LEARNING WITH MISELBO: THE MIXTURE COOKBOOK

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture models in variational inference (VI) is an active field of research. Recent works have established their connection to multiple importance sampling (MIS) through the MISELBO and advanced the use of ensemble approximations for large-scale problems. However, as we show here, an independent learning of the ensemble components can lead to suboptimal diversity. Hence, we study the effect of instead using MISELBO as an objective function for learning mixtures, and we propose the first ever mixture of variational approximations for a normalizing flow-based hierarchical variational autoencoder (VAE) with VampPrior and a PixelCNN decoder network. Two major insights led to the construction of this novel *composite model*. First, mixture models have potential to be off-the-shelf tools for practitioners to obtain more flexible posterior approximations in VAEs. Therefore, we make them more accessible by demonstrating how to apply them to four popular architectures. Second, the mixture components cooperate in order to cover the target distribution while trying to maximize their diversity when MISELBO is the objective function. We explain this cooperative behavior by drawing a novel connection between VI and adaptive importance sampling. Finally, we demonstrate the superiority of the Mixture VAEs' learned feature representations on both image and single-cell transcriptome data, and obtain state-of-the-art results among VAE architectures in terms of negative log-likelihood on the MNIST and FashionMNIST datasets. Code available here: `gitlink`.

1 INTRODUCTION

Adaptive importance sampling (AIS) methods aim at approximating a target distribution by a set of proposal densities, often interpreted as a mixture density (Elvira et al., 2019). However, choosing appropriate proposals is known to be a difficult problem, and so they are sequentially updated according to an adaptation scheme to reduce the mismatch with the target. The adaptive importance sampling methodology (Bugallo et al., 2017) is currently enjoying a great amount of research interest, e.g. regarding its convergence guaranties (Akyildiz and Míguez, 2021; Akyildiz, 2022), and is considered a promising contender for the MCMC methodology (Owen, 2013; Luengo et al., 2020).

Meanwhile, in variational inference (VI), the mismatch between a variational approximation and the target distribution is minimized via optimization of, typically, the Kullback-Leibler (KL) divergence. If one uses this optimization scheme to adapt the proposal densities in AIS, VI appears to be a special case in the AIS methodology. However, whereas VI is well-known in the machine learning (ML) community, AIS has received little attention until very recently.

The signal processing literature has recently started to see an increase of studies at the intersection between AIS and VI (Wang et al., 2019; 2021). For example, in El-Laham et al. (2019a), they obtain a mixture of proposal distributions via auto-differentiation VI (ADVI) (Kucukelbir et al., 2017) as a starting point for an AIS algorithm. After the adaptation, the mixture of proposals is again updated via ADVI.

Similar to AIS, mixture distributions in VI and variational autoencoders (VAEs; Kingma and Welling (2013)) is a research area which is also receiving a lot of attention (Nalisnick et al., 2016; Kucukelbir et al., 2017; Morningstar et al., 2021; Kviman et al., 2022) since they result in increased performance due to more flexible posterior approximations. Despite this, mixture models in VAEs are not considered standard tools. Instead, one would typically use normalizing flows (NFs; (Rezende and

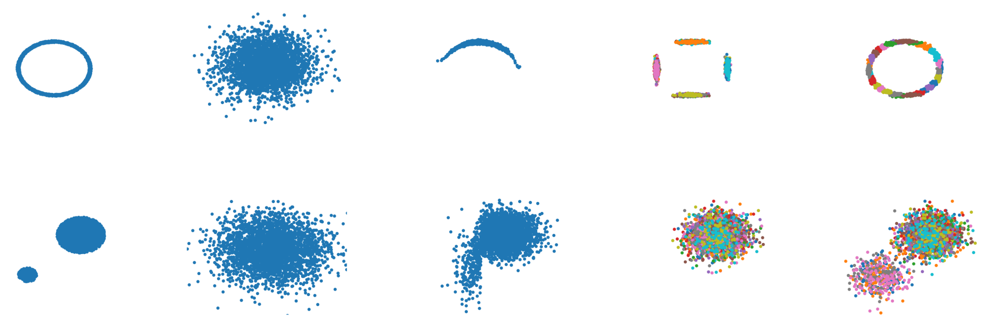

Figure 1: Each row is a different problem. (a) The two true distributions are shown in the leftmost column, followed by (b) a single approximation learned with an importance weighted version of the KL divergence, (c) an IAF, (d) an ensemble and (e) a mixture. The mixture is the only approximation capable of covering the true distribution. Colors represent samples from different components.

Mohamed, 2015; Papamakarios et al., 2021)), hierarchical models (Burda et al., 2015; Sønderby et al., 2016; Vahdat and Kautz, 2020), autoregressive models (Van Oord et al., 2016) or more complex prior distributions (Tomczak and Welling, 2018; Bauer and Mnih, 2019; Aneja et al., 2021). In this work, we argue that mixture models are indeed off-the-shelf solutions in VAEs. To that effect, we demonstrate that mixture models result in increased flexibility when combined with all of the four standard methods mentioned above. We also share our in-detail implementation instructions on how to combine mixtures with other popular tools in what we refer to as the *mixture cookbook*, found in Sec. 3. By doing so, we attempt to lower the barriers to using mixture models in VAEs.

The concepts of the Multiple Importance Sampling Evidence Lower Bound (MISELBO) and ensembles of variational approximations were established in Kviman et al. (2022). MISELBO offers a simple way to compute the ELBO for mixtures or ensembles inspired by the MIS literature (more in Sec. 2.2). To disambiguate between mixtures and ensembles, we refer to ensembles as compositions of independently learned approximations of the same target density (as in Kviman et al. (2022)), while mixtures will be considered to have learned their components jointly. When applied to VAEs, we call them *Ensemble VAEs* and *Mixture VAEs*, respectively. Moreover, we show in Sec. 4.1 and Sec. 4.3.1 that ensemble learning can lead to loss of diversity, while learning with MISELBO enforces diversity, and connect this to findings in the AIS literature.

Using our mixture cookbook, we propose a *composite model*; a hierarchical Mixture VAE with NFs, a VampPrior (Tomczak and Welling, 2018) and a PixelCNN decoder (Van Oord et al., 2016). In Sec. 4.2.1 we incrementally construct this model and successively evaluating the effect of adding each of the encoder-related architectures for different number of mixture components, S. The pattern is clear: adding mixture components improves the marginal log-likelihood estimates for all the considered models. Also, and multiple of the Mixture VAEs, including our composite model, achieve state-of-the-art scores on the MNIST dataset when compared to existing VAE architectures.

Finally, we show that the latent representations learned with Mixture VAEs are superior to those learned with Ensemble VAEs or the Vanilla VAE, measured on the downstream task of linear classification. Apart from images, the classification is also performed on single-cell transcriptome data by applying Ensemble and Mixture VAEs to the single-cell VI algorithm (Lopez et al., 2018).

We now summarize our contributions as follows:

- We show that learning with MISELBO is strongly connected to the adaptive mechanism that AIS algorithms use in order to learn the proposals.
- Based on the connection between AIS and VI, we argue (Sec. 2.3) and show (Sec. 4.1) that the mixture components in a Mixture VAE complement each other by jointly distributing themselves to cover the entire target distribution.

