputs. In case of a random input ordering, the first C conditioning inputs are excluded so that $m^0(d) = d$, for $d = 1, 2, \dots, C$ and $m^0(d) \in \{C + 1, C + 2, \dots, C + D\}$ uniquely and randomly assigned to the inputs $d = C + 1, C + 2, \dots, C + D$. The degrees $m^l(d)$ of the $d-$th hidden unit of layer l should now be larger than C, so that the conditioning inputs are not masked out: the conditioning inputs are connected to all the hidden units. Therefore, $m^l(d)$ are random numbers such that $m^l(d) \in \{C + 1, C + 2, C + D\}$. Equation (12) in Germain et al. (2015) is still valid for the construction of the masks for connections from the input to the first layer hidden units, and from hidden units to next layer hidden units. For the last layer masks (from the hidden units to the output), Equation (13) in Germain et al. (2015) is used, and subsequently the first C, that refer to the conditioning inputs, masks are discarded.

Finally, as suggested in Papamakarios et al. (2017) batch normalization layers between the MAF steps are incorporated. Section B in the Appendix of Papamakarios et al. (2017), provides a description of the batch normalization as a bijector and the `tfp.bijectors.BatchNormalization` of TFP (2018) offers a suggested implementation. In our implementantion, at both training and validation/test time, we maintain averages over minibatches as in Ioffe & Szegedy (2015).

S.V.D Experimental Details of the variational SeRe MAF for CIFAR-10

Table S5 provides the architectural hyperparameters for the experiments in Section 4.2.2 in the main paper. The model consists of 5 layers of 40 latent variables each.

Table S5: Architectural Hyperparameters of the variational SeRe MAF for CIFAR-10

Component	Parameter	Value
Evidence Encoder	`initial # filters`	`16`
	`# ResNet blocks`	`3`
	`ResNet blocks' scale`	$[\downarrow 2, \downarrow 2, \downarrow 2]$
	`feature size`	`128`
	`kernel size`	`3`
	`dropout probability`	`0.5`
	`activation`	`Relu`
	`output activation`	`None`
Latent Encoder	`# hidden layers`	`2`
	`hidden dimension`	`100`
	`feature size`	`80`
	`activation`	`Relu`
	`output activation`	`None`

Posterior Layer (diagonal Gaussian) 2 identical networks (for loc and scale_diag) `MultivariateNormalDiag` in TFP (2018)	`# hidden layers`	`1`
	`hidden dimension`	`512`
	`activation`	`Relu`
	`output activation`	`None`
Posterior Layer - hidden feature maps 2 identical networks (for the residual connections of loc, scale_diag)	`# hidden layers`	`1`
	`hidden dimension`	`256`
	`feature size`	`40`
	`activation`	`Relu`
	`output activation`	`None`
Bijective Layer (diagonal plus unit-rank affine) 3 identical networks (for bin_widths, bin_heights, and knot_slopes) `RationalQuadraticSpline` in TFP (2018)	`# hidden layers`	`2`
	`hidden dimension`	`60`
	`activation`	`tanh`
	`output activation`	`tanh`
	`# bins`	`32`
	`# splines`	`5`
	`mask size`	`5`
	`range_min`	`-20`
Prior Layer (diagonal Gaussian) 2 identical networks (for loc and scale_diag) `MultivariateNormalDiag` in TFP (2018)	`# hidden layers`	`1`
	`hidden dimension`	`256`
	`activation`	`Relu`
	`output activation`	`None`
Decoder - base distribution (unit rank Gaussian) 3 identical networks (for loc, scale_diag,scale_perturb_factor) `MultivariateNormalDiagPlusLowRank` in TFP (2018)	`initial # filters`	`64`
	`# ResNet blocks'`	`3`
	`ResNet blocks' scale`	$[\uparrow 2, \uparrow 2, \uparrow 2]$
	`kernel size`	`3`
	`dropout probability`	`0.5`
	`activation`	`Relu`
	`output activation`	`None`
Decoder - hidden feature maps for the base distribution 3 identical networks (for the residual connections of : loc, scale_diag,scale_perturb_factor)	`# hidden layers`	`2`
	`hidden dimension`	`512`
	`feature size`	`100`
	`activation`	`Relu`
	`output activation`	`None`
Decoder - autoregressive bijector (MAF)	`# flows`	`2`
	`# MADEs/flow`	`5`
	`batch normalization`	`Yes`
	`MADE: # hidden layers`	`2`
	`MADE: hidden dimension`	`1024`
	`MADE: activation`	`Relu`
	`MADE: output activation`	`None`

`MADE: input order`	`random`
`MADE: hidden degrees`	`equal`

REFERENCES

Vladimir I Bogachev. *Measure theory*, volume 1. Springer Science & Business Media, 2007.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a Continuous Space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL*. ACL, 2016.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR, Workshop Track Proceedings*, 2015.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked Autoencoder for Distribution Estimation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, June 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*. PMLR, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR*, 2014.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved Variational Inference with Inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems 29*, 2016.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems 30*, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv preprint arXiv:1912.02762*, 2019.

Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015.

Danilo Jimenez Rezende and Fabio Viola. Taming VAEs. In *arXiv preprint arXiv:1810.00597*, 2018.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning , ICML*, 2014.

Walter Rudin. *Real and complex analysis*. Tata McGraw-hill education, 2006.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder Variational Autoencoders. In *Advances in Neural Information Processing Systems 29*, 2016a.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. How to train deep variational autoencoders and probabilistic ladder networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, 2016b.

E. G. Tabak and Cristina V. Turner. A family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, 66(2), 2013.

Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Commun. Math. Sci.*, 8, 2010. URL `https://projecteuclid.org:443/euclid.cms/1266935020`.

Team TFP. Tensorflow Probability, 2018–2019. *URL https://github. com/tensorflow/probability*, 2018.