A Decoder Suffices for Query-Adaptive Variational Inference (Supplementary Material)

Sakshi Agarwal[1*] **Gabriel Hope**[1*] **Ali Younis**[1] **Erik B. Sudderth**[1]

[1]Department of Computer Science, University of California, Irvine
[*]Equal contribution

The supplementary section is organized as follows. Section 1 focuses on deriving the optimization objective used in QAVI, and elaborates on the tricks used to estimate gradients for the parameters in Mix. QAVI. Section 2 includes some implementation details. Lastly, we discuss some more experiments in section 3.

1 QAVI: DERIVATIONS

1.1 FEAT. QAVI

As specified in the main paper, we define a variational distribution over unobserved features x_M as $q_\lambda(x_M)$ and re-use the encoder to model $q(z|x_O, x_M)$. Hence, the ELBO can be written as :

$$\begin{aligned}
\mathcal{L}_M(\lambda; x_O) &= E_{q_\lambda(x_M, z|x_O)}[\log p_\theta(x_O, x_M, z) - \log q_\lambda(z, x_M|x_O)] \\
&= E_{q_\lambda(x_M, z|x_O)}[\log p_\theta(x_O, x_M|z) + \log p(z) - \log q_\lambda(x_M) - \log q_\phi(z|x_M, x_O)] \\
&= E_{q_\lambda(x_M)}[E_{q_\phi(z|x_O, x_M)}[\log p_\theta(x_O, x_M|z) - \log q_\phi(z|x_M, x_O) + \log p(z)]] - E_{q_\lambda(x_M)}[\log q_\lambda(x_M)] \\
&= E_{q_\lambda(x_M)}[E_{q_\phi(z|x_O, x_M)} \log[p_\theta(x_O, x_M|z)] - KL(q_\phi(z|x_M, x_O)||p(z))] - E_{q_\lambda(x_M)}[\log q_\lambda(x_M)] \\
&= E_{q_\lambda(x_M)}[E_{q_\phi(z|x_O, x_M)} \log[p_\theta(x_O, x_M|z)] - KL(q_\phi(z|x_M, x_O)||p(z))] + H(q_\lambda(x_M)) \qquad (1)
\end{aligned}$$

where $H(q_\lambda(x_M))$ is the entropy of $q_\lambda(x_M)$ and KL is the Kullback-Leibler divergence. Monte-Carlo approximation of Eq. 1 with S samples from $q_\lambda(x_M, z|x_O)$ is equivalent to sampling $x_M^{(s)} \sim q_\lambda(x_M)$ and $z^{(s)} \sim q_\phi(z|x_O, x_M^{(s)})$. We get:

$$\mathcal{L}_M(\lambda; x_O) \approx \frac{1}{S}\sum_{s=1}^{S}[\log p_\theta(x_O, x_M^{(s)}|z^{(s)}) - KL(q_\phi(z|x_O, x_M^{(s)})||p(z))] + H(q_\lambda(x_M)) \qquad (2)$$

1.2 NON-AMORTIZED INFERENCE

Here, we define a variational distribution on the latent space variable z as $q_\lambda(z)$. The ELBO can thus be derived as follows:

$$\begin{aligned}
\mathcal{L}_N(\lambda; x_O) &= E_{q_\lambda(z, x_M|x_O)}[\log p_\theta(x_O, x_M, z) - \log q_\lambda(z, x_M|x_O)] \\
&= E_{q_\lambda(z, x_M|x_O)}[\log p_\theta(x_O, x_M|z) + \log p(z) - \log q_\lambda(z) - \log p_\theta(x_M|z)] \\
&= E_{q_\lambda(z)}[\log p_\theta(x_O|z) + \log p(z) - \log q_\lambda(z)] \\
&= E_{q_\lambda(z)}[\log p_\theta(x_O|z)] - \mathrm{KL}(q_\lambda(z)||p(z)) \qquad (3)
\end{aligned}$$

Approximating the above with S samples, $z^{(s)} \sim q(z)$, we get :

$$\mathcal{L}_N(\lambda; x_O) \approx \frac{1}{S}\sum_{s=1}^{S}[\log p_\theta(x_O|z^{(s)})] - \mathrm{KL}(q_\lambda(z)||p(z)) \qquad (4)$$