- Mixture VAEs should be considered a standard approach for obtaining more flexible posterior approximations. With this in mind, we
    1. empirically show that the negative log-likelihood scores are monotonically non-increasing (the lower the better) as we add more mixture components (Sec. 4.2.2).
    2. make Mixture VAEs more accessible with the mixture cookbook, which shows how to use a mixture of encoders in conjunction with three popular VAE architectures, namely hierarchical models, NFs, and the VampPrior (Sec. 3).
- We put forth a novel composite model: a Hierarchical Mixture VAE with NFs, a VampPrior and a PixelCNN decoder. The composite model outperforms the majority of existing VAE architectures in terms of log-likelihood estimates on the MNIST dataset.
- We extend the theorem from Kviman et al. (2022) to be valid for non-uniform mixture weights (Sec. 2.2).

2 Background

The main topics in this work are AIS, MISELBO and VAEs/amortized VI. In this section, we cover the first two as we expect that these are the least familiar concepts to the reader. For thorough descriptions of VAEs or amortized VI, we refer to Kingma et al. (2019) or Zhang et al. (2018), respectively. We end the section by drawing connections between AIS and MISELBO.

2.1 Adaptive Importance Sampling

AIS is a principled methodology for learning proposal densities in importance sampling (IS). The goal is generally learning the parameters of such proposals. There exist several families of AIS algorithms, depending on how the parameters are learnt. A common approach is to rely on the simulated samples and their IS weights in order to perform moment matching estimators (Cornuet et al., 2012; El-Laham et al., 2019b) or via resampling schemes (Cappé et al., 2004; 2008; Elvira and Chouzenoux, 2022). The IS weights are given by

$$w_{\text{IS}}^{L} = \frac{1}{L}\sum_{\ell=1}^{L}\frac{p_\theta(x, z_\ell)}{q_\phi(z_\ell|x)} \quad z_\ell \sim q_\phi(z|x), \tag{1}$$

if only one proposal $q_\phi(\cdot|x)$ is available, or by

$$w_{\text{MIS}}^{L,s} = \frac{1}{L}\sum_{\ell=1}^{L}\frac{p_\theta(x, z_{s,\ell})}{\sum_{j=1}^{S}\pi_j q_{\phi_j}(z_{s,\ell}|x)} \quad z_{s,\ell} \sim q_{\phi_s}(z|x) \tag{2}$$

if a mixture of proposals is used to simulate the samples (see alternative weighting schemes in Elvira et al. (2019)). The generality of the adaptation scheme provides the practitioner with high modeling flexibility. For instance, common adaptation schemes are gradient based (Elvira et al., 2015; Schuster, 2015) or resampling based (Cappé et al., 2004; 2008) (or both (Elvira and Chouzenoux, 2022)). As a particular example, in deterministic mixture population Monte Carlo (DM-PMC; Elvira et al. (2017a)), an equally weighted mixture proposal is iteratively adapted via sampling, weighting, and resampling steps. Unlike previous schemes, the importance weights are constructed by using the whole mixture in the denominator, i.e., as in the denominator of Eq. 2. This weighting scheme has shown superior performance both for reducing the variance of the estimators (Elvira et al., 2019) and for an increased diversity in the adaptive mechanism (Elvira et al., 2017b).

2.2 MISELBO

For a single mixture component, a tighter bound on marginal log-likelihood than the vanilla ELBO, $\mathcal{L}_{\text{ELBO}}$, is given by the importance weighted ELBO (IWELBO; Burda et al. (2015)) as

$$\mathcal{L}_{\text{IWELBO}}^{L}(x; q_\phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log\frac{1}{L}\sum_{\ell=1}^{L}\frac{p_\theta(x, z_\ell)}{q_\phi(z_\ell|x)}\right], \tag{3}$$

where $\mathcal{L}_{\text{ELBO}}$ is retrieved by letting $L = 1$. Recently Kviman et al. (2022) drew a connection between the extension of Eq. 3 to mixture approximations, and MIS. This resulted in an MIS scheme for computing the ELBO (where simulations are drawn from all mixture components without replacement), leading to the MISELBO objective:[1]

$$\mathcal{L}_{\text{MIS}}^{L}(x; q_\pi(z|x)) = \sum_{s=1}^{S} \pi_s \mathbb{E}_{q_{\phi_s}(z|x)} \left[\log \frac{1}{L} \sum_{\ell=1}^{L} \frac{p_\theta(x, z_{s,\ell})}{\sum_{j=1}^{S} \pi_j q_{\phi_j}(z_{s,\ell}|x)} \right]. \tag{4}$$

When S components are learned independently, it has been shown that forming a uniform mixture and joining their contributions via MISELBO offers a tighter bound than when averaging their individual vanilla-ELBO contributions (Kviman et al., 2022). Our first contribution in this work is to extend this result to weighted mixtures.

Theorem 1. *Given a weighted mixture of variational approximations, $q_\pi(z|x) = \sum_{s=1}^{S} \pi_s q_{\phi_s}(z|x)$, we have*

$$\mathcal{L}_{\text{MIS}}^{L}(x; q_\pi(z|x)) \geq \sum_{s=1}^{S} \pi_s \mathcal{L}_{\text{ELBO}}(x; q_{\phi_s}).$$

Proof. See Appendix A. □

For clarity of writing, we drop the arguments of the lower bounds in subsequent parts of this work.

2.3 An Adaptive Importance Sampling View of MISELBO

A connection between MIS and MISELBO has been recently established in Kviman et al. (2022). This can be seen, for instance by noting that $\mathcal{L}_{\text{MIS}}^{L}$ is a function of the mixture weights w_{MIS}^{L}. Also, we give more details in Sec. 2.2 about how to approximate $\mathcal{L}_{\text{MIS}}^{L}$ through MIS. However, the vision in Kviman et al. (2022) is limited since the set of proposals are given by independent learning processes. In this paper, we take a step further and use MISELBO to also learn the mixture components. We show that this approach is strongly connected to the adaptive mechanism that AIS algorithms use in order to learn the proposals.

DM-PMC mentioned in Sec. 2.1 is an AIS method that, at each iteration, builds an unweighted mixture proposal which approximates the target distribution. Thus, there is a close connection between learning mixtures in DM-PMC (and more generally in AIS) and doing so with MISELBO, when the mixture in Eq. 4 is equally weighted. The resemblance lies in how the samples are simulated from the mixture, but also in that the parameters of the proposals/variational approximations are adapted based on the mixture weights w_{MIS}^{L}. The difference is that, in DM-PMC, the parameters are adapted via resampling, whereas VI is gradient based. Nevertheless, there exist numerous gradient based AIS methods (Elvira et al., 2015; Schuster, 2015; Fasiolo et al., 2018; Elvira and Chouzenoux, 2022), some of them holding connections with variational inference, e.g., El-Laham et al. (2019a).

We now exploit the connection with AIS to better understand the behavior of the mixture components learned via MISELBO. In Elvira et al. (2017b), it was shown that due to the use of the mixture in the denominator of the importance weight, w_{MIS}^{L}, the components cooperate in order to cover the target distribution. In contrast, when adapting S proposal densities using w_{IS}^{L}, they could not cooperate and thus resulted in worse approximations of the target, as it happens for instance in Cappé et al. (2004) (see a broader discussion in Elvira et al. (2017a;b)). Based on this finding from the AIS literature, we expect that the same will happen in the VI setting. That is for a mixture learned with MISELBO, the components will cooperate to cover the target, while an ensemble learned with IWELBO will not. We will see that this is indeed the case in Sec. 4.1.

As in Cappé et al. (2004); Elvira et al. (2017b); Elvira and Chouzenoux (2022), we focus on uniform mixtures, which hold the benefit of not needing to select the mixture weights, simplifying also the sampling process. Uniform mixture weights are also motivated from the perspective of general applicability to VAEs. With uniform weights, the architecture of the encoder network does not need to be revised in order to map x to the mixture weights (e.g., as in Nalisnick et al. (2016)).

