# Uncertainty Visualization via Low-Dimensional Posterior Projections Supplementary material

Omer Yair, Elias Nehme & Tomer Michaeli
Technion - Israel Institute of Technology
Haifa, Israel
omeryair@gmail.com, seliasne@gmail.com, tomer.m@ee.technion.ac.il

## A. Toy model

### A.1. Problem setting

In the main text, we demonstrated PPDE on a 2D denoising task, where samples $\boldsymbol{x}_i$ from a Gaussian mixture model were distorted by additive white Gaussian noise $\boldsymbol{n}_i$ with a standard deviation of $\sigma_n$ to result in noisy measurements $\boldsymbol{y}_i$. The prior distribution $p_{\mathbf{x}}(\boldsymbol{x})$ was comprised of $L$ Gaussians defined by the parameters $\{\boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell, \pi_\ell\}_{\ell=1}^L$, namely,

$$p_{\mathbf{x}}(\boldsymbol{x}) = \sum_{\ell=1}^{L} \pi_\ell \cdot \varphi(\boldsymbol{x}; \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell), \tag{S1}$$

where $\varphi(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the PDF of a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Specifically, in the 2D example of Fig. 5 from the main text, we used $L = 6$ Gaussians with the following parameters:

$$\begin{aligned}
&\boldsymbol{\mu}_1 = (-1, 1)^T && \Sigma_1 = \begin{pmatrix} 0.05 & -0.01 \\ -0.01 & 0.025 \end{pmatrix} && \pi_1 = 0.15 \\
&\boldsymbol{\mu}_2 = (1, 1)^T && \Sigma_2 = \begin{pmatrix} 0.05 & 0.01 \\ 0.01 & 0.025 \end{pmatrix} && \pi_2 = 0.15 \\
&\boldsymbol{\mu}_3 = (0, 0)^T && \Sigma_3 = \begin{pmatrix} 0.02 & 0 \\ 0 & 0.03 \end{pmatrix} && \pi_3 = 0.15 \\
&\boldsymbol{\mu}_4 = (-0.7, -1)^T && \Sigma_4 = \begin{pmatrix} 0.15 & -0.04 \\ -0.04 & 0.04 \end{pmatrix} && \pi_4 = 0.15 \\
&\boldsymbol{\mu}_5 = (0, -1.2)^T && \Sigma_5 = \begin{pmatrix} 0.15 & 0 \\ 0 & 0.025 \end{pmatrix} && \pi_5 = 0.15 \\
&\boldsymbol{\mu}_6 = (0.7, -1)^T && \Sigma_6 = \begin{pmatrix} 0.15 & 0.04 \\ 0.04 & 0.04 \end{pmatrix} && \pi_6 = 0.15.
\end{aligned} \tag{S2}$$

The standard deviation of the additive noise was $\sigma_n = 0.4$.

### A.2. Analytical posterior distribution

Assuming the setting in Sec. A.1, here we derive the analytical posterior distribution. Specifically, we show that the posterior is also a GMM, and that it has the form

$$p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y}) = \sum_{\ell=1}^{L} \tilde{\pi}_\ell(\boldsymbol{y}) \cdot \varphi(\boldsymbol{x}; \tilde{\boldsymbol{\mu}}_\ell(\boldsymbol{y}), \tilde{\boldsymbol{\Sigma}}_\ell(\boldsymbol{y})), \tag{S3}$$

where

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_\ell(\boldsymbol{y}) &= \boldsymbol{\mu}_\ell + \boldsymbol{\Sigma}_\ell(\boldsymbol{\Sigma}_\ell + \sigma_n^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_\ell), \\
\tilde{\boldsymbol{\Sigma}}_\ell(\boldsymbol{y}) &= \boldsymbol{\Sigma}_\ell - \boldsymbol{\Sigma}_\ell(\boldsymbol{\Sigma}_\ell + \sigma_n^2 \boldsymbol{I})^{-1}\boldsymbol{\Sigma}_\ell, \\
\tilde{\pi}_\ell(\boldsymbol{y}) &= \frac{\pi_\ell \varphi(\boldsymbol{y}; \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell + \sigma_n^2 \boldsymbol{I})}{\sum_{\ell'=1}^{L} \pi_{\ell'} \varphi(\boldsymbol{y}; \boldsymbol{\mu}_{\ell'}, \boldsymbol{\Sigma}_{\ell'} + \sigma_n^2 \boldsymbol{I})}.
\end{aligned} \tag{S4}$$

To derive this result, we start by invoking the law of total probability, marginalizing over an auxiliary random variable c that selects one of the $L$ distributions with probabilities $\{\pi_1, \ldots, \pi_L\}$,

$$p_{\mathbf{x}}(\boldsymbol{x}) = \sum_{\ell=1}^{L} p_{\mathbf{x}|\mathrm{c}}(\boldsymbol{x}|\ell) p_{\mathrm{c}}(\ell). \tag{S5}$$

The posterior distribution can then be written as

$$p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y}) = \sum_{\ell=1}^{L} p_{\mathbf{x}|\mathbf{y},\mathrm{c}}(\boldsymbol{x}|\boldsymbol{y}, \ell) p_{\mathrm{c}|\mathbf{y}}(\ell|\boldsymbol{y}). \tag{S6}$$

The term $p_{\mathrm{c}|\mathbf{y}}(\ell|\boldsymbol{y})$ in Eq. (S6) can be written using Bayes' rule as

$$\begin{aligned}
\tilde{\pi}_\ell(\boldsymbol{y}) &\triangleq p_{\mathrm{c}|\mathbf{y}}(\ell|\boldsymbol{y}) \\
&= \frac{p_{\mathbf{y}|\mathrm{c}}(y|\ell) p_{\mathrm{c}}(\ell)}{p_{\mathbf{y}}(\boldsymbol{y})} \\
&= \frac{p_{\mathbf{y}|\mathrm{c}}(\boldsymbol{y}|\ell) p_{\mathrm{c}}(\ell)}{\sum_{\ell'=1}^{L} p_{\mathbf{y}|\mathrm{c}}(\boldsymbol{y}|\ell') p_{\mathrm{c}}(\ell')} \\
&= \frac{\pi_\ell \varphi(\boldsymbol{y}; \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell + \sigma_n^2 \boldsymbol{I})}{\sum_{\ell'=1}^{L} \pi_{\ell'} \varphi(\boldsymbol{y}; \boldsymbol{\mu}_{\ell'}, \boldsymbol{\Sigma}_{\ell'} + \sigma_n^2 \boldsymbol{I})}.
\end{aligned} \tag{S7}$$