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

1.2.1 Gradients for mixture posterior

Implicit Reparameterized Gradients. Graves (2016) and Figurnov et al. (2018) propose a method of computing pathwise gradients that relies on reparameterization where a standardization function $S_\lambda(z)$ is used on samples $z \sim q_\lambda(z)$ to remove the dependence of the parameters λ from those samples: $S_\lambda(z) = \epsilon \sim q(\epsilon)$. For univariate Gaussian mixture models, one such standardization function is the CDF, $F(z|\lambda)$, which transforms samples from the mixture to samples from a uniform distribution with support $[0, 1]$. In the multivariate case, the multivariate distributional transform (Rüschendorf, 2013) can be used:

$$S_\lambda(z) = \Big(F(z_1|\lambda), F(z_2|z_1, \lambda), ..., F(z_D|z_1, ..., z_{D-1}, \lambda)\Big)$$

Implicit differentiation is then performed to derive the following pathwise gradient estimator as shown in Figurnov et al. (2018):

$$\nabla_\lambda z = -\Big(\nabla_z S_\lambda(z)\Big)^{-1} \nabla_\lambda S_\lambda(z) \tag{5}$$

In Figurnov et al. (2018), it is mentioned that this method can be used with mixture models however no empirical results were given. In practice we find that the variance of the pathwise gradient estimates w.r.t. the mixture weights is prohibitively large, preventing learning of the mixture weights. To address this, we use a different estimator for computing gradients w.r.t. the mixture weights.

Importance Sampling Gradient Estimator. Ścibior et al. (2021) propose that samples drawn from $q_\lambda(z)$ may be augmented with a gradient operator that does not change the sample in the forward pass but allows for gradient computation in the backward pass. A stop-gradient operator, $\perp$ is used in this augmentation. When $\perp$ is applied to a function $f\theta(x) = y$, the value of that function does not change in the forward pass: $\perp f\theta(x) = f\theta(x) = y$. In the backward pass, $\perp f\theta(x)$ is treated as a constant value and not a function, yielding a gradient value of 0: $\nabla_\theta \perp f_\theta(x) = 0$. For a given sample $z \sim q_\lambda(z)$, Ścibior et al. (2021) augments that sample using importance sampling:

$$z' = \frac{q_\lambda(z)}{\perp q_\lambda(z)} \cdot z \tag{6}$$

In the forward pass $z' = z$ since the numerator and denominator cancel out. In the backward pass, we compute gradients for the numerator only while treating the denominator as a constant, yielding the following estimator:

$$\nabla_\lambda z' = \frac{z}{\perp q_\lambda(z)} \cdot \nabla_\lambda q_\lambda(z) \tag{7}$$

Unlike the Implicit Reparameterized Gradients estimator, we find that this method computes low variance estimates of the pathwise gradient w.r.t. the mixture weights w_t while having high variance w.r.t. the mixture component means and standard deviations.

Hybrid Approach: We adopt a novel, hybrid approach to pathwise gradient estimation. We use implicit reparameterized gradients to compute gradients w.r.t. the mixture component means and standard deviations, μ_t and Λ_t. We use the importance sampling gradient estimator to compute the gradients with respect to the mixture weights, w_t.

2 IMPLEMENTATION DETAILS

2.1 PRE-TRAINED VAES

MNIST We train a VAE using a WideResNet architecture Zagoruyko and Komodakis (2016), on fully-observed real-valued MNIST. The encoder is a WideResNet with 3 downsampling levels and 2 resnet blocks within each level, with ReLU nonlinearities. The output layer has no non-linearity for the mean and a softplus nonlinearity for the standard deviation. We use a latent dimension of 50. The decoder has a similar upsampling WideResNet. The decoder has no non-linearities at the output for the continuous-Bernoulli distribution Loaiza-Ganem and Cunningham (2019). We train this VAE using Adam Kingma and Ba (2015) with a learning rate of $1e^{-3}$ for 2000 epochs.

SVHN We have a similar architecture for SVHN with Leaky ReLU non-linearities and an output distribution of discretized truncated normal distribution Salimans et al. (2017), considering the range [-1, 1]. When training the VAE for SVHN, we used a learning rate of $1e^{-4}$ and chose a model that has the best validation loss to encourage a more generalized generative model. We trained the model for 500 epochs.