[1]An alternative name of the same lower bound arises naturally in the work of Morningstar et al. (2021) where the sampling scheme is instead viewed as stratified sampling, resulting in the name SELBO or SIWELBO.

3 The Mixture Cookbook

Here we propose the first systematic approach to optimize mixtures in three of the most popular VAE models, namely in (1) hierarchical models, (2) normalizing flows, and (3) VampPrior models. We dedicate a subsection to each of the tools, giving the explicit mixture-related notation that will help practitioners to avoid non-trivial pitfalls. In the following subsections, we consider the case with $L = 1$ without loss of generality, in order to avoid cluttered notation and since this is the most common setting during training.

3.1 How to Apply Hierarchical Models

Let H be the number of levels, then the sample from the hth layer is $z_s^h \sim q_{\phi_s^h}(z^h|z^{h+1})$. We compute the likelihood of the samples $z_s^1, ..., z_s^H$ with respect to the jth hierarchical variational approximation according to

$$q_{\phi_j}(\mathbf{z_s}|x) = q_{\phi_j^H}(z_s^H|x) \prod_{h=1}^{H-1} q_{\phi_j^h}(z_s^h|z_s^{h+1}). \tag{5}$$

Then, the hierarchical version of Eq. 4 is

$$\mathcal{L}_{\text{MIS}} = \sum_{s=1}^{S} \pi_s \mathbb{E}_{z_s^1,...,z_s^H \sim q_{\phi_s}(\mathbf{z_s}|x)} \left[\log \frac{p_\theta(x, \mathbf{z_s})}{\sum_{j=1}^{S} \pi_j q_{\phi_j}(\mathbf{z_s}|x)} \right]. \tag{6}$$

Key observation. In the sth term of Eq. 6, for each layer h, the samples must be drawn from the sth variational approximation. As a rule of thumb, all samples that appear in the sth term need to come from encoder s for the expectation to be properly evaluated.

3.2 How to Apply Normalizing Flows

In normalizing flows, a sample is drawn from a base distribution and then passed through the flow, a sequence of T deterministic and bijective transformations, $\{f_{\gamma_s^t}\}_{t=1}^T$, where γ_s^t are the learnable parameters of the tth mapping specific for the sth encoder network. We here let $q_{\phi_s}(z|x)$ denote the base distribution and $z_{s,j}^T$ be the output of the jth T long flow with $z_s := z_{s,j}^0$ as input. By selecting the transformations such that the corresponding Jacobian matrix is upper or lower triangular[2], we can, via the change-of-variables formula, express the jth mixture member's resulting density's goodness of fit to the sample from component s as

$$q_{\phi_j^T}(z_{s,j}^T|x) = q_{\phi_j}(z_s|x) \prod_{t=1}^{T} \left| \frac{df_{\gamma_j^t}(z_{s,j}^{t-1})}{dz_{s,j}^{t-1}} \right|^{-1}. \tag{7}$$

Key observation. Although the final latent variable is z_s^T, the expectations in Eq. 4 are still approximated by sampling from the base distributions (Rezende and Mohamed, 2015). This implies that when we want to compute the likelihood of z_s for a mixture of NF-based approximations, we only need to keep track of z_s in order to compute Eq. 7 for all j. That is, we do not need $\{z_{s,s}^t\}_{t=1}^T$ to compute the denominator of Eq. 4 in this setting, which drastically reduces the memory requirements.

3.3 How to Apply the VampPrior

The VampPrior models the prior as the variational approximation aggregated over K *pseudo inputs*, u_k, learned representations of the data. If the parametric form of the variational approximation is Gaussian, then the VampPrior is a mixture of Gaussians conditioned on different (pseudo) inputs.

We now show how to apply the VampPrior to ensembles of variational approximations. We get

$$p_\lambda(z) = \frac{1}{K} \sum_{k=1}^{K} q_\phi(z|u_k) = \frac{1}{K} \sum_{k=1}^{K} \sum_{s=1}^{S} \pi_s q_{\phi_s}(z|u_k), \tag{8}$$

[2]We direct the reader to Papamakarios et al. (2021) for in-depth details of NFs.

Table 1: The NLL results for an increasing number of mixtures (S) for the Mixture VAEs considered in Sec. 4.2.1 on the MNIST dataset. Our composite model (bottom row) is a composition of the other four models. A lower NLL metric is better, and the results are averages using three random seeds.

Model	$S = 1$	$S = 2$	$S = 3$	$S = 4$
Vanilla VAE	79.74 ± 0.03	79.15 ± 0.03	78.87 ± 0.03	78.60 ± 0.05
Hierarchical VAE	78.92 ± 0.08	78.25 ± 0.04	77.95 ± 0.04	77.75 ± 0.03
VAE w. NF	79.43 ± 0.03	78.74 ± 0.05	78.31 ± 0.02	78.02 ± 0.03
VAE w. VampPrior	78.69 ± 0.02	78.06 ± 0.02	77.74 ± 0.02	77.55 ± 0.05
Composite Model	78.45 ± 0.11	77.79 ± 0.14	77.35 ± 0.11	$\mathbf{77.03 \pm 0.11}$

where the first equality is the definition of the VampPrior for a generic variational approximation, whereas the second comes from using a mixture approximation. The pseudo inputs are generated by a neural network with learnable parameters and takes a K-dimensional identity matrix as input.

Key observation. We cannot use S separate prior distributions, one for each variational approximation. This is crucial, as it would violate the criterion that all variational approximations need to share the same target distribution (Kviman et al., 2022). In practice, one way to achieve this is by using a single pseudo-input generator and computing the above expression.

4 Experiments

4.1 Two-Dimensional Examples

We now demonstrate the benefits of the type of flexibility obtained with mixtures for density estimation, contrasted to an importance weighted approximation (IWVI), IAF, and ensembles. We show that, in contrast to standard ensemble strategies, the coordinated learning of the mixture allows for the cooperation of the components to jointly approximate the target distribution. This can be easily understood by highlighting the connection between our strategy and relevant AIS algorithms, as discussed in Sec.2.3. The expected cooperative behavior holds in two difficult cases: (i) when $p(z)$ is a ring-shaped density, and (ii) when the target is an unevenly weighted bi-modal Gaussian, as in the second row in Fig. 1.(a) where we have $p(z) = 0.8\mathcal{N}(\mathbf{1}, 0.1I_2) + 0.2\mathcal{N}(-\mathbf{1}, 0.1I_2)$.

In Fig. 1 we compare an $S = 30$-component mixture (e) to ensembles of variational approximations (d) with $S = 30$, trained with $L = 1$; an inverse autoregressive flow (IAF; Kingma et al. (2016)) (c) with $T = 30$ flows and a two-layered MADE (Germain et al., 2015) with 10 hidden units in each layer; IWVI (b), a single variational approximation with $L = 30$; the true distributions, $p(z)$. The IAF is by far the most advanced model of the four approximation methods, and was given time to converge. All approximations and the base distribution for IAF were Gaussian or a mixture of Gaussian's with learnable parameters. See Sec. B for more details.

The results demonstrate that the mixture coordinates its components to spread out and cover $p(z)$, while the ensemble overpopulates certain parts of the target. For example in the bi-modal Gaussian case (lower row in Fig. 1), the mixture distributes its components between the modes proportionally to $p(z)$ (Fig. 1e). Meanwhile, the ensemble fails to cover the lighter mode (Fig. 1d). Interestingly, using $L = 30$ for a single approximation seems to cause it to become indecisive (Fig. 1b). This does not happen when $L = 1$, as is clear in the ensemble case (Fig. 1d). Finally, despite the large complexity of the IAF, it was not flexible enough to capture the target distributions (Fig. 1c).

