A. Additional experimental results

In this section, we provide more qualitative results in Figures 7, 6, 8, 5 as well as a graph showing the convergence properties of the variance for different models in Fig. 9. In order to validate our method with a different architecture, we also report performance of different decoders with a small 5-layer convolutional architecture on the CelebA and CIFAR dataset in Table 3. We see that the ordering of the methods is consistent with this smaller architecture.

B. Experimental details

For the small convolutional network test on SVHN, the encoder has 3 convolutional layers followed by a fully connected layer, while the decoder has a fully connected layer followed by 3 convolutional layers. The β was tuned from 100 to 0.0001 for β-VAE. The number of channels in the convolutional layers starts with 32 and increases 2 times in every layer. The dimension of the latent variable is 20. Adam (Kingma & Ba, 2015) with learning rate of 1e-3 is used for optimization. Batch size of 128 was used and the models were trained for 10 epochs for the experiments with the small convolutional network. We additionally evaluate this small convolutional network on CelebA, CIFAR, and Frey Face[2] datasets in Table 3. Unit Gaussian prior and Gaussian posteriors with diagonal covariance were used. We have attempted to improve the performance of baselines using KL annealing, but didn't find any significant improvement. For the larger hierarchical VAE, we used the official pytorch implementation of (Maaløe et al., 2019). We use the baseline hierarchical VAE with 15 layers of latent variables, without the top-down and bottom-up connections. We train the models for 30 thousand steps on CIFAR and 40 thousand steps on CelebA, with batch size of 48. For the hierarchical VAE and the SVG-LP model, we use the default hyperparameters in the respective implementations. We use the standard train-val-test split for all datasets. All models were trained on a single high-end GPU. We use the official PyTorch implementation of the Inception network to compute FID. All methods are compared on the same hyperparameters.

C. Empirical analyzis of approximations for optimal σ-VAE

The optimal σ-VAE requires computing the following estimate of the variance

$$\begin{aligned}\sigma^* &= \arg\max_\sigma \mathbb{E}_{x\sim\text{Data}}\mathbb{E}_{q(z|x)}\left[\ln p(x|\mu_\theta(z), \sigma^2 I)\right] \\ &= \mathbb{E}_{x\sim\text{Data}}\mathbb{E}_{q(z|x)}\text{MSE}(x, \mu_\theta(z)). \quad (11)\end{aligned}$$

[2]Available at `https://cs.nyu.edu/~roweis/data.html`

This requires computing two expectations, with respect to the data in the dataset, and with respect to the encoder distribution. We use MC sampling with one sample per data point to approximate both expectations. Inspired by common practices in VAEs, we use one sample per data point to approximate the inner expectation. On SVHN, the standard error of this approximation is 0.26% of the value of sigma. We further approximate the outer expectation with a single batch instead of the entire dataset. On SVHN, the standard error of this approximation is 2% of the value of sigma. We see that both approximations are accurate in practice. The second approximation yields a biased estimate of the evidence lower bound because the same batch is used to approximate the variance and compute the lower bound estimate. However, this bias can be corrected by using a different batch, or with a running average of the variance with an appropriate decay. This running average can also be used to reduce the variance of the estimate and to achieve convergence guarantees, but we did not find it necessary in our experiments.

D. Variational per-image variance

The per-image σ can be interpreted as a variational estimate as in Stirn & Knowles (2020). An additional variational distribution for the σ parameter $q(\sigma|x)$ is added as well as a hyperprior $p(\sigma)$. Our method greedily optimizes it to maximize the reconstruction term of the objective, which incurs a penalty term in the form of an additional KL divergence. Using this derivation it is possible to evaluate per-image optimal σ-VAE by solving for optimal σ on test images, as the penalty term ensures fair evaluation:

$$\begin{aligned}&\ln p_\theta(x) \geq \mathbb{E}_{q_\phi(z,\sigma|x)}\left[\ln p_\theta\left(x; \mu_\theta(z), \sigma\right)\right] + \\ &+ \text{D}_{\text{KL}}(q_\phi(z|x)||p_\theta(z)) + \text{D}_{\text{KL}}(q_\phi(\sigma|x)||p_\theta(\sigma)). \quad (12)\end{aligned}$$

Further, using this derivation, it is possible to create a better strategy for setting the optimal sigma by optimizing it with Eq. (12) instead of just the reconstruction loss.

E. Alternative Decoder Choices

We describe the alternative decoders evaluated in Table 2: using the bitwise-categorical, and the logistic mixture distributions.