Tabular datasets We trained a VAE for each UCI dataset with: a latent dimension of 10; both encoder and decoder are multi-layer perceptrons with 3 hidden layers (128 hidden dimension each) and ReLU activations; we use an independent diagonal Normal distribution for the variational family and the observation model. We constrain output standard deviations of the decoder to be larger than 0.001. We trained the model using the Adam optimizer with a learning rate of $2e^{-4}$ for 1000 epochs, saving the parameters of the models with lowest loss on the validation set. Our architectures are chosen to match Mattei and Frellsen (2019). All UCI datasets are normalized to have mean 0 and variance 1. We use 65% of the data for training, 15% for validation and 20% as test set. We corrupt the test set by removing half of the features in each row uniformly at random.

FFHQ HVAE As discussed, for our deep-VAE experiments we adopt the architecture proposed by Child (2021) for the FFHQ-256 dataset. For our quatitative experiments and baselines, we reduce the model size by reducing the internal channel width from 512 to 64 and reducing the number of layers at each resolution by half. This reduces the total number of model parameters from 115M to 5.9M. We retain the original latent dimension channel width of 16. Despite this substantial decrease in parameters, we find that performance is still good, as seen in our reported results.

2.2 BASELINES

Below we discuss the implementation details for the different inference baselines considered in the paper. We refer to x_O as x that has had unobserved features set to zero and x'_O as x_O concatenated with a bitmask that indicates which features are observed.

1. **Fill 0s:** Passing x_O through the pre-trained encoder yields a simple baseline posterior over the latent space, $q(z)$.
2. **pseudo-Gibbs sampling:** Initially, we set $x = x_O$, (i) pass x through the VAE, (ii) sample unobserved features in x from the decoder and (iii) repeatedly perform (i) and (ii) for 300 iterations. In the end, the resulting posterior over the latent space $q(z)$ is the inferred posterior.
3. **Metropolis-within-Gibbs sampling:** This is similar to pseudo-Gibbs sampling, instead here we accept/reject a sample according to Eq. 15 in Mattei and Frellsen (2018) and perform pseudo-Gibbs for the first 20 iterations to avoid early rejections. After a total of 300 iterations, we pick the lower-dimensional latent space posterior for evaluation.
4. **Retuned-Encoder:** We train an inference network (same architecture as base VAE encoder with inputs as x'_O) for each case of partially-observed test data for 100 epochs using the same ELBO in Eq. 2 of the main paper (with x replaced by x_O). We do not consider masking information in the latent space as done in Collier et al. (2020) and instead, exploit the previously trained generative model. Post training, we infer the latent space posterior $q(z)$ using this new encoder model.
5. **Posterior Matching:** In order to train an inference network for this case, we consider the fully-observed train set x and a masking distribution from Zhao et al. (2021) to generate partially observed train data points x'_O. We then consider an external inference network (again similar in architecture to the base (H-)VAE with inputs as x'_O) to approximate the partially-observed posterior in the latent space as $q_\psi(z|x'_O)$. Training this network further maximizes the objective : $E_{z\sim q_\phi(z|x)} \log q_\psi(z|x'_O)$ using the same hyper-parameters as training the base VAE for 1000 epochs, and does not require the generative model. After this general training is complete, we use this inference network to estimate the latent space posteriors ($q(z)$) for each case of partially-observed test data. For both single-layer and hierarchical VAEs, we initialize the partial encoder with the weights of the original encoder, save for the first layer, which is modified to take an additional mask channel. For the HVAE variant of posterior matching, trained on the FFHQ dataset, we train for 300k steps with a batch size of 16.
6. **VAEAC:** We compare a hierarchical VAE trained with the VAEAC objective in our experiments on FFHQ. We do not compare with VAEAC for our standard VAE experiments as the skip connections in the VAEAC architecture makes architecture matching imprecise and posterior matching has been shown to outperform the corresponding VAEAC approach in this setting (Strauss and Oliva, 2022). The "ladder" structure of our chosen HVAE architecture provides a similar "skip" structure to the conditional model. As with posterior matching, we initialize the partial encoder for VAEAC with the weights of the original HVAE encoder. We also initialize the full-encoder and decoder with the corresponding weights from the pre-trained model. For VAEAC, all three networks are trained end-to-end for 300k steps with a batch size of 8 images.

We evaluate each latent-space posterior, $q(z)$, resulted from the above inference methods to compute missing feature log-likelihoods.