4.2 Log-Likelihood Estimation

Here we use the marginal log-likelihood in order to quantify the effect of applying Mixture VAEs to a set of popular VAE algorithms. This is easily achieved through the guidance of our mixture cookbook in Sec. 3. Inspired by Kviman et al. (2022), the mixture components were modelled with separate encoder nets. In the final subsection, we critically analyze the source of the results in Sec. 4.2.1 with some adversary examples. See Sec. C for in-depth experimental details.

Table 2: The NLL and bits-per-dimension results for an increasing number of mixtures (S) for a subset of the Mixture VAEs considered in Sec. 4.2.1 on the FashionMNIST and CIFAR-10 datasets. The lower the better.

Dataset	Model	$S = 1$	$S = 2$	$S = 3$	$S = 4$
FashionMNIST	Vanilla VAE	225.50	224.715	224.39	224.22
FashionMNIST	Composite Model	223.58	222.91	222.37	**222.13**
CIFAR-10	Vanilla VAE	4.85	4.85	4.84	**4.83**

4.2.1 Ablation Study

In this section, we incrementally construct *the composite model*, i.e. a Mixture VAE, which incorporates all popular techniques discussed in this work; hierarchical models, NFs and the VampPrior. Consequently, we perform a large ablation study consisting of 20 different models. We ran each model three times and averaged their results. In Table 1, we report the effect of applying mixtures to each of the architectures for MNIST.

The Vanilla VAE exactly followed the architecture of the PixelCNN (Van Oord et al., 2016) decoder + CNN encoder nets with gating mechanisms (Dauphin et al., 2017) described in Tomczak and Welling (2018), but without hierarchies. The latent space was 40 dimensions. Note that this was the base architecture for all subsequent models. The Hierarchical VAE is the two-layered PixelCNN VAE from (Tomczak and Welling, 2018), without the VampPrior. In the VAE with NF, we employed the IAF with $T = 2$ flows and a two-layered MADE with 320 hidden units in each layer, as in the original paper (Kingma et al., 2016). The VAE with VampPrior is the Vanilla VAE with the addition of a $K = 500$ VampPrior (Tomczak and Welling, 2018). Finally, we implement the composite model. The IAFs were placed in the lower-level latent space, where the prior was a normal distribution with learnable parameters. The VampPrior was implemented in the upper-level latent space. The explicit form of the objective function is in Eq. 16.

As part of our ablation study, we removed the IAFs from the composite model, and, as expected, reproduced the reported performance of the Hierarchical VAE w. VampPrior in Tomczak and Welling (2018), also shown in Table 3.

We report mean and standard deviations (stds) from three runs in Table 1. While the composite model has the largest stds, they are, in general small, and we emphasize that we cannot identify a pattern between increased S and stds. Moreover, the improved NLLs from increasing S are consistently within the error margin of a single std for all architectures (visualized in Fig. 2).

4.2.2 Empirical Monotonicity of Mixture VAEs

Here we describe a novel and impactful finding. In all our experiments, we found that the NLL is a non-increasing function of S. Thus, to improve the NLL scores, it is sensible to simply incorporate a mixture of encoders into an existing model. The power of combining mixtures with popular VAE architectures is clearly demonstrated in tables 1 and 2. These results promote future research on Mixture VAEs, and mixture approximations in VI in general.

Table 3: NLL statistics for different state-of-the-art VAE architectures on the MNIST dataset.

Model	NLL
Hierarchical VAE w. VampPrior (Tomczak and Welling, 2018)	78.45
NVAE (Vahdat and Kautz, 2020)	78.01
PixelVAE++ (Sadeghi et al., 2019)	78.00
MAE (Ma et al., 2019)	77.98
Ensemble NVAE (Kviman et al., 2022)	77.77 ± 0.2
Vanilla VAE ($S = 10$; **our**)	77.82
Composite Model ($S = 4$; **our**)	$\mathbf{77.03 \pm 0.1}$

Table 4: **MNIST:** Accuracy for linear classification using the latent representations for an increasing number of mixture components (S).

Model	$S = 1$	$S = 2$	$S = 3$	$S = 4$	$S = 5$
VAE	94.9±4E-3%	94.8±4E-3%	94.9±4E-3%	94.8±5E-3%	94.8±5E-3%
Ensemble VAE	**95.5±4E-3%**	95.7±4E-3%	95.4±2E-3%	95.9±2E-3%	96.0±2E-3%
Mixture VAE	**95.5±4E-3%**	**95.8±3E-3%**	**96.1±2E-3%**	**96.5±3E-3%**	**96.6±2E-3%**

To stress test the monotonicity claim, we trained Vanilla VAEs with all the way up to $S = 10$ encoder networks, see Fig. 3 for an illustration. For every added component, we saw an improvement in NLL. Although the relative improvement of adding more components decreased after $S = 6$, and the possible improvement is upper bounded in theory, we could not find a case where larger S did not give an improvement in this setup.

Moreover, to the best of our knowledge, impressively many of the Mixture VAEs, especially our composite model, achieve state-of-the-art results on the MNIST dataset compared to other VAE architectures, as shown in Table 3.[3]

4.2.3 BIG IWAE VERSUS MIXTURE VAE

Indeed, the number of encoder network parameters increases as a function of S, and the Mixture VAE uses $S \times L$ samples to approximate its objective function. Hence, it is necessary to explore whether the improved NLL scores obtained by the Mixture VAEs are due to the mixture approximations or due to the increased number of parameters or the number of samples. In general, we found that the Big IWAE could not outperform the corresponding Mixture VAE regardless if we increased the number of hidden layers or the number of hidden units. Conversely, we found that naively increasing the number of parameters in the Big IWAE sometimes resulted in worse NLLs. Meanwhile, naively adding mixture components works well in this regard. See D for more details and results.

4.3 LATENT REPRESENTATIONS

Here we evaluate the quality of the latent representations produced by Mixture VAEs and compare it to the Vanilla VAEs and Ensemble VAEs. Our evaluation is quantitative and, as is common (Sinha and Dieng, 2021; Locatello et al., 2019), assessed by performing a linear classification on the representations. We use two diverse types of data, single cell data and images. The linear classifier was a linear SVM from the Scikit library (Pedregosa et al., 2011).

Table 5: The JSD is upper bounded by $\log S$, and the higher it is the more diverse the mixtures/ensembles are. As expected, the mixtures are the most diverse.

Model	$S = 2$	$S = 3$	$S = 4$	$S = 5$
Ensemble VAE	0.31	0.46	0.54	0.59
Mixture VAE	**0.45**	**0.70**	**0.89**	**1.01**
$\log S$	0.69	1.10	1.39	1.61

A VAE provides the practitioner with mainly two alternatives for representing x in lower dimensions, either via sampling z conditioned on x (Lopez et al., 2018) or via the parameters of the approximate distribution (Dieng et al., 2019). Encouraged by our results in Sec. 4.1, we hypothesise that the Mixture VAE learns complex representations of the data by coordinating its components such that each component contributes in a non-trivial way. To exploit this, we feed the mixture parameters into the classification nets. For the baselines that use single Gaussian approximations, we instead sample multiple times from $q_\phi(z|x)$, such that the number of training features for the classifiers is the same.