Moreover, to explicitly express $p_{\mathbf{x}|\mathbf{y},\mathrm{c}}(\boldsymbol{x}|\boldsymbol{y}, \ell)$ in Eq. (S6), we note that $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian given c, because under a particular choice c $= \ell$, the random vector y is the sum of two independent Gaussian vectors, $\mathbf{x}$ and $\mathbf{n}$. Hence, conditioned on the event c $= \ell$, the joint distribution of $\mathbf{x}$

and $\mathbf{y}$ is given by

$$p_{\mathbf{x},\mathbf{y}|\mathrm{c}}(\boldsymbol{x},\boldsymbol{y}|\ell)=\mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu}_\ell\\ \boldsymbol{\mu}_\ell\end{pmatrix},\begin{pmatrix}\boldsymbol{\Sigma}_{\mathbf{xx}|\mathrm{c}}(\ell) & \boldsymbol{\Sigma}_{\mathbf{xy}|\mathrm{c}}(\ell)\\ \boldsymbol{\Sigma}_{\mathbf{xy}|\mathrm{c}}(\ell) & \boldsymbol{\Sigma}_{\mathbf{yy}|\mathrm{c}}(\ell)\end{pmatrix}\right)$$
$$=\mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu}_\ell\\ \boldsymbol{\mu}_\ell\end{pmatrix},\begin{pmatrix}\boldsymbol{\Sigma}_\ell & \boldsymbol{\Sigma}_\ell\\ \boldsymbol{\Sigma}_\ell & \boldsymbol{\Sigma}_\ell+\sigma_n^2\boldsymbol{I}\end{pmatrix}\right). \quad \text{(S8)}$$

From the joint distribution in Eq. (S8), the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ and c is derived as

$$p_{\mathbf{x}|\mathbf{y},\mathrm{c}}(\boldsymbol{x}|\boldsymbol{y},\ell)=\varphi(\boldsymbol{x};\tilde{\boldsymbol{\mu}}_\ell(\boldsymbol{y}),\tilde{\boldsymbol{\Sigma}}_\ell(\boldsymbol{y})), \quad \text{(S9)}$$

with $\tilde{\boldsymbol{\mu}}_\ell(\boldsymbol{y})$ and $\tilde{\boldsymbol{\Sigma}}_\ell(\boldsymbol{y})$ defined as in Eq. (S4).

### A.3. Projected posterior distribution

Given that the posterior distribution is a GMM as in Eq. (S3), it is straightforward to derive the *projected posterior distribution* (PPD) over any arbitrary 1D subspace defined by a center $\boldsymbol{x}_0$ and a normalized direction $\boldsymbol{w}$. This is done by projecting each of the Gaussians individually, *i.e.,* the PPD for $v=\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{x}_0)$ is given by

$$p_{\mathrm{v}|\mathbf{y}}(v|\boldsymbol{y})=\sum_{\ell=1}^{L}\tilde{\pi}_\ell(\boldsymbol{y})\cdot\varphi(v;\boldsymbol{w}^T(\tilde{\boldsymbol{\mu}}_\ell(\boldsymbol{y})-\boldsymbol{x}_0),\boldsymbol{w}^T\tilde{\boldsymbol{\Sigma}}_\ell(\boldsymbol{y})\boldsymbol{w}). \quad \text{(S10)}$$

During both training and testing, we took $\boldsymbol{x}_0$ to be the posterior mean for a given $\boldsymbol{y}$,

$$\boldsymbol{x}_0(\boldsymbol{y})=\sum_{\ell=1}^{L}\tilde{\boldsymbol{\mu}}_\ell(\boldsymbol{y}), \quad \text{(S11)}$$

and $\boldsymbol{w}$ to be a random normalized direction.

## B. PPD learning with an EBM

For the purpose of training the conditional EBM, we used a variant of contrastive divergence (CD), proposed in [18]. This method attempts to model the log distribution of the dataset by employing a series of distributions that gradually transition between the distribution of the dataset to some reference Gaussian distribution with known parameters. We refer to this method here as *multilevel CD* (MCD) and review it only briefly for completeness. For a more detailed description please refer to [18].

At the core of MCD lies a series of distributions defined by (i) coefficients $\{\alpha_t\}_{t=1}^{T}$ gradually decreasing from $\alpha_1=1$ to $\alpha_T=0$, (ii) a discrete random variable $\mathrm{t}\in\{1,\dots,T\}$, with some chosen prior distribution $p_\mathrm{t}$, and (iii) a reference Gaussian vector $\mathbf{n}$. During training, a new random variable $\tilde{\mathbf{x}}$ is defined as a linear mixture of $\mathbf{x}$ and $\mathbf{n}$ with random linear coefficients associated with $\alpha_\mathbf{t}$ (note that $\mathbf{t}$ is a random variable),

$$\tilde{\mathbf{x}}=\alpha_\mathbf{t}\mathbf{x}+\sqrt{1-\alpha_\mathbf{t}^2}\,\mathbf{n}. \quad \text{(S12)}$$

The surrogate objective of MCD is to model the mixed distribution of the pair $(\tilde{\mathbf{x}},\mathrm{t})$. This is done using a neural network $\boldsymbol{f}(\tilde{\boldsymbol{x}};\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, which receives $\tilde{\boldsymbol{x}}$ as input, and outputs $T$ scalars, where the $t$-th output of the network models the log-probability

$$f_t(\tilde{\boldsymbol{x}};\boldsymbol{\theta})\approx\log p_{\tilde{\mathbf{x}},\mathrm{t}}(\tilde{\boldsymbol{x}},t)=\log p_{\tilde{\mathbf{x}}|\mathrm{t}}(\tilde{\boldsymbol{x}}|t)+\log p_\mathrm{t}(t). \quad \text{(S13)}$$

The distribution of $\mathbf{x}$ can then be extracted from the model at $t=1$ as

$$f_1(\boldsymbol{x};\boldsymbol{\theta})\approx\log p_\mathbf{x}(\boldsymbol{x})+\log p_\mathrm{t}(1). \quad \text{(S14)}$$

The motivation for modeling the pair $(\tilde{\mathbf{x}},\mathrm{t})$, as opposed to just modeling $\mathbf{x}$, is to improve the accuracy of CD in modeling low-density regions of $\log p_\mathbf{x}(\boldsymbol{x})$. During training, the model is only exposed to samples $\boldsymbol{x}$ coming from regions of high probability. Therefore, without further modifications, it usually fails to extrapolate the density function beyond these regions correctly. Training on pairs $(\tilde{\mathbf{x}},\mathrm{t})$ generated by sampling $\boldsymbol{n}$ and $t$ from their known distribution and $\boldsymbol{x}$ from the dataset, MCD can better overcome this challenge.

The update step of MCD combines the standard CD update at any fixed $t$ with a classification update for $t$ given $\tilde{\boldsymbol{x}}$. The CD update at a fixed $t$ is derived by applying the classical CD algorithm to the $t$-th output of the network, which models $\log p_{\tilde{\mathbf{x}}|\mathrm{t}}$ (up to the known term $\log p_\mathrm{t}(t)$). At each training step, the modeled $\log p_{\tilde{\mathbf{x}}|\mathrm{t}}$ is used to produce an MCMC process starting at $\tilde{\boldsymbol{x}}$ and ending at a contrastive sample $\tilde{\boldsymbol{x}}_{\text{neg}}$. The used MCMC process was Langevin dynamics [17] with a Metropolis-Hastings rejection step similar to [2, 11]. The parameters of the model were then updated according to the CD algorithm, following the negative gradient of $f_t(\tilde{\boldsymbol{x}}_{\text{neg}};\boldsymbol{\theta})-f_t(\tilde{\boldsymbol{x}};\boldsymbol{\theta})$.