2.3 FEAT. QAVI

As this approach requires repeated passes of both the encoder and decoder, we evaluate only on the MNIST dataset (for both patches and top-half missingness patterns). Since MNIST is normalized to [0,1], an ideal choice of $q_\lambda(x_M)$ is a continuous bernoulli distribution (Loaiza-Ganem and Cunningham, 2019) with parameter $\lambda \in (0, 1)$. We allow for unconstrained optimization by re-parameterizing as: $\lambda = \sigma(\hat{\gamma})$, where $\sigma(\cdot)$ is the sigmoid function and $\hat{\gamma}$ is an unconstrained parameter. We initialize the parameters randomly, and then, optimize the ELBO in Eq. 5 above with $S = 100$ samples for 300 iterations using Adam with a learning rate of 1.0. We decrease the learning rate by a factor of 10 every 100 steps. To sample from the latent space distribution $q_\lambda(z)$ here means we first draw k samples $\hat{x} \sim q_\lambda(x_M)$, followed by one sample $z \sim q(z|\hat{x})$.

2.4 GAUS./FLOW/MIX. QAVI

Initializations to the mean and standard deviations of Gaussian distributions for each variational posterior considered in the paper is crucial. Means for the Gaussian posterior, base Gaussian distribution for IAF and individual Gaussian components in the Mixture is carried out via the outputs of the amortized inference model $q_\phi(z|x_O)$. The standard deviation for each posterior is initialized to 1 per latent dimension. For the mixture case, we introduce a [-1,1] uniform random noise to the means to break symmetry among its individual components and keep the initial components' weights uniform.

The variational parameters in the Gaussian Posterior are optimized using Adam with a learning rate of 1.0, decreasing it by a factor of 2 every 30 iterations.

We follow Kingma et al. (2016) and use a 2-layer MADE network (Germain et al., 2015) with 320 dimensions (or 50 dimension in case of UCI datasets) per layer to implement each IAF transformation. We stack 2 IAF transformations on top of a simple Gaussian distribution in the latent space. For computational reasons, we share the parameters of the two transformations across a batch (size 100) of test images. We optimize the base Gaussian distribution using Adam with a learning rate of 0.1 and the parameters of the inverse autoregressive neural network using Adam with a learning rate of 0.01.

When the variational posterior is a Gaussian Mixture, we use 10 Gaussian components. Training of all parameters (means, standard deviations, weights) is done via Adam optimizer, involving 3 phases. The first phase consists of the first 50 iterations, where we optimize all parameters with a learning rate of 0.1 and at an interval of every 10 iterations, attempt to re-initialize those components whose weights fall below a threshold $t = 0.7$. We reset the weights for all components to be equal when this re-initialization occurs. This method of re-initializing allows us to throw away any "bad" initializations. The second phase consists of 150 iterations, where only the means and standard deviation in the individual components are optimized with a learning rate of 1.0, decreased every 30 iterations by a factor of 2. During the last 100 iterations, we optimize all parameters, where the unconstrained weights of individual components are optimized with a learning rate of 0.1, decreasing every 30 iterations by a factor of 10. The idea behind doing this is to let the individual components converge first, followed by adjusting weights to these individual components.

2.5 HIERARCHICAL QAVI

Figures 2 and 3, show the amortized and QAVI posterior achitectures respectively. Figures 5-7 show further results using the HVAE FFHQ QAVI model.

Modifications to P-IDS and U-IDS metrics. In order to reduce sensitivity to the test set size we modify the original P-IDS and U-IDS metrics. In the original formulation used by Zhao et al. (2021), P-IDS and U-IDS accuracies are computed on the same set of images used to train the discriminative SVM model. In our modified approach, we use 10-fold cross validation: training the SVM and evaluating the scores on distinct subsets.

3 OTHER EXPERIMENTAL DETAILS

Number of samples Increasing the number of samples to estimate the loss in Eq. 7 & 8 has a sub-linear effect on the runtime since they can be computed in parallel on modern GPUs. Gauss. QAVI suffers virtually no performance penalty from reducing the number of samples to 10 (as measured by IWAE), while our flow and mixture based posteriors perform similar to Gauss. QAVI with fewer samples: they require more samples to capture all the modes of the approximate posterior.