4.3.1 IMAGES

The VAE architectures here are two-layered MLP versions from (Tomczak and Welling, 2018), and are trained according to the same scheme as described in Sec. C. The experiment is carried out on the MNIST and FashionMNIST datasets. In addition to the obtained accuracies, we also quantify the

[3]Explicitly comparing with VAE architectures excludes some brilliant works, such as consistency regularization in VAEs (Sinha and Dieng, 2021), which is a regularizing term in the training objective function.

Table 6: **scVI:** Accuracy for linear classification using the latent representations for an increasing number of mixture components (S).

Model	$S = 1$	$S = 2$	$S = 3$	$S = 4$	$S = 5$
VAE (scVI)	94.2±.02%	96.1±.02%	95.9±.01%	95.5±.01%	96.1±.02%
Ensemble VAE	**94.7±.02%**	**96.4±.02%**	**96.2±.01%**	96.5±.02%	96.8±.01%
Mixture VAE	**94.7±.02%**	95.9±.01%	**96.2±.01%**	**97.0±.01%**	**97.1±.01%**

diversity of the mixtures and ensembles in terms of the general Jensen-Shannon divergence (JSD) for uniform mixtures, using Eq. 19. The JSD is non-negative, upper bounded by $\log S$, and is maximized when all components have non-overlapping supports.

From Tables 4 and 8 it is clear that the Mixture VAEs are superior on this task for all $S > 1$, and in Table 5 we show that the Mixture VAEs obtains more diverse approximations than the ensemble VAE as measured by in JSD on the MNIST dataset. Using the JSD results, we are also able to explain the stagnating accuracies by the Ensemble VAEs. The JSDs are barely increasing with S, meaning that more components are not necessarily contributing with more information.

4.3.2 Single-cell data

The scVI algorithm (Lopez et al., 2018) is a deep generative model for single-cell data, and is a state-of-the-art VAE algorithm that incorporates a probabilistic model for single-cell transcriptomics. We implement it using scVI-tools (Gayoso et al., 2022). This experiment is carried out on the mouse cortex cells dataset (CORTEX) (Zeisel et al., 2015), which consists of 3005 cells and labels for seven distinct cell types. We used the top 1.2k genes, ordered by variance. Table 6 shows the accuracy results for the linear classifier on the latent representations from scVI. Mixture VAE outperforms the baselines for $S > 3$, however, the Ensemble VAE is also able to utilize its representations for lower S. where it performs the best. In Sec. E.2 we show the supremacy of Mixture VAEs on clustering tasks.

5 Related Work

Mixture VAEs have previously appeared in the literature as means to obtain more flexible posterior approximations (Pires and Figueiredo, 2020; Morningstar et al., 2021). As far as we are aware, they appeared first in Nalisnick et al. (2016) who introduced a Gaussian mixture model latent space placed a Dirichlet prior on the mixture weights. The mixture weights for the mixture approximation were then inferred via a neural network mapping from x to the simplex. In another example (Roeder et al., 2017), a weighted mixture ELBO, more similar to MISELBO with $L = 1$, was used to train a VAE. They showed that stop gradients could be applied to Mixture VAEs, and discussed how to sample from the mixture in order to approximate the ELBO. The latter Mixture VAE was only applied to a toy data set. Although these are important works on Mixture VAEs, there are no earlier examples in the literature where the general applicability of Mixture VAEs has been demonstrated. We show that Mixture VAEs can be off-the-shelf tools, and provide the *mixture cookbook*, the first of its kind.

Outside the VAE literature, mixture learning with variational objectives have been presented in for example automatic differentiation VI (Kucukelbir et al., 2017) and Thompson sampling based bandits (Wang and Zhou, 2020). Finally, our composite model build on the architectures presented in Tomczak and Welling (2018) and (Kingma et al., 2016), however our model is a novel composition of the two methods applied to Mixture VAEs.

6 Conclusions

In this work we have shown that mixture components cooperate to cover the target distribution when learned with MISELBO, and connect this to the AIS methodology. Furthermore, we have made Mixture VAEs more accessible by providing the mixture cookbook, which shows how to train mixtures of encoders for advanced VAE methods. Finally, we put forth a novel Mixture VAE, the composite model, which outperforms most of the state-of-the-art methods on the MNIST dataset.

REFERENCES

Akyildiz, Ö. D. (2022). Global convergence of optimized adaptive importance samplers. *arXiv preprint arXiv:2201.00409*.

Akyildiz, Ö. D. and Míguez, J. (2021). Convergence rates for optimised adaptive importance samplers. *Statistics and Computing*, 31(2):1–17.

Aneja, J., Schwing, A., Kautz, J., and Vahdat, A. (2021). A contrastive learning approach for training variational autoencoder priors. *Advances in Neural Information Processing Systems*, 34.

Bauer, M. and Mnih, A. (2019). Resampled priors for variational autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 66–75. PMLR.

Bugallo, M. F., Elvira, V., Martino, L., Luengo, D., Miguez, J., and Djuric, P. M. (2017). Adaptive importance sampling: The past, the present, and the future. *IEEE Signal Processing Magazine*, 34(4):60–79.

Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

Cappé, O., Douc, R., Guillin, A., Marin, J. M., and Robert, C. P. (2008). Adaptive importance sampling in general mixture classes. *Statistics and Computing*, 18:447–459.

Cappé, O., Guillin, A., Marin, J. M., and Robert, C. P. (2004). Population Monte Carlo. *Journal of Comp. and Graphical Statistics*, 13(4):907–929.

Cornuet, J., Marin, J.-M., Mira, A., and Robert, C. P. (2012). Adaptive multiple importance sampling. *Scandinavian Journal of Statistics*, 39(4):798–812.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

Dieng, A. B., Kim, Y., Rush, A. M., and Blei, D. M. (2019). Avoiding latent variable collapse with generative skip models. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2397–2405. PMLR.

El-Laham, Y., Djurić, P. M., and Bugallo, M. F. (2019a). A variational adaptive population importance sampler. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5052–5056. IEEE.

El-Laham, Y., Martino, L., Elvira, V., and Bugallo, M. F. (2019b). Efficient adaptive multiple importance sampling. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE.

Elvira, V. and Chouzenoux, E. (2022). Optimized population monte carlo. *IEEE Transactions on Signal Processing (to appear in)*.

Elvira, V., Martino, L., Luengo, D., and Bugallo, M. F. (2017a). Improving population monte carlo: Alternative weighting and resampling schemes. *Signal Processing*, 131:77–91.

Elvira, V., Martino, L., Luengo, D., and Bugallo, M. F. (2017b). Population monte carlo schemes with reduced path degeneracy. In *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 1–5. IEEE.

Elvira, V., Martino, L., Luengo, D., and Bugallo, M. F. (2019). Generalized multiple importance sampling. *Statistical Science*, 34(1):129–155.

Elvira, V., Martino, L., Luengo, L., and Corander, J. (2015). A gradient adaptive population importance sampler. In *IEEE International Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4075–4079.

Englesson, E. and Azizpour, H. (2021). Generalized jensen-shannon divergence loss for learning with noisy labels. *Advances in Neural Information Processing Systems*, 34.

Fasiolo, M., de Melo, F. E., and Maskell, S. (2018). Langevin incremental mixture importance sampling. *Stat. Comput.*, 28(3):549–561.