As for the classification update, it was derived by viewing the network as a classifier, minimizing the cross-entropy loss. This holds since a $\mathrm{Softmax}$ operation on the network outputs results in the estimation of the conditional distribution $p_{\mathrm{t}|\tilde{\mathbf{x}}}$

$$\mathrm{Softmax}(\boldsymbol{f}(\tilde{\boldsymbol{x}};\boldsymbol{\theta}))_t\approx\frac{p_{\tilde{\mathbf{x}},\mathrm{t}}(\tilde{\boldsymbol{x}},t)}{\sum_{t'=1}^{T}p_{\tilde{\mathbf{x}},\mathrm{t}}(\tilde{\boldsymbol{x}},t')}=p_{\mathrm{t}|\tilde{\mathbf{x}}}(t|\tilde{\boldsymbol{x}}). \quad \text{(S15)}$$

The combined update rule of MCD was therefore given by

$$\boldsymbol{\theta}^{(k+1)}=\boldsymbol{\theta}^{(k)}-\eta\nabla_{\boldsymbol{\theta}}\left[f_t(\tilde{\boldsymbol{x}}_{\text{neg}};\boldsymbol{\theta})-f_t(\tilde{\boldsymbol{x}};\boldsymbol{\theta})\right.$$
$$\left.+\mathrm{Softmax}(\boldsymbol{f}(\tilde{\boldsymbol{x}};\boldsymbol{\theta}))_t\right], \quad \text{(S16)}$$

where $\eta$ is the algorithm's learning rate.

Note that thus far the derivation of MCD assumed an unconditional setting where our goal was to model $\log p_\mathbf{x}(\boldsymbol{x})$.

To adopt this training scheme for modeling the PPD, we need to replace $\boldsymbol{x}$ with its projection $\boldsymbol{v} = \boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{x}_0)$ and make the EBM depend on the distorted sample $\boldsymbol{y}$ and the selected subspace $\mathcal{A}(\boldsymbol{y}) = \{\boldsymbol{x}_0(\boldsymbol{y}), \boldsymbol{W}(\boldsymbol{y})\}$. As described in Sec. 3.3 of the main text, this dependency is achieved using a separate feature extractor which outputs a feature vector $\boldsymbol{h}$ conditioning the layers of the EBM through feature normalization.

## C. Datasets and restoration tasks

We evaluated our method on a range of datasets and tasks spanning common scenarios in low-level vision. Specifically, we used the MNIST dataset [1] ($28 \times 28$), the CelebA dataset [9] (cropped and resized to $160 \times 160$ as in SR-Flow [10]), the CelebA-HQ dataset [6] ($256 \times 256$), the ImageNet 1K dataset [14] ($128 \times 128$) and an a cells microscopy dataset presented in [15]. The tasks we experimented with were:

- **MNIST inpainting**. Recovering digit images from only the bottom 8 rows.
- **MNIST denoising**. Denoising digit images contaminated with additive white Gaussian noise with $\sigma_n = 1$ and clipping.
- **CelebA-HQ inpainting**. Recovering a rectangular area of size $70 \times 175$ around the eyes in CelebA-HQ images of size $256 \times 256$.
- **CelebA-HQ colorization**. Recovering RGB images from grayscale images obtained by averaging color channels.
- **CelebA super-resolution**. $8\times$ super-resolution of face images downsampled with a bicubic kernel to $20 \times 20$ pixels as described in [10].
- **ImageNet colorization**. Recovering RGB images from grayscale images obtained by averaging color channels.
- **Biological image-to-image**. Transforming a given microscopic biological imaged specimen with one fluorescent dye appears as if it was imaged by another. We have broken this task into transforming patches of size $64 \times 64$ between the 2 domains.

## D. Architectures and training hyperparameters

The estimation of the projected posterior distribution (PPD) requires the selection of a subspace for each distorted image on which the posterior is to be projected. As described in Sec. 3.2 of the main text, we selected in this work to use a subspace that contains the minimum MSE (MMSE) estimator and is spanned by the first two principal components (PCs) of the posterior. For the purpose of predicting these values we trained two auxiliary networks. Overall, we have sequentially trained three networks for each of the evaluated tasks.

The first network was trained using an MSE loss to output the MMSE estimate for a given distorted image $\boldsymbol{y}$. The second network was trained using NPPC [12] to predict the first five principal components of the posterior given both the distorted image and the MMSE estimate (predicted by the first network). The third network was trained using the method described in this paper to model the PPD.

Each network required a different architecture depending on the task at hand. Overall, for every task, we used a specific subset of the following architectures:

- **U-Net-S**. A small version of the original U-Net architecture from [13] with three encoder and decoder levels containing a single convolution layer each. The number of channels in each level is 32, 64 and 128, respectively. The bottleneck has two convolutions of 256 channels. In addition, we used a group-norm layer after each convolution, LeakyReLU as a non-linearity layer, and a nearest-neighbor interpolation for upsampling.
- **U-Net-L**. A U-Net architecture similar to the one used in DDPM [4] for the CelebA generative task, but with half the number of channels. We also removed all network parts related to the time index dependency.
- **U-Net-M**. A network similar to U-net-L with one level less in the encoder and decoder (the top level, in terms of the number of channels).
- **EDSR**. An architecture similar to the EDSR architecture described in [8] using 16 residual blocks and a width of 64 channels.
- **cEBM-S**. This network is comprised of two parts, as described in Sec. 3.3 in the main text. The first part is a feature extractor based on a ResNet18 architecture [3], fitted for smaller images. We removed the stride in the input convolution and the following pooling layer and reduced the number of channels by $2\times$. In addition, we replaced the BatchNorm layers with GroupNorm layers and performed the downsampling using average pooling layers (instead of stride convolutions). The output of this network is a feature vector of length 256. The second part of the network is a conditional MLP with 6 hidden layers of width 128. We used SiLU as the non-linearity followed by an AdaIN [5] layer that performs adaptive scaling and shifting. The scaling and shifting parameters are extracted from the feature vector using a linear layer.
- **cEBM-L**. This network is similar to the cEBM-S except that the architecture of the feature extractor is replaced by the encoder part of the U-Net-L, using a global average pooling layer at the end of the bottleneck to produce a feature vector of length 512.

In general, the U-Net architectures were used for the MMSE estimator and the PCs, and the conditional EBMs was used for modeling the PPD. The only exceptions were the networks in the super-resolution task, in which the distorted image was smaller than all other images and therefore was fed into the EDSR architecture either to directly produce the desired output (as in the case of the MMSE) or as a preprocessing step before concatenating it with the other inputs.

All networks were trained using the Adam optimizer, stopping the training process once it reached a minimal value of the objective over a dedicated validation set. For the training of the PC predictor and the PPD model, the learning rate was kept fixed. On the other hand, for training the MMSE estimator, the learning rate was reduced by half every 5000 steps. Table S1 summarizes the architectures and training hyperparameters used for the networks in each task/dataset.

## E. Additional Results

### E.1. PPDs

Figures S1-S7 provide more examples of PPDs captured by PPDE for the datasets and tasks presented in the main text.

### E.2. Comparisons to NPPC and KDE

Figures S8-S13 provide further negative log-likelihood comparisons of PPDE with the respective baselines.