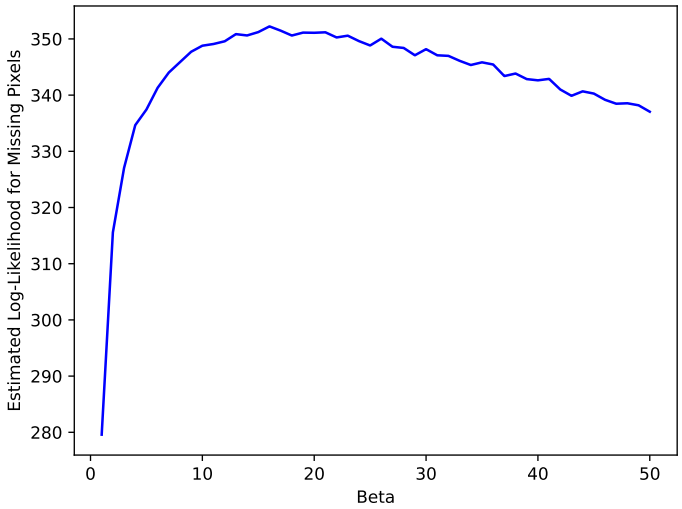


Figure 1: The plot shows the estimated log-likelihood for missing pixels for our non-amortized Gaussian posterior method when $1 \leq \beta \leq 50$. This is performed for the random missing patches in MNIST test images. We observe that up-weighting the KL term by the hyper-parameter β in Eq. 8 from the main paper boosts performance until the value (≈ 15) and slowly decreases the log-likelihood with higher values. This translates clearly to the fact that some variance in the Gaussian posterior is helpful to capture uncertainty, but after a threshold this variance might result in inaccurate imputations.

β hyper-parameter Figure 1 shows the estimated log-likelihood for missing pixels for our non-amortized Gaussian posterior method when $1 \leq \beta \leq 50$. The plot suggests that using $\beta = 20$ is optimal.

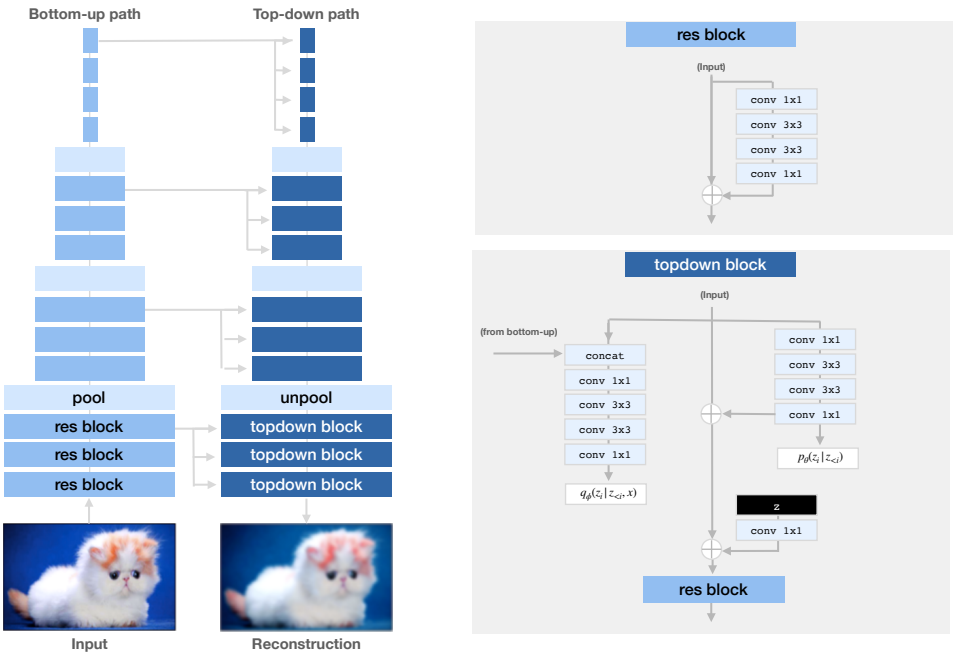


Figure 2: Architecture of the original (amortized inference) "very-deep" VAE. Adapted from figure 3 of Child (2021).