Bitwise-categorical VAE While the 256-way categorical decoder described in Section 3.2 is very powerful due to the ability to specify any possible intensity distribution, it suffers from high computational and memory requirements. Because 256 values need to be kept for each pixel and channel, simply keeping this distribution in memory for

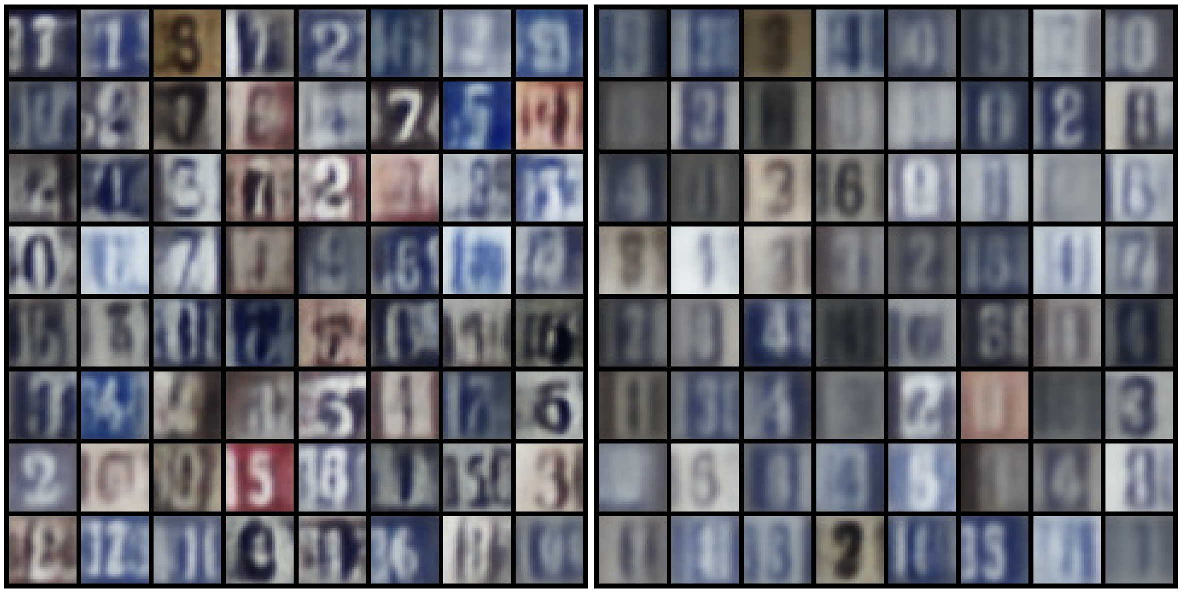

Figure 5. Samples from the σ-VAE (left) and the Gaussian VAE (right) on the SVHN dataset. The Gaussian VAE produces blurry results with muted colors, while the σ-VAE is able to produce accurate images of digits.