Table S1. Architectures, learning rates, and number of steps used per task/dataset.

| Task | MMSE Arch. | MMSE LR | MMSE # of steps | PCs Arch. | PCs LR | PCs # of steps | PPD Arch. | PPD LR | PPD # of steps |
|---|---|---|---|---|---|---|---|---|---|
| **MNIST** | | | | | | | | | |
| Inpainting | U-Net-S | $10^{-4}$ | 3,510 | U-Net-S | $10^{-4}$ | 5,000 | cEBM-S | $10^{-4}$ | 36,504 |
| Denoising | U-Net-S | $10^{-3}$ | 57,330 | U-Net-S | $10^{-4}$ | 408,000 | cEBM-S | $10^{-3}$ | 1,026,558 |
| **CelebA** | | | | | | | | | |
| Inpainting Eyes | U-Net-L | $10^{-4}$ | 7,555 | U-Net-L | $10^{-5}$ | 22,500 | cEBM-L | $10^{-4}$ | 94,000 |
| Colorization | U-Net-L | $3 \cdot 10^{-5}$ | 9,066 | U-Net-L | $3 \cdot 10^{-5}$ | 38,500 | cEBM-L | $10^{-4}$ | 44,500 |
| Super-resolution | EDSR | $10^{-4}$ | 287,651 | EDSR + U-Net-L | $10^{-5}$ | 113,500 | EDSR + cEBM-L | $10^{-4}$ | 203,250 |
| **ImageNet** | | | | | | | | | |
| Colorization | U-Net-M | $3 \cdot 10^{-4}$ | 187,200 | U-Net-M | $10^{-5}$ | 37,600 | cEBM-L | $10^{-4}$ | 102,500 |
| **Biological** | | | | | | | | | |
| Image-to-image | Pretrained | | | U-Net-M | $10^{-4}$ | 81,100 | cEBM-L | $10^{-4}$ | 402,800 |

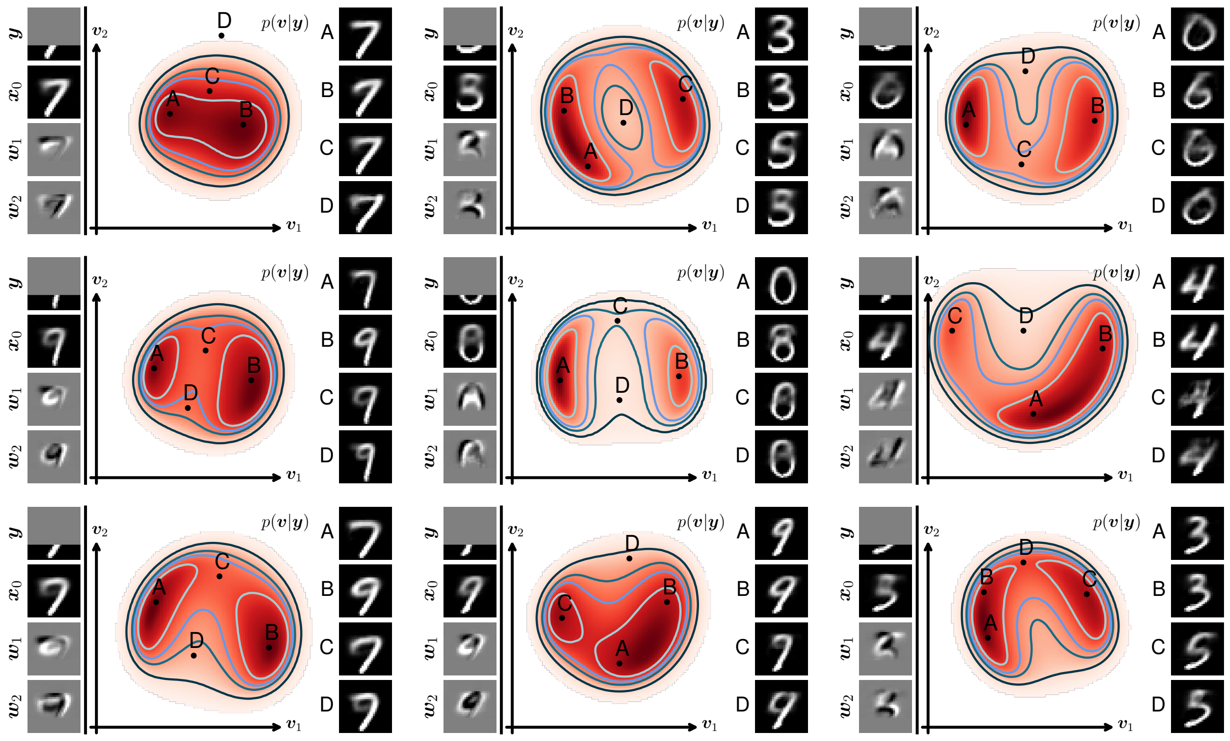

Figure S1. More examples of MNIST inpainting PPDs.

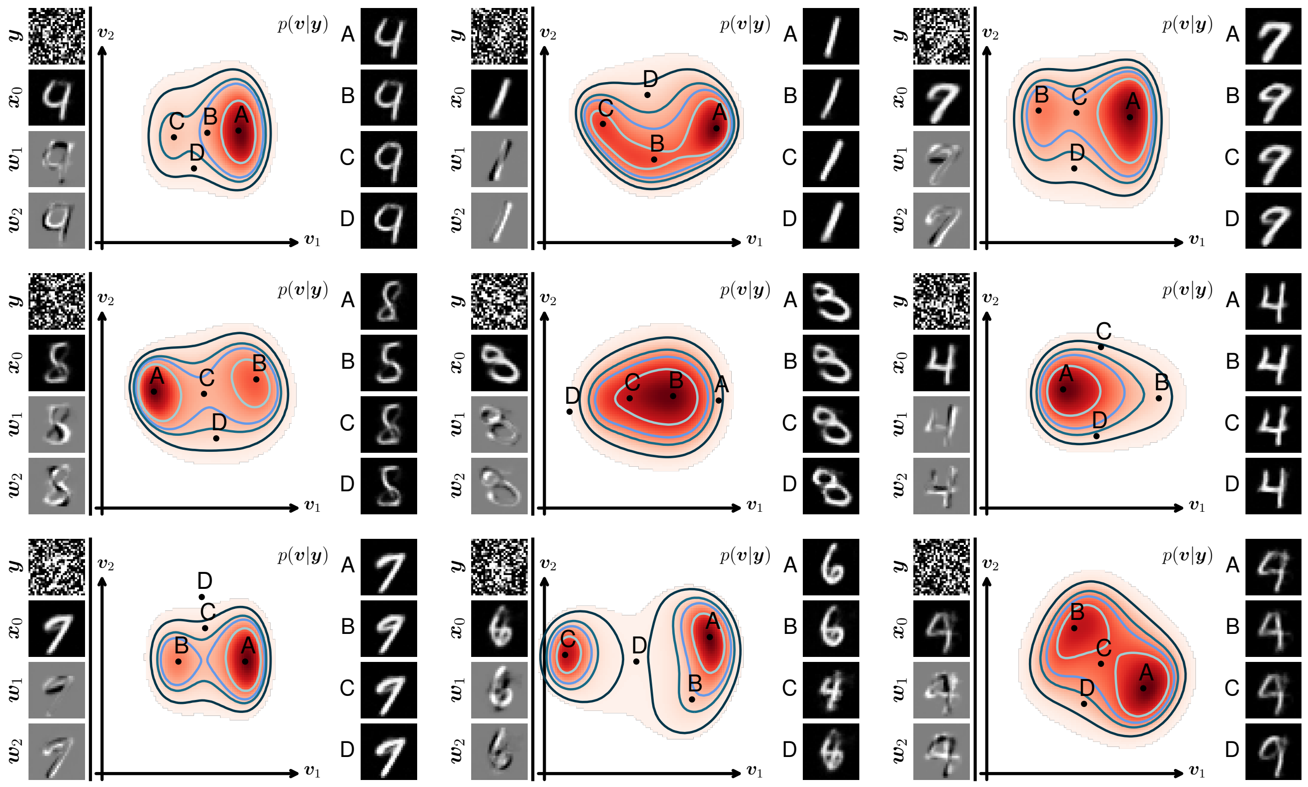

Figure S2. More examples of MNIST denoising.