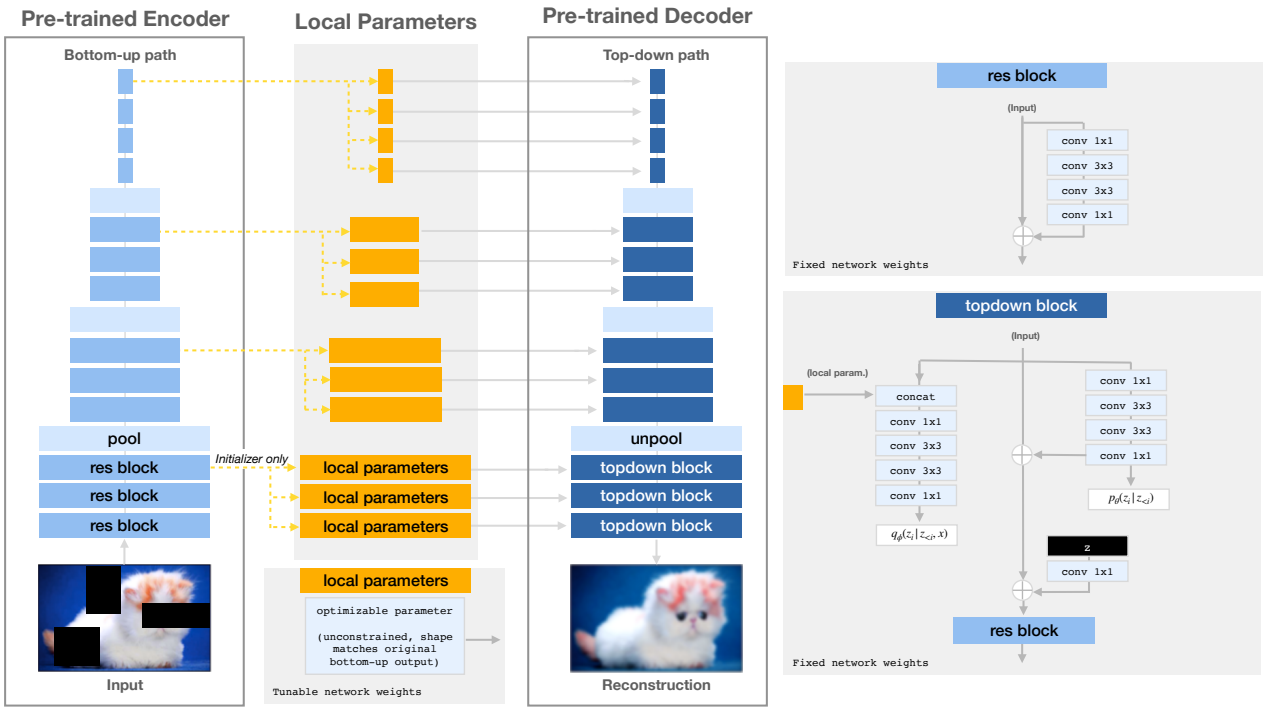


Figure 3: Architecture of the local variational distribution ($q_\lambda(z)$) for QAVI with the "very-deep" VAE. We show how intermediate outputs of the bottom-up encoder are replaced with trainable network parameters to definite a local variational distribution.

Figure 4: Digit completion results on real-valued MNIST (left) and SVHN (right) images for the rotating-half and random-patches missing criteria.

Figure 5: Comparison of QAVI inpainting using the reduced-size (5.9M parameter) very deep HVAE model used for our main experiments with QAVI inpainting using the original FFHQ-256 model of Child (2021) (115M parameters).