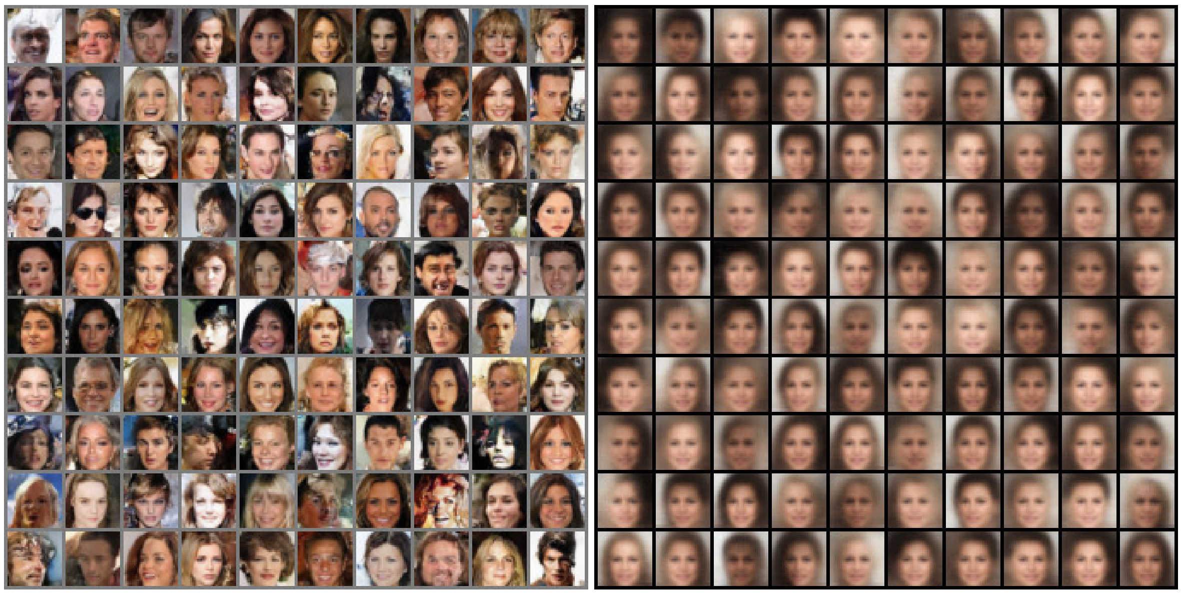

Figure 6. Samples from the σ-VAE (left) and the Gaussian VAE (right) on the CelebA dataset, images cropped to the face for clarity. The Gaussian VAE produces blurry results with indistinct face features, while the σ-VAE is able to produce accurate images of faces.