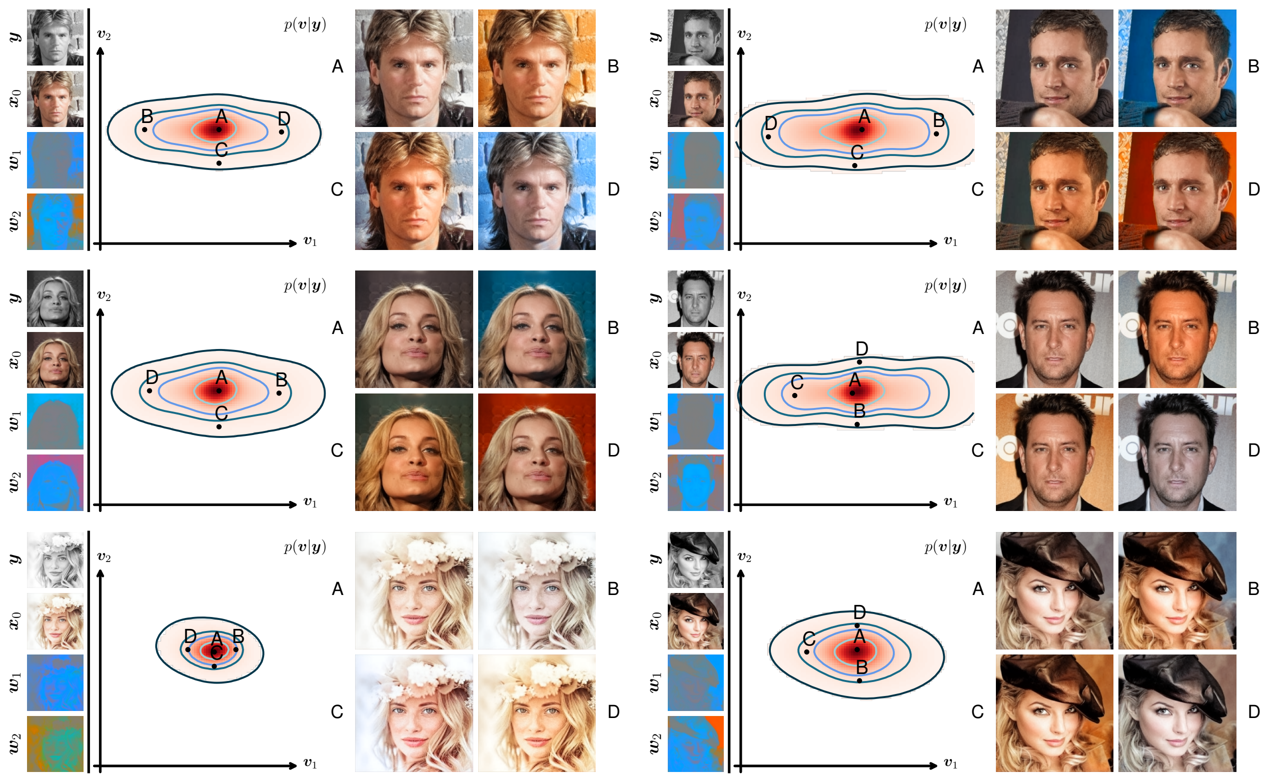

Figure S3. More examples of CelebA-HQ colorization.

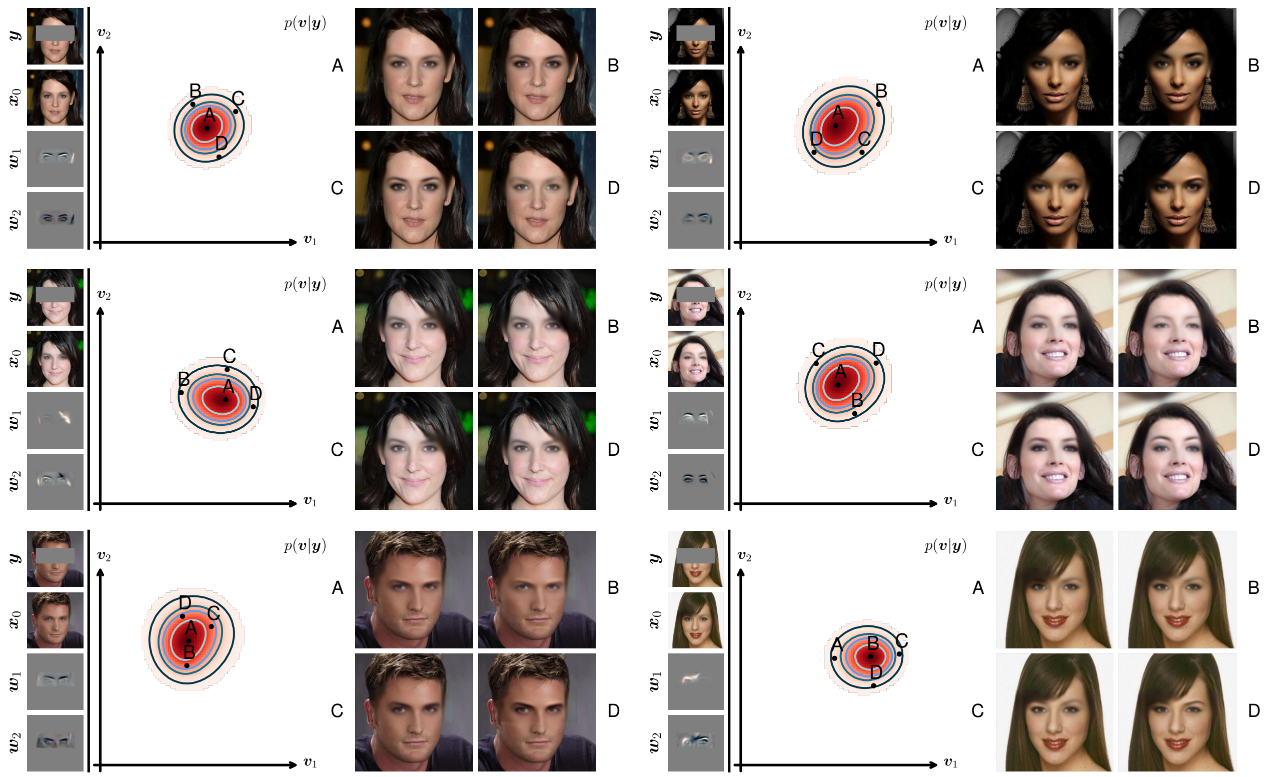

Figure S4. More examples of CelebA-HQ inpainting.