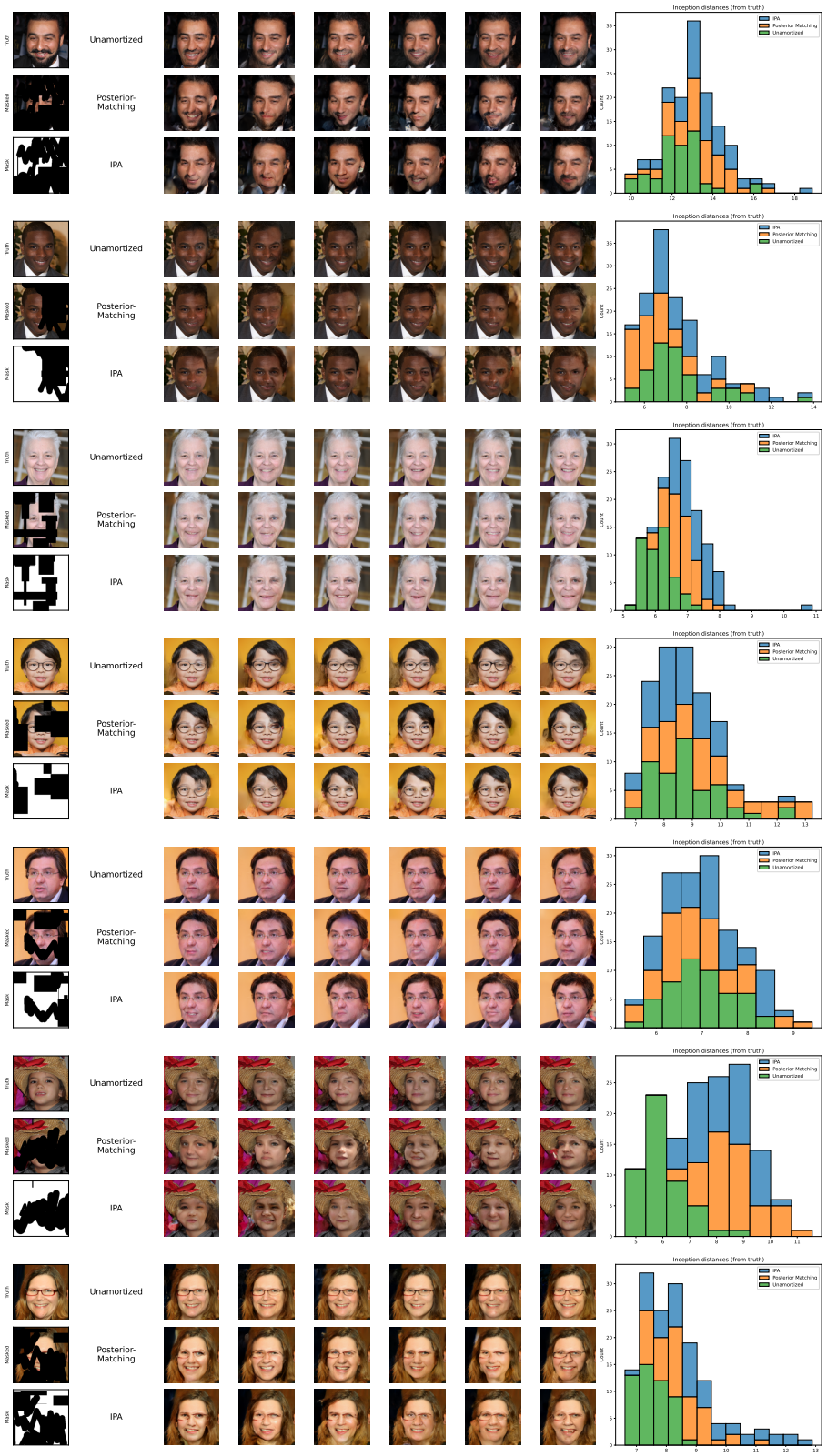
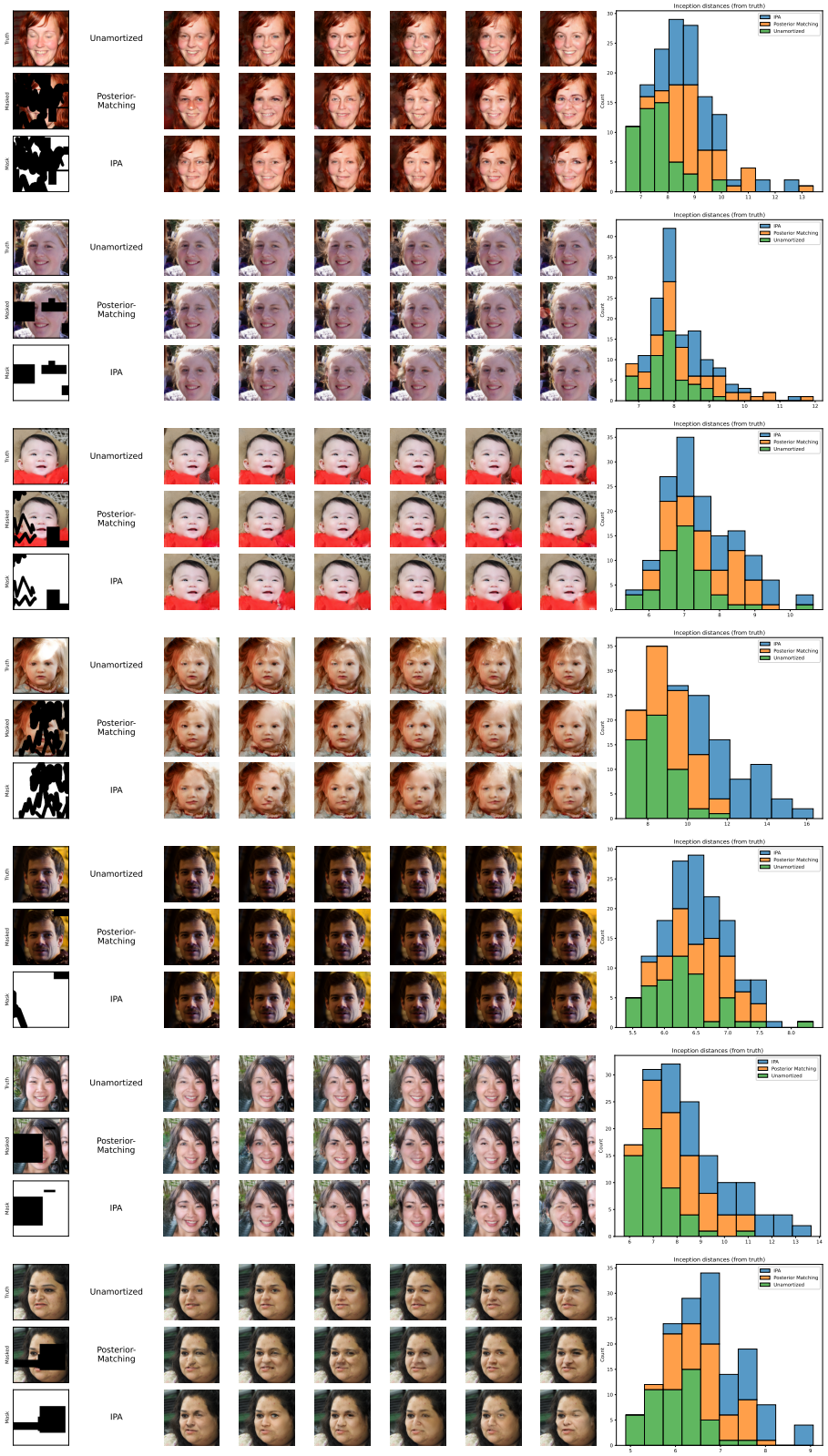