Gayoso, A., Lopez, R., Xing, G., Boyeau, P., Valiollah Pour Amiri, V., Hong, J., Wu, K., Jayasuriya, M., Mehlman, E., Langevin, M., Liu, Y., Samaran, J., Misrachi, G., Nazaret, A., Clivio, O., Xu, C., Ashuach, T., Gabitto, M., Lotfollahi, M., Svensson, V., da Veiga Beltrame, E., Kleshchevnikov, V., Talavera-López, C., Pachter, L., Theis, F. J., Streets, A., Jordan, M. I., Regier, J., and Yosef, N. (2022). A python library for probabilistic analysis of single-cell omics data. *Nature Biotechnology*.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kingma, D. P., Welling, M., et al. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. (2017). Automatic differentiation variational inference. *Journal of machine learning research*.

Kviman, O., Melin, H., Koptagel, H., Elvira, V., and Lagergren, J. (2022). Multiple importance sampling elbo and deep ensembles of variational approximations. In *International Conference on Artificial Intelligence and Statistics*, pages 10687–10702. PMLR.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. (2019). Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR.

Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., and Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. *Nature methods*, 15(12):1053–1058.

Luengo, D., Martino, L., Bugallo, M., Elvira, V., and Särkkä, S. (2020). A survey of monte carlo methods for parameter estimation. *EURASIP Journal on Advances in Signal Processing*, 2020(1):1–62.

Ma, X., Zhou, C., and Hovy, E. (2019). Mae: Mutual posterior-divergence regularization for variational autoencoders. *arXiv preprint arXiv:1901.01498*.

Morningstar, W., Vikram, S., Ham, C., Gallagher, A., and Dillon, J. (2021). Automatic differentiation variational inference with mixtures. In *International Conference on Artificial Intelligence and Statistics*, pages 3250–3258. PMLR.

Nalisnick, E., Hertel, L., and Smyth, P. (2016). Approximate inference for deep latent gaussian mixtures. In *NIPS Workshop on Bayesian Deep Learning*, volume 2, page 131.

Owen, A. B. (2013). Monte carlo theory, methods and examples.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Pires, G. G. F. and Figueiredo, M. A. (2020). Variational mixture of normalizing flows. In *ESANN*.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

Roeder, G., Wu, Y., and Duvenaud, D. K. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. *Advances in Neural Information Processing Systems*, 30.

Sadeghi, H., Andriyash, E., Vinci, W., Buffoni, L., and Amin, M. H. (2019). Pixelvae++: Improved pixelvae with discrete prior. *arXiv preprint arXiv:1908.09948*.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*.

Schuster, I. (2015). Gradient importance sampling. *arXiv:1507.05781*, pages 313–316.

Sinha, S. and Dieng, A. B. (2021). Consistency regularization for variational auto-encoders. *Advances in Neural Information Processing Systems*, 34.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). Ladder variational autoencoders. *Advances in neural information processing systems*, 29.

Thin, A., Kotelevskii, N., Doucet, A., Durmus, A., Moulines, E., and Panov, M. (2021). Monte carlo variational auto-encoders. In *International Conference on Machine Learning*, pages 10247–10257. PMLR.

Tomczak, J. and Welling, M. (2018). Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR.

Vahdat, A. and Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679.

Van Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR.

Wang, H., Bugallo, M. F., and Djurić, P. M. (2019). Adaptive importance sampling supported by a variational auto-encoder. In *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 619–623. IEEE.

Wang, H., Bugallo, M. F., and Djurić, P. M. (2021). Adaptive importance sampling via auto-regressive generative models and gaussian processes. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5584–5588. IEEE.

Wang, Z. and Zhou, M. (2020). Thompson sampling via local uncertainty. In *International Conference on Machine Learning*, pages 10115–10125. PMLR.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Zeisel, A., Muñoz-Manchado, A. B., Codeluppi, S., Lönnerberg, P., La Manno, G., Juréus, A., Marques, S., Munguba, H., He, L., Betsholtz, C., et al. (2015). Cell types in the mouse cortex and hippocampus revealed by single-cell rna-seq. *Science*, 347(6226):1138–1142.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. (2018). Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026.

A Proof of Theorem 1

What we set out to prove is the following

$$\sum_{s=1}^{S} \pi_s \mathbb{E}_{q_{\phi_s}(z|x)} \left[\log \frac{p_\theta(x, z_s)}{\sum_{j=1}^{S} \pi_j q_{\phi_j}(z_s|x)} \right] \geq \sum_{s=1}^{S} \pi_s \mathbb{E}_{q_{\phi_s}(z|x)} \left[\log \frac{p_\theta(x, z_s)}{q_{\phi_s}(z_s|x)} \right]. \tag{9}$$

Proof. We will make use of the non-negativity property of the (generalized) JSD (Englesson and Azizpour, 2021). Rearranging the terms, we get

$$\sum_{s=1}^{S} \pi_s \mathbb{E}_{q_{\phi_s}(z|x)} \left[\log \frac{p_\theta(x, z_s)}{\sum_{j=1}^{S} \pi_j q_{\phi_j}(z_s|x)} \right] - \sum_{s=1}^{S} \pi_s \mathbb{E}_{q_{\phi_s}(z|x)} \left[\log \frac{p_\theta(x, z_s)}{q_{\phi_s}(z_s|x)} \right] = \tag{10}$$

$$- \sum_{s=1}^{S} \pi_s \mathbb{E}_{q_{\phi_s}(z|x)} \left[\log \sum_{j=1}^{S} \pi_j q_{\phi_j}(z_s|x) \right] + \sum_{s=1}^{S} \pi_s \mathbb{E}_{q_{\phi_s}(z|x)} \left[\log q_{\phi_s}(z_s|x) \right] = \tag{11}$$

$$\mathbb{H} \left[\sum_{s=1}^{S} \pi_s q_{\phi_s}(z_s|x) \right] - \sum_{s=1}^{S} \pi_s \mathbb{H} \left[q_{\phi_s}(z_s|x) \right] = \text{JSD}(\{q_{\phi_s}, \pi_s\}_{s=1}^{S}) \geq 0, \tag{12}$$

where the final inequality is due to the non-negativity of the JSD. □

B Two Dimensional Experiments

In both experiments we consider a complicated $p(z)$. All approximations were initialized at $z_1 = z_2 = 0$ and their parameters were optimized using the ADAM optimizer (Kingma and Ba, 2014) with learning rate 0.1.

The objective function to be minimized was either the KL divergence between the approximation and $p(z)$, or, for the single approximation with $L = 30$, an importance-weighted variant of the KL

$$\mathcal{D}_L = -\mathbb{E}_{z_\ell \sim q_\phi(z)} \left[\log \frac{1}{L} \sum_{\ell=1}^{L} \frac{p(z_\ell)}{q_\phi(z_\ell)} \right]. \tag{13}$$

We believe that the effects of minimizing $\mathcal{D}_L$ are not well studied, although, one instance of $\mathcal{D}_L = 0$ is clearly when the approximation exactly matches $p(z)$. Future work will hopefully bring interesting insights and enable us to explain the behavior exhibited in the experiments by the importance weighted approximation (IWVI).

All approximations were initialized at $(0, 0)$, but the same behavior/solutions were obtained for the mixture approximation for alternative starting points. In some cases, a reduced learning rate helped the mixture approximations.

Finally, when plotting $p(z)$ we uniformly sampled z in a square which captured their supports. We then plotted the points with likelihood more than 0.1, i.e. $p(z) > 0.1$.

C Log-Likelihood Estimation Details

For MNIST (LeCun and Cortes, 2010) and FashionMNIST (Xiao et al., 2017) we used $p_\theta(x|z) = \text{Bernoulli}(x; \theta)$, assuming independence across the pixels, while a 3-component mixture of discretized Logistic distributions (Salimans et al., 2017) was used for CIFAR-10 (Krizhevsky et al., 2009). To implement the mixture of discretized Logistic distributions in PyTorch, we used the provided code from Vahdat and Kautz (2020). Otherwise in general, our code was largely inspired by the one provided by Tomczak and Welling (2018).