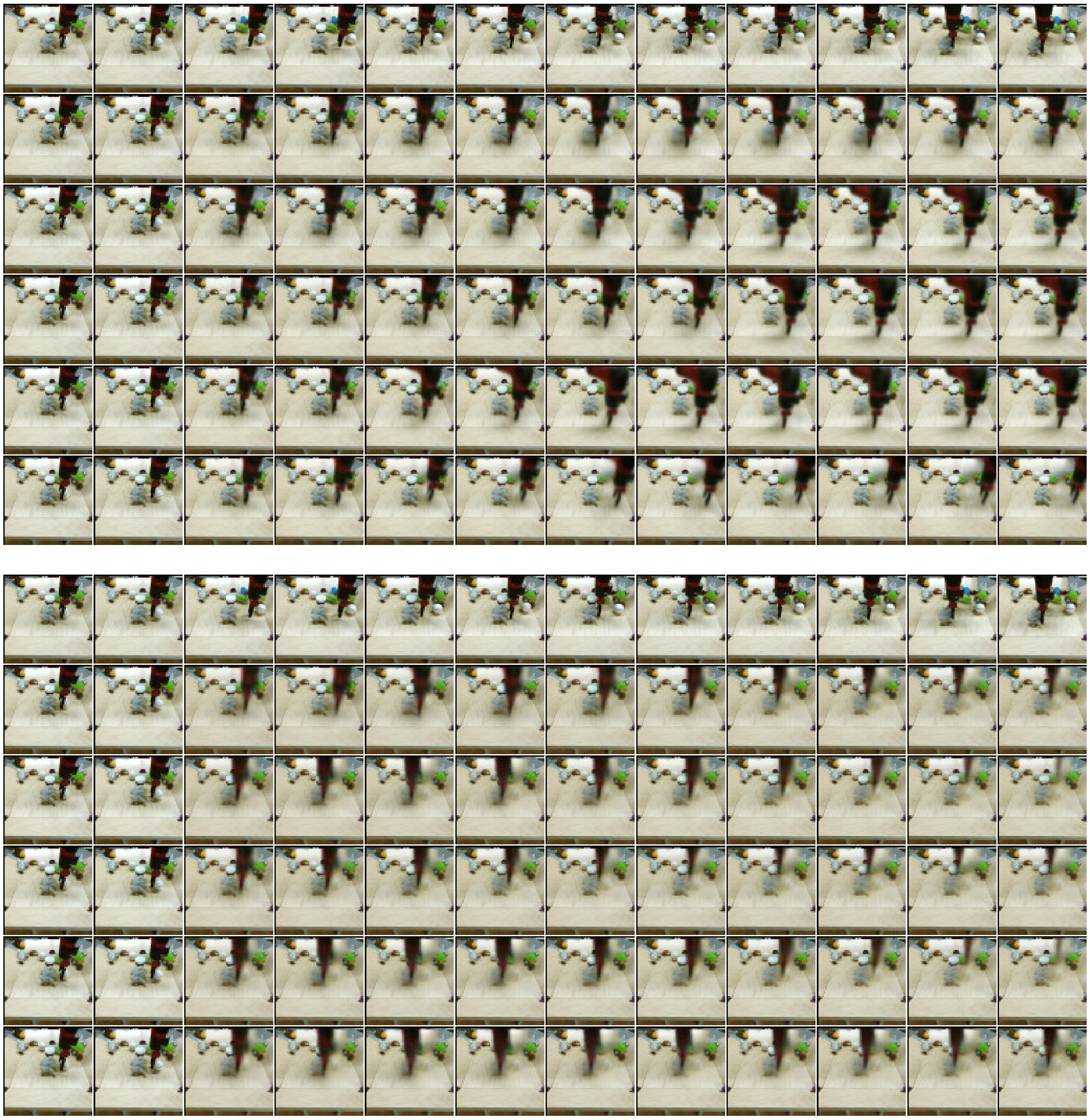

Figure 7. Samples from the σ-VAE (top) and the Gaussian VAE (bottom) on the BAIR dataset. Sampled sequences conditioned on two initial frames are shown, and the ground truth sequence is shown at the top. The Gaussian VAE produces blurry robot arm texture and the arm often disappears towards the end of the sequence, while the σ-VAE is able to produce sequences with realistic motion and model the details of the arm texture, such as the gripper.

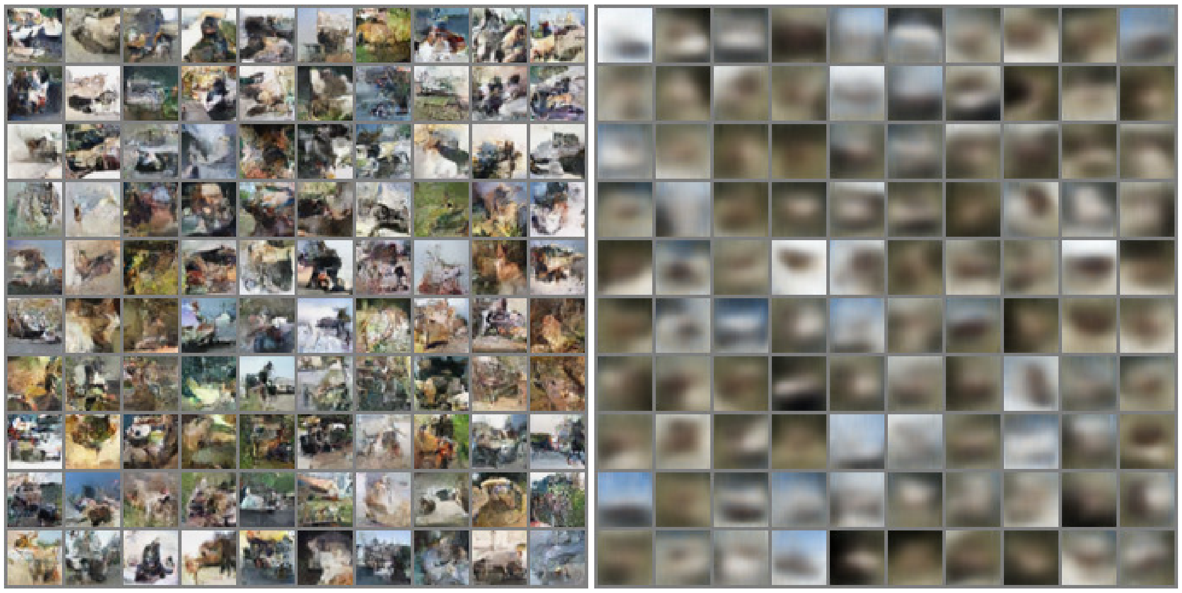

Figure 8. Samples from the σ-VAE (left) and the Gaussian VAE (right) on the challenging CIFAR dataset. The Gaussian VAE produces blurry results with muted colors, while the σ-VAE models the distribution of shapes in the CIFAR data more faithfully.

Table 3. Generative modeling performance of the proposed σ-VAE on CelebA, CIFAR, and Frey Face with a smaller model. We see that uncalibrated decoders such as mean-only Gaussian perform poorly. β-VAE allows to calibrate the decoder but needs careful hyperparameter tuning. Calibrated decoders such as categorical or σ-VAE perform best.

	CelebA VAE		CIFAR VAE		Frey Face VAE	
	$-\log p \downarrow$	FID $\downarrow$	$-\log p \downarrow$	FID $\downarrow$	$-\log p \downarrow$	FID $\downarrow$
Bernoulli VAE (Gregor et al., 2015)		102.7		165.1		47.7
Categorical VAE	< 10195	**50.45**	< 10673	124.1	$< \mathbf{2454}$	50.16
bitwise-categorical VAE	< 11019	56.36	< 11604	99.65	< 3173	66.77
Logistic mixture VAE	$< \mathbf{10154}$	61.81	$< \mathbf{10648}$	100.2	< 2562	50.28
Gaussian VAE	< 2201	144.8	< 1409	205.8	< 726.4	80.17
β-VAE (Higgins et al., 2017)	$< \mathbf{-1942}$	58.73	< -1318	117.9	< -420.0	**37.61**
Shared σ-VAE (Ours)	$< \mathbf{-1939}$	73.27	$< \mathbf{-1830}$	137.8	< -49.78	**42.86**
Optimal σ-VAE (Ours)	$< \mathbf{-1951}$	61.27	$< \mathbf{-1832}$	**80.9**	$< \mathbf{-1622}$	53.36
Opt. per-image σ-VAE (Ours)		**53.13**		89.88		56.07