Figure 6: Comparison of inpainting results on the FFHQ-256 dataset. We compare our non-amortized deep VAE inpainting with IPA and Posterior Matching. For each image we show the true image, the masked image and 6 inpainted samples from each method. We also extract a 2048-dimensional feature vector for each image using the Inception-V3 network (as for FID score). We plot the distribution of distances for 100 sampled images from the true image using euclidean distance in this extracted feature space. We see that in most cases, the IPA model produces more varied imputations, while non-amortized inference produces more consistent results.


Truth
Masked
Mask
Unamortized
Posterior-Matching
IPA
Inception distances (from truth)
IPA
Posterior Matching
Unamortized
Count

References

Child, R. (2021). Very deep {vae}s generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*.

Collier, M., Nazabal, A., and Williams, C. (2020). VAEs in the presence of missing data. In *ICML Workshop on the Art of Learning with Missing Values (Artemiss)*.

Figurnov, M., Mohamed, S., and Mnih, A. (2018). Implicit reparameterization gradients. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 881–889, Lille, France. PMLR.

Graves, A. (2016). Stochastic backpropagation through mixture density distributions. *arXiv preprint arXiv:1607.05690*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4743–4751, Red Hook, NY, USA. Curran Associates Inc.

Loaiza-Ganem, G. and Cunningham, J. P. (2019). The continuous bernoulli: fixing a pervasive error in variational autoencoders. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Mattei, P.-A. and Frellsen, J. (2018). Leveraging the exact likelihood of deep latent variable models. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Mattei, P.-A. and Frellsen, J. (2019). MIWAE: Deep generative modelling and imputation of incomplete data sets. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4413–4423. PMLR.

Rüschendorf, L. (2013). *Copulas, Sklar's Theorem, and Distributional Transform*, pages 3–34. Springer Berlin Heidelberg, Berlin, Heidelberg.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*.

Ścibior, A., Masrani, V., and Wood, F. (2021). Differentiable particle filtering without modifying the forward pass. *International Conference on Probabilistic Programming (PROBPROG)*.

Strauss, R. and Oliva, J. (2022). Posterior matching for arbitrary conditioning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *BMVC*.

Zhao, S., Cui, J., Sheng, Y., Dong, Y., Liang, X., Chang, E. I., and Xu, Y. (2021). Large scale image completion via co-modulated generative adversarial networks. In *International Conference on Learning Representations (ICLR)*.