As is standard, we computed the bits per dimension scores as

$$\text{BPD}(x; \theta) = \frac{\log p_\theta(x)}{d_x \log(2)}, \tag{14}$$

where $d_x = 32^2 \cdot 3$ for CIFAR-10 and the marginal log-likelihood was approximated via MISELBO.

Regarding data augmentation, we used horizontal flips for the CIFAR-10 training data (as in Vahdat and Kautz (2020)), and Bernoulli draws for the pixel values for the MNIST and FashionMNIST data (as in Tomczak and Welling (2018)).

To approximate the marginal log-likelihood, we used $L = 5000$ and $L = 100$ importance samples for MNIST/FashionMNIST and CIFAR-10, respectively. These choices were based on examples in the literature (Tomczak and Welling, 2018; Vahdat and Kautz, 2020).

For all models when training on the MNIST or FashionMNIST datasets, we utilized a linearly increasing KL warm-up (Sønderby et al., 2016) for the first 100 epochs, early stopping with 100 epochs look ahead, and the ADAM optimizer (Kingma and Ba, 2014) with a learning rate of $0.5 \cdot 10^{-4}$ and no weight decay. Observe that this is the training scheme used in Tomczak and Welling (2018) but with longer look ahead. For CIFAR-10 we employed an identical scheme but trained for a maximum of 1000 epochs with no look-ahead mechanism. In all setups here, we used a training and validation batch size of 100.

The models used for MNIST and FashionMNIST had 40-dimensional latent spaces, while the CIFAR-10 related models had 124-dimensional latent spaces (as in Thin et al. (2021)). Furthermore, the "vanilla" VAEs used on CIFAR-10 had CNN encoder networks along with a PixelCNN decoder.

For all large-scale experiments we used a single 32-GB Tesla V100 GPU, or a desktop computer with a 10-GB RTX 3080 GPU.

As mentioned in Sec. 4.2.1, during training of the models we employed KL warm-up. For the vanilla Mixture VAE, this results in the following objective function

$$\mathcal{L}_{\text{MIS}}^{\beta} = \sum_{s=1}^{S} \pi_s \mathbb{E}_{q_{\phi_s}} \left[\log p_\theta(x|z_s) \right] + \beta \sum_{s=1}^{S} \pi_s \mathbb{E}_{q_{\phi_s}} \left[\log \frac{p_\theta(z_s)}{\sum_{j=1}^{S} \pi_j q_{\phi_j}(z_s|x)} \right], \tag{15}$$

where β linearly increased from zero to one over the first 100 epochs.

Unlike NVAE (Vahdat and Kautz, 2020), to name another large VAE model, our models were trained on single GPU nodes. Parallelizing the encoder operations, for instance, over multiple GPUs is easily achieved and will speed up the training process considerably. The $S = 1$ composite model trained for 9 hours, while the biggest model, the $S = 4$ composite model, trained for 52 hours and had 5.2 million network parameters. For more info regarding the NVAE architecture, hyperparameters and run time, see here `https://github.com/NVlabs/NVAE#running-the-main-nvae-training-and-evaluation-scripts.`.

Finally, we provide the formulation of MISELBO with $L = 1$ for the composite model

$$\mathcal{L}_{\text{MIS}} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{E}_{z_s^1, z_s^2 \sim q_{\phi_s}(\mathbf{z_s}|x)} \left[\log \frac{p_\theta(x|z_{s,s}^{T,1}, z_s^2) p_\theta(z_{s,s}^{T,1}|z_s^2) \frac{1}{KS} \sum_{k=1}^{K} \sum_{j=1}^{S} q_{\phi_j}(z_s^2|u_k)}{\frac{1}{S} \sum_{j=1}^{S} q_{\phi_j^2}(z_s^2|x) q_{\phi_j^1}(z_{s,j}^{T,1}|z_s^2, x)} \right]. \tag{16}$$

In this work we used uniform mixture weights, however the expression above can trivially be extended to non-uniform weights.

D Big IWAE versus Mixture VAE

Indeed, the number of encoder network parameters increases as a function of S, and the Mixture VAE uses $S \times L$ samples to approximate its objective function. Hence, it is necessary to explore whether the improved NLL scores by the Mixture VAEs are due to the mixture approximations, or due to the increased number of parameters and the number of importance samples. In general, we found that the Big IWAE could not outperform the corresponding Mixture VAE regardless if we increased the number of hidden layers or the number of hidden units.

In order to do restrict the hyper-parameter space, we perform this analysis in isolation from the experiments in the main text and consider simpler VAEs. Namely, we employ the vanilla MLP VAE from (Tomczak and Welling, 2018), without the gating mechanisms. This gives the encoder networks of the Mixture VAE two hidden layers with 300 hidden units each. We trained the Mixture VAEs

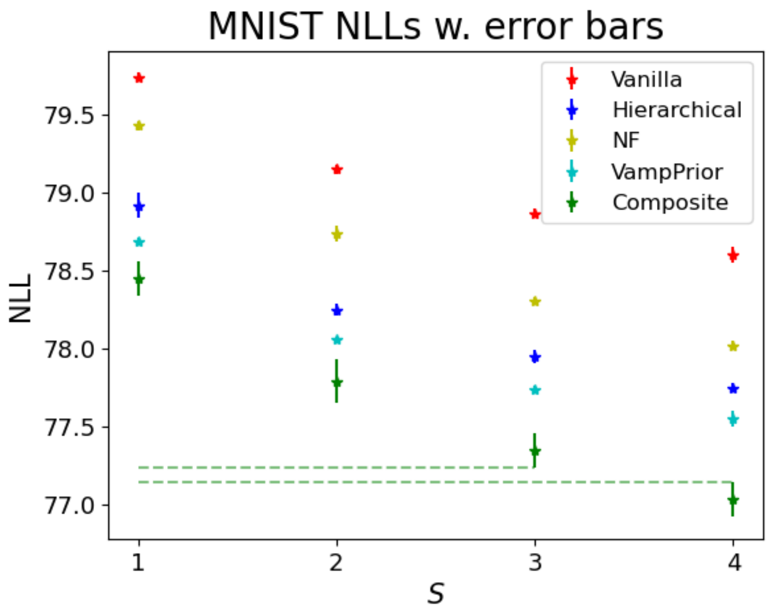

Figure 2: Error bars using a single standard deviation for the NLL scores on the MNIST dataset. Note that none of the error bars are overlapping for any model as S increases. This is emphasized with the dashed green line, showing that the composite model's results for $S = 3$ and $S = 4$ do not overlap.