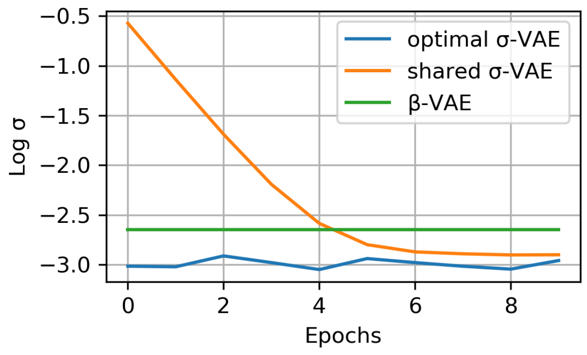

Figure 9. Variance convergence speed on SVHN. We see that the shared σ-VAE which optimizes the variance with gradient descent has an initial period of convergence when the variance converges to the region of the optimal value. In contrast, σ-VAE with analytical (optimal) variance quickly learns a good estimate of the variance, which leads to better performance. The unit variance Gaussian β-VAE can be interpreted as having a constant variance determined by β, shown here. Since the variance doesn't change throughout training, it achieves suboptimal performance.

Table 4. ELBO on discretized data. All distributions except categorical have scalar scale parameters. The σ-VAE performs well on the discretized ELBO metric, performing similarly to a discrete distribution parametrized as a discretized Gaussian or discretized Logistic. Full categorical distribution attains highest likelihood due to having the most statistical power.

	CIFAR VAE		
	$-\log \text{pdf} \downarrow$	$-\log p \downarrow$	FID $\downarrow$
Categorical VAE		$< \mathbf{10673}$	**137.6**
Gaussian VAE	< 740.5	< 15131	212.7
Gaussian σ-VAE	< -896.1	< 11120	**136.7**
Disc. Gaussian σ-VAE		< 11117	**136.9**
Disc. Logistic σ-VAE		< 11103	**136.7**

one 3-channel 1024×1024 image would require 3 GiB of memory, compared to 0.012 GiB for the Gaussian decoder. Therefore, training deep neural networks with this full categorical distribution is impractical for high-resolution images or videos. The bitwise-categorical VAE improves the memory complexity by defining the distribution over 256 values in a more compact way. Specifically, it defines a binary distribution over each bit in the pixel intensity value, requiring 8 values in total, one for each bit. This distribution can be thought of as a classifier that predicts the value of each bit in the image separately. In our implementation of the bitwise-categorical likelihood, we convert the image channels to binary format and use the standard binary cross-entropy loss (which reduces to binary log-likelihood since all bits in the image are deterministically either zero or one). While in our experiments the bitwise-categorical distribution did not outperform other choices, it often performs on par with our proposed method. We expect this distribution to be useful due to its generality as it is able to represent values stored in any digital format by converting them into binary.

Logistic mixture VAE For this decoder, we adapt the discretized logistic mixture from Salimans et al. (2017). To define a discrete 256-way distribution, it divides the corresponding continuous distribution into 256 bins, where the probability mass is defined as the integral of the PDF over the corresponding bin. (Kingma et al., 2016) uses the logistic distribution discretized in this manner for the decoder. Salimans et al. (2017) suggests to make all bins except the first and the last be of equal size, whereas the first and the last bin include, respectively, the intervals $(-\infty, 0]$ and $[1, \infty)$. Salimans et al. (2017) further suggests using a mixture of discretized logistics for improved capacity. Our implementation largely follows the one in Salimans et al. (2017), however, we note that the original implementation is not suitable for learning latent variable models, as it generates the channels autoregressively. This will cause the latent variable to lose color information since it can be represented by the autoregressive decoder. We therefore adapt the mixture of discretized logistics to the pure latent variable setup by removing the mean-adjusting coefficients from (Salimans et al., 2017). In our experiments, the logistic mixture outperformed other discrete distributions.