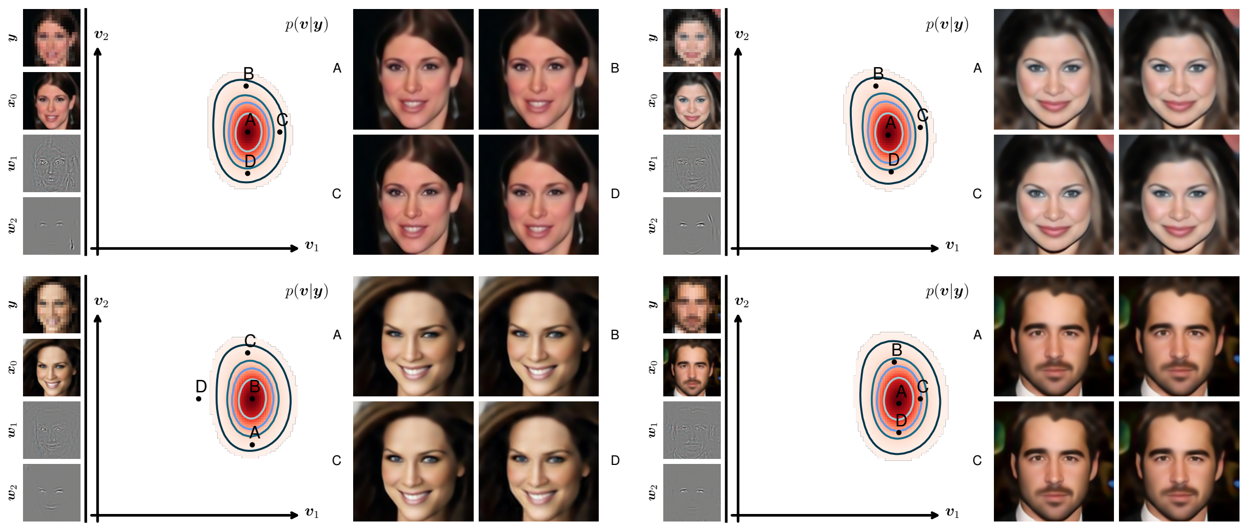

Figure S5. More examples of CelebA 8× super-resolution.

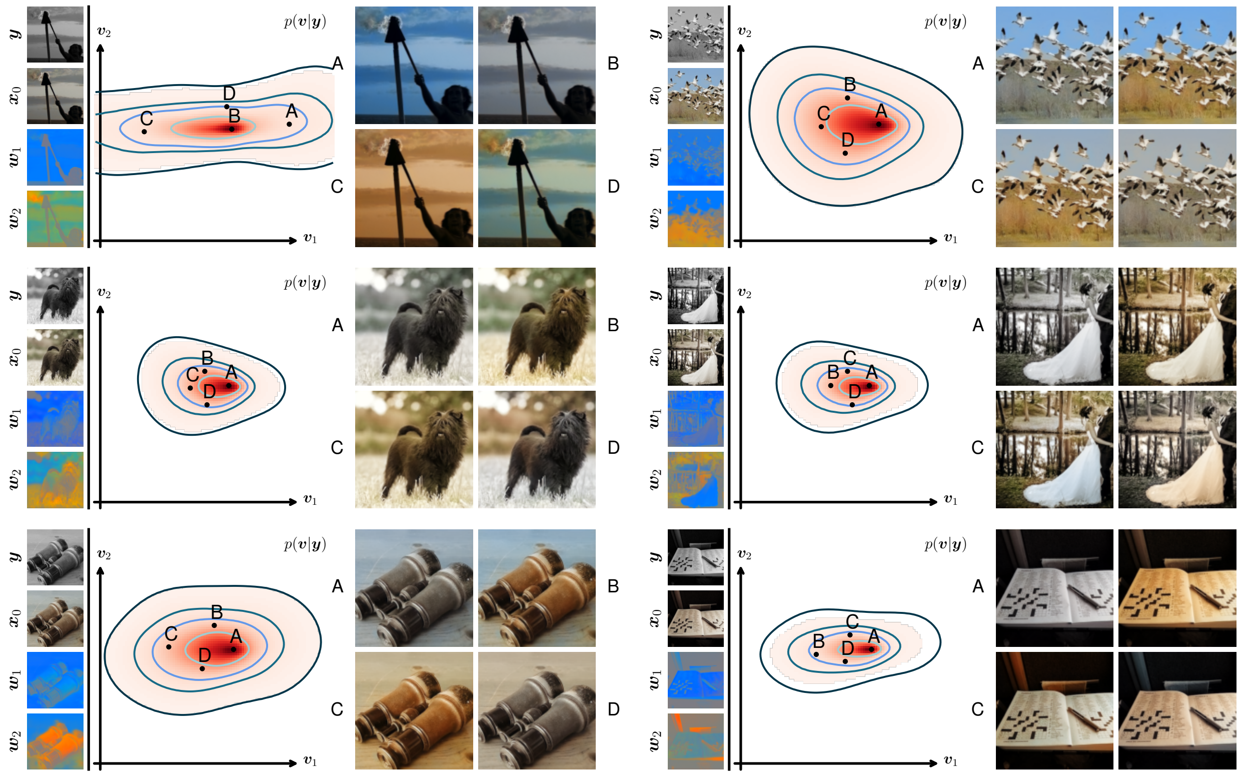

Figure S6. More examples of ImageNet colorizaion.

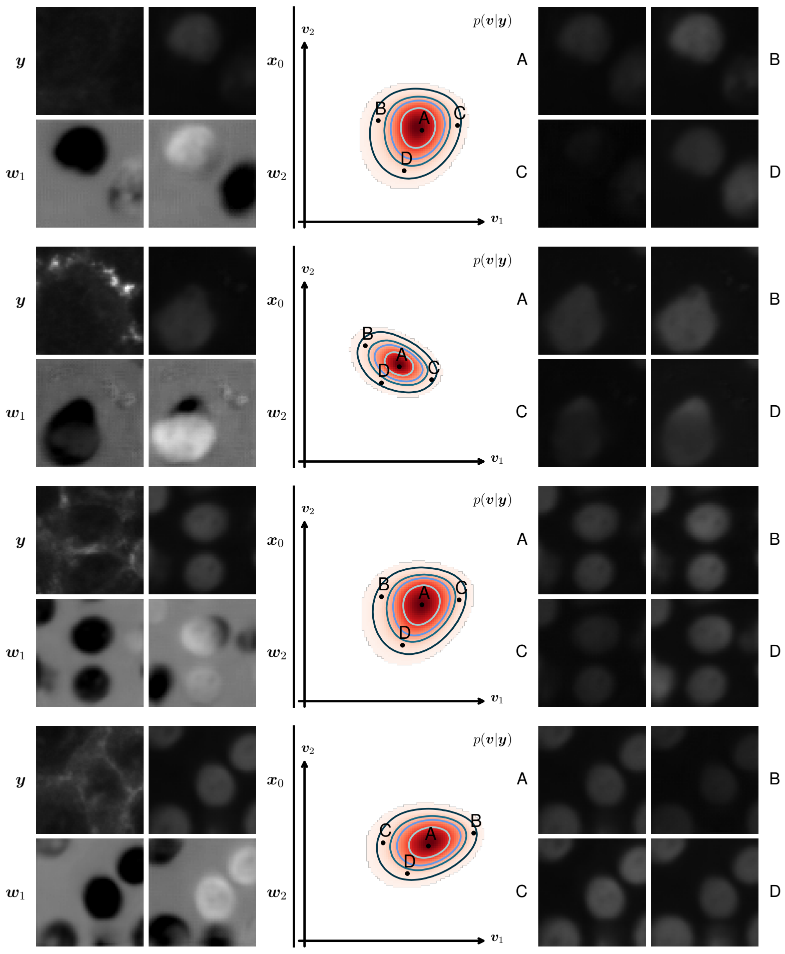

Figure S7. More examples of Biological image-to-image transfer.