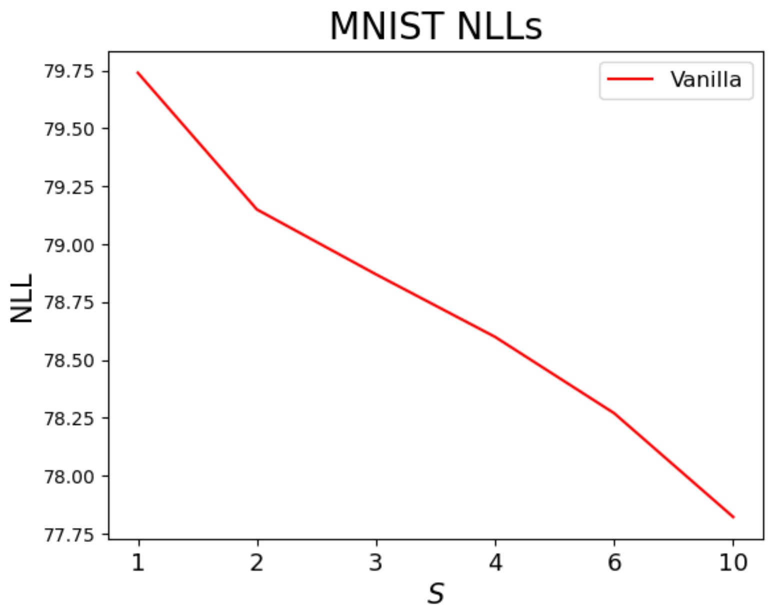

Figure 3: Monotonic NLL improvement (the lower the better) as S increases. The results are from the MNIST dataset using the Vanilla VAE from Sec. 4.2.1.

Table 7: Comparison between IWAEs trained with $L = S$ importance samples and at least as many network parameters as the Mixture VAEs for $S = 1, 2, 3$. In the experiments we either increase the number of hidden layers N for a fixed number of hidden units n, or the other way around. In all cases we round off upwards, such that the IWAEs enjoy more parameters than Mixture VAE. The performances are measured in NLL, the lower the better.

Model	$S = 1$	$S = 2$	$S = 3$
Mixture VAE			
NLL	85.24	85.18	84.77
# encoder parameters	349,200	698,400	1,047,600
# parameters in total	790,384	1,139,584	1,488,784
IWAE (increase N)	$N = 1$	$N = 5$	$N = 9$
NLL	85.24	97.45	101.62
# encoder parameters	349,200	709,200	1,069,200
# parameters in total	790,384	1,150,384	1,510,384
IWAE (increase n)	$n = 300$	$n = 509$	$n = 679$
NLL	85.24	86.28	85.16
# encoder parameters	349,200	698,857	1,047,697
# parameters in total	790,384	1,140,041	1,488,881

with $S = 1, 2, 3$ and $L = 1$, and we adjusted the number of parameters in the encoder networks of the importance weighted autoencoder (IWAE; Burda et al. (2015)) to get a corresponding number of parameters with $L = S$.

In general, we found that the Big IWAE could not outperform the corresponding Mixture VAE regardless if we increased the number of hidden layers or the number of hidden units. In most cases, we found the opposite; more parameters in the Big IWAE resulted in worse NLL. This does not mean that single encoder networks cannot be scaled to outperform Mixture VAEs in general, but it shows that naively increasing the number of parameters does not necessarily lead to better log-likelihood estimates. On the other hand, naively adding mixture components work wells in this regard.

To make the negative log-likelihood (NLL) comparison between Big IWAE and the mixture VAE fair, we estimated the marginal log-likelihood by performing Monte Carlo sampling either via

$$\log p_\theta(x) \approx \frac{1}{S}\sum_{s=1}^{S} \log \frac{1}{L}\sum_{\ell=1}^{L} \frac{p_\theta(x, z_{s,\ell})}{\frac{1}{S}\sum_{j=1}^{S} q_{\phi_j}(z_{s,\ell}|x)}, \quad z_{s,\ell} \sim q_{\phi_s}(z|x), \tag{17}$$

if the variational approximation is a mixture distribution, or

$$\log p_\theta(x) \approx \frac{1}{S}\sum_{s=1}^{S} \log \frac{1}{L}\sum_{\ell=1}^{L} \frac{p_\theta(x, z_{s,\ell})}{q_\phi(z_{s,\ell}|x)}, \quad z_{s,\ell} \sim q_\phi(z|x), \tag{18}$$

in the $S = 1$ case. That is, MISELBO and IWELBO were both approximated by sampling L importance samples S times. We report the scores in terms of NLL which were obtained using $L = 5000$ following Tomczak and Welling (2018).

E Latent Representations

E.1 Image data

The expression for the JSD is given here,

$$\mathrm{JSD}(\{q_{\phi_s}\}_{s=1}^{S}) = \mathbb{H}\left[\frac{1}{S}\sum_{s=1}^{S} q_{\phi_s}(z|x)\right] - \frac{1}{S}\sum_{s=1}^{S} \mathbb{H}\left[q_{\phi_s}(z|x)\right]. \tag{19}$$

Table 8: **FashionMNIST:** Accuracy for linear classification using the latent representations for an increasing number of mixture components (S).

Model	$S=1$	$S=2$	$S=3$	$S=4$	$S=5$
VAE	79.5±3E-3%	79.7±3E-5%	79.6±2E-3%	79.7±2E-3%	79.5±4E-3%
Ensemble VAE	**80.6±2E-3**%	81.1±3E-3%	81.7±1E-3%	82.5±6E-4%	82.6±1E-3%
Mixture VAE	**80.6±2E-3**%	**81.8±2E-3**%	**82.3±1E-3**%	**82.7±1E-3**%	**83.5±2E-3**%

E.2 Single-Cell Data

We have implemented and trained scVI models based on scVI-tools Gayoso et al. (2022). We have randomly split CORTEX data into train, validation, test sets with the following ratio: 80%, 7%, 13%. Each model has been trained 1000 epochs with an early stop condition of having non-decreasing loss for 100 epochs. All the classification/clustering experiments were performed with the latent representations of test set.

In the scVI latent space, we also do clustering using k-means from the Scikit-learn library (Pedregosa et al., 2011). Then following (Lopez et al., 2018), we quantitatively assess the quality of clustering by applying two clustering metrics: adjusted rand index (ARI) and normalized mutual information (NMI). The results below show that the latent representations produced by Mixture VAE extension of scVI perform better compared to the latent representations produced by the vanilla and Ensemble VAE extensions of scVI.

We assess the quality of latent features using two clustering metrics, which are employed by scVI Lopez et al. (2018) to compare against baselines. These metrics are adjusted rand index (ARI) and normalized mutual information (NMI).

Adjusted rand index (ARI): is the rand index adjusted for chance and used as a similarity measure between the predicted and true clusterings. It is mathematically defined as:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]\Big/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]\Big/\binom{n}{2}}$$

where n_{ij}, a_i, b_j are values from the contingency table. The higher ARI score means the more the two clusterings agree. ARI can be at most 1, which indicates that the two clusterings are identical.

Table 9: ARI: adjusted rand index (higher is better)

Model	$S=1$	$S=2$	$S=3$	$S=4$	$S=5$
VAE (scVI)	0.82	0.82	0.82	0.81	0.82
Ensemble VAE	0.82	0.83	0.83	0.82	0.83
Mixture VAE	0.82	0.82	0.84	0.85	0.85

Normalized Mutual Information (NMI): is the normalised version of MI score, which is a similarity measure between two different labels of the same data. Normalisation is done in a way so that the results would be between 0 (no mutual information) and 1 (perfect correlation):

$$NMI = \frac{I(P;T)}{\sqrt{\mathbb{H}(P)\mathbb{H}(T)}}$$

where P and T denote the empirical categorical distributions of the predicted and true labels, respectively. I is the mutual entropy, and $\mathbb{H}$ is the Shannon entropy. The higher NMI score indicates the better match between two label distributions.

Table 10: NMI: normalized mutual information (higher is better)

Model	$S = 1$	$S = 2$	$S = 3$	$S = 4$	$S = 5$
VAE (scVI)	0.81	0.81	0.81	0.81	0.81
Ensemble VAE	0.82	0.82	0.82	0.81	0.81
Mixture VAE	0.82	0.82	0.83	0.84	0.84