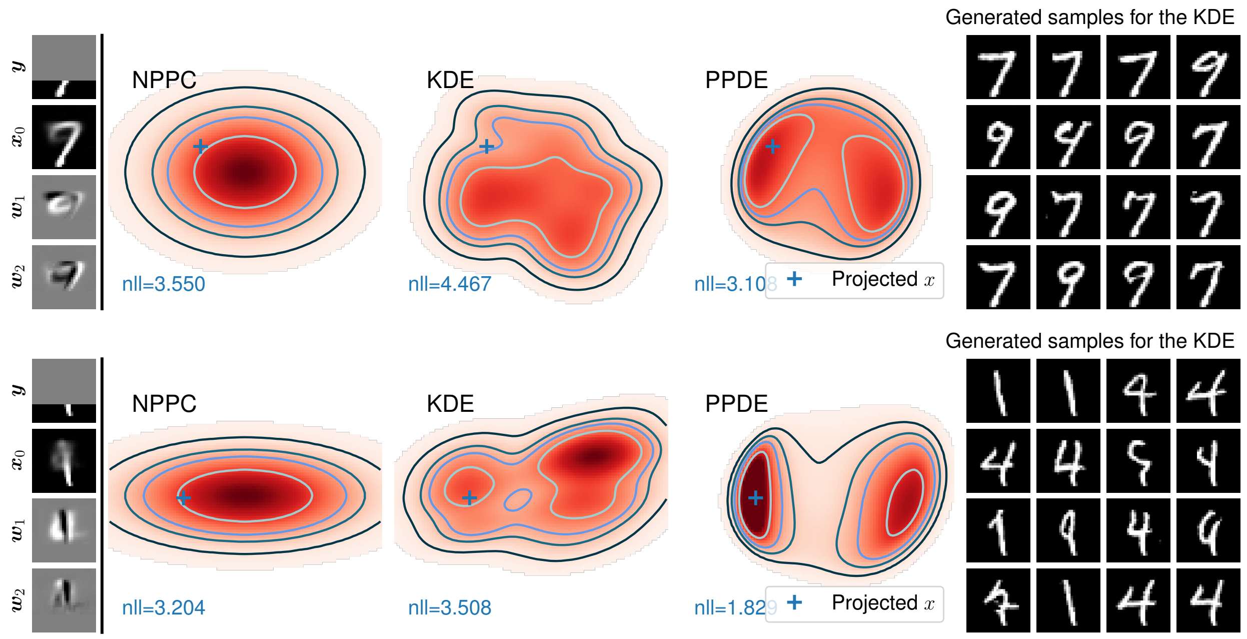

Figure S8. More comparisons of MNIST inpainting PPDs. The posterior samples on the right were obtained using an EBM trained on MNIST.

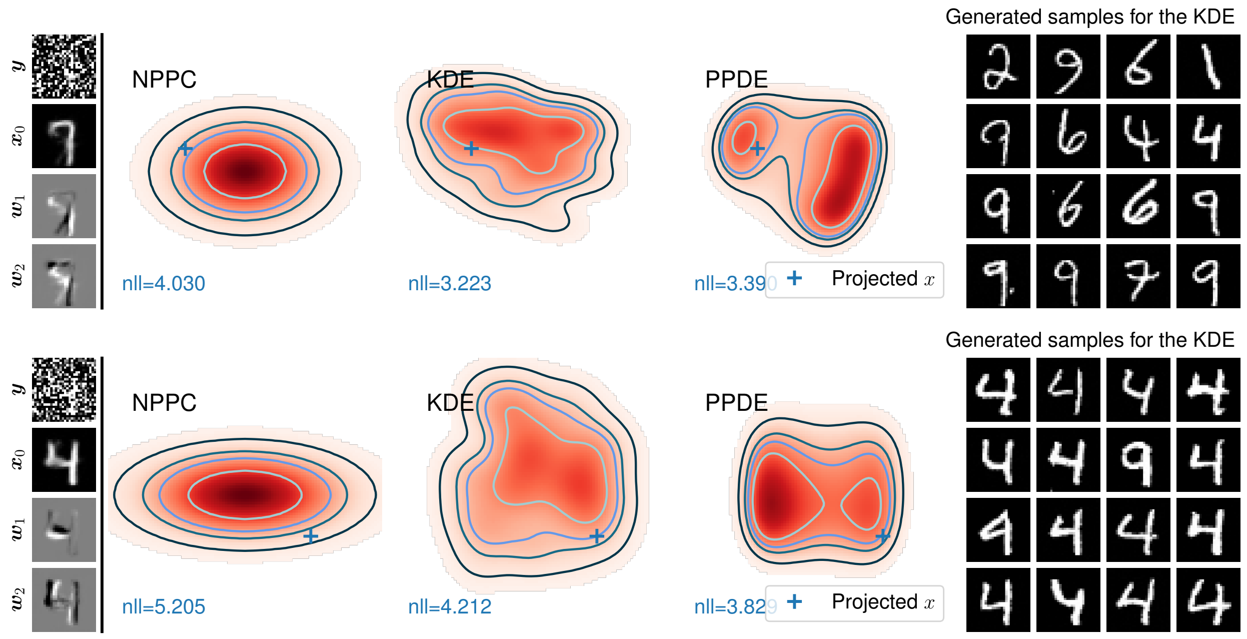

Figure S9. More comparisons of MNIST denoising. The posterior samples on the right were obtained using an EBM trained on MNIST.

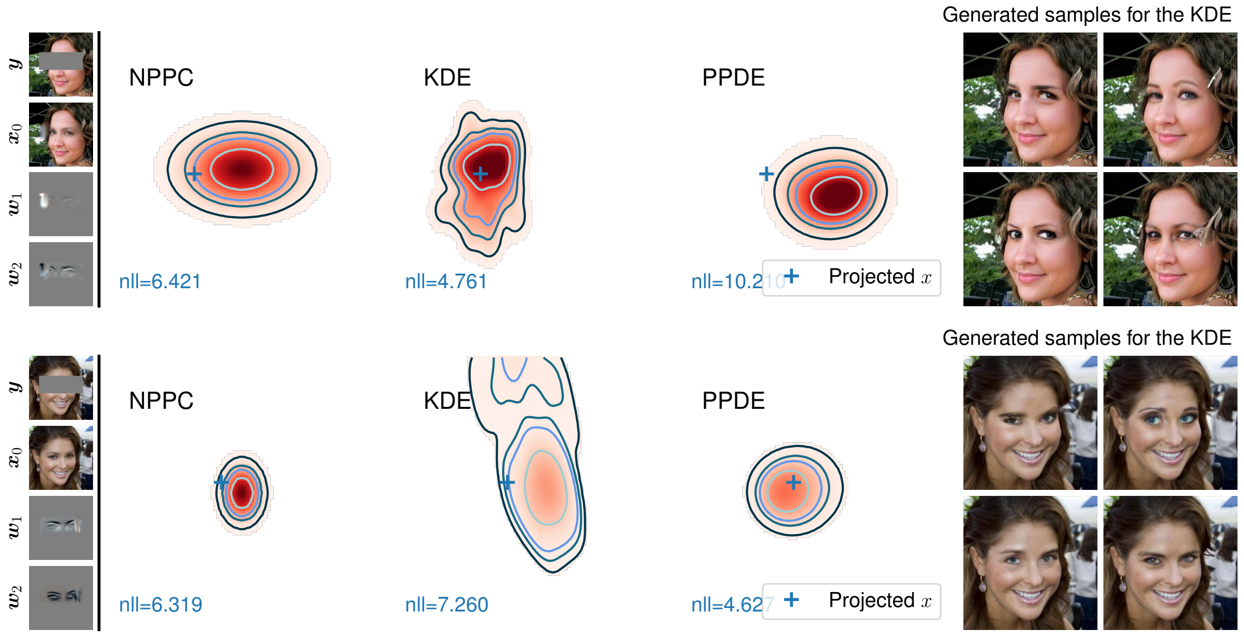

Figure S10. More comparisons of CelebA-HQ inpainting. The posterior samples on the right were obtained using DPS [7].

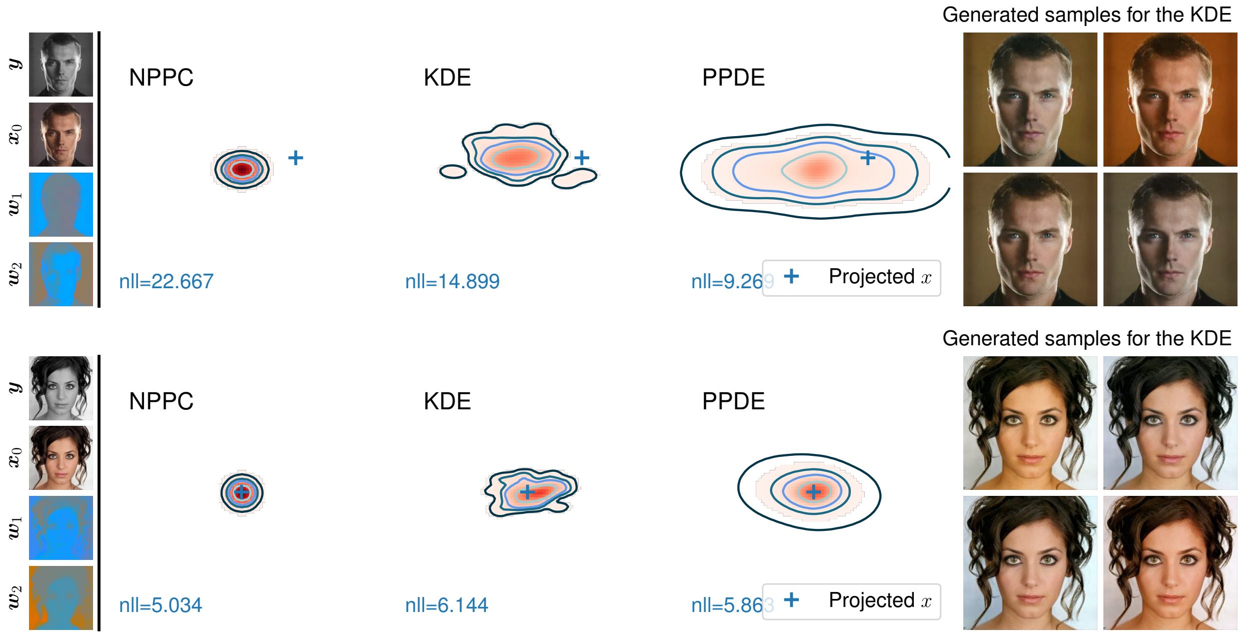

Figure S11. More comparisons of CelebA-HQ colorization. The posterior samples on the right were obtained using DDNM [16].

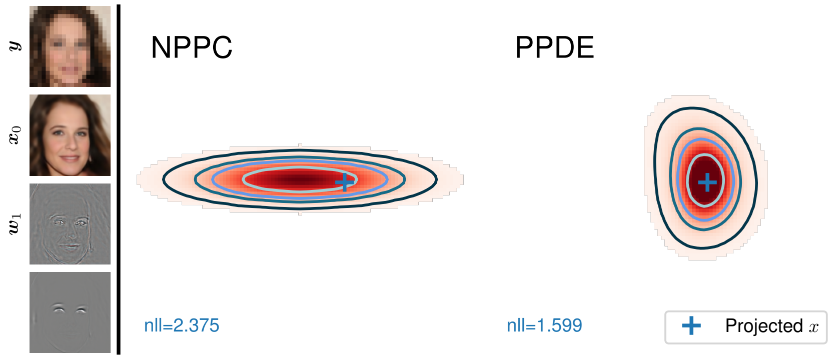

Figure S12. More comparisons of CelebA 8× super-resolution.

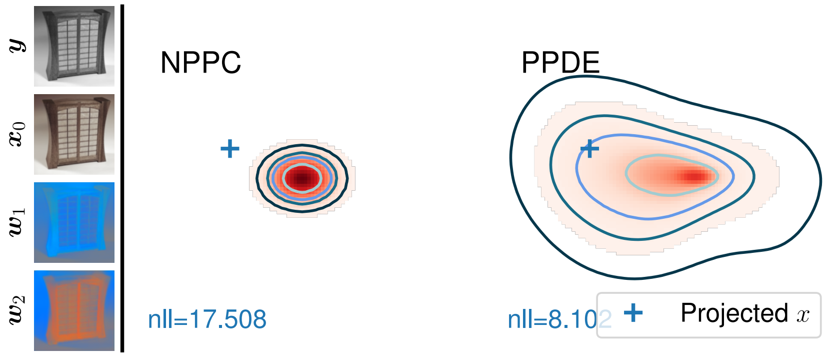

Figure S13. More comparisons of ImageNet colorization.

## References

[1] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 3

[2] WK HASTINGS. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–97, 1970. 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3

[5] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017. 3

[6] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 3

[7] Wenbo Li, Zhe Lin, Kun Zhou, Lu Qi, Yi Wang, and Jiaya Jia. Mat: Mask-aware transformer for large hole image inpainting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10758–10768, 2022. 11

[8] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 3

[9] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 3

[10] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. Srflow: Learning the super-resolution space with normalizing flow. In *ECCV*, 2020. 3

[11] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. 2

[12] Elias Nehme, Omer Yair, and Tomer Michaeli. Uncertainty quantification via neural posterior principal components. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 3

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 3

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 3

[15] Lucas von Chamier, Romain F Laine, Johanna Jukkala, Christoph Spahn, Daniel Krentzel, Elias Nehme, Martina Lerche, Sara Hernández-Pérez, Pieta K Mattila, Eleni Karinou, et al. Democratising deep learning for microscopy with zerocostdl4mic. *Nature communications*, 12(1):2276, 2021. 3

[16] Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. *The Eleventh International Conference on Learning Representations*, 2023. 11

[17] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011. 2

[18] Omer Yair and Tomer Michaeli. Thinking fourth dimensionally: Treating time as a random variable in ebms. *https://openreview.net/forum?id=m0fEJ2bvwpw*, 2022. 2